INFORMATYKA KOMPUTERY SYSTEMY



CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 1/1986

W numerze:

- Świat minikomputerów z bliska
- Cztery programy w BASIC—u na SPECTRUM
- Komputer dla kolekcjonerów
- Liga Myślących
- „Iks" pomaga w poznaniu komputera



NICZEGO już uzasadniać nie trzeba. To co było tylko przyszłością i to odległą, jak dodawali złośliwi, stało się dniem powszednim. Prawdziwą komputerową eksplozję spowodowały mini - i mikrokomputery. Zaczęło się od gier; zabawnych i pasjonujących, wymagających refleksu i logicznego myślenia. Minikomputer sprawił, że rozmowa z „bezduszną maszyną" stała się prostsza i łatwiejsza.

Informatyka jest dziś przedmiotem nauczania w wielu szkołach średnich i we wszystkich liceach wojskowych. Dialog z maszyną zaczyna się wcześnie; komputer jest pomocą dydaktyczną — staje się niezbędny tak jak kreda i tablica. To mała informatyka. Ta wielka — to duże systemy, o rozbudowanych bazach danych, dostarczające informacji szybko i sprawnie.

Cybernetyk to już nie „Wielki Mag" dwudziestego pierwszego wieku, ale inżynier — specjalista jak wielu innych. Jego obowiązkiem jest służyć wiedzą tym, którzy jej potrzebują. My także chcemy mieć w tym swój skromny udział, dlatego też powstał „IKS", który ma uczyć i bawić. Znajdziesz w nim, Czytelniku, programy komputerowe, artykuły problemowe, informacje poświęcone konstrukcyjnym nowinkom, a także możesz wziąć udział w startującej „Lidze Myślących". To oczywiście nie wszystko. Również sam możesz być współredaktorem pisma — redakcja czeka na Twoje propozycje.

Chcemy, takie mamy ambicje, stać się pomostem między marzeniami a możliwościami. Rzeczywistość zweryfikuje nasze plany, ale już obiecać możemy, że najważniejszym kryterium naszego działania będzie szerzenie komputerowej wiedzy, wiedzy, która już jutro będzie wszystkim tak potrzebna jak dziś tabliczka mnożenia.

Komputer może wszystko — a tak naprawdę może tylko tyle, ile pozwoli mu człowiek. Martwe układy elektroniczne trzeba „nauczyć" każdej wykonywanej przez nie czynności — zrobią tylko tyle, ile potrafi zażądać od nich programista — i to właśnie on jest najważniejszy.

Komputer mówi prawdę, a przynajmniej myli się rzadko. Pracuje szybko i nie ulega stresom, potrafi również sprawnie określić, w jakim stanie znajduje się jego rozumny partner. Biorytm dla „IKS-a" to prognoza naszej kondycji na cały rok. Dokładnie 29 stycznia zrodził się pomysł jego wydania. Pierwsze „plusy" we wszystkich rytmach sprawiły, że po kilku tygodniach byliśmy gotowi. Przeczekaliśmy jeszcze dni krytyczne i okres słabszej formy intelektualnej — starannie unikaliśmy jakiejkolwiek koncepcyjnej pracy w „słabsze" dni. Przestrzegać tej zasady będziemy również w przyszłości. I choć uznać można biorytmiczną prognozę za jeden z komputerowych żartów — to właśnie jest on jednym ze sposobów wykorzystania elektronowego mózgu; zamiast kartki papieru, długich obliczeń, wystarczy podać datę swojego urodzenia i cierpliwie poczekać na wynik działania programu.

„IKS" chce być partnerem wszystkich, których zainteresowała informatyka: miłośników nowoczesnej techniki i logicznego myślenia. Oddając „IKS-a" do Twoich, Czytelniku, rąk liczymy, że znajdziemy w Tobie partnera naszej pracy. Jednocześnie podkreślamy, że wszystkie błędy obarczają wyłącznie redakcję, zaś to co uznacie w „IKS-ie" za wartościowe jest owocem pracy wszystkich — redakcji i jej sympatyków, którym w tym miejscu serdecznie dziękujemy.

Życzymy przyjemnej lektury!

REDAKCJA

IKS
ur. 1986.01.29

BIORYTMY NA ROK 1986

+++ DNI POMYSLNE
--- DNI NIEPOMYSLNE
0 DNI FERALNE

	DNI MIESIACA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
01	STAN FIZYCZNY																													+	+	+
	STAN PSYCHICZNY																													+	+	+
	STAN INTELEKTUALNY																													+	+	+
02	STAN FIZYCZNY	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+			
	STAN PSYCHICZNY	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+			
	STAN INTELEKTUALNY	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
03	STAN FIZYCZNY	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-
	STAN PSYCHICZNY	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+
	STAN INTELEKTUALNY	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-
04	STAN FIZYCZNY	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	
	STAN PSYCHICZNY	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	
	STAN INTELEKTUALNY	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	
05	STAN FIZYCZNY	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+
	STAN PSYCHICZNY	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+
	STAN INTELEKTUALNY	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-
06	STAN FIZYCZNY	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	
	STAN PSYCHICZNY	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	
	STAN INTELEKTUALNY	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	
07	STAN FIZYCZNY	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0
	STAN PSYCHICZNY	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-
	STAN INTELEKTUALNY	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-
08	STAN FIZYCZNY	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+
	STAN PSYCHICZNY	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-
	STAN INTELEKTUALNY	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0
09	STAN FIZYCZNY	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	
	STAN PSYCHICZNY	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	
	STAN INTELEKTUALNY	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
10	STAN FIZYCZNY	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0
	STAN PSYCHICZNY	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-
	STAN INTELEKTUALNY	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+
11	STAN FIZYCZNY	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	
	STAN PSYCHICZNY	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	
	STAN INTELEKTUALNY	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	
12	STAN FIZYCZNY	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-
	STAN PSYCHICZNY	-	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+
	STAN INTELEKTUALNY	+	+	+	+	+	+	+	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	+	+	+	+

Problemy mikroelektroniki

W ostatnich latach jesteśmy świadkami dynamicznego rozwoju techniki mikroelektronicznej, która rewolucjonizuje wszystkie dziedziny życia. To lawinowo narastające zjawisko wpływa decydująco na ogólny poziom wiedzy naszego społeczeństwa — dziś ocenia się, że co 8 lat następuje podwojenie naszej wiedzy.

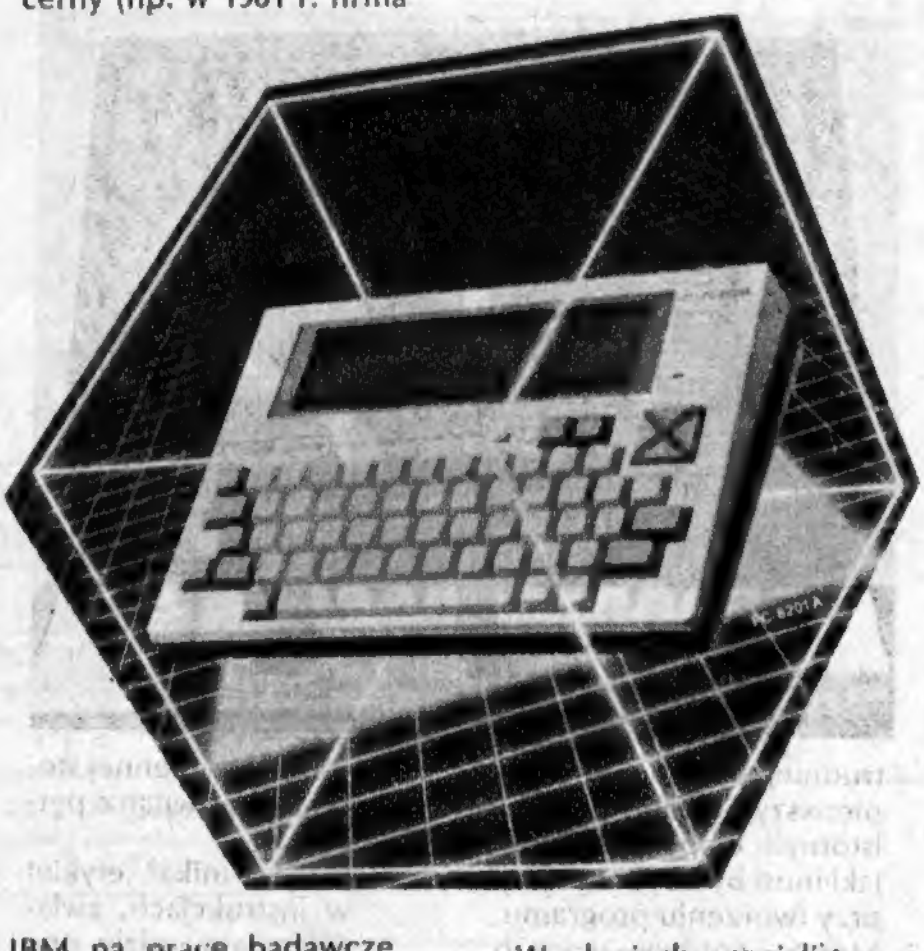
Hasło postęp

W odróżnieniu od pierwszej rewolucji naukowo-technicznej, w wyniku której praca fizyczna została zastąpiona pracą zmechanizowaną, rozwój elektroniki, umożliwiający zastąpienie pracy umysłowej pracą urządzeń elektronicznych, nazwano drugą rewolucją naukowo-techniczną.

Prognozy zawarte w raporcie Klubu Rzymskiego z 1982 r. przewidują, że rewolucja ta wywrze większy wpływ na rozwój cywilizacyjny świata niż pierwsza. Ma to być głównie następstwem takich pozytywnych efektów elektronizacji jak: wzrost wydajności pracy, eliminacja prac niebezpiecznych i uciążliwych, automatyzacja prac rutynowych, wzrost efektywności kontroli całego procesu produkcji i poszczególnych stanowisk pracy.

Mikroelektronika jest dziedziną nauki i techniki zajmującą się zminiaturyzowanymi układami elektronicznymi, wykonywanymi w postaci scalonej przy całkowitym lub częściowym braku elementów indywidualnych.

Kolejnymi etapami w rozwoju mikroelektroniki było uruchomienie produkcji tranzystorów (1948 r.), układów scalonych (1960 r.), układów scalonych wielkiej skali integracji (1970 r.) oraz mikroprocesorów (1972 r.). Obecnie prace nad technikami i technologiami mikroprocesorowymi są rozwijane dzięki dużym subwencjom rządowym. Mimo wielkich możliwości finansowych, jakimi dysponują elektroniczne koncerny (np. w 1981 r. firma



IBM na prace badawcze przeznaczyła 1,6 mld dolarów tj. 5,5 proc. wartości sprzedaży swoich wyrobów) rządy Stanów Zjednoczonych, Japonii i Francji przeznaczają dodatkowo wielkie nakłady na przyspieszenie rozwoju przemysłu elektronicznego.

Informatyka, której rozwój jest w coraz większym stopniu stymulowany wprowadzaniem układów mikroeletronicznych o wielkim i bardzo wielkim stopniu scalenia, staje się jednym z głównych czynników postępu naukowo-technicznego. Już obecnie jako wskaźnik uprzemysłowienia przyjmuje się liczbę skutecznie eksploatowanych komputerów (mini- i mikrokomputerów). Średnie nakłady ponoszone na wdrażanie oraz zastosowanie metod i środków informatyki we Francji i w Stanach Zjednoczonych wynosiły w latach 1976—1980 odpowiednio 2,3 proc. i 3,1 proc. produktu narodowego brutto, zaś w 1985 r. wzrosły do 3,5 proc. i 4 proc., mimo że gospodarki tych krajów są w wysokim stopniu nasycone sprzętem komputerowym.

W krajach socjalistycznych wyraźne przyspieszenie postępu w dziedzinie mikroelektroniki nastąpiło w latach siedemdziesiątych, przede wszystkim w wyniku wzrostu nakładów inwestycyjnych oraz zwiększenia zdolności produkcyjnych przemysłu elektronicznego. Było to szczególnie widoczne w Polsce po opracowaniu i wprowadzeniu w życie *„Programu elektronizacji gospodarki narodowej do 1990 r."* (Uchwała nr 175/75 Rady Ministrów z 26 września 1975 r.). Zjawiska kryzysowe przełomu lat siedemdziesiątych i osiemdziesiątych przerwały na pewien czas rozwój przemysłu elektronicznego, a inwestycje w tej dziedzinie zostały wstrzymane. W tym samym czasie obserwuje się dynamiczny rozwój bazy elektronizacji w innych krajach socjalistycznych. Na przykład w NRD w drugiej połowie lat siedemdziesiątych, zgodnie z decyzjami najwyższych władz partyjnych i państwowych, nastąpiła dalsza intensyfikacja prac nad mikroelektroniką ze szczególnym uwzględnieniem układów mikroprocesorowych. Aktualnie w NRD nad rozwojem nowoczesnych układów scalonych wielkiej skali integracji pracuje trzy razy większy potencjał ludzki niż w Polsce. Bezpośrednia współpraca z innymi partnerami z RWPG, a zwłaszcza z CSRS i LRB wskazuje, że w krajach tych środki finansowe przeznaczone w ostatnich 5—8 latach na rozwój mikroelektroniki są znacznie wyższe niż w PRL. To samo dotyczy zastosowań informatyki — w czym prawie wszystkie kraje socjalistyczne nas wyprzedziły; liczba komputerów przypadających na 1 mln mieszkańców wynosiła w 1980 r. w CSRS — 81 sztuk, w WRL — 51; NRD — 47, LRB — 28, w Polsce — 24 sztuki.

Miernikiem postępu w elektronice jest poziom produkcji układów scalonych wielkiej i bardzo wielkiej skali integracji, czyli mikroprocesorów. Mikroprocesory 4-bitowe mogą służyć do konstruowania prostych mikrokomputerów przeznaczonych do sterowania sprzętem powszechnego użytku, jak odbiorniki radiowe i telewizyjne, pralki automatyczne, maszyny do szycia, kalkulatory itp. Mikroprocesory 8-bitowe służą do stosunkowo prostych zastosowań profesjonalnych, takich jak sterowanie perwferyjnym

cd. na str. 8

Co trzeba wiedzieć?

Przystępując do pisania programu, każdy człowiek realizujący to zadanie powinien sobie wyobrazić wielkość pracy koncepcyjnej, jaką musi wykonać, aby produkt finalny, program był napisany szybko i dokładnie oraz aby odzwierciedlał pierwotne zamierzenie twórcy.

PRZEDE wszystkim trzeba zrozumieć problem. Sformułowanie zadania nie może bowiem pozostawiać wątpliwości co do jego sensowności, poprawności formalnej i merytorycznej oraz kompletności sformułowań. Zdefiniowanie problemu określa i zawęża zagadnienie do wymagań, jakie przed nim stawiamy.

Drugim, niezmiernie ważnym etapem jest analiza problemu i dobór metody rozwiązania. Od człowieka, który zna problem mający być oprogramowany zależy wybór właściwej i najlepszej metody jego rozwiązania. Kolejnym etapem, bardzo często zaniedbywanym lub wręcz lekceważonym, jest opracowanie algorytmu (przepisu) rozwiązania problemu. Jest to pierwszy bardzo poważny moment przybliżający człowieka i jego problem do komputera. Tu bowiem problem zostaje w oparciu o znajomość sprzętu komputerowego i języka programowania przeniesiony na konkretny, sformalizowany sposób interpretacji rozwiązania (tablica decyzyjna, wzór matematyczny, schemat blokowy, opis słowny). Opracowując algorytm każdy człowiek piszący program (programista) powinien bardzo dokładnie zweryfikować, czy to, co zostało przez niego przedstawione w postaci sformalizowanej interpretacji zadania odpowiada w zupełności problemowi, który miał oprogramować. Każde odstępstwo lub różnicę należy natychmiast skorygować — nieuczynienie tego spowoduje przeniesienie błędu na dalsze etapy, w których jego wykrycie może być o wiele trudniejsze. Etap ten jest pierwszym, ale bardzo istotnym etapem kontroli, jaki musi być wykonywany przy tworzeniu programu. Mając gotowy sposób rozwiązania problemu przystępuje się do jego kodowania, czyli przenoszenia na język komputerowy. W trakcie kodowania programu należy pamiętać o pewnej prostej zasadzie: **zadaniem programisty jest napisanie kodu, którego odbiorcą jest przede wszystkim człowiek, a dopiero później komputer.** Trzeba zatem skupić się na jasności, prostocie i czytelności kodu. W trakcie pisania programu:



1) należy używać znaczących nazw zmiennych — każda nazwa zmiennej powinna być unikalna tak, aby wykluczyć możliwości pomyłki w tym zakresie;
2) jeżeli w identyfikatorach występują cyfry, należy umieścić je na końcu nazwy;
3) nie należy używać jako identyfikatorów słów kluczowych języka programowania;
4) nie należy używać zbyt dużo zmiennych roboczych;
5) dla jednoznaczności określeń używać należy nawiasów;
6) należy pisać tylko jedną instrukcję w wierszu;
7) nie należy zmieniać wartości zmiennej sterującej wewnątrz pętli;
8) należy unikać etykiet w instrukcjach, zwłaszcza tam, gdzie one są zbędne — jeżeli do instrukcji nie prowadzi skok, nie należy poprzedzać jej etykietą;
9) należy wykorzystywać w maksymalnym stopniu oprogramowanie standardowe;
10) należy komentować wszystkie zmienne programu;
11) należy deklarować wszystkie zmienne programu w sposób jawny;
12) nie należy używać jednej zmiennej dla różnych celów;
13) nie należy używać różnych nazw dla tego samego obszaru pamięci;

KOLEJNY etap to wprowadzenie programu do komputera. Podczas tej czynności należy zwracać baczną uwagę na to, czy postać wprowadzonej procedury (instrukcji) jest właściwa — błąd trzeba natychmiast korygować. Od tej pory można przyjąć, że podstawowe błędy w składni będzie wykrywał komputer, co nie zwalnia programisty od stałej weryfikacji poprawności programu. Z chwilą otrzymania poprawnego (pod względem formalnym) programu następuje najbardziej pracochłonna czynność procesu programowania: testowanie i uruchomienie programu. Pamiętać tu należy, że od jakości testu oraz rzetelności analizy jego wyników przez programistę zależy jakość programu, a także czas poświęcony na jego weryfikację.

Jednym z mierników oceny biegłości programisty jest jego zdolność do znajdowania i poprawiania własnych błędów. Początkujący programiści nie potrafią szukać i poprawiać swoich błędów, umieją to natomiast programiści doświadczeni. Programistów uczy się często programowania, rzadko natomiast uruchamiania programu. Uruchomienie programu trwa dłużej i jest o wiele trudniejsze niż samo pisanie programu. Oto wskazówki, które programiści powinni przeanalizować przed przystąpieniem do testowania i uruchamiania programu:

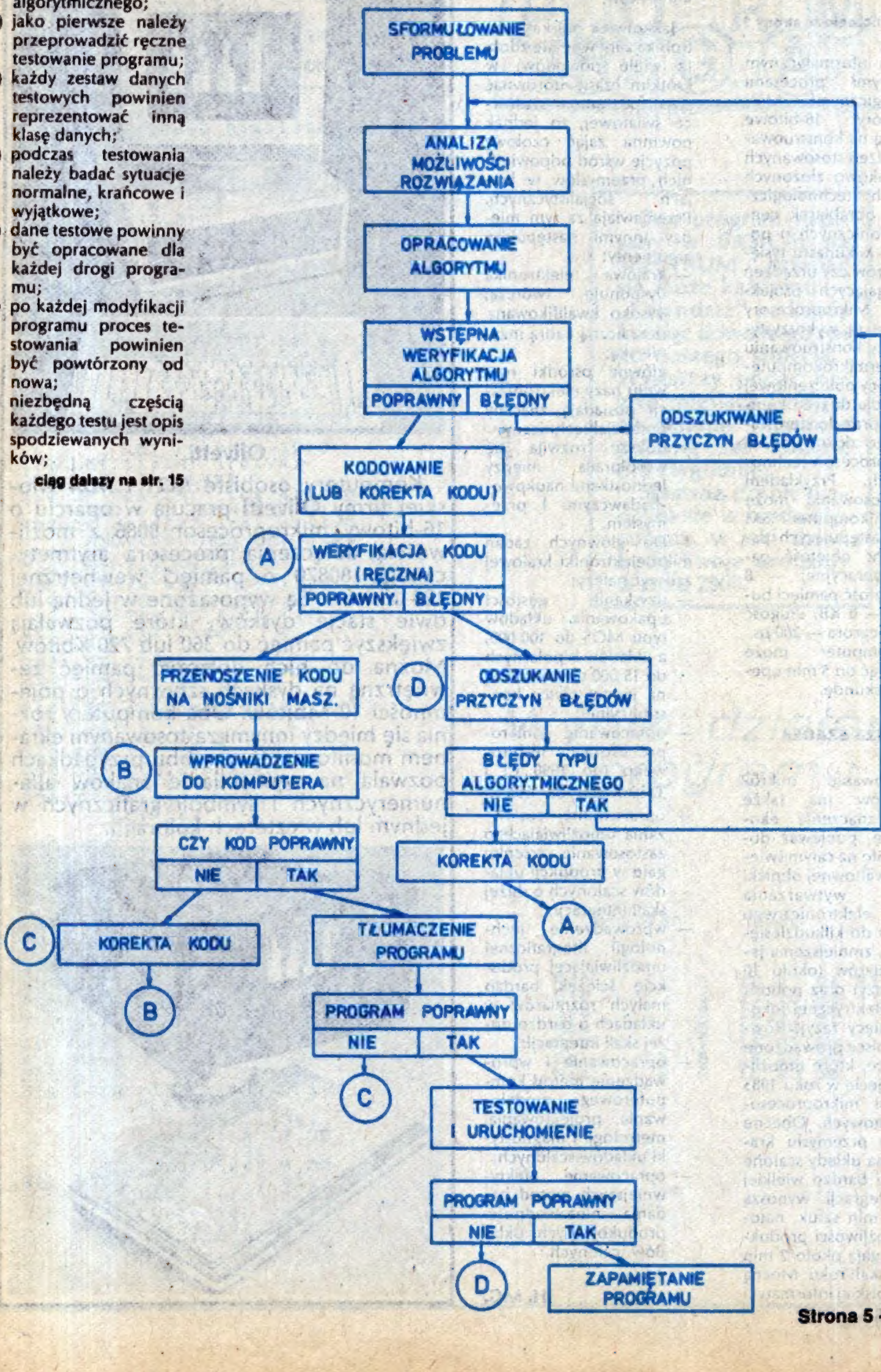
1) należy stosować minimalną liczbę zestawów testowych;
2) ponieważ gruntowne testowanie jest bardzo żmudne, należy do problemu testowania podchodzić z rozwagą;
3) testowanie zaczynać

należy już na etapie tworzenia rozwiązania algorytmicznego;

4) jako pierwsze należy przeprowadzić ręczne testowanie programu;
5) każdy zestaw danych testowych powinien reprezentować inną klasę danych;
6) podczas testowania należy badać sytuacje normalne, krańcowe i wyjątkowe;
7) dane testowe powinny być opracowane dla każdej drogi programu;
8) po każdej modyfikacji programu proces testowania powinien być powtórzony od nowa;
9) niezbędną częścią każdego testu jest opis spodziewanych wyników;

ciąg dalszy na str. 15

ETAPY TWORZENIA PROGRAMU



Problemy mikroelektroniki

dokończenie ze strony 3

sprzętem informatycznym i prostymi procesami technologicznymi. Mikroprocesory 16-bitowe pozwalają na konstruowanie urządzeń stosowanych w stosunkowo złożonych procesach technologicznych np. obrabiarek, central telefonicznych o pojemności kilkunastu tysięcy numerów czy urządzeń wspomagających projektowanie. Mikroprocesory 32-bitowe są wykorzystywane przy konstruowaniu tzw. supermikrokomputerów o mocy obliczeniowej typowej dla dużych komputerów oraz do sterowania bardzo złożonymi zespołami procesów technologicznych. Przykładem ich zastosowania może być minikomputer SM 1505 o następujących parametrach: objętość pamięci operacyjnej — 8 MB. Objętość pamięci buforowej — 8 KB, długość cyklu procesora — 200 μs.

Minikomputer może wykonywać do 5 mln operacji na sekundę.

Nasza szansa

Zastosowanie mikroprocesorów ma także wielkie znaczenie ekonomiczne, ponieważ doprowadziło na całym świecie do gwałtownej obniżki kosztów wytwarzania sprzętu elektronicznego (od kilku do kilkudziesięciu razy), zmniejszenia jego rozmiarów (około 10 tysięcy razy) oraz poboru energii elektrycznej (około 10 tysięcy razy). Również w Polsce prowadzone były prace, które umożliwiły podjęcie w roku 1983 produkcji mikroprocesorów 8-bitowych. Obecne potrzeby przemysłu krajowego na układy scalone wielkiej i bardzo wielkiej skali integracji wynoszą około 5 mln sztuk, natomiast możliwości produkcyjne sięgają około 2 mln sztuk w skali roku. Mocną stroną polskiej informatyki jest dobrze przygotowana kadra programistów i matematyków.

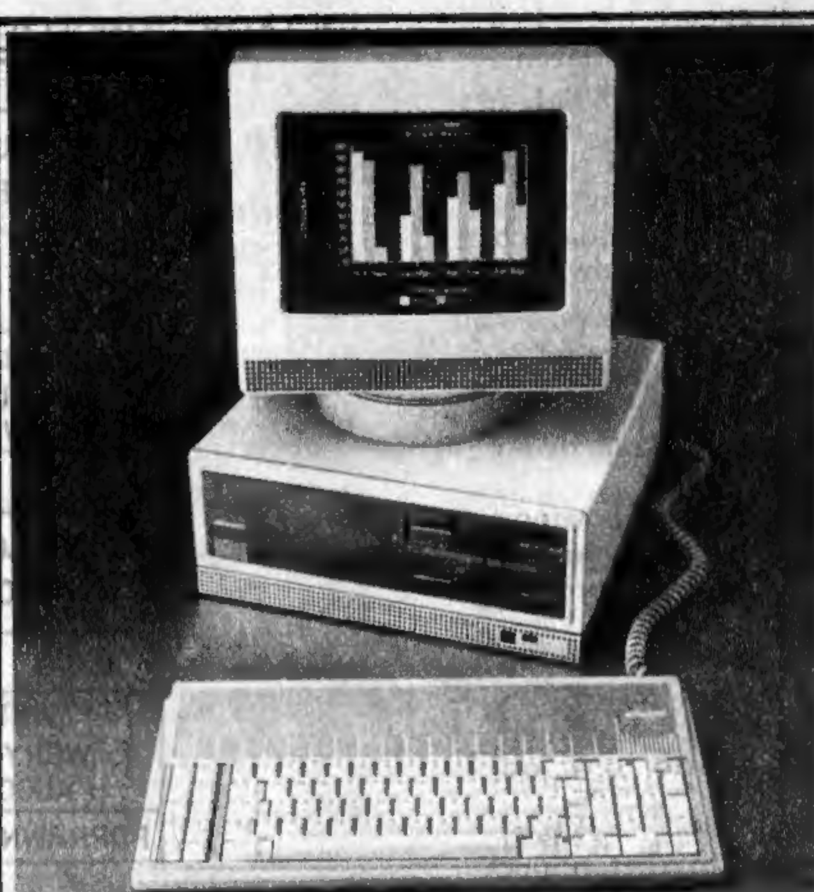
Jakkolwiek polska elektronika zapewne nie zdoła (z wielu powodów) w krótkim czasie dorównać swym poziomem czołówce światowej, to jednak powinna zająć czołową pozycję wśród odpowiednich przemysłów w krajach socjalistycznych. Przemawiają za tym między innymi następujące argumenty:

— krajowa elektronika dysponuje twórczą, wysoko kwalifikowaną oraz liczną kadrą inżynierską;
— główne ośrodki rozwoju bazy elektronizacji posiadają znaczny potencjał techniczny;
— dobrze rozwija się współpraca między jednostkami naukowo-badawczymi i przemysłem.

Do głównych zadań mikroelektroniki krajowej zaliczyć należy:

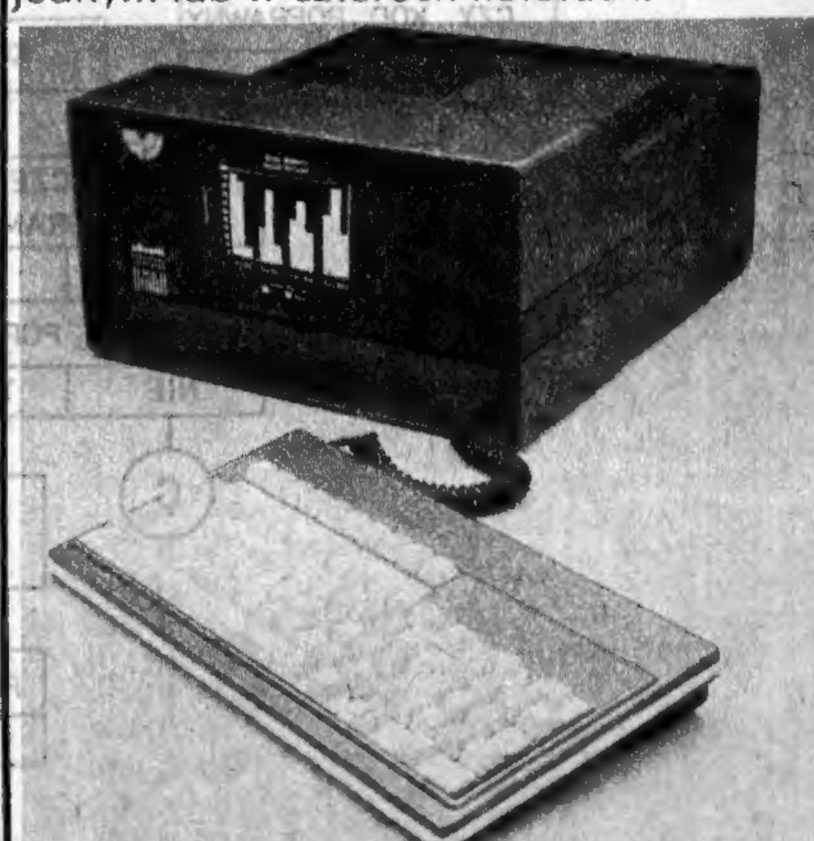
— uzyskanie gęstości upakowania układów typu MOS do 100 000, a układów bipolarnych do 15 000 tranzystorów na jednej płytce konstrukcyjnej;
— opracowanie mikroprocesora 16-bitowego (do 1990 r.) i 32-bitowego (do 1995 r.);
— opracowanie rozwiązania umożliwiającego zastosowanie arsenku galu w produkcji układów scalonych o dużej skali integracji;
— wprowadzenie technologii litograficznej umożliwiającej produkcję ścieżek bardzo małych rozmiarów w układach o bardzo dużej skali integracji;
— opracowanie i wprowadzenie metod komputerowego modelowania, projektowania, metrologii i diagnostyki układów scalonych;
— opracowanie efektywniejszych metod badania niezawodności produkowanych układów scalonych.

JH. MG.



Olivetti

Komputery osobiste **M21** i **M24** włoskiej firmy **Olivetti** pracują w oparciu o 16-bitowy mikroprocesor 8086 z możliwością dołączenia procesora arytmetycznego 80870 o pamięci wewnętrznej 128 Kbitów. Są wyposażone w jedną lub dwie stacje dysków, które pozwalają zwiększyć pamięć do 360 lub 720 Kbitów. Można do nich dołączyć pamięć zewnętrzną na dyskach sztywnych o pojemności 10 Mbitów. Oba komputery różnią się między innymi zastosowanym ekranem monitora, który w obu przypadkach pozwala na wyświetlanie znaków alfanumerycznych i symboli graficznych w jednym lub w czterech kolorach.



PROGRAM 1

```
 10 REM
        Kalendarz
       ************
 20 BORDER 6: INK 0: PAPER 6:
    CLS
 30 PRINT AT 3,11;"KALENDARZ"
 40 PRINT AT 6,3;
    "Program wyswietla dni tygo
dni  okreslonego miesiaca zadane
go   roku"
 50 PAUSE 50
 60 PRINT AT 21,5; FLASH 1;
  "Nacisnij dowolny klawisz"
 70 PAUSE 0: CLS
 80 DIM c(7,6)
 90 LET  d$="PonWtoSroCzwPioSob
Nie"
 100 INPUT "Podaj miesiac ",m$
 110 LET m=0
 120 READ q$,f,d
 130 LET m=m+1
 135 IF m>11 THEN
            PRINT AT 10,0;"Zle w
prowadzony miesiac": PAUSE 50:
            CLS : RESTORE :
            GO TO 100
 140 IF m$<>q$ THEN GO TO 120
 150 INPUT "Podaj rok ",rok
 160 IF rok>1900 THEN
         LET rok=rok-1900
 170 LET sk=INT (rok/4)
 180 LET of=rok+sk
 190 IF rok-INT (rok/4)*4=0
      AND m<3 THEN LET of=of-1
 200 IF rok-INT (rok/4)*4=0
      AND m=2 THEN LET d=29
 210 LET of=of-INT (of/7)*7
 220 LET f=f+of
 230 LET f=f-INT (f/7)*7
 240 IF f=0 THEN LET f=7
 250 FOR I=1 TO f-1
 260 LET c(I,1)=0
 270 NEXT I
 280 LET dzien=1
 290 FOR I=f TO 7
 300 LET c(I,1)=dzien
 310 LET dzien=dzien+1
 320 NEXT I
 330 FOR J=2 TO 6
 340 FOR I=1 TO 7
 350 LET c(I,J)=dzien
 360 LET dzien=dzien+1
 370 IF dzien>d THEN
        LET I=7:
        LET J=6
 380 NEXT I
 390 NEXT J
 400 CLS
 410 PRINT AT 4,8;m$:
     PRINT AT 4,20;rok+1900
 420 FOR I=1 TO 7
 430 PRINT AT I+7,5;
       d$((I-1)*3+1 TO I*3)
 440 NEXT I
 450 FOR J=1 TO 6
 460 FOR I=1 TO 7
 470 IF c(I,J)<>0 THEN
     PRINT AT I+7,3*J+10;c(I,J)
 480 NEXT I
 490 NEXT J
 500 STOP
 510 DATA "Styczen",1,31,
          "Luty",4,28,
          "Marzec",4,31,
          "Kwiecien",7,30,
          "Maj",2,31,
          "Czerwiec",5,30
 520 DATA "Lipiec",7,31,
          "Sierpien",3,31,
          "Wrzesien",6,30,
          "Pazdziernik",1,31,
          "Listopad",4,30,
          "Grudzien",6,31
```

Ten program napisany w BASIC-u na SPECTRUM przydać się może każdemu — „Kalendarz" na każdy miesiąc dowolnego roku naszego stulecia. Wystarczy podać nazwę miesiąca i rok, a na ekranie monitora pojawią się kolejne dni i tygodnie.

Sprawdź, czy nie urodziłeś się w niedzielę, zobacz, w jaki dzień będą Twoje urodziny w roku 2000.

Wynik działania programu

```
      Styczen        2000

Pon          3  10  17  24  31
Wto          4  11  18  25
Sro          5  12  19  26
Czw          6  13  20  27
Pio          7  14  21  28
Sob      1   8  15  22  29
Nie      2   9  16  23  30
```

PROGRAM 2

Ważne dla uczniów szkół średnich, a szczególnie dla maturzystów. Program rysuje wykresy wartości zadanych wyrażeń algebraicznych w określonym zakresie wartości i przedziale zmienności argumentu x. Wprowadzane wyrażenia mogą być dowolnymi wyrażeniami algebraicznymi złożonymi z operatorów arytmetycznych, stałych, zmiennej x, arytmetycznych funkcji standardowych dostępnych w BASIC-u na SPECTRUM. Po adaptacji program można też uruchomić na innym mikrokomputerze. Radzimy spróbować — badanie funkcji, często największa trudność uczniów, z pewnością okaże się łatwiejsze.

```
  10 REM
      Wykresy
       *********

  20 BORDER 1: INK 2: PAPER 6:
     BRIGHT 1: CLS
  30 PRINT AT 4,8;
      "WYKRESY FUNKCJI
          y = f(x)."
  40 PRINT AT 20,4; FLASH 1;
     "Nacisnij dowolny klawisz"
  50 PAUSE 0
  60 REM
     Uczytanie funkcji, zakresu
     zmian jej argumentu i
     wartosci
  70 BORDER 7: PAPER 7: INK 9:
     BRIGHT 0: CLS
  80 INPUT "Podaj funkcje";F$
  90 INPUT    "Minimalna wartosc
zmiennej x =     ?";XMIN
 100 INPUT    "Maksymalna wartosc
 zmiennej x =     ?";XMAX
 110 IF XMIN>=XMAX THEN
              PRINT 10,0;"Blednie
zadany zakres  wartosci zmiennej
 x":          PAUSE 100: CLS :
              GO TO 90
 120 INPUT    "Minimalna wartosc
funkcji y =          ?";YMIN
 130 INPUT   "Maksymalna wartosc
funkcji y =          ?";YMAX
 140 IF YMIN>=YMAX THEN
              PRINT AT      10,0;"B
lednie zadany zakres wartosci  f
unkcji y": PAUSE 100: CLS :
             GO TO 120
 150 REM
 160 CLS
 170 LET XZAKRES=XMAX-XMIN:
     LET YZAKRES=YMAX-YMIN:
     LET DX=XZAKRES/199:
     LET DY=YZAKRES/143
 180 LET CY=ABS (YMIN*(YMIN<0)):
     LET CY=CY/DY+16
 190 LET CX=ABS (XMIN*(XMIN<0)):
     LET CX=CX/DX+32
 200 PRINT AT 0,5;"y = ";F$
 210 REM
     Rysowanie osi wspolrzednych

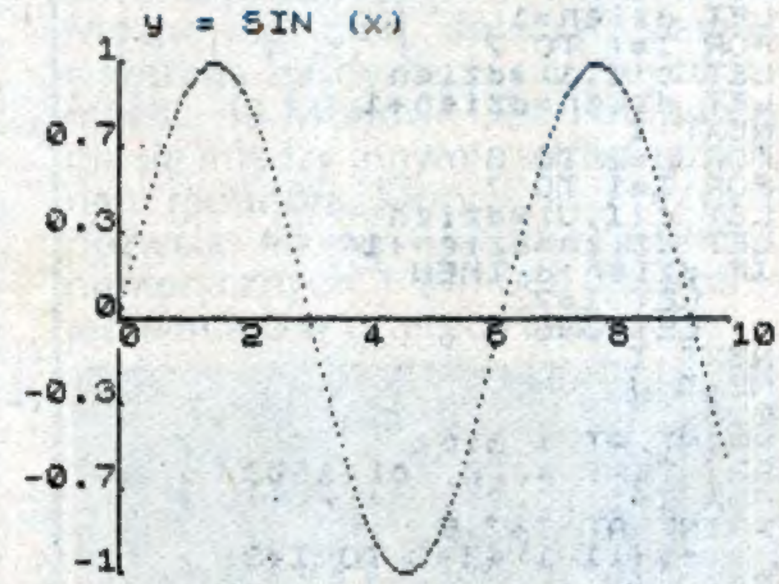
 220 PLOT 32,CY: DRAW 200,0
 230 PLOT CX,16: DRAW 0,144
 240 REM
 250 LET YSK=INT ((175-CY)/8)+1
 260 LET XSK=INT (CX/8)-4
 270 REM   Opis osi X

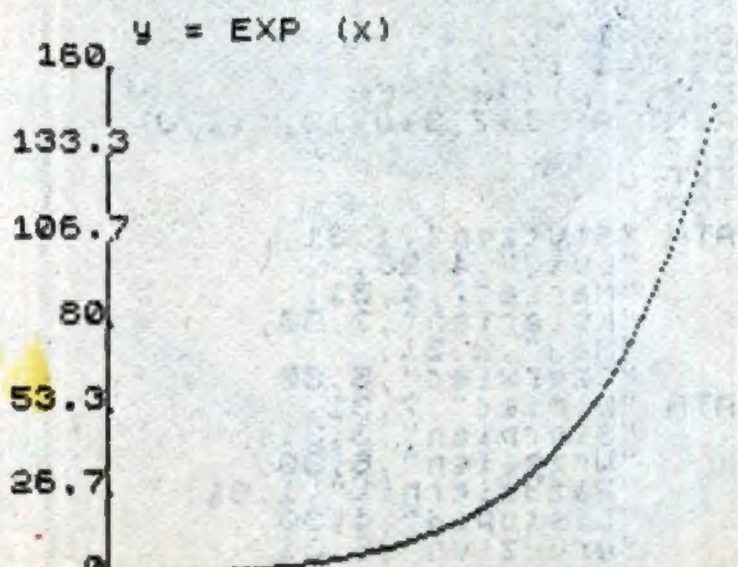
 280 FOR N=0 TO 5
 290 LET SX=(INT (10*(XMIN+
         N*XZAKRES/5)+.5))/10
 300 PRINT AT YSK,5*N+4;SX:
     PLOT (5*N+4)*8,CY-1
 310 NEXT N
 320 REM   Opis osi y

 330 FOR N=0 TO 6
 340 LET SY=(INT (10*(YMAX-N
         *YZAKRES/6)+.5))/10
 350 LET G$=STR$ SY
 360 IF LEN G$<4 THEN
        FOR C=LEN G$ TO 3:
        LET G$=" "+G$:
        NEXT C
 370 PRINT AT 3*N+1,XSK;G$:
     PLOT CX+1,(3*N+2)*8:
 380 NEXT N
 390 REM
       Rysowanie funkcji

 400 FOR N=0 TO 199
 410 LET x=XMIN+N*DX
 420 LET y=(VAL (F$)-YMIN)/DY
 430 PLOT N+32,y+16
 440 NEXT N
 450 INPUT   "Nastepna funkcja ?
(T/N)",Z$
 460 IF CODE Z$=84 OR CODE Z$=
     116 THEN CLS :
           GO TO 80
 470 STOP
```

Wynik działania programu





PROGRAM 3

Nie jest to program o finezyjnej konstrukcji; punkt po punkcie odwzorowuje granice naszego państwa. Możliwości graficzne mikrokomputera SPECTRUM nie są duże — stąd każdy rysunek „przędzie się" mozolnie. Cierpliwych zachęcamy do wspólnej pracy. Może razem uda nam się przygotować komputerową mapę Polski, na której zaznaczycie miejscowości, w których działają kluby komputerowe.

```
1 REM MAPA POLSKI
2 BORDER 1:PAPER 7:INK 9:BRIGHT 1:CLS
10 REM GRANICA ZACHODNIA
20 PLOT 63,60:DRAW 3,6,PI/4
30 DRAW 1,3
40 DRAW -1,3
50 DRAW -3,6,PI/2
60 DRAW -2,6
70 DRAW 2,3
80 DRAW 0,6,PI/4
90 DRAW -2,3,PI/4
100 DRAW 2,9,-PI/4
110 DRAW -6,6,PI/4
120 DRAW 0,3
130 DRAW 3,6,PI/4
140 DRAW 0,12,PI/4
150 DRAW -2,9
200 REM GRANICA POLNOCNA
210 DRAW 3,-2
220 DRAW 11,5
230 DRAW 12,1
240 DRAW 12,8,PI/4
250 DRAW 12,6
260 DRAW 12,2
270 DRAW 3,0
280 DRAW 9,-5
290 DRAW -2,-2,-PI
300 DRAW -5,4,PI/8
310 DRAW 1,-6,PI/8
320 DRAW 18,-1,PI/2
330 DRAW 48,-2,PI/8
340 DRAW 4,-2,PI/2
400 REM GRANICA WSCHODNIA
410 DRAW 8,-9,-PI/2
420 DRAW 6,-18,PI/4
430 DRAW 3,-12
440 DRAW -3,-4,-PI/2
450 DRAW -9,-9,PI/8
460 DRAW 7,-7
470 DRAW 0,-6
480 DRAW 2,-10,PI/4
490 DRAW 0,-5
500 DRAW 6,-7,-PI/4
510 DRAW 0,-5,-PI/2
520 DRAW 0,-9,-PI/4
530 DRAW -20,-26,PI/4
540 DRAW 1,-4,-PI/4
550 DRAW 1,-5,PI/4
560 DRAW 0,-4,-PI
600 REM GRANICA POLUDNIOWA
610 DRAW -18,9,-PI/4
620 DRAW -5,1,PI/2
630 DRAW -6,-1,PI/2
640 DRAW -9,0,-PI
650 DRAW -9,-5,PI
660 DRAW -2,1,-PI
670 DRAW -3,0,PI
680 DRAW -3,6,-PI
690 DRAW -4,3
700 DRAW -4,-3,PI/2
710 DRAW -4,2
720 DRAW -2,4
730 DRAW -3,6,-PI
740 DRAW -10,4,-PI/2
750 DRAW -4,0,-PI
760 DRAW -2,4
770 DRAW -2,2,PI
780 DRAW -11,5
790 DRAW 2,-6
800 DRAW -3,0
810 DRAW -4,0,-PI
820 DRAW -5,7
830 DRAW 3,5,PI/2
840 DRAW -6,0,PI/2
850 DRAW -11,6
860 DRAW -2,5
870 DRAW -5,0
880 DRAW 0,-3
890 DRAW -3,1,-PI
900 REM MIEJSCOWOSCI
910 PLOT 165,92:PLOT 165,93:
PLOT 166,92:PLOT 166,93:
CIRCLE 166,93,3
920 PRINT AT 11,18;"WARSZAWA"
```

PROGRAM 4

„Wyścigi psów" są grą. Wygrywa ten, kto trafnie przewidzi kolejność psów na mecie. Nie jest to sprawa prosta, bowiem o zwycięstwie decyduje generator liczb losowych. Powodzenia.

```
  10 REM   WYSCIGI PSOW
  20 REM *********************
  30 DATA 4,6,132,252,124,202,
        169,169
  40 REM Wczytanie graficznego
        znaku psa
  50 FOR n=1 TO 7
  60 READ bit
  70 POKE USR "B"+n,bit
  80 NEXT n
  90 DIM d(10): DIM c(10):
     DIM p(10)
 100 REM Kolory psow
 110 FOR n=1 TO 4:
     LET c(n)=n:
     LET c(n)=n-1:
     NEXT n
 120 FOR n=5 TO 7:
     LET c(n)=n:
     NEXT n
 130 FOR n=8 TO 10:
     LET c(n)=n-8:
     NEXT n
 140 BORDER 0: PAPER 7: INK 0:
     CLS
 150 PRINT AT 3,10; FLASH 1;
     "WYSCIGI PSOW"
 160 PAUSE 100
 170 PRINT AT 21,4; FLASH 1;
     "Nacisnij dowolny klawisz"
 180 PAUSE 0
 190 PAPER 4: CLS
 200 REM Psy na stanowiskach
        startowych
 210 FOR n=1 TO 10
 220 PRINT AT 2*n-1,0;n;
     AT 2*n-1,2; INK c(n);"■"
 230 NEXT n
 240 INPUT ;"Po  nacisnieciu
ENTER   psy  wystartuja";p$
 250 FOR n=1 TO 10
 260 PRINT AT 2*n-1,2;
     INK c(n);"■";"B"
 270 LET d(n)=3
 280 NEXT n
 290 LET mal=0
 300 LET p=INT (RND*10)+1:
     LET fl=0
 310 FOR n=1 TO 10
 320 IF p=p(n) THEN LET fl=1
 330 NEXT n
 340 IF fl=1 THEN GO TO 300
 350 PRINT AT 2*p-1,d(p);" ";
     INK c(p);AT 2*p-1,d(p)+1;
     "B"
 360 LET d(p)=d(p)+1
 370 IF d(p)=31 THEN
      LET p(mal+1)=p:
      LET mal=mal+1:
      BEEP .1,3
 380 IF mal<10 THEN GO TO 300
 390 BEEP 2,3
 400 PRINT AT 21,0; FLASH 1;
     "Nacisnij dowolny klawisz a
by    uzyskac wyniki gonitwy"
 410 PAUSE 0: CLS
 420 PRINT AT 2,5;
     "WYNIKI GONITWY"
 430 PRINT AT 4,5;
     "1 miejsce - pies nr ";
     p(1);AT 5,5;
     "2 miejsce - pies nr ";
     p(2);AT 6,5;
     "3 miejsce - pies nr ";
     p(3)
 440 PRINT AT 9,0; "Rezultaty p
ozostalych zawodnikow"
 450 FOR n=4 TO 10
 460 PRINT AT 8+n,5;"Pies nr ";p
(n)
 470 NEXT n
 480 STOP
```

Wynik działania programu



Mikrokomputer dla kolekcjonerów

Prawie każdy kolekcjoner i zbieracz w miarę wzrostu swoich zbiorów napotykać zaczyna trudności. Nadmiar informacji utrudnia kolekcjonowanie, a żmudne procesy ewidencji i katalogowania zbiorów, pochłaniają wiele czasu. Przy wykorzystaniu mikrokomputera *Commodore-C64* **postanowiono „rozpracować" tematykę widniejącą na okolicznościowych datownikach pocztowych, aby dobrać właściwy materiał do budowy zbiorów filatelistycznych.**

WYSZUKIWANIE właściwego materiału wśród kilkunastu tysięcy pozycji zewidencjonowanych w *„Katalogu polskich stempli okolicznościowych"* stanowi już problem. Przysłowiową „deskę ratunku" dostrzeżono w zaprowadzeniu komputerowej ewidencji i okolicznościowych datowników pocztowych, którym przypisane zostaną cechy znacznikowe związane z występującą tematyką zbiorów filatelistycznych.

W tym przypadku postanowiono opracować typowy, dostosowany do zbiorów filatelistycznych (pod kątem datowników pocztowych) program.

Przejdźmy jednak do konkretów. W tekście wykorzystywać będziemy pojęcia:

Numer katalogowy

99999 W

- wyróżnik numeru katalogowego
- numer pozycji w katalogu



Wyróżnikiem numeru katalogowego może być:

— (spacja) oznaczająca, że pozycja występuje tylko w katalogu (brak w zbiorze);

+ — oznaczający, że pozycja katalogowa występuje w zbiorze;

* — informuje, że pozycja katalogowa występuje wielokrotnie w zbiorze.

Numer pozycji w katalogu zgodny jest z systemem *„Katalogu polskich stempli okolicznościowych"*. Może to być kolejny pięcio- lub trzycyfrowy numer w powiązaniu z dwucyfrową końcówką roku.

Słownik

X nazwa

- Nazwa słownika
- cecha — jedno- lub dwuliterowy kod słownika

KAŻDEMU numerowi katalogowemu można przyporządkować ciąg cech z zakresu własnego, odpowiadającego profilowi działalności kolekcjonerskiej słownika. Dla stempli pocztowych czy też innych walorów słownik może zawierać zapisy przedstawione poniżej. Na przykład:

P Przemysł
PB Budowlany
PC Chemiczny
PE Energetyczny

N Nobliści
NC Curie Skłodowska
ND Dunant
NR Reymont

S Sport
SA Akrobatyka
SB Boks
SK Koszykówka
ST Tenis
SZ Sporty zimowe

W Wojskowość
WA Awiacja
WB Broń
WC Cedynia
WG Grunwald
WN Okres Napoleoński

Jednoliterowa pierwsza cecha jest ogólnym kodem tematu (używanym lub nie do opisu elementów zbioru), zaś dwuliterowe cechy dokładnie precyzują tematykę.

Zapisy numerów katalogowych, ich występownie (lub nie) w zbiorze oraz przyporządkowanie tematyczne ilustruje kolejny przykład:

67177 + LA NS
67224 + NC
67233 * WA
72259 * WC TG
72280 + WC SL
74052 * SN
77127 * WB WG
78300 WA EW

Powyższe przykłady dokładnie odzwierciedlają strukturę danych wprowadzanych do komputera.

Jeżeli zapis słownika lub numeru katalogowego z cechami występuje po raz pierwszy, to jest wprowadzany do pamięci komputera. Ponowne wystąpienie zapisu na przykład:

72359 * WC

powoduje wycofanie tej informacji ze zbioru (prawdopodobnie błędnie wprowadzonej), bo pozostaje zapis:

72259 * TG

Częściej dokonywana jest jednak operacja zmiany wyróżnika w numerze katalogowym. Zapis:

67177 *

świadczy o pozyskaniu dodatkowych elementów zbioru (występujących dotychczas w pojedynczym egzemplarzu) i aktualizuje zbiór zarówno dla słownika LA jak i NS.

ZAŁOŻENIE ewidencji i zabezpieczenie aktualizacji (kasowanie, zmiana) to tylko pierwszy etap wykorzystania komputera. Efekt pracy włożonej w uciążliwe kodowanie numerów katalogowych, wyróżników i ich przyporządkowanie tematyczne widać dopiero później, na etapie wyszukiwania danych.

Dla podanych wcześniej przykładów zapis:

67224

wyprowadzi informację:

67224 + NC

cd. na str. 14

Klasyfikacja i zastosowanie sprzętu mikrokomputerowego

CORAZ powszechniejsze zastosowanie mikrokomputerów w różnych dziedzinach zrodziło potrzebę ich klasyfikacji. W literaturze fachowej oraz wśród producentów istnieje wielka różnorodność poglądów dotyczących kryteriów podziału tego sprzętu. Powoduje to wiele nieporozumień i niejednoznaczności w określaniu obszarów ich zastosowań. Mając na uwadze potrzebę ujednolicenia fachowego słownictwa oraz formułowania wymagań technicznych dla poszczególnych grup mikrokomputerów Komisja Międzynarodowa RWPG do Spraw Techniki Obliczeniowej zaproponowała następujący podział mikrokomputerów: ręczne, kieszonkowe, domowe i personalne. Ponadto mikrokomputery personalne podzielono na 8-bitowe, 16-bitowe oraz personalne mikro-EMC.

Mikrokomputery ręczne są przedłużeniem produkcji kalkulatorów elektronicznych. Typowym przedstawicielem tej klasy sprzętu jest mikrokomputer **HP41CV serii 40** firmy Hewlett Packard. Mikrokomputer ten cechuje miniaturowa klawiatura alfanumeryczna, 15-znakowy alfanumeryczny wyświetlacz na ciekłych kryształach, pamięć dynamiczna (RAM) zawierająca 319 rejestrów. Ponadto mikrokomputer ten wyposażony jest w układ sprzężenia video, pozwalający na podłączenie do niego między innymi drukarki, magnetofonu i czytnika kodów paskowych.

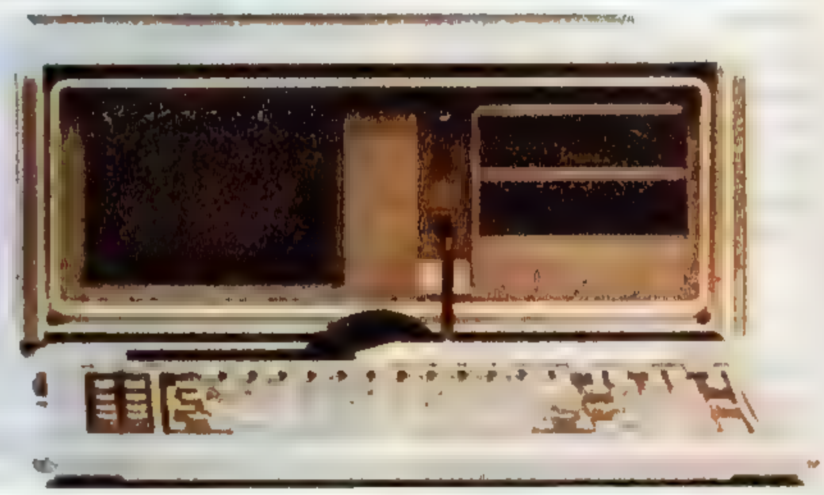
Mikrokomputery kieszonkowe charakteryzuje wysoki stopień scalenia i miniaturyzacji jednostki centralnej oraz urządzeń peryferyjnych. W swoim zestawie mają najczęściej dołączoną drukarkę oraz układ sprzęgający umożliwiający podłączenie magnetofonu kasetowego.

Niektóre spośród mikrokomputerów kieszonkowych mają wbudowane czytniki kart magnetycznych oraz układy umożliwiające współpracę z magnetofonem minikaset cyfrowych, drukarką, ploterem oraz monitorem telewizyjnym. W najbardziej popularnych mikrokomputerach tej grupy wykorzystywane są procesory 8-bitowe lub dwa procesory 4-bitowe, a ponadto pamięć stała (ROM) o pojemności do 48 KB, pamięć dynamiczna (RAM) o pojemności do 16 KB, klawiatura kalkulatorowa, a także 32-znakowy wyświetlacz zbudowany na diodach luminescencyjnych. Przedstawicielami tej grupy mikrokomputerów są **PC-1500 SHARP** oraz **HHC-PANASONIC**.

Mikrokomputery domowe przeznaczone są do współpracy z domowymi odbiornikami telewizyjnymi, magnetofonami kasetowymi oraz innymi mikrokomputerami personalnymi dzięki wbudowanym układom asynchtonicznego sprzężenia szeregowego. Mikrokomputery domowe cechują: pojemność pamięci operacyjnej do 64 KB, standardowa klawiatura o układzie podobnym do występującego w maszynach do pisania, zastosowanie odbiornika telewizyjnego jako monitora (ekran podzielono na 16—24 linie, a w każdej po 40—64 znaki pisarskie), a także m liwość wykorzystania magnetofonów kasetowych oraz pamięci na dyskach elastycznych jako pamięci masowych. Do typowych przedstawicieli mikrokomputerów domowych należy zaliczyć **ATARI 400** oraz **ZX81** firmy **Sinclair**.

Mikrokomputery personalne 8-bitowe z dyskowym systemem operacyjnym są konstruowane głównie z myślą o zastosowaniu w różnych konfiguracjach urządzeń przeznaczonych do nauki. Podstawowy zestaw zawiera: pamięć operacyjną o pojemności 64 KB, dyskowy system operacyjny, złącza do drukarki wierszowej, magnetofonu kasetowego i pamięci zewnętrznej na dyskach elastycznych. Ponadto zestaw zawiera monitor ekranowy, na którym można wyświetlać informację w 24 wierszach 80-znakowych, przy czym można używać znaków i liter alfabetu łacińskiego oraz cyrylicy. Jako typowy mikrokomputer tej grupy można uznać: **APPLE II** i **TRS-90**.

Mikrokomputery personalne 16-bitowe są wykorzystane w pracach naukowo-badawczych, w projektowaniu inżynierskim, w automatyzacji prac biurowych i innych. Podstawowy zestaw tego typu mikrokomputerów zawiera: moduły procesora z pamięcią stałą (ROM) i dynamiczną (RAM), kontrolery monitora ekranowego, pamięć na dyskach twardych i elastycznych oraz kontrolery komunikacyjne. Wśród urządzeń najczęściej podłączanych do mikroprocesora można wymienić czarno-białe lub kolorowe monitory ekranowe, pamięci na dyskach twardych i elastycznych, drukarki oraz plotery. Typowymi przedstawicielami tej grupy mikrokomputerów są: **APPLE-LISA, IBM PC/XT** oraz **HP-150**.



IBM Portable PC

Personalne mikro-EMC mogą być wykorzystywane we wszystkich gałęziach gospodarki narodowej, głównie do: obliczeń naukowo-technicznych, zbierania i obróbki danych, opracowywania i uruchamiania programów, obsługi stanowisk inżynierskich, budowy i rozbudowy komputerowych sieci abonenckich. Klawiatura i monitor są tu elementami przenośnymi. Do najczęściej podłączanych urządzeń peryferyjnych należą minidyski elastyczne, dyski magnetyczne typu **WINCHESTER**, drukarki, klawiatury alfanumeryczne, monitory ekranowe oraz urządzenia służące do wprowadzania i wyprowadzania danych. Typowymi przedstawicielami mikrokomputerów tej grupy są **MBC 4050** firmy SANYO oraz **RAINBOW 100** firmy DEC.

Mikrokomputery personalne 8- i 16-bitowe znajdują zastosowanie przede wszystkim jako procesory obróbki tekstów i procesory zarządzania bazami danych, końcówki sieci lokalnych, ułatwiają wykorzystanie grafiki komputerowej, a także są narzędziami obliczeniowymi w komputerowych systemach księgowości i edukacyjnych. Rozważając strukturę zastosowań znanych mikrokomputerów można stwier-

dzić, że są one najczęściej wykorzystywane jako procesory obróbki tekstów. Za ich pomocą można wyświetlać, łączyć i formułować teksty na ekranie monitora oraz przechowywać je na dyskietkach, wykorzystywanych jako nośniki standardowych fragmentów tekstu (np. nagłówki).



Polski mikrokomputer Meritum 1

OBECNIE na krajowym rynku mikrokomputerowym spotyka się wyroby przedsiębiorstw państwowych oraz firm prywatnych i polonijnych. Do mikrokomputerów domowych zaliczyć można: **MERITUM-1**, **MERITUM-2**, **IMZ-80** oraz **ZX 81**.

Mikrokomputer MERITUM-1, wzorowany na mikrokomputerze **TRS-80**, produkowany jest w Zakładach Urządzeń Komputerowych MERA-ELZAB. W skład zestawu mikrokomputera wchodzą: moduł klawiatury wraz z mikroprocesorem oraz układami sprzęgającymi moduł zasilacza, odbiornika telewizyjnego NEPTUN 156 oraz magnetofon kasetowy.

Mikrokomputer IMZ-80, produkowany przez firmę IMPOL-II, może współpracować z monitorem telewizyjnym, magnetofonem kasetowym oraz drukarką znakowo-mozaikową. Do omawianej grupy należy również zaliczyć mikrokomputer **SPECTRUM** wykorzystywany w szkolnictwie średnim.

WŚRÓD mikrokomputerów 8-bitowych wymienić leży mikrokomputery serii **ELWRO 500** i **MK4500** oraz **MERA 660** i **ComPAN-8**. Mikrokomputer ELWRO 500 produkowany jest w Zakładach Elektronicznych MERA-ELWRO; w skład zestawu wchodzą moduły jednostki centralnej, klawiatury, pamięci na dyskach elastycznych, monitora ekranowego oraz drukarki.

Mikrokomputery serii MK 4500 produkowane są przez Krakowską Fabrykę Aparatów Pomiarowych MERA-KFAP oraz składają się z modułów: jednostki centralnej, monitora ekranowego, pamięci na dyskach elastycznych oraz klawiatury.

Mikrokomputer MERA 660 jest produkowany przez Centrum Naukowo-Produkcyjne Systemów Sterowania MERA-STER w dwóch wersjach konstrukcyjnych autonomicznej — jako samodzielne stanowisko pracy projektanta oraz sieciowej — jako terminal sieci **SN-60** lub **ETHERNET**.

Producentem mikrokomputerów ComPAN-8 są Zakłady Urządzeń Komputerowych MERA-ELZAB i jest on przeznaczony do automatyzacji badań naukowych oraz do pracy jako programowana końcówka systemów komputerowych. W skład mikrokomputera wchodzą moduły: procesora, odwzorowania zawartości pamięci na ekranie monitora (tzw. moduł Video-RAM), kontrolerów drukarki **DZM-180/KSRE** i pamięci na dyskach elastycznych oraz specjalizowane.

DO tej grupy mikrokomputerów należy zaliczyć także **IMP-85** produkowany przez firmę **IMPOL-II** oraz mikrokomputery specjalizowane **PSPD-90** i **RTDS-8**. Komputeryzacja może dać oczekiwane rezultaty tylko wtedy, gdy z żelazną konsekwencją będzie się przestrzegać zasady, że odpowiednie narzędzie ma znaleźć się w odpowiednich rękach. W myśl tej zasady nie należy stosować mikrokomputerów w rozwiązaniach wymagających operowania na dużych zbiorach danych w długich okresach. W takich wypadkach mikrokomputery mogą być wykorzystywane tylko marginalnie, dla uzyskania efektów krótkotrwałych i wspomagających, nie należy też oczekiwać, że w wyniku zastosowania mikrokomputerów nastąpi zdecydowana poprawa w danej dziedzinie.

Aby wykluczyć wypadki nieuzasadnionego stosowania mikrokomputerów, producenci tego sprzętu oferują profesjonalne mikrokomputery przygotowane do wykonywania różnorodnych, ale wzajemnie uzupełniających się zadań obliczeniowych. Jednocześnie precyzują oni najbardziej typowe obszary zastosowań, wychodząc z założenia, że adaptacje do określonych warunków eksploatacji można zapewnić przez dobór odpowiedniej konfiguracji technicznej mikrokomputera oraz instalację właściwego pakietu oprogramowania.

Zasadniczym sposobem zapewnienia różnorodnych zastosowań mikrokomputerowych jest wyposażenie ich w emulatory typowych końcówek sieci komputerowych oraz pakiety oprogramowania telekomunikacyjnego umożliwiającego współpracę z mikrokomputerami — końcówkami sieci teleprzetwarzania.



IBM Personal Computer XT

JAKKOLWIEK obecnie mikrokomputery personalne służą do przetwarzania danych cyfrowych i obróbki tekstów, przewiduje się, że przyszłościowym obszarem zastosowań tych urządzeń będzie digitalizacja danych analogowych (dane podawane głosem będą przekształcone w postać cyfrową) i obrazów ponieważ około 90% informacji przetwarzanych przez człowieka ma taką postać. Ocenia się, że możliwości zastosowania mikrokomputerów są obecnie nieograniczone, liczba rozwiązań wykorzystujących technikę mikroprocesorową urasta coraz szybciej, czy będzie jej kres?

Marian GNIŁKA **Juliusz HASCHKA**

Mikrokomputer dla kolekcjonerów

dokończenie ze strony 11

co oznacza, że ten numer katalogowy jest w zbiorze, występuje w pojedynczym egzemplarzu i należy do tematu **Nobliści—Curie-Skłodowska** (patrz przykład powyżej).

Bazując na tym samym przykładzie zapis WA wyprowadzi informację:
67233 * 78300
a zapis WC+ wyprowadzi numer katalogowy
72280+

Analogicznie do wybierania można zadać cechę z wyróżnikiem „*". Cecha ze spacją (brak wyróżnika) wyprowadza wszystkie elementy z „+" tylko występujące w zbiorze z „*" — występujące w zbiorze wielokrotnie.

Można wybierać także wg jednoliterowej cechy wiodącej tematu. Zapis „W" spowoduje wyprowadzenie wszystkich numerów katalogowych z wyróżnikami i tematami **Wojskowość** w następujący sposób:

WA Awiacja
67233 * 78300
WB Broń
77127 *
WC Cedynia
72259 * 72280 +
WG Grunwald
77127 *

Jednorazowo można podać do wybierania nawet kilkaset numerów katalogowych lub cech (ewentualnie z wyróżnikami), ale trzeba zdawać sobie sprawę z tego, że im większa ilość danych, tym dłuższy jest czas przeszukiwania zbiorów.

W PROGRAMIE zapewniona została także możliwość wyprowadzenia słownika założonego zbioru. Polecenie „SLOW" dla komputera wyprowadza wszystkie wprowadzone do pamięci cechy z nazwami. Dodatkowo — polecenie „WYDR" wyprowadza w sposób uporządkowany wszystkie dane zgromadzone w pamięci komputera (cechy z nazwami i przyporządkowane im numery katalogowe). Opcjonalnie wszystkie informacje (z aktualizacji i wyszukiwania) można wyprowadzać zarówno na monitor — ekran telewizyjny — jak i na drukarkę.

Dotychczas posługiwaliśmy się pojęciem pamięci komputera. Jest to duże uproszczenie, wprowadzone w celu pokazania idei systemu. W rzeczywistości prezentowane funkcje (wprowadzania, aktualizacji i wyszukiwania danych) realizuje napisany w języku *BASIC* program dla mikrokomputera *Commodore-C64*. Program wykorzystuje mikrodrajw (do zapamiętywania danych) i opcjonalnie drukarkę.

Szeregowa transmisja „do" i „z" mikrodrajwera jest też stosunkowo wolna, wydłuża czas pracy programu. Dopiero *Commodore-128* z mikrodrajwerem pracującym równolegle (lub inny mikrokomputer z takim rozwiązaniem) powinien zabezpieczyć szybkie otrzymywanie wyników z uruchomionego programu.

Ze względu na dużą ilość danych (kilkanaście tysięcy pozycji) bardzo istotnym problemem jest również sposób gospodarowania pamięcią komputera i dyskietki.

PRZYJĘTE założenia nie powodują *„rezerwacji miejsca"* dla cech w ramach numeru katalogowego i nie ograniczają ilości cech dla dowolnego numeru katalogowego. Podobną organizację mogą mieć *„komputerowe katalogi"* innych zbiorów. Kolekcjonerom radzimy spróbować.

Krzysztof ŚWIĘTOŃ
Zdzisław SZAŁKOWSKI

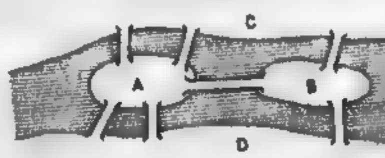
Liga Myślących

Zadanie nr 1

— W każdym z trzech zamkniętych koszy znajdują się po dwie kule: biała z czarną, biała z białą i czarna z czarną. Przyczepione do koszy kartki wskazują kolory kul znajdujących się w każdym koszu. Ile trzeba wykonać „ciągnięć", aby się dowiedzieć, jakie kule są w jakim koszu przy założeniu, że wszystkie kartki przyczepiono do koszy błędnie.

Zadanie nr 2

— Dwie wyspy A i B łączy między sobą i z obu brzegami C i D siedem mostów (rys.) Proszę spróbować „przejść" przez wszystkie mosty dokładnie jeden raz, tak aby powrócić do punktu wyjścia.



Zadanie nr 3

— W jednostce wojskowej trwają przygotowania do mającej się odbyć odprawy związanej z omówieniem dotychczasowych wyników i postawieniem nowych zadań. Przy stole prezydialnym ustawiono 5 krzeseł. Należy podać, na ile sposobów może usiąść na nich pięcioosobowa komisja.

Zadanie nr 4

— Przed treningiem strzeleckim por. Saniak dał chor. Kwiatkowi pudełko z amunicją mówiąc, aby podzielił tę amunicję równo na każde stanowisko ogniowe w ten sposób, aby na pierwsze stanowisko dał 1 nabój i 1/6 część reszty amunicji z pudełka, na drugie stanowisko — 2 naboje i 1/6 część tego co zostało, na trzecie — 3 naboje i 1/6 część kolejnej reszty i tak do końca, aż pudełko będzie puste. Należy podać, ile było sztuk amunicji przeznaczonej na każde stanowisko ogniowe i ile było stanowisk ogniowych.

Rozwiązania zadań prosimy przysyłać pod adres redakcji do końca kwietnia br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.



Polski mikrokomputer ComPAN—8

dokończenie ze str. 5

10) dane testowe powinny obejmować zarówno poprawne jak i błędne warunki wejściowe;
11) wyniki każdego testu powinny być dokładnie zbadane i przeanalizowane;
12) wraz ze wzrostem liczby wykrytych błędów w programie rośnie prawdopodobieństwo istnienia pozostałych nie wykrytych błędów;
13) nigdy nie należy modyfikować programu jedynie w celu ułatwienia jego testowania.

Testowanie programu można określić mianem miniatury całego cyklu jego budowy. Trzeba bowiem zaprojektować, zapisać i przetestować dane testowe oraz wykonać program z ich wykorzystaniem, a następnie przeanalizować jego rezultaty.

OMÓWIONE problemy dotyczyły kodowania i testowania oraz uruchamiania programu. Nie wyczerpują one całkowicie problemu. Dzieje się tak dlatego, że proces programowania można traktować jako ciąg kolejnych czynności, z których każda ma możliwość wprowadzenia błędów.

Pełna lista błędów, jakie można popełnić w trakcie procesu programowania jest długa i może być przedmiotem oddzielnego opracowania.

Zapamiętaj!

1) pisząc program należy myśleć o jego uruchomieniu;

2) przed kodowaniem programu powinno się opracować choćby jeden schemat blokowy;

3) podczas uruchamiania dobrze mieć listę zmiennych i stałych użytych w programie;

4) problemy należy rozwiązywać pojedynczo;

5) zmiany w programie powinny być dokonywane wyłącznie na poziomie języka źródłowego;

6) podczas uruchamiania programu należy pamiętać o zachowywaniu kolejnych wydruków programów źródłowych oraz danych testowych, na których był program uruchomiony;

7) należy bacznie zwracać uwagę czy przy usuwaniu jednego rodzaju błędów nie wprowadzono innych — nowych błędów.

H. J.

W naszym komputerlandzie

W naszym komputerlandzie, gdzie zwykle jest bardzo spokojnie i słychać tylko szum taśm perforowanych, zapanował wielki zamęt. Zbliżały się bowiem dni doniosłych rocznic i poważnych świąt. Pracowały rozmaite komitety organizacyjne i zespoły. Zespół najstarszych robotów kierowany przez pracowitego Spektrusia miał najwięcej pracy. Spektruś wychudł i zmizerniał, bowiem nijak nie mógł się z robotami dogadać. Każde niemal spotkanie kończyło się kłótnią: co komu i za co się należy. Reprezentacje rozmaitych generacji i typów robotów wciąż przypominały kogo, kiedy i przy jakiej okazji pominięto. Mnożyły się pretensje. I choć na robocich piersiach powoli, ale ciągle medali przybywało (u niektórych — jak za co najmniej dwie wygrane komputerlandzkie wojny), ulubionym tematem zebrań były metalowe krążki i baretki.

Rada komputerlandu, widząc kłopoty Spektrusia, postanowiła zająć oficjalne stanowisko. Uchwaliła więc kolejny regulamin i odznaczenie — „Medal za skromność". Dopiero rozgorzały dyskusje i spory. Do robotów przyłączyły się także inne dziełoroby. Każdy chciał nosić ów „dowód skromności".

I tylko pracowite mikrokomputerki pracowały, pracowały, pracowały...

Podglądał:
Eugeniusz MLECZAK

Czekamy na Twoją opinię!

ANKIETA

Zostań redaktorem IKS-a!
Bądź naszym ekspertem!

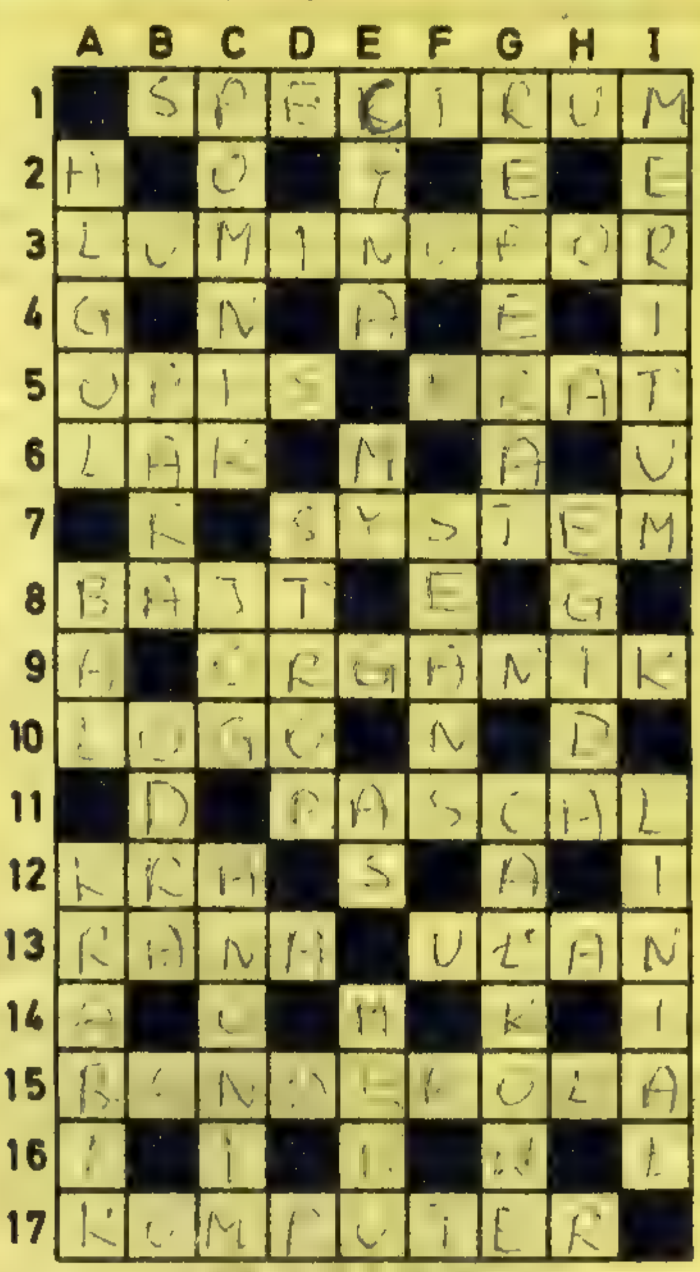
(podkreśl problemy, które Cię interesują)

A) **Co Twoim zdaniem powinniśmy w nim publikować?**
1) **programy:**
— gier;
— graficzne;
— użytkowe.
2) **artykuły problemowe dotyczące:**
— szczegółowych rozwiązań stosowanych w technice komputerowej;
— przeglądu współczesnych mikrokomputerów;
— oprogramowania;
— systemów informatycznych;
— zastosowań elektronicznej techniki obliczeniowej;
— metod rozwiązywania problemów.

B) **Zgłoś własne propozycje** . . . .
. . . .
. . . .
. . . .

DZIĘKUJEMY!

Krzyżówka nr 1



Poziomo: 1-B popularny minikomputer, 3-A część lampy kineskopowej, 5-A rysopis, charakterystyka, 5-F Placek dla Jacka, 6-A do pieczętowania, 7-D dziesiętny lub dwójkowy, 8-A element pamięci, 9-C chemik specjalista, 10-A język mikrokomputera, 11-D matematyk lub język programowania, 12-A tafle lodu, 13-A uszkodzenie ciała, 13-F żołnierz na koniu, 15-A naklejka, 17-A mózg elektronowy.

Pionowo: 1-C posąg, 1-E składnik stopu lutowniczego, 1-G może być sprawozdawczy, 1-I mikrokomputer z Zabrza, 2-A język algorytmiczny, 5-B on i ona, 6-E ja i ty, 7-D sklepienie, 7-F filmowy lub spirytystyczny, 7-H cudowna tarcza Zeusa, 8-A wielka zabawa, 8-C zwolennik jogi, 10-B informatyczna rzeka, 11-E znany pies, 11-G równanie różniczkowe lub..., 11-I przyrząd kreślarski, 12-A mały krab, 12-C nie podpisany, 14-E restauracyjny spis.

Hasło: 3-D, 6-C, 11-F, 7-I, 17-B, 4-I, 3-C, 17-D, 12-I, 5-D, 1-I, 8-F, 14-E.

Rozwiązania (tylko hasło) należy przesłać pod adresem redakcji do końca kwietnia br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Mikro-ciekawostki





„Apple II" trafił do szkół. Prosta lista komend umożliwia dialog z komputerem najmłodszym, a ekran okazuje się być o wiele wygodniejszy od tablicy. „Myszka" podłączona do komputera sprawia, że grafika nie wymaga skomplikowanych programów — wystarczy odrobina wyobraźni — jej nigdy dzieciom nie brakowało.



„Iks" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera, Ryszard Rogoń. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon 22-21-27 i 359-34. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 7585 P-91

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 2/1986

W numerze:

Grafika komputerowa
BASIC dla początkujących
Programy na:
— Commodore
— Spectrum
— Atari



Foto: Jan Zelman

FASCYNACJA informatyką trwa nadal. Od kiedy sprzęt elektroniczny osiągnął rozmiary domowych telewizorów, a jego ceny spadły wystarczająco, szybko dostał się do rąk entuzjastów — amatorów. Komputer w domu nie jest niczym nadzwyczajnym — po prostu jest i trzeba z niego korzystać.

Konieczność zatem zmusza do poznania nie tylko nowych technik programowania, ale także przełamania niepraktycznych schematów myślenia. „Bezmyślne" przecież narzędzie może się przydać wtedy i tylko wtedy, kiedy znajdzie się dla niego właściwe miejsce. Sprawa zatem pierwsza, to znaleźć owo najwłaściwsze miejsce dla komputera. I to właśnie trzeba wiedzieć.

Zaczyna się wcześnie — pierwszą pasją są gry. O tym już pisaliśmy. Ale ten okres przecież przemija, musi przeminąć, choć przecież rozrywka zawsze będzie dziedziną, w której informatyka może mieć swój udział. Krok następny, to codzienne wykorzystanie komputerów. Najpierw są to sprawy małe, drobne codzienne problemy, które „wrzucono" do komputera. Później przychodzi czas na zadania poważniejsze. Wkrótce okazuje się, że kilkunastoletni młodzi ludzie do perfekcji opanowują nową technikę. W ich rękach komputer staje się posłuszny. Stąd już tylko krok do perfekcjonizmu i do sytuacji, w której informatyka nie będzie celem, lecz jedynie dyscypliną dostarczającą metod i narzędzi do każdej innej. I tak właśnie być powinno.

Ale najpierw trzeba się uczyć myśleć; myśleć jasno, przejrzyście i jednoznacznie formułować swe poglądy, maszyna bowiem nie toleruje niedomówień i choć rozkazy wykonuje szybko i bezbłędnie, to jednak nie myśli!

Informatyka, to przede wszystkim jej zastosowania profesjonalne — tu liczby mówią prawdę: jedynie 10 procent wartości światowego handlu komputerami stanowią komputery osobiste — reszta to sprzęt profesjonalny, a choć i na takim można zagrać (na przykład w szachy), to służy on raczej innym celom.

Trudno zatem sądzić, że dzisiejsi miłośnicy informatyki, która za sprawą mikrokomputerów dotarła do wielu, spędzą życie przed ekranem monitora, z zapartym tchem śledząc coraz to nowsze gry. Krąg entuzjastów byłby znacznie większy, gdyby sprzęt ten był tańszy. Nie każdy może wydać kilkadziesiąt tysięcy na tę ekskluzywną „zabawkę", którą przecież mieć wypada. Brak bowiem, do tej pory, w naszym kraju prostego i przede wszystkim taniego mikrokomputera, którego masowa produkcja usprawiedliwiałaby niską cenę, a w konsekwencji jego popularność sprzyjałaby szybkiemu powstawaniu przeróżnych programów na to urządzenie.

Tymczasem każdy bawi się we własnym kącie. A co bardziej pomysłowi robią nawet na tym pieniądze. Nawet państwowe firmy zdecydowały się wprowadzić na krajowy rynek obcy sprzęt, stosując przelicznik, który nawet „specjalistów" z Czackiego i Kredytowej przyprawiłby o zawrót głowy.

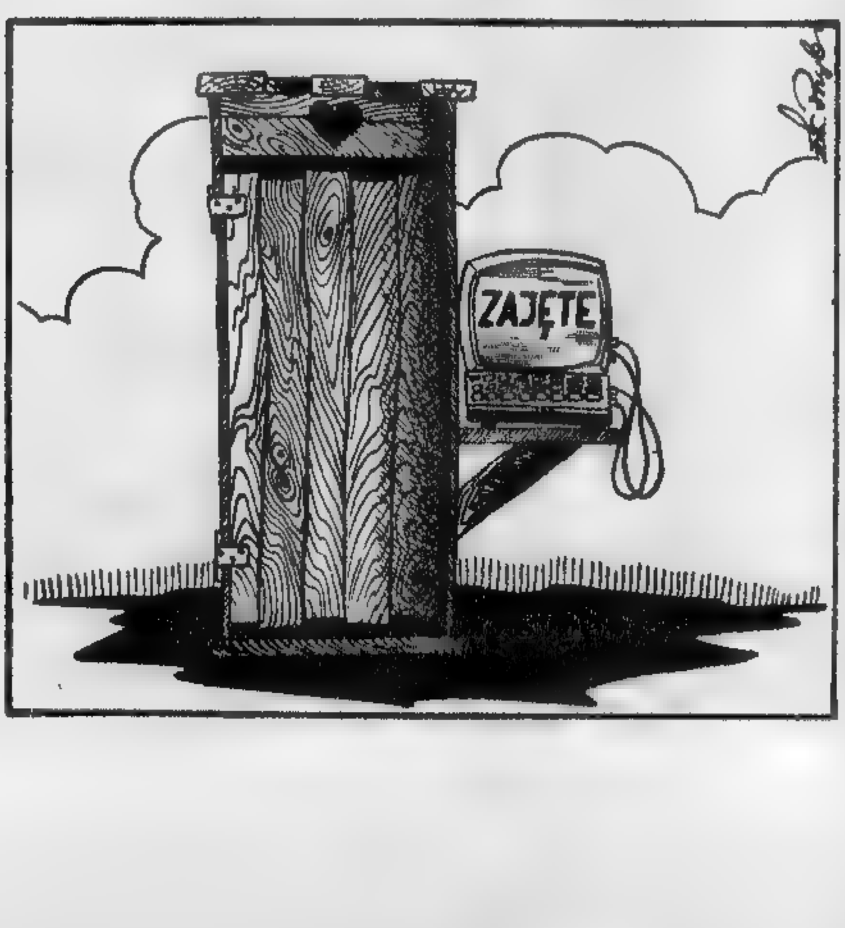
Odbieramy zatem telefony; co lepsze: Spectrum czy Commodore, czym różni się Amstrad od Atari, albo: dlaczego kasety z programem wyprawy gwiezdnej, działającym na BBS, nie można uruchomić na Spectrum? Co można wtedy odpowiedzieć? Pytamy zatem do czego przeznaczony ma być komputer. „No, chciałbym zagrać" — słyszymy najczęściej. Cóż, jeśli tylko o to chodzi, to wybranie komputera najdroższego jest niestety pierwszą przegraną w tej „grze".

Zabawa to krok pierwszy, od tego zaczynamy, ale nie wyobrażam sobie, aby nie pójść dalej. Idźmy zatem.

Wyruszamy w dalszą drogę po przygodę z informatyką. Nasze łamy prezentują przed wami, drodzy Czytelnicy, świat współczesnej informatyki. Nie zawsze jest on prosty, nie zawsze zaczyna się od zabawy — najczęściej zacząć trzeba od pracy i nauki. Na przykład prezentowane przez nas programy. Każdy z nich pisany jest na konkretny mikrokomputer, ale wystarczy poznać reguły programowania, aby zaadaptować je do własnych potrzeb. Albo problemy komputerowej grafiki, jakże fantastyczne figury można otrzymać na ekranie — spróbuj Czytelniku poznać z nami reguły animacji, to przecież może się przydać i przyda się z pewnością.

LICZYMY, że mozaika problemów, prezentowanych na naszych łamach przy pomocy Czytelników, stanie się z czasem przejrzystą drogą polskiej informatyki.

REDAKCJA



Historia oprogramowania

Komputer bez oprogramowania nie jest wiele wart. Oprogramowanie czyni z „żelastwa" narzędzie „przydatne i rozumne", wspomagające człowieka. Wielkie zasługi w rozwoju konstrukcji maszyn liczących położył BABBAGE, twórca pierwszej maszyny różnicowej. Dzieło jego udokumentowała córka angielskiego poety, AUGUSTA BYRON.

Po drugiej wojnie światowej nastąpił burzliwy rozwój sprzętu komputerowego i w konsekwencji oprogramowania. W 1946 roku **Konrad Züse**, nie mając możliwości zbudowania odpowiedniego sprzętu, dokonuje opracowania języka PLANKALKUL, który zawiera konstrukcje umożliwiające operowanie na zmiennych o różnej wartości. Można więc uznać, że rok 1946 to data utworzenia pierwszego języka algorytmicznego.

W 1949 roku **Francis Holberton** opracował ideę generowania programów sortowania i łączenia, zaś rok 1951 przynosi opracowanie pierwszego kompilatora AUTOCODE (dla komputera Ferranti MARK I), powstają również pierwsze koncepcje wykorzystywania w programowaniu podprogramów (Wilkes, Wheeler, Gill).

W 1955 roku powstaje idea stworzenia standardów dla systemów operacyjnych, kompilatorów, translatorów itp. W roku 1957 uruchomiono pierwszy program napisany w języku FORTRAN przez użytkownika.

Zastosowania komputerów i języków programowania przestają służyć tylko problemom gospodarczym, statystycznym i naukowym. Komputery wkraczają również do zastosowań wojskowych. W 1958 roku rozpoczyna działanie system obrony powietrznej SAGE. Oprogramowanie tego systemu obejmuje takie elementy jak: konwersacyjny dostęp do monitora graficznego, programową korektę przekłamań, transmisję danych oraz symulatory otoczenia.

Użytkownicy korzystający z języka FORTRAN uznali, że ma on wiele ograniczeń. Tworzą więc nowy język o nazwie ALGOL. Jest to język dobrze i starannie opracowany, o strukturze blokowej i rozwiniętych możliwościach wykonywania operacji matematycznych, całkowicie pomijający zagadnienia wprowadzania danych i wypisywania wyników. W 1960 roku na spotkaniu, które odbyło się w Paryżu, podjęto wiążące decyzje dotyczące ALGOLU oraz opracowano raport, który do dziś służy jako wzór definicji języka.

W tym samym roku zostaje uruchomiony w firmie SDS język JOVIAL, którego wprowadzenie oznacza dla wielu programistów pożegnanie z językiem typu asembler.

Rok 1960 jest bardzo obfity w wydarzenia; po konferencji paryskiej (dotyczącej ALGOLU) i wprowadzeniu języka JOVIAL, w grudniu tego roku opracowano i przekazano do wykorzystania język COBOL. Program napisany w tym języku zostaje wykonany na komputerach UNIVAC II i RCA 501 — otrzymane wyniki są identyczne.

Firma BORROUGHS opracowała nowy komputer B-5500, oparty na zastosowaniu języka ALGOL. Dla ułatwienia kompilacji i wykonania zadań zastosowano specjalne instrukcje obsługi stosu.

Rok 1962 przynosi opracowanie i oddanie do użytku (John Mc Carthy) języka LIPS przeznaczonego do zagadnień związanych ze sztuczną inteligencją i operowaniem symbolami. Kolejny rok to opracowanie przez firmę GENERAL ELECTRIC pierwszego systemu zarządzania bazą danych.

W 1964 roku John Kemeny oraz Tom Kurtz oddają do użytku BASIC — język, którego celem było ułatwienie korzystania z komputera tym użytkownikom, którzy nie są programistami. Język ten powstał na podstawie ALGOL-u.

W 1965 roku opracowano język APL, przeznaczony do definiowania i opisywania różnych funkcji komputera. W tym samym roku holenderski profesor Edsger W. Dijkstra podejmuje problem tzw. zakleszczania, który występuje wówczas, gdy dwa (lub więcej) programy zgłaszają akces do tych samych zbiorów danych.

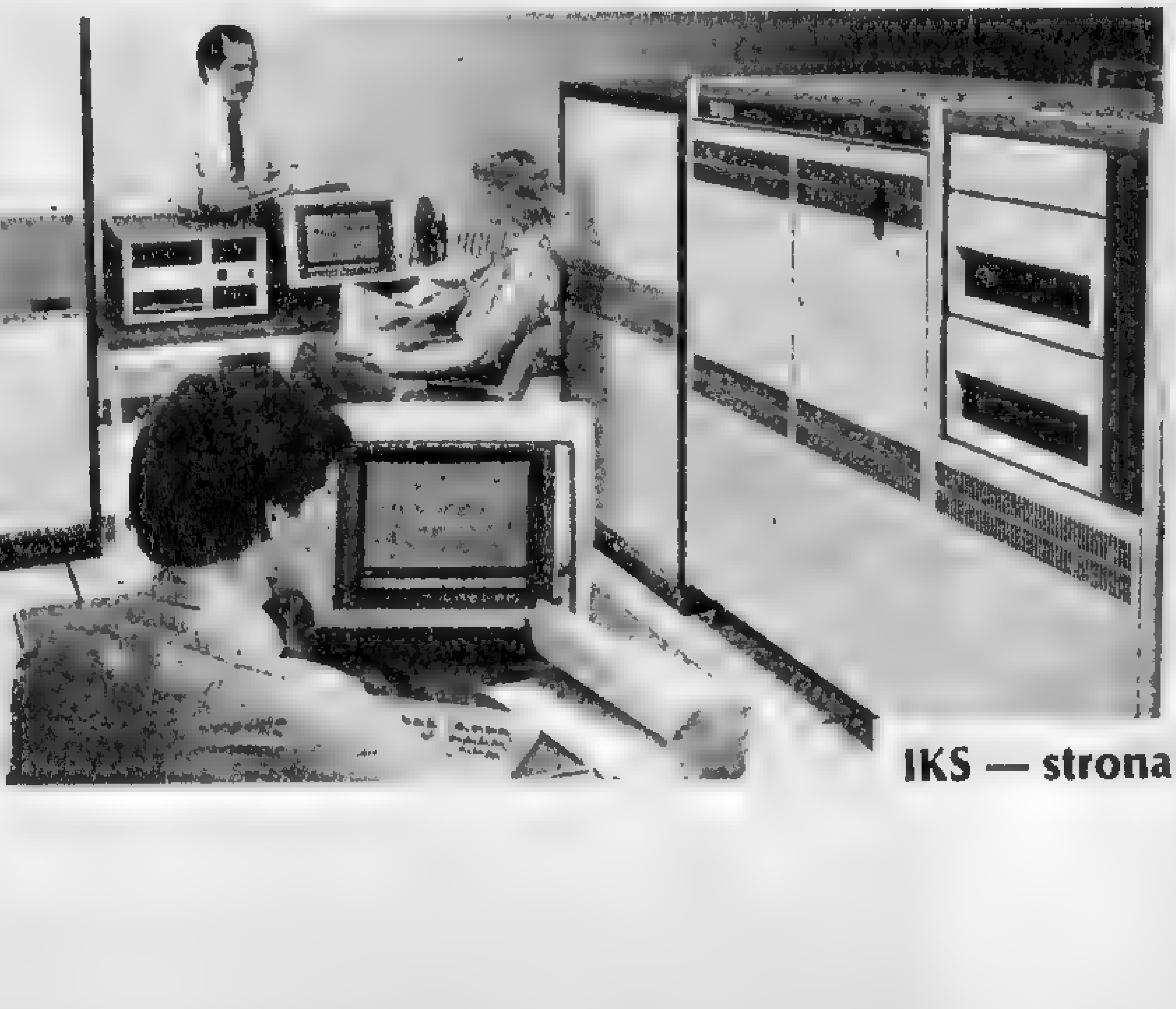
Dijkstra z problemem uporał się i w konsekwencji opracował system operacyjny THE, który problem podziału resursów rozwiązywał.

Rok 1968 przynosi słynny list prof. E. Dijkstry, skierowany do czasopisma Communications of the ACM. „*Jakość programisty* — pisze profesor — *jest odwrotnie proporcjonalna do liczby instrukcji GOTO użytych w jego programach. Ostatnio odkryłem, dlaczego użycie instrukcji GOTO ma tak fatalne skutki i uświadomiłem sobie, że instrukcja ta powinna być usunięta ze wszystkich języków wysokiego poziomu*". W ten sposób ogłoszono koncepcję tzw. programowania strukturalnego.

W roku 1971 firma IBM ogłosiła opracowanie tzw. pamięci wirtualnej, dzięki której programiści będą zwolnieni od potrzeby optymalizacji wykorzystania pamięci oraz segmentacji programów (idea ta była zrealizowana w 1958 r. w Anglii — komputer ATLAS oraz w 1961 r. w USA — komputer B-5500).

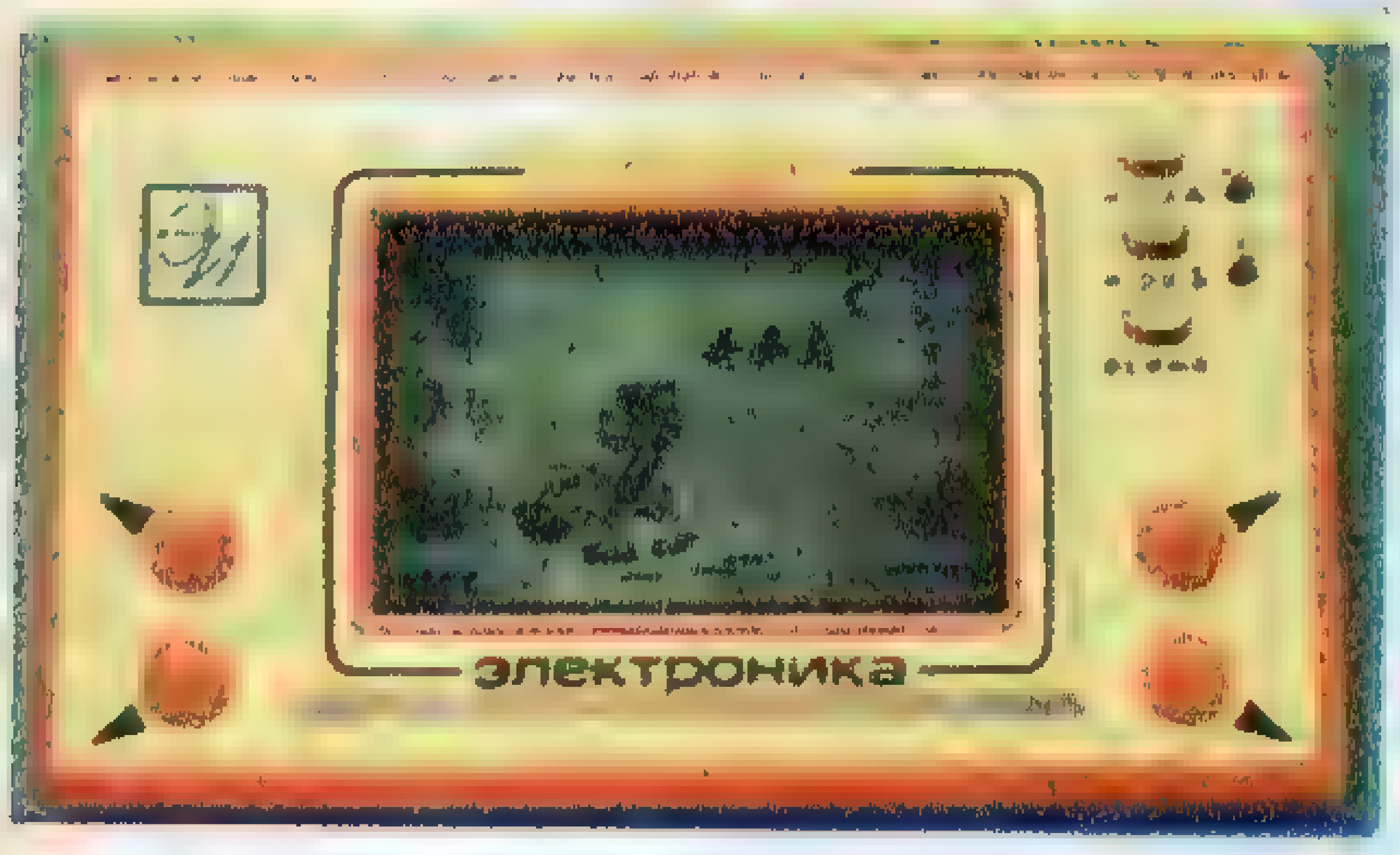
Rok 1979 to eksplozja mini-komputerów i początek fascynacji mikroinformatyką — ale to jeszcze nie jest historią.

Opracowano na podstawie „Magic Moments in Software" Datamation



„Nu, pogodi!"

Elektronika, a szczególnie jej najmłodsze dziecko, mikrokomputery, są przedmiotem zainteresowań wielu z nas. Profesjonaliści konstruują nowe ich generacje, stosują mikroprocesory w urządzeniach powszechnego użytku, wymyślają oprogramowanie zadowalające nawet wybrednych koneserów, a zarazem proste w użyciu. Dla niektórych, nie interesujących się specjalnie jego budową, mikrokomputer stał się prawdziwym pomocnikiem w codziennej pracy nie tylko zawodowej, przyczyniając się do podniesienia jej efektywności, nowatorskich rozwiązań, ale i do oszczędności czasu i wydatkowanej energii. Najwięcej jednak emocji wzbudza zabawa, gra, w której mikrokomputer jest przeciwnikiem. Godnym przeciwnikiem, który sprawia, że taka zabawa wymaga wysiłku intelektu, skupienia i szybkiego refleksu.



fot. 1

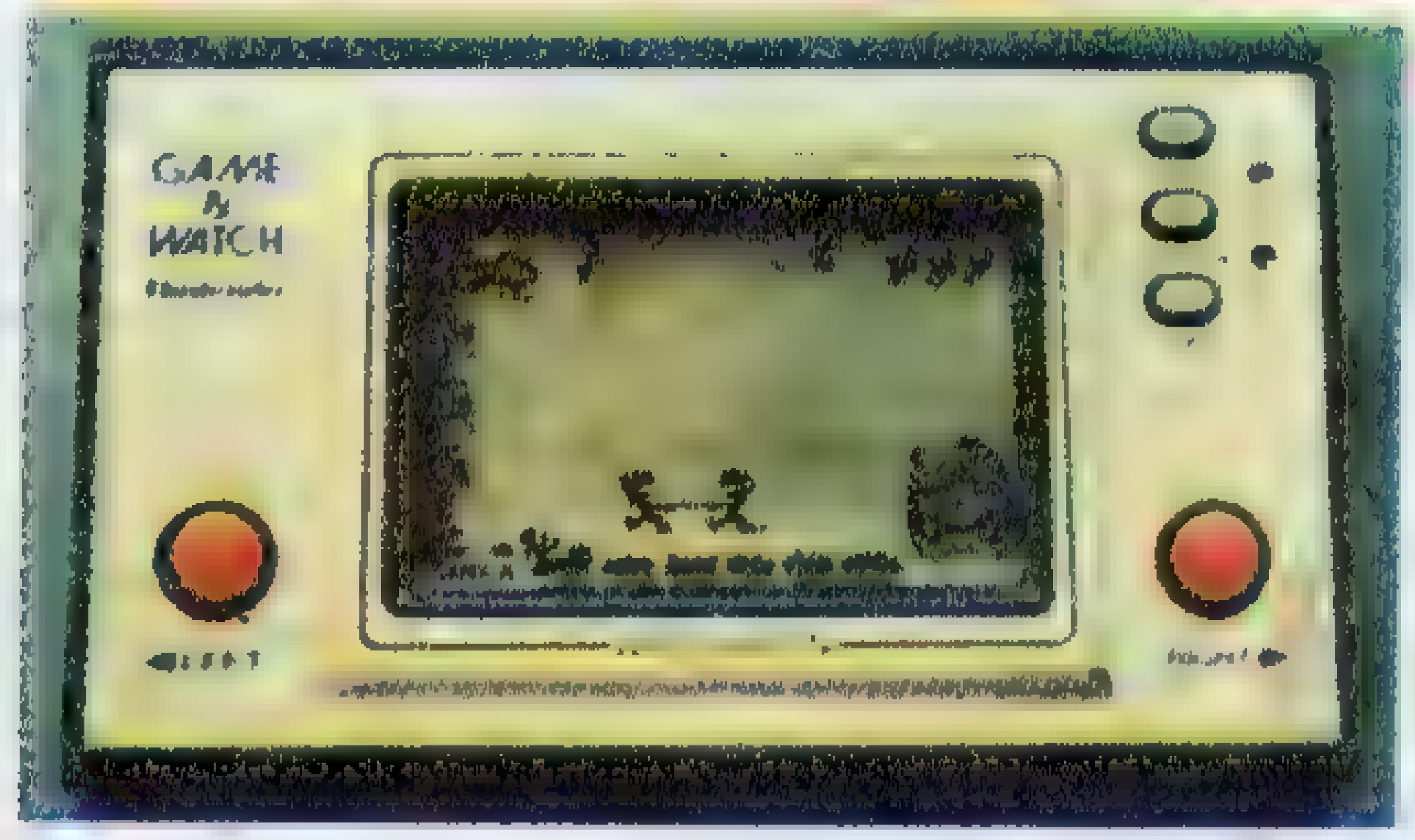
Taka rozrywka i zabawa jest najchętniej widzianą formą kontaktu z urządzeniem elektronicznym. Pomimo dużego zainteresowania i wypieków na twarzy nie każdy może być tym szczęśliwcem, w którego zasięgu jest mikrokomputer, przede wszystkim ze względu na jego cenę.

Jednak i tym elektronika sprawiła pewną niespodziankę — mikroprocesorową grę elektroniczną w cenie samochodu zdalnie sterowanego.



fot. 2

Podstawą tego urządzenia jest układ scalony LSI zawierający podstawowe elementy mikrokomputera tj. mikroprocesor, pamięć ROM i RAM, układy WE — WY. Możliwość programowania mikroprocesora, a także dowolny projekt układu gry, sprawiły, że powstały liczne tego typu zabawki, które dają równie dużo zadowolenia i emocji, co niejedna gra np. na ZX SPECTRUM, ale kosztują tanio i dają się nosić w piórniku. W wielu krajach w sklepach z zabawkami można kupić taką grę. Może niedługo nasz przemysł przygotuje mikroprocesorową grę np: „Bolka i Lolka" dla naszych milusińskich.



fot. 3

Produkowana przez Moskiewskie Zakłady Elektroniczne zabawka „Elektronika", z budzikiem, jest mikroprocesorową grą elektroniczną (fot. 1). Przeznaczona jest dla dzieci i młodzieży i kosztuje tylko 25 rubli.

Na ekranie przedstawiona jest jedna z sytuacji z filmu „Nu, pogodi!": wilk nadstawił koszyczek i czeka na zsuwające się z półki jajko. Wilk musi nałapać jak najwięcej jajek do koszyka. Jeżeli jajko wpadnie do koszyka, to liczba punktów wzrasta o jeden. Wynik wyświetlany jest na ekranie w prawym górnym rogu.

W grze można uzyskać maksymalnie 999 punktów. Jeżeli jednak, na skutek złego naciśnięcia klawisza, wilk nie złapie jajka, spadnie ono na ziemię i wyskoczy z niego kurczaczek. W tym przypadku grający otrzymuje punkt karny, zobrazowany na ekranie jako migający kurczaczek pod wynikiem gry.



Jeżeli nie uda się wilkowi złapać trzech jajek, to gra się kończy. Jeśli jednak uzyska się 200 i 500 punktów, powoduje to automatyczne skasowanie punktów karnych, a umieszczony wewnątrz głośniczek wyda sympatyczny, wielotonowy dźwięk. Utrudnieniem w grze jest zasada, że w miarę jak zwiększa się liczba zdobytych punktów, szybkość zsuwania się jajek wzrasta.

„TOM I JERRY" to inna, wyprodukowana w Japonii, mikroprocesorowa gra (fot. 2). Są tu dwa warianty: A i B.

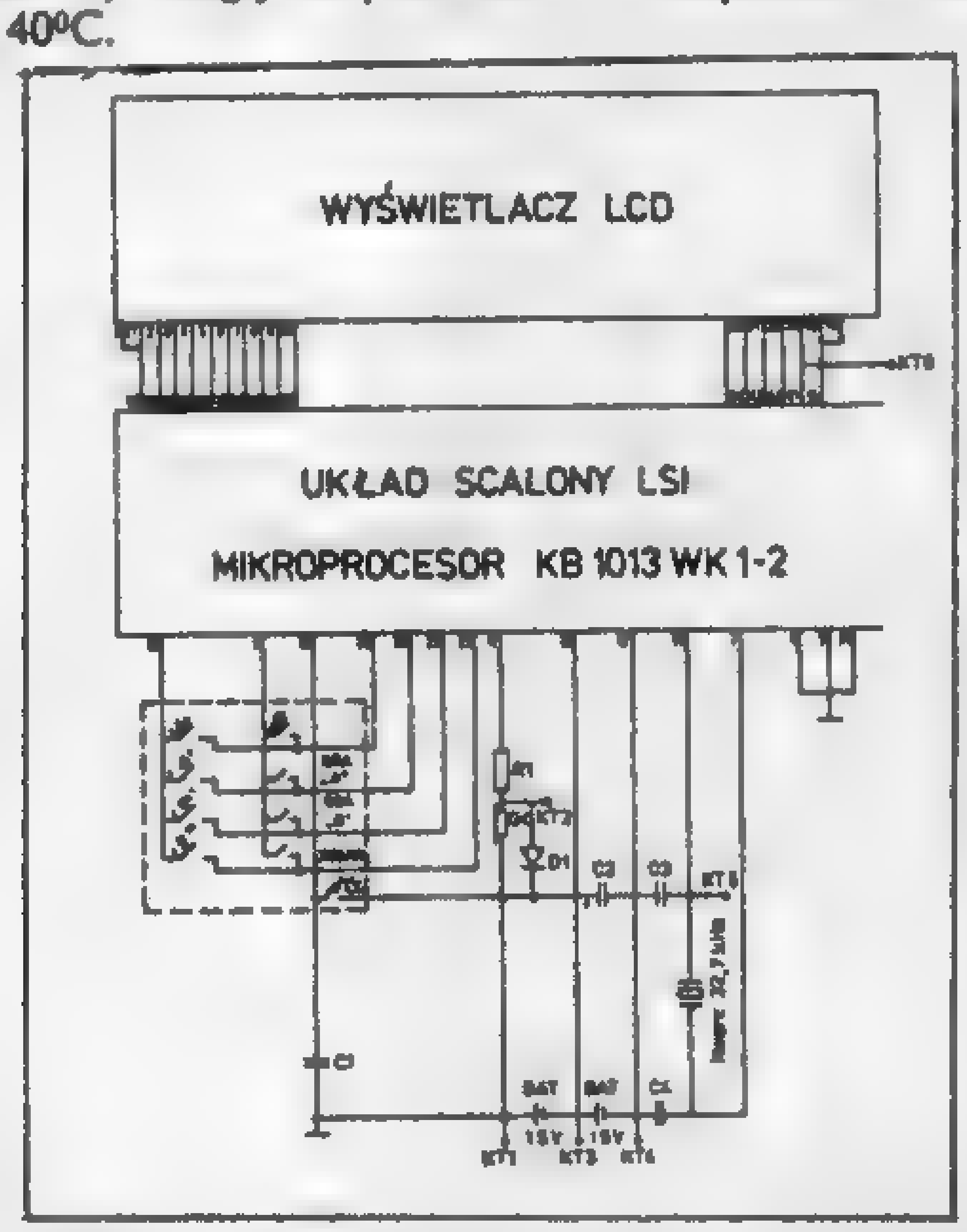
W wariancie „A" kot, machając łapami, broni się przed atakującymi go z dwóch stron myszkami. Zbicie przez kota myszki daje graczowi 10 punktów. Kota atakują również myszki z pułapkami. Jeżeli przypadkowo gracz naciśnie klawisz powodujący ruch łapy kota, tak że zbije on myszkę z pułapką, to kot staje w bezruchu, mysz spada na dół, a gracz uzyskuje punkt karny. Punkty karne, przedstawione na ekranie jako łapy kota, wyświetlane są w prawym dolnym rogu. Uzyskanie trzech punktów karnych kończy grę.

W wariancie „B", naciskając odpowiedni przycisk, można wylosować, mając szczęście, punkty dodatkowe lub punkty karne.

Uzyskanie większej liczby punktów powoduje automatyczne utrudnienie gry — kota zaczyna atakować więcej myszek.

Kolejną mikroprocesorową zabawą jest, pochodząca również z Kraju Kwitnącej Wiśni, gra „POŻAR" (fot. 3); z palącego się budynku wyskakują ludziki, sanitariusze muszą tak ustawić nosze, aby ludzik, kilkakrotnie odbijając się od nich, znalazł się w karetce. Za każde poprawne ustawienie noszy gracz otrzymuje jeden punkt. Jeżeli jednak nie zdąży prawidłowo ustawić noszy, otrzymuje punkt karny, a na ekranie ukazuje się „ludzik z zaświatów". Otrzymanie trzech punktów karnych kończy grę.

Gra ma dwa warianty. W pierwszym ludziki spadają z trzeciego, a w wariancie drugim z drugiego piętra. Tak jak poprzednio w miarę zdobywania punktów liczba ludzików się zwiększa, co utrudnia grę. Można również, uzyskując 200 i 500 punktów, skasować punkty karne. Zasada działania prezentowanych gier jest bardzo podobna. W podobny sposób są zasilane (dwie baterie LR 44), zużywają podobną ilość energii elektrycznej (ok. 0,00005 W), mogą też pracować w temperaturze od +5 do +40°C.



Na rys. 1 przedstawiony jest schemat układu elektronicznego takiej gry. Jak łatwo zauważyć, składa się on zaledwie z kilku elementów: układu wyświetlacza, miniaturowego układu scalonego mikroprocesora, kwarcu sterującego zegarem, brzęczyka, a także kilku typowych elementów elektronicznych. Aż trudno uwierzyć, że tak prosta konstrukcja może dać tyle zadowolenia dzieciom.

Andrzej SOBOLEWSKI
Piotr SOBOLEWSKI

PS Autorzy serdecznie dziękują koleżance Oli i koledze Mikołajowi ze Szkoły Podstawowej nr 261 w Warszawie za udostępnienie gier i cenne wskazówki.

Chochlik zamieszał...

Nam też zdarzył się błąd. Drobna pomyłka i program rysujący kalendarz działa błędnie. W wyniku błędu występującego w sekwencji instrukcji w wierszach od 110 do 140 poprawnie zadany miesiąc — grudzień — jest odrzucony z wypisaniem komunikatu: „Źle wprowadzony miesiąc".

Najprościej poprawne działanie programu uzyskamy zmieniając warunek w instrukcji IF z wiersza o numerze 135 do postaci:

```
135 IF m>11 AND m$<>q$
         THEN PRINT AT 10,0;"Zle
wprowadzony miesiac": PAUSE 50:
         CLS : RESTORE :
         GO TO 100
```

Błąd powstał w wyniku niezbyt przejrzystej konstrukcji pętli powodującej porównywanie wczytanego miesiąca. Prostsza i bardziej czytelna konstrukcja pętli wygląda następująco:

```
110 FOR m=0 TO 11
115 READ q$,f,d
120 IF m$=q$ THEN GO TO 150
125 NEXT m
130 PRINT AT 10,0;"Zle wprowadz
ony miesiac": PAUSE 50: CLS :
    RESTORE : GO TO 100
```

lub jeszcze lepiej:

```
110 LET m=0
115 READ q$,f,d
120 IF m$=q$ THEN GO TO 150
125 LET m=m+1
130 IF m>11 THEN
         PRINT AT 10,0;"Zle wpro
wadzony miesiac": PAUSE 50:
         CLS : RESTORE :
         GO TO 100
140 GO TO 115
```

Teraz z pewnością jest dobrze.

PRZEPRASZAMY!

PROGRAM 5

SPECTRUM

Compact jest programem na 16/48 KB Spectrum, pozwalającym na zmniejszenie zajętości pamięci i zwiększenie szybkości działania innych programów napisanych w **BASIC-u.** Umożliwia wykonywanie trzech podstawowych operacji:

– skracanie zapisu liczb bez wpływu na dokładność obliczeń;

– skracanie długich nazw zmiennych do nazw dwuznakowych postaci: AO, CO, CO... ZZ;

– kompresja (łączenie) wierszy programu, jeśli jest to możliwe.

W celu wykorzystania programu **Compact** załaduj go do pamięci instrukcjami:

CLEAR adres: LOAD "" CODE

gdzie adres jest początkowym adresem obszaru pamięci, która może wykorzystać program **Compact**. W zależności od posiadanych wersji mikrokomputera Spectrum możesz podać wartość adresu np. 32108 lub 64876.

Uruchom program **Compact** komendą RUN. Gdy otrzymasz pytanie o adres startowy, wprowadź wcześniej wybraną wartość adresu. Na zakończenie program **Compact** podaje adresy startowe operacji możliwych do wykonania (zapamiętaj je lub zapisz). Wykonaj NEW.

Wprowadź do pamięci program, który chcesz poddać obróbce programem **Compact**. Jeśli chcesz wykonać operacje, które umożliwia **Compact**, podaj komendy postaci:

RAND USR adres operacji

(adres operacji miałeś wcześniej podany). Przekształcone postacie swojego programu możesz obejrzeć wykonując LIST programu.

```
0 REM *** COMPACT ***
5 DATA 205,427+START,254,46,40,1
9,254,196,40,15,205,136,44,48,25
,254,65,56,6,205,65,41,223,24,14
,54,46,84,93,35,126,254,14,32,25
0,19,205,399+START,43
6 DATA 205,452+START,24,213,237,
123,61,92,205,179,30,33,65,48,34
,146,92,205,427+START,205,141,44
,48,64,235,231,235,205,136,44,48
,56,34,93,92,229,217,229,205
7 DATA 31,28,35,48,30,237,91,146
,92,205,182,42,123,28,254,90,32,
10,30,65,122,20,254,57,32,2,22,6
5,237,83,146,92,205,255,42,217,2
25,217,209,35,237
8 DATA 160,237,160,223,205,399+S
TART,43,205,452+START,24,182,42,
83,92,205,366+START,32,101,231,2
46,128,42,69,92,254,235,40,73,25
4,237,229,245,204,347+START,241,
225,40
9 DATA 25,35,254,236,40,242,254,
247,40,238,254,250,40,49,254,234
,40,45,254,254,40,41,254,229,32,
40,231,205,72,32,1,0,0,254,46,40
,7,40,22,205,27
10 DATA 45,56,117,62,14,71,237,1
77,205,180,51,126,205,72,32,32,1
03,205,153,30,96,105,205,347+STA
RT,223,254,234,40,5,205,57,30,48
,160,42,85,92,24,150,42
11 DATA 83,92,205,321+START,205,
184,25,235,205,321+START,229,205
,360+START,203,126,225,40,240,22
9,213,235,17,0,0,205,136,25,35,2
05,136,25,62,254,146,209,225,250
,103
12 DATA 254,19,19,213,84,93,35,3
5,78,35,70,35,197,205,229,25,43,
54,58,193,225,94,35,86,235,9,235
,114,43,115,43,43,24,186,126,230
,192,200,42,83,92
13 DATA 126,230,192,194,176,27,2
29,205,360+START,203,254,225,205
,184,25,235,24,237,205,110,25,12
6,230,192,192,205,360+START,203,
190,201,35,35,35,195,171,40,126,
230,192
14 DATA 192,50,71,92,86,35,94,23
7,83,69,92,35,229,221,225,94,35,
86,34,93,92,25,35,34,85,92,237,9
8,217,201,205,221,25,197,196,232
,25,217,193,9,217
15 DATA 201,126,205,182,24,254,3
4,34,93,92,192,35,71,237,177,24,
240,42,83,92,205,366+START,194,1
76,27,231,253,52,13,254,206,32,9
,35,205,411+START,254,41,32
16 DATA 248,35,205,411+START,254
,58,40,232,254,203,40,228,254,23
4,40,3,254,13,192,217,221,94,0,2
21,86,1,235,237,82,221,117,0,221
,116,1,42,85,92,24,194
20 INPUT "adres startowy?";START
30 IF START+491>PEEK 23732+256*P
EEK 23733 THEN GOTO 20
40 RANDOMIZE START: CLEAR START-
1: LET START=PEEK 23670+256*PEEK
 23671
50 FOR A=0 TO 491: READ B: IF B>
255 THEN RANDOMIZE B: POKE START
+A,PEEK 23670: LET A=A+1: POKE S
TART+A,PEEK 23671: GOTO 70
60 POKE START+A,B
70 NEXT A
80 PRINT "skracanie zapisu liczb
: ";START'"skracanie nazw zmienn
ych: ";START+46'"laczenie linii:
 ";START+136
```

PROGRAM 6

SPECTRUM

```
10 REM ZEGAR
20 LET MECHANIZM=270:
   LET CZAS=1000:
   LET SEKUNDA=2000:
   LET MINUTA=3000:
   LET GODZINA=4000
30 BORDER 1: PAPER 5: INK 9:
    BRIGHT 1: CLS
40 REM POMIAR CZASU JEST

       DOKONYWANY Z
       DOKLADNOSCIA 20 MSEK
50 INPUT "PODAJ GODZ.";T
60 LET N=T*3600*50
70 INPUT "PODAJ MIN.";MIN
80 LET N=N+MIN*60*50
90 INPUT "PODAJ SEK.";SEK
100 LET N=N+SEK*50
110 REM USTAW LICZNIK CZASU
120 LET A=INT (N/65536)
130 LET B=INT ((N-A*65536)/256)
140 LET C=N-65536*A-256*B

150 POKE 23764,A:
     POKE 23763,B:
      POKE 23762,C
160 REM TARCZA
170 FOR N=1 TO 12
180 PRINT AT 10-9*COS(N*PI/6),
        15+9*SIN(N*PI/6);N
190 NEXT N
200 CIRCLE 123,90,82
210 GOSUB CZAS
220 LET FIRST=1
230 GOSUB SEKUNDA
240 GOSUB MINUTA
250 GOSUB GODZINA
260 LET FIRST=0:
     LET FLAGG=0
270 REM MECHANIZM
280 GOSUB CZAS
290 IF SEK<>S THEN GOSUB     SEKUNDA
300 IF MIN<> THEN GOSUB     MINUTA
310 IF MIN/12<>INT(MIN/12)
     THEN LET FLAGG=0
320 IF MIN/12=INT(MIN/12)
     AND FLAGG=0 THEN GOSUB     GODZINA
330 GOTO MECHANIZM
950 REM **SUBROUTINES**
```

Po uruchomieniu tego programu na ekranie monitora pojawi się zegar. Dokładność wskazań zależy od precyzji danych wejściowych, czyli czasu „nakręcania" oraz sprawności układów elektronicznych mikrokomputera Spectrum.

dokończenie na stronie 8

dokończenie ze strony 7

```
1000 REM CZAS
1010 LET N=INT((65536*PEEK
        23764+256*PEEK 23763
        +PEEK 23762)/50)
1020 LET K=INT((65536!*PEEK
        23674+256*PEEK 23763
        +PEEK 23762)/50)
1030 LET N=N*(N>K)+K*(N<=K)
1040 LET T=INT(N/3600)
1050 IF T>=12
     THEN POKE 23764,0:
          POKE 23763,0:
          POKE 23762,0
1060 LET MIN=
        INT((N-T*3600)/50)
1070 LET SEK= N-60*MIN-3600*T
1080 RETURN
2000 REM SEKUNDA
2010 LET S=SEK
2020 IF FIRST=0
      THEN PLOT 123,90:
           DRAW OVER 1;SX,SY:
           REM ZGAS SEKUNDNIK
2030 LET GRAD=SEK*PI/30:
      LET SY=70*COS GRAD:
       LET SX=70*SIN GRAD
2040 PLOT 123,90:
      DRAW OVER 1;SX,SY:
         REM SEKUNDNIK
2050 BEEP .02,12
2060 RETURN
3000 REM MINUTA
3010 LET M*MIN
3020 IF FIRST=0
      THEN PLOT 123,90:
           DRAW OVER 1;MX,MY:
           REM ZGAS MIN.
3030 LET GRADM=MIN*PI/30:
      LET MY=60*COS GRADM:
       LET MX=60*SIN GRADM
3040 PLOT 123,90:
      DRAW OVER 1;MX,MY:
      REM MINUTY
3050 RETURN
4000 REM GODZINY
4010 LET FLAGG=1
4020 IF FIRST=0
      THEN PLOT 123,90:
           DRAW OVER 1;
            TX,TY,.3:
           DRAW OVER 1;
            -TX,-TY,.3:
      REM ZGAS GODZ.
4030 LET GRADG=
        (T+MIN/60)*PI/6:
     LET TY=50*COS GRADG:
      LET TY=50*SIN GRADG
4040 PLOT 123,90:
     DRAW OVER 1;TX,TY,.3:
       DRAW OVER 1;-TX,-TY,.3:
        REM GODZINY
4050 RETURN
```

PROGRAM 7

ATARI

```
10 REM ODWRACANKA
25 ? "[CLEAR]"
30 DIM A(5,5),D(100),E(100),B$(2),C$(2)
35 OPEN =1,4,0,"K:"
40 DIM L1$(16),L2$(16),L3$(16),L4$(16)
45 DIM O$(2)
50 B$ "  ":C$ "   "
60 L1$ "[O][R][R][W][R][R][W][R][R][W][R][R]
   [W][R][R][E]"
70 L2$ " !     !     !     !     !     ! "
80 L3$ "[A][R][R][S][R][R][S][R][R][S][R][R][S][R][R][D] "
90 L4$ "[Z][R][R][X][R][R][X][R][R][X][R][R][X][R][R][C] "
95 X 12:Y 2
100 FOR I 1 TO 5:FOR J 1 TO 5
110 A(I,J) 0: NEXT J:NEXT I:N 0
120 GOSUB 5000:REM STRONA TYTULOWA
130 GR.0:SET.2,1,1:SET.4,12,6:POKE 752,1
135 GOSUB 1000:REM KONTUR:GOSUB 2000
140 N N+1
150 POS.1,20:PRINT "PODAJ POZYCJE ODWRACAN
    EGO POLA:   "
153 POS.1,21:? "         "
155 GET =1,U1:IF U1 49 OR U1 53 THEN 155
156 POS.1,21:U U1—48:? U
157 GET =1,V1:IF V1 49 OR V1 53 THEN 157
158 POS.3,21:V V1—48:? V
160 POS.1,20:? "                              "
170 IF U 1 OR U 5 THEN 150
180 IF V 1 OR V 5 THEN 150
190 U INT(U):V INT(V):D(N) U:E(N) V
200 GOSUB 2000:GOSUB 1500:REM ODWRACANIE
210 FOR I 1 TO 5:FOR J 1 TO 5
220 IF A(I,J)  2 THEN 140
230 NEXT J:NEXT I
240 GOSUB 2500:REM WYGRANA
1000 REM KRESLENIE KONTURU
1005 POS.X,Y:? L1$
1010 POS.X,Y+1:? L2$
1020 POS.X—1,Y+2:?"1";L2$
1025 FOR I 2 TO 5
1030 POS.X,Y+3*I—3:? L3$
1040 POS.X,Y+3*I—1:? L2$
1050 POS.X—1,Y+3*I—1:? I;L2$
1055 NEXT I
1060 POS.X,Y+15:? L4$
1070 POS.X,Y—1:? "  1  2  3  4  5"
1080 X X+1:Y Y+1
1090 RETURN
1500 REM WYDRUK ODWROCENIA
1510 FOR I 1 TO 5:FOR J 1 TO 5
1520 IF A(I,J) 3 THEN GOSUB 1600:A(I,J) 2
1530 IF A(I,J) 1 THEN GOSUB 1650:A(I,J) 0
1540 NEXT J:NEXT I
1550 RETURN
1600 REM CZARNE NA BIALE
1610 SOUND 0,121,10,8
```

W tej grze komputer rysuje na ekranie planszę składającą się z 25 kwadratów. Każdy z nich ma dwie strony: ciemną i jasną. Twoim zadaniem jest zamienienie wszystkich ciemnych pól na jasne, w jak najmniejszej ilości ruchów. Dla utrudnienia, oprócz wskazanego kwadratu, odwracane są zawsze cztery sąsiednie pola (o ile mieszczą się na planszy). Zagraj sam z sobą. Powodzenia!

```
1620 POS.X+3*(J—1),Y+3*(I—1):? B$
1630 POS.X+3*(J—1),Y+3*(I—1)+1:? B$
1640 SOUND 0,0,0,0:RETURN
1650 REM BIALE NA CZARNE
1660 SOUND 0,60,10,8
1670 POS.X+3*(J—1),Y+3*(I—1):? C$
1680 POS.X+3*(J—1),Y+3*(I—1)+1:? C$
1690 SOUND 0,0,0,0:RETURN
2000 REM ODWRACANIE
2010 GOSUB 2100
2020 U U+1:GOSUB 2100
2030 U U—2:GOSUB 2100
2040 U U+1:V V+1:GOSUB 2100
2050 V V—2
2100 IF U 1 OR U 5 THEN 2160
2110 IF V 1 OR V 5 THEN 2160
2120 IF A(U,V) 2 THEN 2150
2130 IF A(U,V) 0 THEN A(U,V) 3
2140 RETURN
2150 A(U,V) 1
2160 RETURN
2500 REM WYGRANA
2510 GR.2
2515 SET.0,2,14:SET.1,3,2:SET.2,8,12:SET.3,4,4:SET.4,11,2
2520 POS.6,4:? #6; "WYGRANA"
2530 POS.1,9:? #6;" w ";N;" krokach "
2534 ?
2535 ? "NOWA GRA ? ( SELECT / OPTION )"
2540 IF PEEK(53279) 3 THEN GOSUB 3000
2550 IF PEEK(53279) 5 THEN GOTO 2700
2560 GOTO 2540
2700 REM NOWA GRA
2710 FOR I 1 TO 5:FOR J 1 TO 5
2720 A(I,J) 0:NEXT J:NEXT I:N 0
2730 X 12:Y 2
2740 GOTO 130
3000 REM KROKI PARTII
3010 ? "CZY CHCESZ OBEJRZEC KROKI WYGRANEJ
     PARTII ? (T/N)"
3020 INPUT O$
3030 IF O$ "N" THEN NEW
3040 IF O$ "T" THEN 3020
3050 GR. 0
3060 FOR I 1 TO N
3070 ? D(I);",";E(I);" ":
3080 NEXT I
3090 FOR WA 1 TO 5000:NEXT WA
3100 NEW
3500 OPIS ZASAD GRY
3505 GR.0:POS.2,5
3520 ? "    CELEM GRY JEST ZAMIANA CZARNYCH
     POL NA BIALE W KWADRACIE 5 5. JEDNO-
     CZESNIE ODWRACA SIE ";
3530 ? "PIEC POL W KSZTAL-CIE KRZYZA, KTOREGO
     SRODKIEM JEST PO-LE O PODANYCH WSPO
     LRZEDNYCH. ";
3540 ? "JESLI ZA-DANIE POLE LEZY PRZY BRZEGU
     KWADRATU   ODWRACANA JEST TYLKO
     CZESC KRZYZA.     GRA KONCZY SIE PO";
3550 ? " ODWROCENIU WSZYST-KICH POL. MOZE
     SZ OTRZYMAC WYDRUK KRO-KOW WYKON
     ANYCH W GRZE."
3555 ? :? " UZYWAJ TYLKO KLAWISZY 1 2 3 4 5 ."
3557 ? "
3560 FOR I 1 TO 2000:NEXT I
3570 RETURN
5000 REM STRONA TYTULOWA
5010 GR.2+32
5020 SET.0,1,2:SET.1,3,2:SET.2,8,12:SET.3,4,6:SET.4,11,2
5030 POS.5,5:? #6;"ODWRACANKA"
5045 ? " START      POCZATEK GRY"
5052 ?
5055 ? " OPTION     ZASADY GRY"
5058 POKE 752,0
5060 IF PEEK(53279) 6 THEN RETURN
5065 SET.3,14,2:SET.0,2,14
5067 FOR W 1 TO 100:NEXT W
5070 IF PEEK(53279) 3 THEN 3500
5075 SET.3,14,4:SET.0,2,6
5077 FOR W 1 TO 50:NEXT W
5080 GOTO 5060
```


UWAGA!

Niektóre znaki występujące na ekranie monitora nie są dostępne w druku, dlatego też przyjęliśmy poniższe zasady zapisu:

- pojedyńcze znaki w nawiasach kwadratowych odpowiadają symbolom graficznym, wprowadzane są przy naciśniętym klawiszu CTRL;
- ciąg znaków w nawiasie kwadratowym odpowiada następującym czynnościom: klawisz ESC, a potem klawisz CTRL i ujęty w nawiasie jednocześnie.

PROGRAM 8

COMMODORE C-64

```
5 POKE53280,0 POKE53281,0
10 PRINT" "
15 PRINT"S"TAB(7)"Z Y C I E  M A K L E
 R A"TAB(36)"C.S."
20 PRINT"    JESTES WLASCICIELEM KOPA
LNI. WYKO
25 PRINT"RZYSTUJAC OPERACJE NA GIELDZIE
SPROBOJ"
30 PRINT"ZARZADZAC, JAK NAJDLUZEJ POSIAD
ANYM"
35 PRINT"MAJATKIEM "
40 PRINT"   U W A G A !!!"
45 PRINT"NIE MOZE PRACOWAC MNIEJ NIZ 10
GORNIKOW"
50 PRINT"W JEDNEJ KOPALNI."
55 PRINT"POZIOM ZYCIA GORNIKOW OKRESLA W
YDAJNOSC"
60 PRINT"ICH PRACY  I WIELKOSC ZATRUDNIEN
IA."

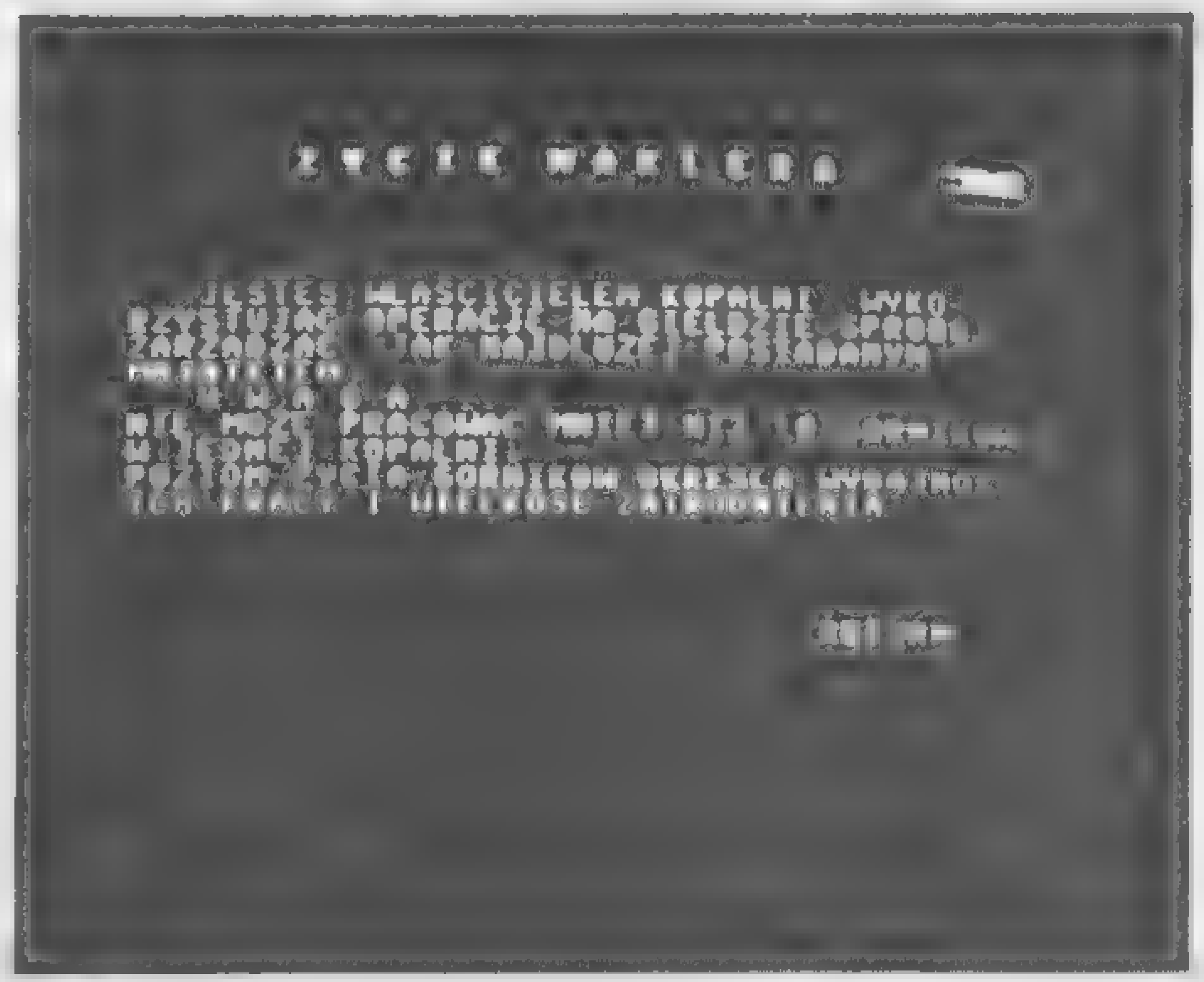
65 PRINT"SQQQQQQQQQQQQQQQQQQQQ"TAB(30)"RE
TURN"
70 PRINT"SQQQQQQQQQQQQQQQQQQQQQ"TAB(30)"RE
TURN"
75 GETR$ IFR$ ""THEN15
100 LK=INT(RND(1)*4+5)
110 LG INT(RND(1)*30+50)
120 SK=INT(RND(1)*50+10)*LG
130 RW INT(RND(1)*40+80)
140 Z=0 PZG=1 R=1 ZM=0
200 PRINT"   "TAB(7)"Z Y C I E  M A K L
 E R A"
205 PRINT"  "TAB(10)"ROK",R
210 PRINT"  OBECNIE POSIADASZ"
215 PRINT"STAN KONTA        ......",SK,"$"
220 PRINT"LICZBA GORNIKOW.........",LG
225 PRINT"LICZBA KOP         .....",LK

230 PRINT"ROCZNE WYDOBYCIE.......",RW,"TO
N KAZDA"
235 Z=Z+RW*LK
240 PRINT"ZAPASY.....................",Z,"TON
 RUDY"
245 PRINT"POZIOM ZYCIA GORNIKOW.",PZG
250 PRINT"  G I E L D A "
255 CR INT(RND(1)*12+7)
260 CK=INT(RND(1)*2000+2000)
265 CZ=INT(RND(1)*40+80)
270 PRINT"CENA TONY RUDY.............
..",CR,"$"
275 PRINT"CENA KOPALNI...............
..",CK,"$"
280 PRINT"CENA ZYWNOSC DLA GORNIKA....
..",CZ,"$"
285 PRINT"  KUPNO  SPRZEDARZ"
290 GOSUB1000
295 INPUT"ILE TON RUD  SPRZEDAJESZ",SR

300 IFSR<0ORSR>ZTHEN290
305 IFSR<>0THENZ=Z-SR:SK=SK+SR*CR GOSUB1
010 GOSUB1030
310 GOSUB1000
315 INPUT"ILE KOPALNI  SPRZEDAJESZ",KS
320 IFKS<0ORKS>LKTHEN310
325 IFKS<>0THENLK=LK-KS SK=SK+KS*CK GOSU
B1010 GOSUB1020
330 IFLK 0THEN2000
335 GOSUB1000
340 INPUT"ILE KOPALNI CHCESZ KUPIC",KK
345 IFKK<0ORKK*CK>SKTHEN355
350 IFKK<>0THENLK=LK+KK SK=SK-KK*CK GOSU
B1010 GOSUB1020
355 GOSUB1000
360 INPUT"ILE $ WYPLACASZ GORNIKOM",WG
365 IFWG<0ORWG>SKTHEN355
370 IFWG<>0THENSK SK WG GOSUB1010

400 IFWG/LG>CZ+20THENPZG=PZG+0.1 GOSUB10
40
405 IFWG/LG<CZ-20THENPZG=PZG-0.1 GOSUB10
40
410 IFPZG<0.6THEN2010
415 IFPZG>1.1THENRW=RW+INT(RND(1)*20+5)
420 IFPZG>=1THEN435
425 IFRW>25THENRW=RW-INT(RND(1)*20+5) IF
RW<0THENRW=0
430 GOSUB1050
```

Trudne jest życie maklera. Masz szansę stać się milionerem. Twój majątek to akcje kopalni rudy, czy zrobisz na nich fortunę? — wszystko zależy od Ciebie. Program na Commodore-64 przeniesie Cię na giełdę. Czy potrafisz zarobić?



Pierwszy komunikat poprawnie działającego programu...

```
435 IFLG/LK<10THEN2020
440 IFPZG>1.1THENLG=LG+INT(RND(1)*10+1)
GOSUB1060
445 IFPZG<0.9THENLG-LG INT(RND(1)*10+1)
GOSUB1060
450 IFRND(1)>0.2THEN465
455 GOSUB1500 PRINT"RADIOAKTYWNE MINERAL
Y.....WIELU ZGINELO"
460 LG INT(LG/2) GOSUB1060 ZN=1

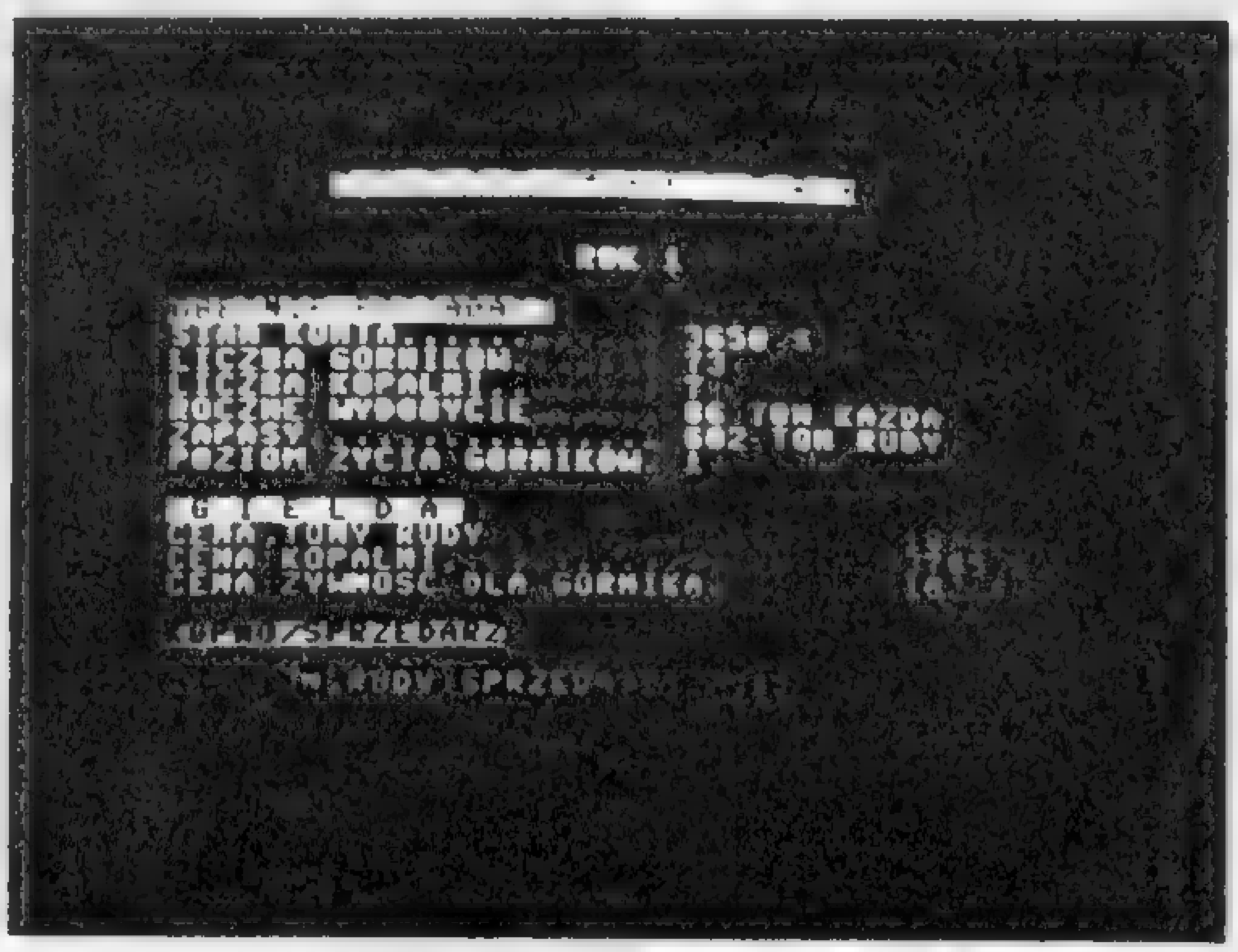
465 IFRW<150THEN500
470 IFZN 1THENZN 0 GOSUB1500
475 PRINT"KRYZYS NADPRODUKCJI.....WYDOBY
CIE SPADA"
480 RW=INT(RW/2) GOSUB1050
500 R=R+1
510 FORI=1TO1500 NEXT
520 GOTO200
1000 FORI=1824TO2023 POKEI,32 NEXT
1001 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"; RETUR
N
1010 FORI=1287TO1303 POKEI,32:NEXT
1011 PRINT"SQQQQQQQ"TAB(22),SK;"S":RETURN

1020 FORI=1367TO1383 POKEI,32:NEXT
1021 PRINT"SQQQQQQQQQ"TAB(22),LK:RETURN
1030 FORI=1447TO1463 POKEI,32:NEXT
1031 PRINT"SQQQQQQQQQQQ"TAB(22),Z,"TON RU
DY" RETURN

1040 FORI 1487TO1503 POKEI,32 NEXT
1041 PRINT"SQQQQQQQQQQQQQ"TAB(22),PZG RETU
RN
1050 FORI=1407TO1423 POKEI,32 NEXT
1051 PRINT"SQQQQQQQQQQQ"TAB(22),RW,"TON KA
ZDA" RETURN
1060 FORI 1327TO1343 POKEI,32 NEXT
1061 PRINT"SQQQQQQQQQ"TAB(22),LG RETURN
1500 FORI 1TO10
1505 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQQ"
1510 PRINT" S Y T U A C J E " FORJ 1T
050 NEXT
1515 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQQ"
1520 PRINT" S Y T U A C J E " FORJ 1T
050 NEXT
1525 NEXTI RETURN
2000 GOSUB1500 PRIN "NIE MASZ KOPALNI.
..KONIEC GRY" GOTO3000

2010 GOSUB1500:PRINT"STRAJKI GORNIKOW...
..KONIEC GRY" GOTO3000
2020 GOSUB1500 PRINT"ZA MALO GORNIKOW...
..KONIEC GRY" GOTO3000
3000 FORI=1TO1500:NEXT
3010 PRINT"  KONIEC GRY "
3020 PRINT"  BYLES WLASCICIELEM KOPALNI
LAT",R
3030 INPUT" CZY CHCESZ JESZCZE GRAC (T/N
)",R$
3040 IFR$="N"THENSTOP
3050 IFR$="T"THENRUN100
3060 GOTO3030
```

Co kupić?
Co sprzedać?
Decyzja należy do Ciebie!
Zdecyduj!



WIDEOTEKST

Włodzimierz Gogołek

Trudno sobie wyobrazić współczesny dom bez odbiornika telewizyjnego. Fakt ten z coraz większym powodzeniem wykorzystywany jest w technice wideotekstowej. Już w 1976 roku brytyjska firma BBC zaproponowała użytkową eksploatację systemu, który pozwala na to, aby telewizor zastąpił gazetę. Jej strony zmieniają się po naciśnięciu odpowiedniego klawisza. Zatem, aby przeczytać wieczorną gazetę, wystarczy włączyć telewizor, nastawić go na odpowiedni kanał, a zgodnie z naszym życzeniem na ekranie ukażą się kolejne kolumny wybranego pisma.

DROGA uzyskania przedstawionego efektu jest stosunkowo prosta i nie wymaga od użytkownika dużych nakładów. Wystarczy bowiem dostęp do linii telefonicznej i przystawka (dekoder) do typowego telewizora. Po wykręceniu odpowiedniego numeru telefonu dekoder zostaje połączony z centralnym komputerem. Jest on źródłem oczekiwanych informacji. Dotyczą one różnorodnych tematów. Na przykład:

— aktualne informacje gospodarcze,
— ceny, taryfy,
— katalogi,
— wiadomości bieżące
— poczta elektroniczna,
— dane medyczne (symptomy chorób, charakterystyki lekarstw — system AMNET).

Warto w tym miejscu przypomnieć, iż obraz telewizyjny składa się z ponad 600 linii. Są one rysowane przez wiązkę elektronów, emitowaną przez działo elektronowe kineskopu. Wiązką sterują układy elektroniczne telewizora (zgodnie z sygnałem odbieranym przez antenę). Zależnie od treści obrazu linie te mają różny poziom czerni (w przypadku telewizji czarno-białej). Gdy na ekranie wyświetlany jest tekst, wiązka rysuje tylko czerń lub biel, bez odcieni pośrednich. Innymi słowy zera lub jedynki. Stąd już tylko jeden „krok" do komputerów.

Aby zachować złudzenie stałości obrazu, jest on „rysowany" linia po linii 50 razy w ciągu jednej sekundy. Czas rysowania jednej linii wynosi 64 mikrosekundy. Ale po narysowaniu każdej linii, wiązka musi wrócić do lewej krawędzi ekranu. Właśnie podczas tych powrotów przesyłane są przez centralny komputer odpowiednio pogrupowane zera i jedynki. Oczekuje na nie wspomniana przystawka telewizora — dekoder. Gromadzi on w swojej pamięci tak „przemycane" zera i jedynki i tworzy z nich oczekiwany przez nas wideotekst. Zatem transmisja tego typu informacji nie przeszkadza w ciągłym oglądaniu telewizyjnego obrazu.

Gdyby pamięć dekodera była podobna do posiadanej przez domowy mikrokomputer, można by było zapamiętać w niej 40 typowych stron wideotekstu. Praktycznie pamięci te są znacznie większe. Zatem konstrukcja dekoderów umożliwia nam

dokończenie na stronie 14

dokończenie ze strony 13

natychmiastowe wyświetlenie dowolnej strony aktualnej gazety. Czas odczytu znaku z tego typu pamięci jest bowiem mniejszy od 0,000003 sekundy.

Omawiana metoda dostarczania informacji ma bezwzględną przewagę nad tradycyjnymi formami masowych środków informacji w kwestii aktualności danych. Często jest to decydujące w komercyjnych zastosowaniach, gdzie cena tego typu usług bywa znacznie mniejsza od szacunkowych zysków wynikających ze zmniejszenia okresu odszukiwania odpowiednich informacji. Zachęcają do tego typu zastosowań różne sy-



stemy wideotekstowe. Na przykład system PRESTEL świadczy poza ogólnodostępnymi usługi w zakresie gromadzenia i dystrybucji (drogą wideotekstu) informacji poufnych, dostępnych po podaniu obowiązującego hasła.

Jednym z drobnych, ale niezmiernie istotnych elementów techniki wideotekstowej jest klawiatura, która pozwala odbiorcy na wybór odpowiedniej informacji oraz formy jej prezentacji. Jest to zazwyczaj poręczne urządzenie o wymiarach i konstrukcji przypominającej typowy kalkulator kieszonkowy.

Systemy teletekstowe stają się coraz popularniejsze, aczkolwiek dzięki ekspansji mikrokomputerów zmieniają swój charakter. Wydaje się, że technika mikro w znacznym stopniu udoskonali dotychczasowe formy informacji odbieranych przez abonentów wideotekstów.

Firma Satellite Syndicated Systems zapewnia, iż transmisje wideotekstów mają między innymi następujące parametry:

— szybkość transmisji — 5,554195 Mbit/s,
— format strony — 19 wierszy, 40 kolumn
— kolory — czarny, czerwony, zielony, żółty, niebieski, magenda, siny jako dodatek do białego,
— specjalne znaki kontrolne — pulsujące litery, litery o podwójnej wysokości,
— możliwość drukowania tekstów.

Dane te dowodzą możliwości bezpośredniego sprzęgnięcia techniki wideotekstowej z domowym sprzętem mikrokomputerowym.

Mikrokomputery dzięki swej autonomii zmieniają dotychczasowy charakter wideotekstu. Umożliwiają między innymi trudne dotychczas do wyobrażenia doskonalenie formy informacji teletekstowej. Urządzeniem odbierającym staje się nie przystosowany telewizor, lecz praktycznie nie mniej popularny domowy komputer. Poza wspomnianymi udoskonaleniami pozwala on nie tylko na gromadzenie, ale także przetwarzanie uzyskiwanych drogą teletekstu informacji.

Powstaje w ten sposób nowy twór, który praktycznie ma jedną wspólną cechę z historyczną już wersją wideotekstu BBC z 1976 roku — ekran telewizora. Okazuje się, iż stanowi on często ukrywaną przez firmy komputerowe istotną barierę w upowszechnianiu techniki wideotekstowej — coraz częściej w publikacjach autorzy zauważają niepokojący wzrost liczby osób, których choroby związane są z korzystaniem z obrazu telewizyjnego, jako pośrednika w przekazie informacji wyświetlanej w postaci znaków. Ponad 50% osób korzystających profesjonalnie z obrazów TV skarży się na różnego rodzaju dolegliwości, których przyczyna tkwi w szkodliwości tych urządzeń.

PRZYCZYN tego stanu rzeczy należy upatrywać między innymi w złej formie informacji prezentowanej na ekranie TV. Badania dowodzą dużego znaczenia także wielu innych czynników mających wpływ na komfort odbioru tego typu informacji. Na przykład:

— rozdzielczość, liczba punktów, tak zwanych pikseli, z których tworzone są znaki (nie powinna być mniejsza niż 28 pikseli na centymetr kwadratowy ekranu),

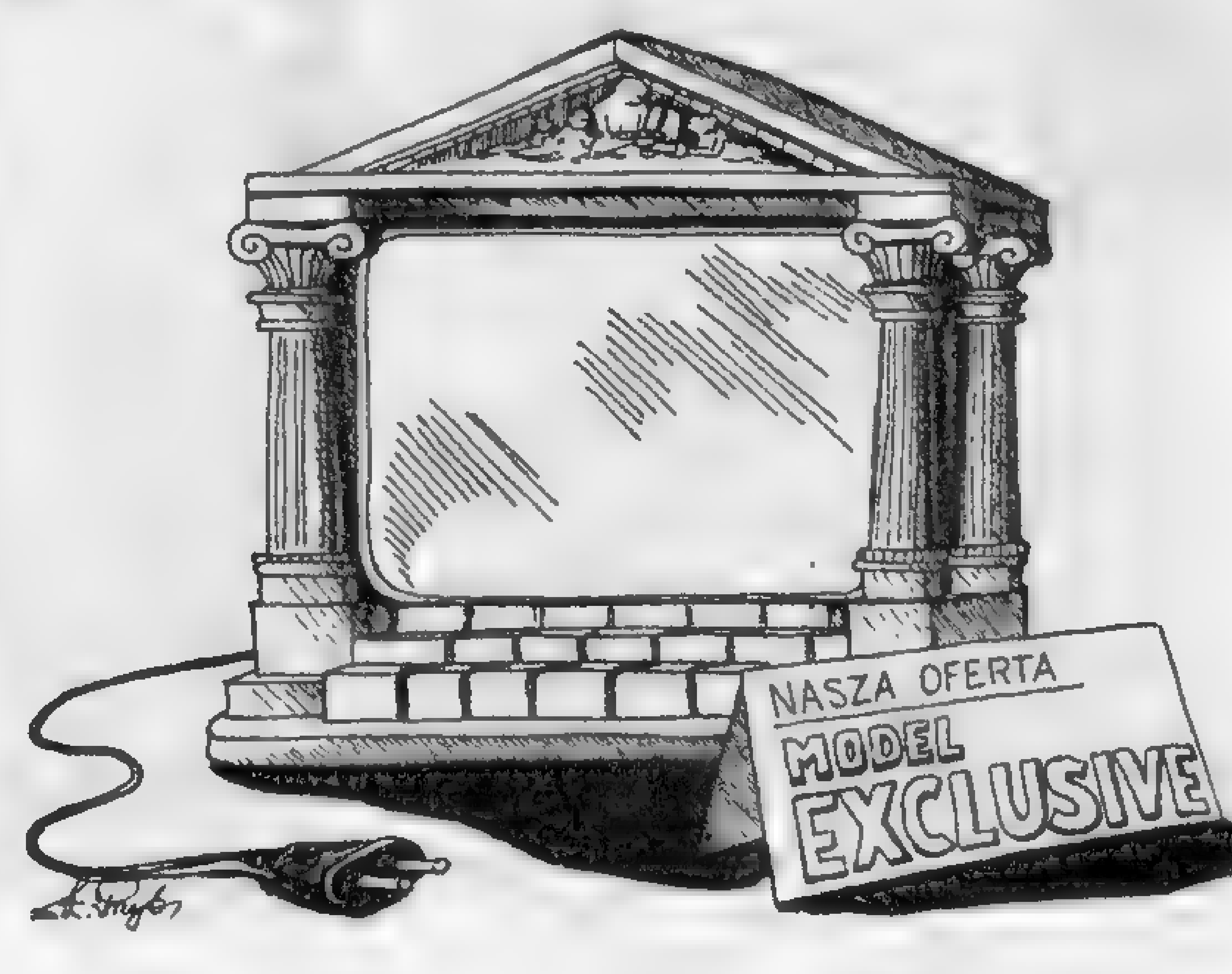
— jasność, zależy od otaczającego oświetlenia i powinna wynosić 70—240 cd/m^2,

— kontrast, optimum 94%,

— częstotliwość drgań obrazu (jego kreślenia) powinna być jak największa. Stosowana w standardach światowych jest zbyt niska i najczęściej bywa przyczyną kłopotów zdrowotnych (bólu głowy, oczu).

Stąd też wiele firm poświęca wiele wysiłku na doskonalenia monitorów komputerowych, które zastępują w technice wideotekstowej typowe odbiorniki telewizyjne. Coraz częściej rozdzielczość monitorów wynosi około 1000x1500 pikseli, a częstotliwość jest większa od 60 Hz. Wydaje się, że już w krótkim czasie pokonane zostaną bariery korzystania z „elektronicznych gazet". Monitory będą gwarantowały obraz podobny do obrazu tradycyjnej gazety, a koszt eksploatacji zostanie wkalkulowany w opłatę telefoniczną. Okaże się wówczas, że słowa wideotekst, teletekst, staną się nieaktualne. Zastąpią je po prostu komputery. Uzupełnione o syntezery mowy ludzkiej przypomną nam czasy, gdy pojawiło się radio...

Włodzimierz GOGOŁEK





System rozliczeń finansowych PRO-LOHN

Na rynku zachodnioniemieckim pojawił się mikrokomputerowy system księgowania PRO-LOHN opracowany przez firmę Victor Technologies. W podstawowej konfiguracji składa się on z mikroprocesora, klawiatury, zawierającej dodatkowo siedem klawiszy funkcyjnych, stacji dysków, monitora i drukarki. Oprogramowanie tego systemu umożliwia uzyskanie bieżących, statystycznych zestawień różnego rodzaju wynagrodzeń (za godziny pracy, nadgodziny, za skrócony czas pracy, pracę akordową itp.) i potrąceń finansowych pracowników (podatki, składki, przelewy, oszczędności itp.), tworzenie kwartalnych i rocznych sprawozdań rozliczeń finansowych (prowadzenie księgi rozliczeń finansowych), jak również generowanie w ustalonej formie przekazów i czeków dla urzędów finansowych, ubezpieczalni, banków itp.

W fazie osadzania systemów wszystkie wymagane dane stałe o pracownikach muszą zostać oczywiście wprowadzone do systemu. Automatycznie natomiast dokonywane jest obliczanie odpowiednich procentów potrąceń, zgodnie z obowiązującymi w danej części RFN przepisami. Praca użytkownika z systemem odbywa się w sposób konwersacyjny, za pomocą klawiszy funkcyjnych. Ich znaczenie jest każdorazowo wyświetlane na ekranie monitora. Istnieje możliwość wykorzystania tego systemu dla sieci zakładów produkcyjnych, jak również włączenia go do komputerowego systemu komunikacyjnego poczty.

CS











To, co udało się podpatrzeć naszemu fotoreporterowi, podziwiać można na zdjęciach. „Home, Office, Personal Computer '86" — tak zatytułowana była wystawa sprzętu informatycznego, zorganizowana w dniach 8—10 kwietnia 1986 roku, przez „Agpol", w hotelu „Victoria". Ze znanych marek swoje wyroby prezentowały między innymi: „IBM", „Olivetti", „Hewlett-Packard" oraz „Commodore". Obecne były również firmy polonijne („Impol", „Polbrit", „Apina") oraz niektóre tylko przedsiębiorstwa państwowe („Mera-Błonie", „Zakład Informatyki Przemysłu Okrętowego", „Mikrotech").

Oferta obejmowała zarówno złożone systemy komputerowe do sterowania całymi liniami technologicznymi, jak i również sprzęt dla indywidualnego odbiorcy.

Wystawa cieszyła się ogromnym powodzeniem. Nic dziwnego — tak bogatej ekspozycji jeszcze u nas nie prezentowano.









Foto: Jan Zelman

GRAFIKA KOMPUTEROWA

Jerzy Chmurzyński

Rozwój komputerów przebiega w dwóch zasadniczych kierunkach. Pierwszy z nich polega na poszukiwaniu rozwiązań zapewniających coraz większą moc obliczeniową. Chodzi tu przede wszystkim o wzrost szybkości pracy procesorów, pamięci operacyjnej i kanałów transmisji danych. W tym kierunku badań rozwiązywane są problemy łączenia komputerów w struktury wieloprocesorowe lub wielokomputerowe, problemy wieloprocesowych systemów operacyjnych oraz współbieżnej organizacji obliczeń.

Drugi kierunek rozwoju komputerów polega na doskonaleniu procesu komunikowania się człowieka z komputerem, dostosowywaniu urządzeń zewnętrznych komputerów do potrzeb i możliwości człowieka. Czynnikiem stymulującym rozwój w tym kierunku jest ograniczony czas pozostawiony człowiekowi na analizę dużej ilości dynamicznie zmieniającej się informacji. Przykładowo, analiza sytuacji powietrznej w rejonie lotniska nie może być prowadzona według tabulogramów dostarczanych przez dalekopisy lub drukarki. Czas wyprowadzenia tabulogramu z komputera i jego analizy przez człowieka jest na tyle długi, że sytuacja w powietrzu może się zasadniczo zmienić. W wyniku tego wydana decyzja staje się nieaktualna lub wręcz błędna. Inny przykład dotyczy projektowania wspomaganego komputerem. Wyniki obliczeń mogą być wyprowadzane w postaci tabulogramów zawierających kolumny liczb, jakże trudno jednak jest analizować takie dane. Brak jest przy tym elementu niezbędnego projektantom — rysunku. Właśnie rysunek — graficzna forma prezentacji danych, pozwala na skrócenie czasu analizy danych generowanych przez komputer. Graficzna prezentacja sytuacji powietrznej okazuje się najbardziej właściwą formą przedstawienia operatorom wyników funkcjonowania komputera. Stosowanie graficznej prezentacji danych jest konieczne w wielu innych sytuacjach.

Dane w postaci graficznej mogą być zarówno wyprowadzane z komputerów, jak i wprowadzane. Odpowiednio do kierunku przesyłania danych graficznych, istnieją specjalizowane urządzenia zewnętrzne komputerów. Do wprowadzania danych graficznych służą czytniki rysunków, zaś wyprowadzanie danych graficznych zapewnia pisak cyfrowy. Zarówno czytnik rysunków, jak i pisak cyfrowy należą do stosunkowo wolnych urządzeń elektromechanicznych, a to nie pozwala odwzorować szybkich zmian danych. Właściwością taką charakteryzują się grafoskopy, wyposażone w monitory ekranowe z lampą elektronopromieniową.

Grafoskopy są to urządzenia zewnętrzne komputerów, przystosowane do zobrazowania rysunków złożonych z punktów, odcinków linii prostej, łuków itp. oraz do zobrazowania tekstów alfanumerycznych. Grafoskopy wyposażone są w szereg urządzeń operatorskich, takich jak: pióro świetlne, manipulatory kulowe i drążkowe, klawiatury alfanumeryczne i funkcyjne. Dzięki tym urządzeniom operator może wprowadzać do komputera zarówno dane alfanumeryczne, jak i graficzne. Dane graficzne, wymieniane za pośrednictwem grafoskopów, pozwalają odwzorować dynamikę zmian tych danych w czasie rzeczywistym. Dzięki temu grafoskopy umożliwiają organizowanie interakcyjnej współpracy człowieka z komputerem, zapewniając wymianę danych w postaci graficznej. Ta forma współpracy człowieka z komputerem zapewnia wysoką efektywność wykorzystania komputerów.

Grafika komputerowa jest dziedziną techniki należącą do informatyki i obejmuje problemy budowy i oprogramowania urządzeń komputerowych, służących do przesyłania i przetwarzania danych w postaci graficznej.

Metody kreślenia obrazów

Istnieją dwie podstawowe metody kreślenia obrazów na ekranie lampy elektronopromieniowej. Pierwsza metoda polega na swobodnym przemieszczaniu strumienia elektronów i nosi nazwę kreślenia konturowego. Na rys. 1a przedstawiono ekran lampy z odcinkiem AB wykreślonym metodą kreślenia konturowego. Wykreślenie odcinka AB polega na cyklicznym wykonywaniu następujących operacji:

1/ pozycjonowanie układu odchylania strumienia elektronów w punkcie A;
2/ odblokowanie strumienia elektronów;
3/ sterowanie za pomocą układów odchylania strumienia elektronów po linii prostej do punktu B;
4/wygaszenie strumienia elektronów.

Częstotliwość powtarzania przedstawionych powyżej operacji zależy od czasu poświaty luminoforu: im czas poświaty jest krótszy, tym częściej należy wyświetlać obraz na ekranie, aby był widoczny dla obserwatora jako obraz stały.

Druga, powszechnie stosowana metoda kreślenia obrazów, nazywana jest metodą kreślenia rastrowego z wybieraniem liniowym (rys. 1b). Strumień elektronów przebiega w tym wypadku w sposób zdeterminowany, całe pole ekranu, niezależnie od postaci wyświetlanego obrazu. Sam obraz powstaje wskutek sterowania jasności plamki świetlnej na ekranie, przy czym sterowanie to jest współbieżne z chwilowym położeniem strumienia na ekranie. Na tej samej zasadzie powstaje obraz na ekranie lampy kineskopowej odbiornika telewizyjnego. Odcinek AB (rys. 1b) kreślony metodą rastrową z wybieraniem liniowym złożony jest zatem z punktów A, 1, 2, 3, 4, B, wyświetlanych poprzez zwiększenie jasności plamki

świetlnej podczas przemieszczania się strumienia elektronów przez linie wyblerania b, c, d, e, f, g.

Obrazy kolorowe

Obrazy kolorowe uzyskuje się, stosując w monitorach ekranowych lampy chromatyczne. Powszechne zastosowanie mają lampy z luminoforem mozaikowym, stosowane również w odbiornikach telewizji kolorowej.

Kineskop maskowy z luminoforem mozaikowym ma trzy niezależne działa ekranowe, wytwarzające na wspólnym ekranie trzy obrazy w podstawowych kolorach, tj. czerwonym (R), zielonym (G) i niebieskim (B). Każdy obraz powstaje w wyniku pobudzenia przez odpowiednią wiązkę elektronów punktowo rozłożonego luminofora, przy czym wszystkie trzy wiązki są jednocześnie odchylane za pomocą jednego układu odchylającego. Przy małych wymiarach poszczególnych punktów świetlnych następuje w oku obserwatora synteza przestrzenna odbieranych zjawisk świetlnych, wskutek czego oprócz kolorów podstawowych obserwować można inne kolory.

Wprowadzenie kolorów do obrazów wyświetlanych na monitorach ekranowych wydatnie zwiększa czytelność obrazu, a dzięki temu można zwiększyć liczbę jednocześnie zobrazowanych danych.

Wymiar rastrowy ekranu

W celu oznaczenia punktów na powierzchni ekranu jest on dzielony umowną kwadratową siatką linii. W ten sposób uzyskuje się możliwość adresowania punktów znajdujących się w punktach przecięć siatki. Współrzędne X, Y punktu są to liczby określające odpowiednio numer linii pionowej i linii poziomej, licząc od ustalonego punktu początkowego o współrzędnych 0, 0. Początek układu współrzędnych w typowych rozwiązaniach znajduje się w lewym dolnym lub w lewym górnym rogu ekranu.

Odległość pomiędzy dwiema sąsiednimi liniami kwadratowej siatki ekranu nazywana jest rastrem. Odległość ta jest wprost proporcjonalna do średnicy ekranu i odwrotnie proporcjonalna do liczby linii pionowych lub poziomych siatki kwadratowej.

Wymiar rastrowy ekranu jest to para liczb oznaczających ilość linii poziomych i pionowych siatki ekranu, np. 1024 x 1024. Rastrowy wymiar ekranu ulega zazwyczaj zmianie przy stosowaniu kolorów. W miarę zwiększenia liczby kolorów wymiar rastrowy maleje. Przykład zmian wymiaru rastrowego przedstawiono poniżej:

LICZBA KOLORÓW	WYMIAR RASTROWY
2	640 x256
4	320 x 256
16	160 x 256

Zmniejszenie wymiaru rastrowego ekranu związane jest ze zwiększeniem pamięci potrzebnej do zapamiętania kodów stosowanych kolorów.

Ważnym parametrem monitora ekranowego jest rozdzielczość. Rozdzielczość charakteryzuje zdolność zobrazowania na ekranie drobnych szczegółów obrazu, które można jeszcze rozróżnić. Rozdzielczość wyznacza się przez pomiar plamki świetlnej, uzyskanej na ekranie za pomocą strumienia elektronów lub też przez określenie liczby rozróżnialnych punktów wyświetlanych na długości jednego centymetra. Nie należy mylić rastrowego wymiaru ekranu, czyli liczby adresowalnych punktów, z rozdzielczością ekranu. Zwykle rozdzielczość ekranu jest większa niż jeden raster, co oznacza, że na odcinku o długości jednego rastra można zobrazować kilka rozróżnialnych punktów świetlnych.

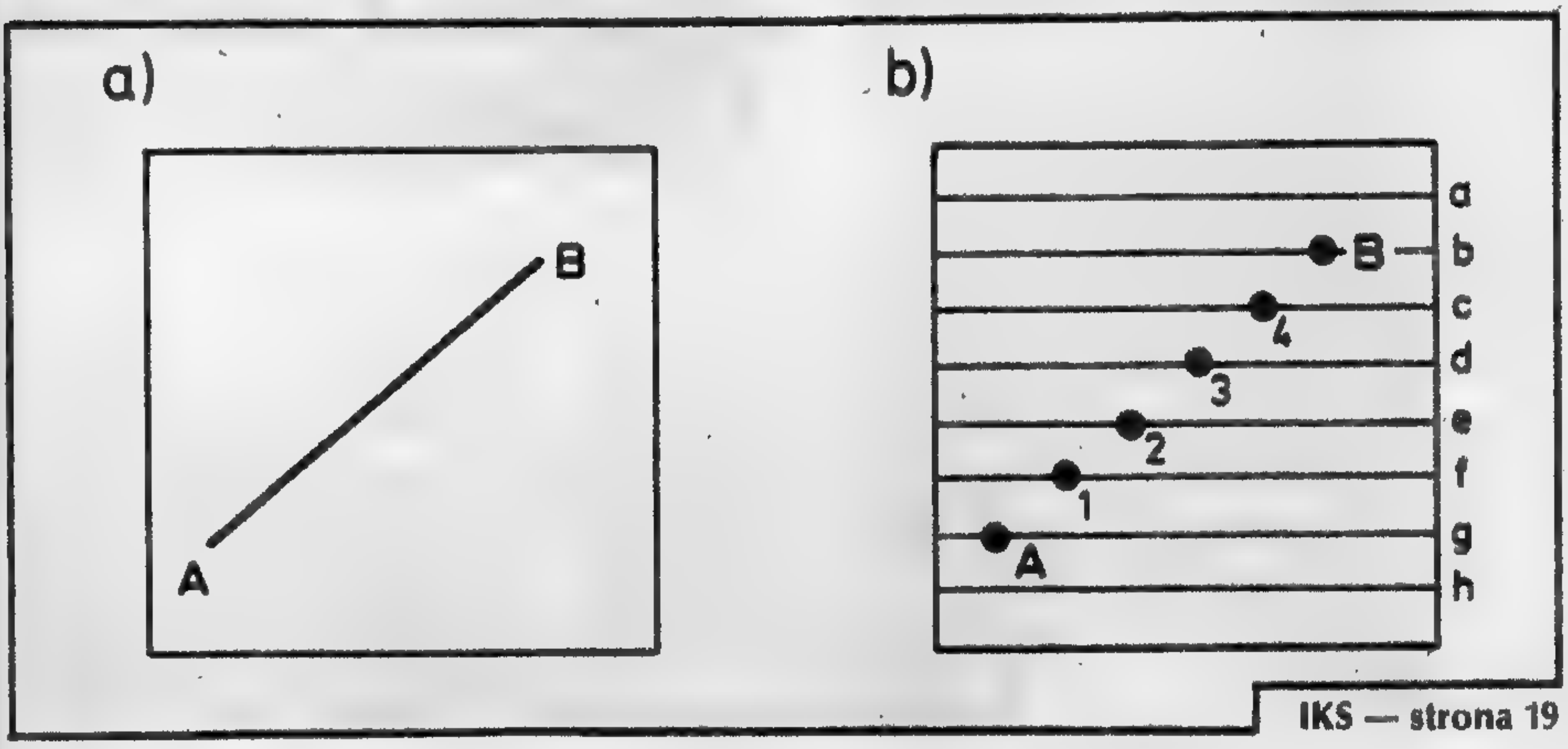
Budowa grafoskopu

Grafoskop złożony jest z następujących elementów składowych (rys. 2):

a/ monitora graficznego, w którego skład wchodzi lampa elektronopromieniowa z układami odchylania poziomego i pionowego oraz układ podświetlania,
b/ procesora graficznego, zwanego też sterownikiem graficznym,
c/ urządzeń operatorskich, do których należą:
- pióro świetlne,
- manipulator kulowy lub drążkowy,
- klawiatura funkcyjna,
- klawiatura alfanumeryczna.

dokończenie na stronie 30

rys. 1



SYMULATORY LOTNICZE

BARDZO trudno o dokładniejszą definicję pojęcia symulacji, pojęcia, które w ostatnich latach zrobiło w lotnictwie — zarówno wojskowym, jak i cywilnym — karierę, wprost oszałamiającą. Ogólnie rzecz biorąc nastąpił przełom w metodzie szkolenia i treningu pilotów. Okazało się, że warunki rzeczywistego lotu samolotu można, prawie z wszystkimi jego cechami i szczegółami, symulować na ziemi — w specjalnie do tego celu skonstruowanych symulatorach. Rzecz może wydać się niewtajemniczonym banalna, niemniej faktem jest, że powstanie tych urządzeń stało się możliwe dopiero po zastosowaniu w nich komputerów.



Pomysł zbudowania naziemnego urządzenia do szkolenia pilotów nie jest pomysłem nowym i powstał w latach dwudziestych, kiedy to Amerykanin o nazwisku **Link** zaprezentował takie właśnie urządzenia. Nazywano je wówczas trenażerem i bez istotniejszych ulepszeń przetrwało dość długo. Jeszcze piloci z okresu II wojny światowej szkolili się na urządzeniach bardzo mało przypominających współczesne systemy symulacji lotu sterowane komputerowo.

Prawdziwa pogoń za możliwie największą wiernością odtwarzania warunków rzeczywistego lotu rozpoczęła się w latach pięćdziesiątych — wraz z pojawieniem się niezbędnego do tego celu środka: specjalizowanego komputera analogowego. Nadal jednak nie dawały się przezwyciężyć niektóre trudności, mające dla jakości symulacji kluczowe znaczenie. Nie można było bowiem marzyć o skutecznym symulowaniu lotu bez zadowalającej precyzji odtwarzania jego dynamiki, czyli też bez wystarczającej dla wytworzenia złudzenia autentyczności jakości wizualizacji sytuacji zewnętrznej.

Powyższe kłopoty przestały stanowić nierozwiązywalny problem po wprowadzeniu komputerów cyfrowych. Sercem współczesnego symulatora lotniczego jest właśnie taki komputer. Oprócz zarządzania funkcjonowaniem symulatora jako całości odbiera on bezpośrednio dyspozycje pilota (np. ruch drążka sterowego), aby następnie — po ich przetworzeniu — przekazać je do komputerów — satelitów. To one sterują pracą poszczególnych podsystemów całego urządzenia, z których dla szkolącego się pilota największe znaczenie ma rzecz jasna zespół przyrządów pomiarowych, wskaźników i monitorów.

W celu zachowania maksymalnego złudzenia lotu prawdziwym samolotem, wszystkie przyrządy umieszczone są w dokładnej imitacji kabiny — tak jak w rzeczywistym samolocie.

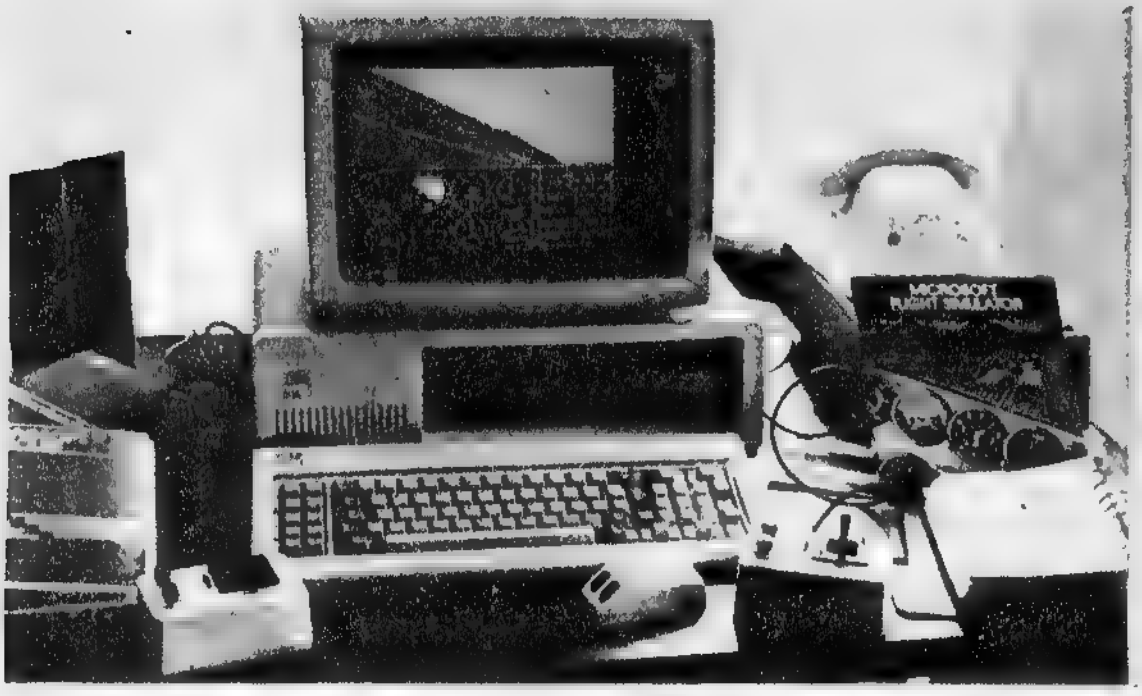
Warunkiem pojawienia się u szkolonego pilota złudzenia ciągłości „lotu" jest wykonywanie przez komputer obliczeń w zawrotnym tempie 50—60 cyklów na sekundę. Dopiero przy tej szybkości rozwiązywania przez maszynę całego szeregu równań matematycznych możliwe staje się płynne manewrowanie w „przestrzeni powietrznej". System symulacji przewiduje współdziałanie w procesie szkolenia instruktora. Może on układać dowolne scenariusze „lotu", uwzględniając na przykład złą widoczność, silny boczny wiatr, a nawet awarię jednego z systemów samolotu.

OBECNIE zakłada się, że rozwiązania konstrukcyjne współczesnych symulatorów lotniczych powinny opierać się na tzw. modułach. Pozwalają na to

Szkolący się pilot widzi na ekranie symulowany obraz sytuacji powietrznej.







Zestaw firmy IBM pracujący z oprogramowaniem Microsoft.

osiągnięcia ostatnich lat w dziedzinie mikroelektroniki. Konstrukcja modułowa umożliwia dowolne podłączanie i odłączanie od symulatora różnych urządzeń peryferyjnych. Każde z nich imituje inny system wyposażenia samolotu. Ostatnio też pojawiła się koncepcja, która w jeszcze większym stopniu uelastyczni system symulatora, dopuszczając oprócz modyfikacji strukturalnych także modyfikowanie oprogramowania. Powyższa koncepcja wymagać będzie zastąpienia komputera centralnego zespołem mikrokomputerów.

Zadaniem symulatora jest zapewnienie pilotowi także „kompletu" doznań fizycznych, nieodłącznie towarzyszących w czasie prawdziwego lotu. Zamontowana na hydraulicznych siłownikach kabina symulatora może zmieniać swe położenie we wszystkie strony, co sprawia, że szkolony pilot odczuwa najzupełniej obiektywnie występujące przyspieszenia, przechyły, kołysanie boczne, a także turbulencje powietrza i drgania powstające przy kołowaniu na pasie startowym.

JEDNYM z najbardziej kosztownych podsystemów współczesnego symulatora lotniczego jest układ wizualizacji, zapewniający pilotowi odebranie wrażenia poruszania się w fizycznej, trójwymiarowej przestrzeni. Układ ten nabiera szczególnego znaczenia w ćwiczeniach manewrów stosowanych w walce powietrznej. Obecnie systemy wizualizacji polegają na rzutowaniu obrazu o wysokiej rozdzielczości na specjalne ekrany — kopuły umieszczone w polu widzenia kabiny pilota. Obrazy wyświetlane wewnątrz kopuł mogą dawać pilotowi tak silne złudzenie własnego ruchu, że w wielu wypadkach rezygnuje się wręcz z wyposażenia kabiny w układ imitacji ruchu, zachowując jedynie sam fortel symulacji przyśpieszeń.

Można by zapytać o przyczyny coraz częstszego wprowadzania symulatorów w procesie szkolenia załóg lotniczych. Czy może mamy do czynienia z jeszcze jedną zabawą, wymyśloną jakby przy okazji burzliwego rozwoju komputerów? Okazuje się, że powstanie i wprowadzenie symulatorów podyktował dokładny rachunek ekonomiczny. Co prawda kompletne systemy symulacji lotów są bardzo drogie, lecz same samoloty wraz z kosztami ich eksploatacji jeszcze droższe. W grę wchodzą ponadto liczne zalety szkolenia na symulatorze, które nie występują w wypadku prowadzenia treningu na samolocie. I tak symulatory pozwalają m.in. na wielokrotne kontrolowanie w identycznych warunkach zdobytych przez pilota umiejętności, co ma istotne znaczenie przy ich ocenianiu. Symulator pozwala ponadto na prowadzenie ćwiczeń zachowania się załóg w razie zaistnienia awarii bez narażania życia i sprzętu, stwarza możliwość szkolenia w dowolnie przyjętych warunkach taktycznych i pogodowych.

TYMCZASEM perspektywy postępu technologicznego w budowie samych symulatorów są obiecujące. Oprócz przemian wynikających z postępu technicznego, zwiększa się coraz bardziej zakres zadań, jakie stawia się przed symulatorami. Według opinii fachowców, należy w związku z tym spodziewać się ich „specjalizowania", dzielącego je na grupy urządzeń wzajemnie uzupełniających się pod względem jakości wykonywanych funkcji. Przemawiają za tym także względy ekonomiczne. Z kolei inna, równolegle występująca tendencja rozwojowa zmierza do tego, by na jednym symulatorze możliwe było ćwiczenie użycia różnych systemów broni. Te i wiele innych zagadnień, których rozwiązania domagają się przyszli użytkownicy każą spodziewać się dalszego wzrostu niemałego już znaczenia symulatorów w lotnictwie, choć całkowite wyparcie przez nie tradycyjnych metod szkolenia jest mało prawdopodobne.

opr. P.P.



Szkolenie pilota w symulatorze jest tańsze i bezpieczniejsze.

PROGRAM 9

COMMODORE C-64

Jeśli masz ochotę i starcza Ci odwagi, stań do walki na ringu z wybranym losowo przez siebie przeciwnikiem. Jeśli będziesz miał szczęście w losowaniu przeciwnika, będziesz walczył trzy rundy, o ile, oczywiście, nie powali cię na deski morderczy nokaut — pozostałe informacje w komunikatach programu tym razem przeznaczonego na Commodore-64.

```
20 REM ***BOKS***
21 POKE53280,12:POKE53281,9
22 S1=0:S2=0:V=53248
23 POKEV,0:POKEV+1,0:POKEV+2,0:POKEV+3,0
24 GOSUB7000:POKE53280,2:POKE53281,2
25 GOSUB2500
26 RO=1:PRINTCHR$(8);:POKEV+21,3
30 GOSUB9500
62 PRINT"SQQQQQQQQQQ"TAB(33)RO
63 POKESO+15,37:POKESO+4,21
70 X=32:Y=60:TM=3:TS=1
71 F=224:G=220:E=0
72 POKE2040,195:POKE2041,192:POKESO+4,20
100 POKEV,X:POKEV+1,Y:POKEV+2,F:POKEV+3,
G
130 TS=TS-1:IFTS=0THENTM=TM-1:TS=60:IFTM
=-1THEN8500
150 PRINT"SQQQQQ";:PRINTTAB(31)TM;
160 IFTS>9THENPRINT" ",TS:GOTO400
170 PRINT"  ";TS
400 IFPEEK(653)=2THENGOSUB1000
410 IFPEEK(653)=1THENGOSUB1100
450 IFF<X+20THENIFRND(1)>SKTHENGOSUB2000
500 KE=PEEK(197)
505 Y=Y-8*((KE=23ANDY<220)-(KE=12ANDY>60
))
510 X=X-8*((KE=44ANDX<224)-(KE=47ANDX>32
))
550 F=F+8*INT(RND(1)*2.5)*-((F<X+4)-(F>X
+4))
552 IFF<X+6THENF=X+6
560 G=G+8*INT(RND(1)*2.5)*-((G<Y)-(G>Y))
563 IFG=YTHENG=Y+8*(INT(RND(1)*3)-1)
565 IFG>220THENG=220
570 IFE=0THENIFX>210THENE=1
580 IFE>0THENE=E+(X<100)+1:IFE=10THENGOS
UB3000
590 GOTO100
990 REM ***UDERZENIA***
1000 POKE2040,197:POKESO+8,10
1020 IFY<G+12ANDY>G+4THENIFX>F-12ANDX<F+
4THENS1=S1+1:GOSUB1200:POKESO+11,33
1030 IFS1=K2THEN6000
1040 FORT=1TO40:NEXT
1050 POKE2040,195:POKESO+11,0
1090 RETURN
1100 POKE2040,196:POKESO+8,10
1120 IFY<G-4ANDY>G-12ANDX>F-12ANDX<F+4TH
ENS1=S1+1:GOSUB1200:POKESO+11,33
1130 IFS1=K2THEN6000
1140 FORT=1TO40:NEXT
1150 POKE2040,195:POKESO+11,0
1190 RETURN
1200 PRINT"SQQQQQQQQQQQQQQQQQQQ"TAB(33)S1
1210 RETURN
2000 IFG>YTHEN2100
2010 POKE2041,194:POKESO+8,12
2020 IFG>Y-12ANDG<Y-4THENIFF>XANDF<X+8TH
ENS2=S2+1:GOSUB2200:POKESO+11,33
2035 IFS2=K1THEN6500
2040 FORT=1TO40:NEXT
2050 POKE2041,193:POKESO+8,12
2100 POKE2041,193:POKESO+8,12
2130 IFG>Y+4ANDG<Y+12THENIFF>XANDF<X+8TH
ENS2=S2+1:GOSUB2200:POKESO+11,33
2135 IFS2=K1THEN6500
2140 FORT=1TO40:NEXT
2150 POKE2041,192:POKESO+11,0
2190 RETURN
2200 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQ"TAB(33
)S2
2210 RETURN
2500 REM ***DZWIEKI***
2505 SO=54272
2506 FORT=0TO28:POKESO+T,0:NEXT
2510 POKESO+1,146:POKESO+5,7:POKESO+6,12
3:POKESO+15,19:POKESO+24,15
2520 POKESO+8,10:POKESO+12,9:POKESO+13,1
23
2550 RETURN
3000 REM ***ODRZUCENIE OD LINY***
```

```
3010 FORT=200TO150STEP-1 POKEV,T NEXT
3040 E=0:X=T:RETURN
6000 X=X-10:F=F+10 POKEV+2,F:POKE2041,19
6:POKEV,X:POKESO+11,0
6010 POKESO+15,19 FORT=1TO10 POKESO+4,21
 FORDL=1TO100 NEXT POKESO+4,20
6020 FORDL=1TO100 NEXT NEXTT
6050 PRINT"SQQQQQQQQQ"TAB(7)"DOBRZE BOKSU
JESZ"
6060 POKE2040,199 FORDL 1TO5000:NEXT
6100 GOTO6625

6500 X=X-10:F=F+10:POKEV,X:POKE2040,198
POKEV+2,F POKESO+11,0
6510 POKESO+15,19:FORT=1TO10 POKESO+4,21
 FORDL=1TO100 NEXT:POKESO+4,20
6520 FORDL=1TO100 NEXT NEXTT
6550 PRINT"SQQQQQQQQQ NIE MASZ SZCZESCI
A"
6600 POKE2041,199 FORDL=1TO5000 NEXT
6625 POKEV+21,0:PRINT" ";:IFK1=S2ORS2>S1
THEN6700
6630 PRINT" DOBRZE WALCZYSZ!!!"
6640 PRINT"QUZYSKANE WYNIKI:"S1"DLA CIEB
IE" PRINTTAB(16)S2"DLA "B$(XX)
6660 GOTO6780
6700 PRINT" SLABO WALCZYLES!!!":GOTO6640

6780 PRINT"QOTO TWOJI PRZECIWNICY:Q"
6790 FORF=1TO10 PRINTTAB(7)F"...";:IFF<>
10THENPRINT".",

6795 PRINTB$(F) NEXT POKE198,0
6797 INPUT"QWYBIERZ PRZECIWNIKA (0 - KON
IEC)",XX IFXX<>0THEN6810
6800 END
6810 PRINT"QQQQQQQQQQQQBEDZIE Z TOBA WALC
ZYC "B$(XX)
6815 PRINT"QPRZYCISNIJ JAKIS KLAWISZ"
6820 GETW$ IFW$=""THEN6820
6830 S1=0 S2=0 SK=(11-XX)/12
6831 K2=50+XX+INT(RND(1)*30)
6832 K1=80-XX+INT(RND(1)*50)
6840 GOTO26
7000 RESTORE:REM ***STRONA TYTULOWA***
7005 PRINT"QQQQQQQQQQQQQ"TAB(15)" B O K
S "
7010 FORI=1TO1000 NEXT
7100 PRINT"SQQQQQQQQQQQQQQQ PODAJ SWOJE
IMIE (MAX 6 LITER)"
7150 INPUTC$:IFLEN(C$)>6THEN7100

7500 DATA"HENRY HOPELESS","ARNOLD AWFUL"
,"NIGEL NO-GOOD","PHILIP FAIR"
7502 DATA"ANDREW AVERAGE","ROGER RIGHT-H
OOK","BRIAN BRICKWALL","SIMON SUPER"
7504 DATA"EDWARD EXCELLENT","FLYNN FANTA
STIC"
7510 PRINT"QQQ"TAB(15)" B O K S "
7520 PRINT"Q CHCIALBYS ZOSTAC MISTRZEM,
NIEPRAWDAZ?"
7522 PRINT"Q JESLI TAK, TO TO JEST GRA D
LA CIEBIE."
7524 PRINT"Q OTO TWOI PRZECIWNICY:Q"
7530 FORF=1TO10:READA$:PRINTTAB(7)F"..."
;:IFF<>10THENPRINT".",
7535 PRINTA$:B$(F)=A$ NEXT
7540 INPUT"Q KTOREGO WYBIERASZ";XX:IFXX<
1ORXX>10THENPRINT"QQ" GOTO7540
7545 SK=(11-XX)/12
7546 K2=50+XX+INT(RND(1)*30)

7547 K1=80-XX+INT(RND(1)*50)
7550 B$=LEFT$(B$(XX),6)
7560 PRINT"Q BARDZO DOBRZE, TWOIM PRZE
CIWNIKIEM JEST" PRINT" "B$(XX)
7562 PRINT"Q PORUSZASZ SIE PO RINGU PRZY
CISKAJAC":PRINT"Q KLAWISZE:"
7563 PRINTTAB(12)"X   - DOL"
7564 PRINTTAB(12)"Z   - GORA"
7565 PRINTTAB(12)"<   - W LEWO"
7566 PRINTTAB(12)">   - W PRAWO"
7567 PRINTTAB(11)"CBM   - LEWA REKA"
7568 PRINTTAB(10)"SHIFT   PRAWA REKA"

7570 PRINT"Q MOZLIWE JEST ZAKONCZENIE W
ALKI PRZED" PRINT" CZASEM"
7575 GOSUB9000
7580 PRINT"Q PRZYCISNIJ JAKIS KLAWISZ,
JEZELI JESTES" PRINT" GOTOW"
7590 GETX$ IFX$=""THEN7590

7595 RETURN
8500 REM ***KONIEC RUNDY***
8505 PRINT"SQQQQQQ"TAB(31)" 0   0 "
8510 POKESO+4,21
8520 POKE2040,192 POKE2041,195
8530 FORT=1TO500 NEXT POKESO+4,20
8540 X=X+4*(X>32) Y=Y+4*(Y>60) F=F-4*(F<
234):G=G-4*(G<220)
8550 POKEV,X POKEV+1,Y POKEV+2,F POKEV+3
,G
8560 POKE2040,192 POKE2041,195
8570 IFX<=32ANDY<=60ANDF>=234ANDG>=220TH
EN8580
8575 GOTO8540
8580 RO=RO+1 IFRO=4THENRO=3 GOTO6625
8585 POKESO+15,19 FORT 1TO10 POKESO+4,21

8590 PRINT"SQQQQQQQQQQQ"TAB(7)" NASTEPNA
RUNDA "

8595 FORDL=1TO200 NEXT POKESO+4,20
8600 PRINT"SQQQQQQQQQQQ"TAB(7)"
 "
8605 FORDL=1TO200:NEXT
8610 NEXTT
8620 GOTO62
9000 REM ***CZYTANIE SPRITE***
9010 FORT=0TO511:READA POKE192*64+T,A:NE
XTT
9100 POKE2040,192 POKE2041,195:POKEV+39,
0:POKEV+40,7
9400 RETURN
9500 REM ***RYSOWANIE RINGU***
9510 AA$="%":BB$=AA$
9520 FORT=0TO27:AA$=AA$+"%":BB$=BB$+" "
NEXT
9530 AA$=AA$+"%" BB$=BB$+"%"
9540 PRINT" "AA$ FORT=0TO22 PRINTBB$ NEX
T:PRINTAA$;

9600 PRINT"S"SPC(153)"CZAS"SPC(195)"RUND
A"SPC(195)"PUNKTY "
9605 PRINT"QQ"TAB(32)LEFT$(C$,6)
9610 PRINT"QQQ"TAB(32)LEFT$(B$(XX),6):RE
TURN
10000 REM ***SPRITES***
10001 DATA0,56,0,0,127,252,0,127,254,0,1
27,2
10002 DATA0,127,2,0,56,63,0,0,127,0,0,12
7
10003 DATA0,0,99,0,0,65,0,0,0,0,0,0
10004 DATA0,0,65,0,0,99,0,0,127,0,56,127

10005 DATA0,127,63,0,127,2,0,127,254,0,1
27,252
10006 DATA0,56,0,0
10007 DATA112,0,0,255,255,252,255,255,25
4,254,0,2
10008 DATA254,0,10,112,0,62,0,0,127,0,0,
99

10009 DATA0,0,65,0,0,65,0,0,1,0,0,65
10010 DATA0,0,65,0,0,99,0,0,126,0,56,62
10011 DATA0,127,10,0,127,2,0,127,254,0,1
27,252
10012 DATA0,56,0,0
10013 DATA0,56,0,0,127,252,0,127,254,0,1
27,2
10014 DATA0,127,10,0,56,62,0,0,127,0,0,9
9
10015 DATA0,0,65,0,0,65,0,0,1,0,0,65
10016 DATA0,0,65,0,0,99,0,0,127,112,0,62

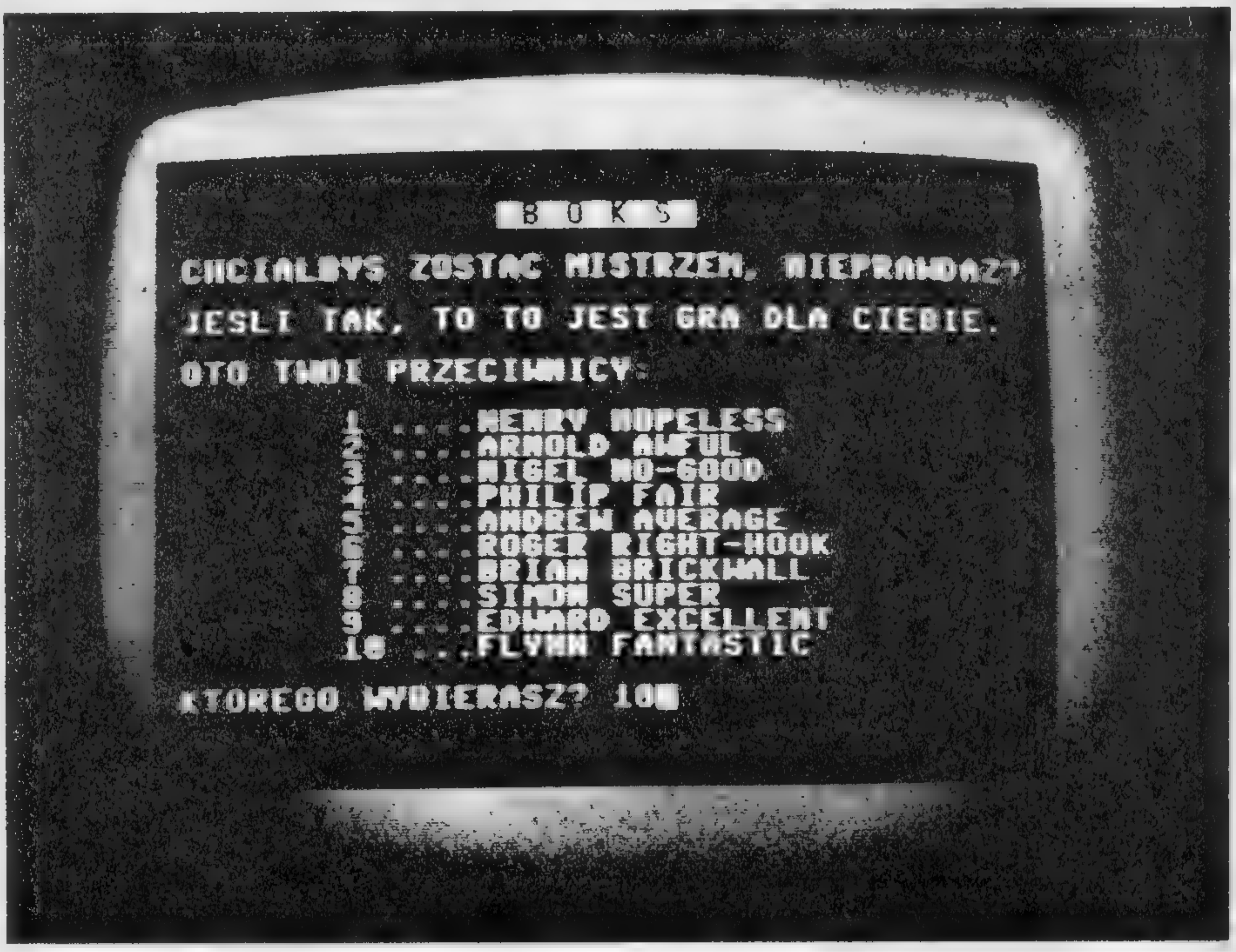
10017 DATA254,0,10,254,0,2,255,255,254,2
55,255,252
10018 DATA112,0,0,0
10019 DATA0,56,0,63,252,0,127,252,0,65,2
52,0
10020 DATA73,252,0,124,56,0,126,0,0,198,
0,0

10021 DATA131,0,0,131,0,0,129,0,0,131,0,
0
10022 DATA171,0,0,198,0,0,254,0,0,124,56
0
10023 DATA73,252,0,65,252,0,127,252,0,63
252,0
10024 DATA0,56,0,0
10025 DATA0,28,0,63,254,0,127,254,0,64,2
54,0
10026 DATA72,126,0,124,28,0,126,0,0,198,
0,0
10027 DATA131,0,0,131,0,0,129,0,0,131,0,
0
10028 DATA131,0,0,198,0,0,126,0,0,124,0,
14
10029 DATA72,0,63,64,0,127,123,255,255,6
3,255,255
10030 DATA0,0,14,0
10031 DATA0,0,14,63,255,255,127,255,255,
64,0,127

10032 DATA72,0,63,124,0,14,126,0,0,198,0
0
10033 DATA131,0,0,131,0,0,129,0,0,131,0,
0
10034 DATA131,0,0,198,0,0,126,0,0,124,28
0
10035 DATA72,126,0,64,254,0,127,254,0,63
254,0
10036 DATA0,28,0,0
10037 DATA0,0,0,0,0,24,0,0,24,0,48,0
10038 DATA1,241,240,2,49,0,2,1,0,2,1,0
10039 DATA115,253,0,251,255,0,255,254,0,
```

```
255,254,0
10040 DATA251,255,0,115,252,128,2,0,64,2
 0,126
10041 DATA98,0,2,124,0,6,96,0,6,0,0,0
10042 DATA0,0,0,0
10043 DATA112,0,224,112,0,224,32,240,64
33,248,64

10044 DATA35,168,64,35,168,64,35,252,64,
19,12,128
10045 DATA9,249,0,4,242,0,3,252,0,3,252,
0
10046 DATA3,252,0,1,248,0,0,240,0,1,152,
0
10047 DATA1,152,0,3,156,0,1,152,0,13,155
 0
10048 DATA15,159,0,0
```



Tak zaczyna się mecz.

PROGRAM 10

SPECTRUM

```
9000 REM  PAMIEC PROGRAMU
9005 REM
9010 CLS :
     LET s=PEEK 23635+PEEK 23636
         *256-1:
     GO TO 9210
9020 LET p=PEEK s
9030 IF p=13 THEN GO TO 9200
9040 IF p=14 THEN GO TO 9300
9050 IF p<10 THEN PRINT s;"      "
     ;"   ";
9060 IF p>9 AND p<100 THEN PRINT
     s;"     ";p;"    ";
9070 IF p>99 THEN PRINT s;"    "
     ;p;"    ";
9080 IF p>31 THEN PRINT PAPER 6;
     CHR$ p
9090 IF p<32 THEN PRINT
9100 LET s=s+1:
     GO TO 9020
9200 PRINT s;"     ";p;"     ";
     PAPER 4;"koniec wiersza"
9205 PRINT
9210 LET s=s+1:
     LET p=PEEK s*256+PEEK (s+1)
9220 PRINT s;"     "; PAPER 4;
     "Wiersz nr: ";p;"      "
9230 LET s=s+2:
     LET p=PEEK s+PEEK (s+1)*256
9240 PRINT s;"     "; PAPER 6;
     "Ilosc bajtow = ";p
9250 LET s=s+1:
     GO TO 9100
9300 PRINT s;"     ";p;"     ";
     PAPER 4;"Stala liczbowa:"
9310 PRINT s+1;"     ";
9320 FOR f=s+1 TO s+5
9330 PRINT PEEK f;" , ";
9340 NEXT f
9350 PRINT : LET s=s+5:
     GO TO 9100
```

Tylko dla dociekliwych! Dzięki temu programowi można zajrzeć do pamięci „Spectrum" i sprawdzić jej wykorzystanie przez dowolny inny program napisany w BASIC-u.

Najpierw należy wczytać instrukcję LOAD poniższy program, następnie instrukcję MERGE program, którego sposób zapisu w pamięci chcemy obejrzeć na ekranie. Instrukcja GOTO 9000 spowoduje wyświetlenie zawartości komórek pamięci i ich adresy, począwszy od pierwszej instrukcji (wiersza) programu. Naciśnięcie dowolnego klawisza (oprócz n) spowoduje wyświetlenie zawartości kolejnych komórek pamięci.

Wynik działania programu.

```
23912     190     PEEK
23913     115     s
23914     13      koniec wiersza

23915     Wiersz nr: 9030
23917     Ilosc bajtow = 24
23919     250     IF
23920     112     p
23921     61      =
23922     49      1
23923     51      3
23924     14      Stala liczbowa:
23925     0 , 0 , 13 , 0 , 0 ,
23930     203     THEN
23931     236     GO TO
23932     57      9
23933     50      2
23934     48      0
23935     48      0
23936     14      Stala liczbowa:
23937     0 , 0 , 240 , 35 , 0 ,
23942     13      koniec wiersza
```

1 BASIC

(Podstawy programowania)

Miłośników zabaw z mikrokomputerami, przyszłych informatyków, zachęcamy do nauki programowania. Zaczynamy od wykładów języka BASIC na mikrokomputer ZX Spectrum.

Wykład pierwszy:
— klawiatura ZX Spectrum
— ZX Spectrum jako kalkulator
— ekran telewizyjny

Zabawa z komputerem polega na wydawaniu mu rozkazów, przekazywanych najczęściej za pomocą klawiatury, które on posłusznie wykonuje. Wykład nasz rozpoczniemy od zapoznania się z klawiaturą ZX Spectrum i jej obsługą. Włączamy komputer i zaczynamy.

Klawiatura ZX Spectrum jest podobna do maszyny do pisania. Każdy klawisz ma 5 różnych oznaczeń. Oznaczenia te napisane są na, pod i nad klawiszem w różnych kolorach. Symbole używane w ZX Spectrum obejmują nie tylko pojedyncze znaki (litery, cyfry itd.), ale także symbole złożone (takie jak słowa kluczowe, nazwy funkcji itd.), które powinny być wprowadzane właściwymi klawiszami, a nie literowane. W celu realizowania wszystkich tych funkcji i rozkazów używane są odpowiednie oznaczenia wyróżnione na klawiszach, wprowadzane częściowo przez użycie shift'ów (odpowiednich klawiszy), a częściowo przez zmianę trybu pracy maszyny.

Tryb pracy oznaczony jest kursorem, jest to migająca literka wskazująca miejsce na ekranie, gdzie zostanie wpisany na ekran kolejny symbol z klawiatury. Rozróżniamy pięć trybów pracy ZX Spectrum.

Po włączeniu Spectrum, na ekranie (w dolnej linii) pojawia się napis: c 1982 Sinclair Research Ltd.

Od tego momentu Spectrum gotowy jest do przyjmowania informacji (rozkazów) i jest w trybie K (kursor K).

Tryb K (od Keywords — słowa kluczowe) — komputer oczekuje rozkazu lub linii programu, a z położenia kursora w linii wie, że powinien otrzymać numer linii lub słowo kluczowe. Sytuacja taka zachodzi na początku linii, po THEN, lub po dwukropku (o ile nie jest w tekście). Słowa kluczowe umieszczone są na klawiszach w kolorach białym i czerwonym, lub pod klawiszem w kolorze czerwonym lub nad klawiszem w kolorze zielonym, są to np. PRINT, INPUT, IF, DIM, FOR, STEP, TO, THEN itd. Jeżeli literka K — kursor K znajduje się w lewym dolnym rogu ekranu, lub w środku linii (na dole ekranu), to naciśnięty klawisz jest interpretowany jako słowo kluczowe lub liczba. Po naciśnięciu klawisza ze słowem kluczowym na ekranie zobaczymy to słowo oraz literę L. Jest to najczęściej spotykany tryb pracy — **tryb L** (kursor L). Bez shift'ów »klawisze: CAPS SHIFT (CS) oraz SYMBOL SHIFT (SS)« naciśnięty klawisz wypisze swój podstawowy symbol tzn.: cyfry lub litery (ale tylko małe).

Tryb C (od Capitals — duże litery) — kursor C otrzymujemy przez naciśnięcie jednocześnie klawiszy CAPS SHIFT i 2 (cyfra). Jest on wariantem trybu L i służy do wprowadzania dużych liter.

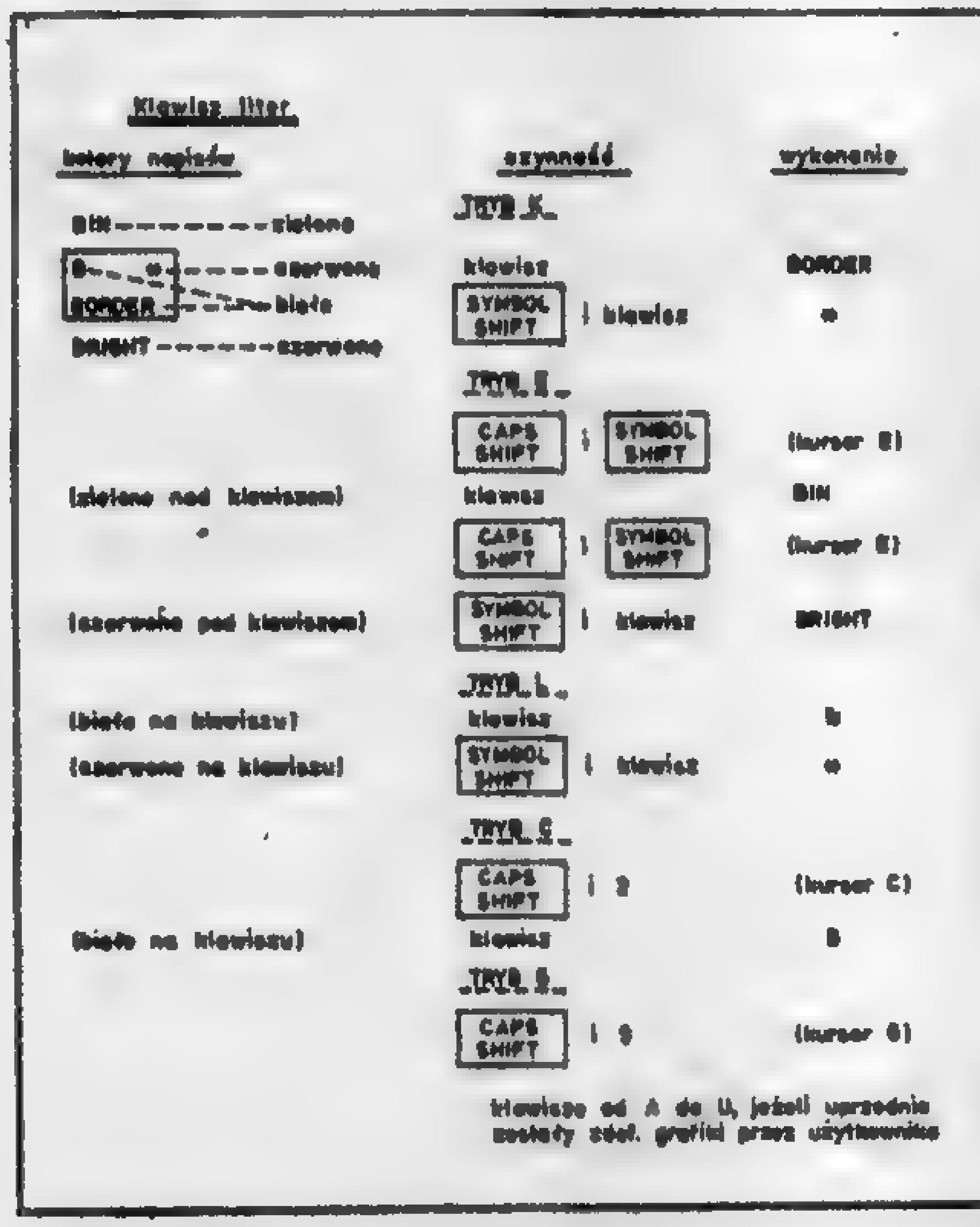
Tryb E (EXTEND MODE — rozszerzony) — kursor E otrzymujemy przez naciśnięcie jednocześnie klawiszy CAPS SHIFT i SYMBOL SHIFT. Tryb E służy do wprowadzenia symboli napisanych nad klawiszem (w kolorze zielonym), a z klawiszem SYMBOL SHIFT — do wprowadzania symboli napisanych pod klawiszem (w kolorze czerwonym).

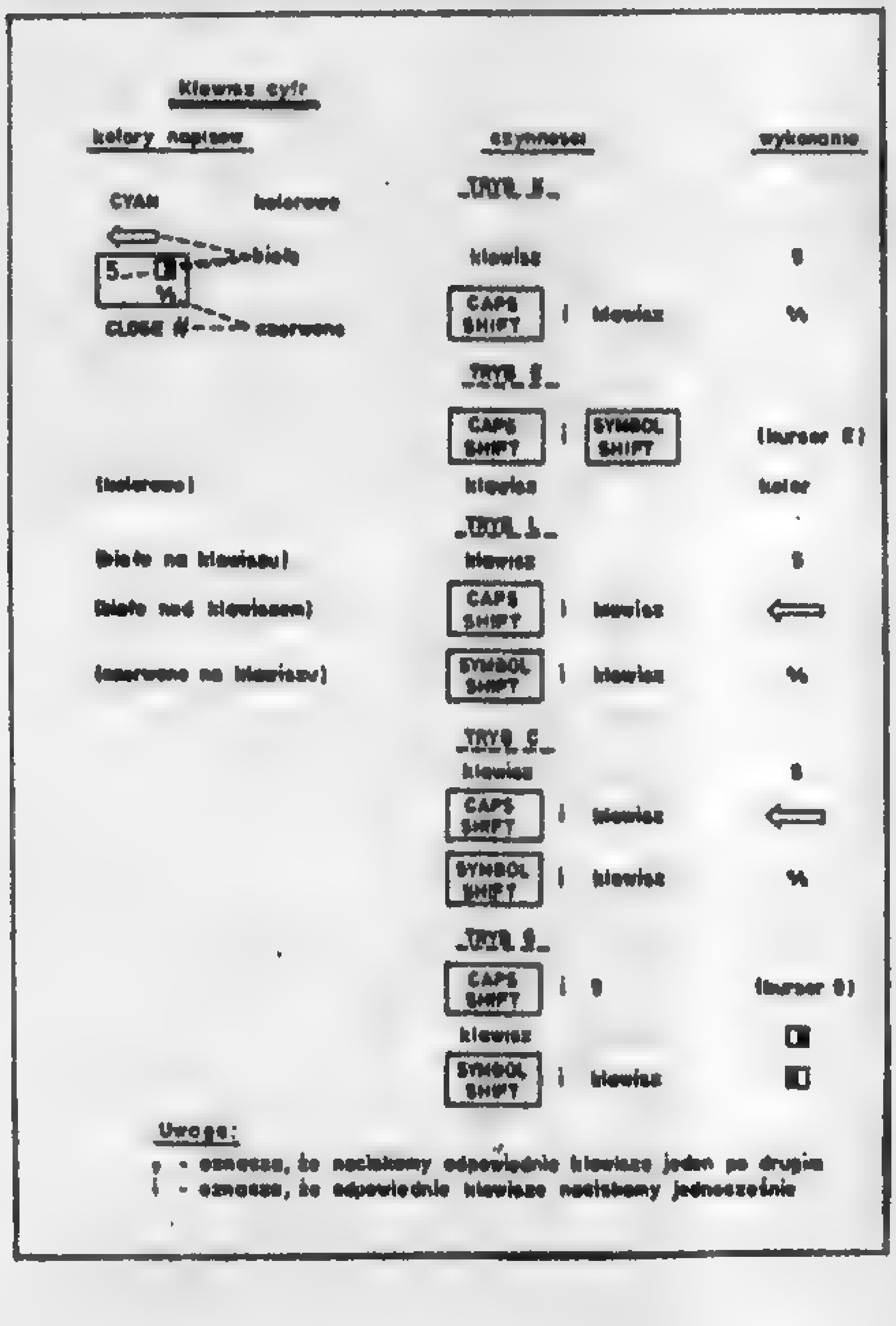
Tryb G (GRAPH — grafika) — kursor G otrzymujemy przez naciśnięcie jednocześnie klawiszy CAPS SHIFT i 9 (cyfra). Tryb G używany jest w grafice komputerowej. Naciśnięte klawisze cyfrowe (1÷8) dadzą odpowiednie symbole graficzne narysowane na kluczach.

Powrót do trybu L lub K otrzymujemy przez ponowne naciśnięcie klawiszy dla utworzenia trybów C, E, G.

Weźmy dla przykładu dwa klawisze i pokażemy, jak należy się nimi posługiwać:

rys. 1





rys. 2

Posługując się słowem kluczowym PRINT, symbolami +, –, *, /, ↑, ", ;, „ ' literami i cyframi sprawdzimy jak **ZX Spectrum „pracuje" jako kalkulator**, a tym samym nabierzemy wprawy w obsłudze klawiatury.

Słowo kluczowe PRINT (napisz) jest rozkazem dla komputera, aby coś na ekranie napisał.

Zaczynamy pracę od obliczenia wartości wyrażenia:

$$2 + 6\,(12 - \frac{14}{2})^2$$

Komputer jest przygotowany do wprowadzania tzn. jest w trybie pracy K.

Naciskamy klawisz P, na ekranie pojawi się nam słowo PRINT, po nim wprowadzamy powyższe wyrażenie, stosując podane niżej reguły:

Symbol	Klawisz	Funkcja	Przykład
+	K	dodaj dwie liczby	12+6=18
–	J	odejmij dwie liczby	18–12=6
*	B	pomnóż dwie liczby	12*4=48
/	V	podziel dwie liczby	28/14=2
↑	H	podnieś do potęgi (np. 2 do potęgi 3)	2↑3=8

Każdy z tych symboli otrzymamy, naciskając równocześnie klawisze: SYMBOL, SHIFT i odpowiedni klawisz (K lub J, lub B lub V lub H).

Nawiasy ( ) są takie same jak w matematyce, brak jest nawiasów [ ], { }itp. Jeżeli nawiasów jest więcej, to piszemy odpowiednią liczbę par nawiasów ( ).

Wprowadzajmy dalej:

PRINT 2 + 6 * (12 — 14/2)↑ 2

i wreszcie klawisz ENTER. W lewym górnym rogu ekranu otrzymamy liczbę 152. Sprawdzimy otrzymany wynik.

$$2 + 6 \cdot 5^2 = 2 + 6 \cdot 25 = 2 + 150 = 152$$

Analizując zapisany wzór zapamiętajmy następującą regułę: przed nawiasem należy postawić odpowiedni znak (tu znak*) jeżeli przed nim stoi liczba (lub litera). Popatrzmy jak by to wyglądało, gdybyśmy napisali z „błędami".

PRINT 2 + 6 (12 — 14/2) ↑ 2 i ENTER

Komputer nie „wprowadzi" naszej instrukcji i zasygnalizuje błąd: między 6 i (pokaże się migający znak zapytania: PRINT 2 + 6 ⍰ (12 — 14/2) 2

i komputer „czeka" na poprawę błędu.

Poszukajmy nad klawiszami białych znaków

a) ← b) →

Naciskając ← (CAPS SHIFT i 5) przesuwamy kursor L w lewo na miejsce znaku ? i wstawiamy znak * i ENTER, komputer zareagował poprawnie.

Jak zauważyliśmy ← służy do przesuwania w lewo danego kursora w linii, natomiast → do tego samego celu, ale w prawo.

Możemy ten błąd poprawić w inny sposób: znajdźmy DELETE (klawisze CAPS SHIFT i Ø) naciśnijmy je 9 razy: za każdym naciśnięciem tych klawiszy kursor L przesuwa się o jedno miejsce w lewo, kasując napisane uprzednio znaki; otrzymamy więc na ekranie:

PRINT 2 + 6 ⍰

w tym momencie zaczynamy wprowadzać „od nowa" znaki począwszy od *.

Zastanówmy się co się bardziej opłaca w tym wypadku używać ← (strzałka w lewo) czy komendy DELETE.

Obliczmy jeszcze wartość wyrażenia:

$$120 - 2 \cdot 90 + \frac{14}{7} \cdot 9$$

a więc napiszmy:

PRINT 120 — 2 * 90 + 14/7 * 9 i ENTER

otrzymamy wynik = —42

Przeprowadzimy teraz analizę wykonania obliczeń przez komputer:

1) 120—2*90+14/7*9
2) 120—180+14/7*9
3) 120—180+2*9
} mnożenie i dzielenie wg priorytetów od lewej do prawej

4) 120—180+18
5) —60+18
6) —42
} dodawanie i odejmowanie wg priorytetów od lewej do prawej

Podane wyżej symbole +, —, *, /, ↑ nazywają się operatorami arytmetycznymi. Działania w danym wyrażeniu arytmetycznym wykonują się od lewej do prawej strony wg ustalonych priorytetów:

potęgowanie ↑
mnożenie * i dzielenie / są równoważne
dodawanie + i odejmowanie — są równoważne

Ponadto pierwsze wykonują się działania zamknięte w największą liczbę nawiasów, wewnątrz nawiasów obowiązują podane wyżej priorytety.

Wprowadźmy do komputera następujący zapis (pamiętajmy o kursorze K):

PRINT „TO JEST PIERWSZY WYKŁAD"

W lewym górnym rogu ekranu otrzymamy TO JEST PIERWSZY WYKŁAD. Każdy ciąg znaków (dowolnych) napisany w cudzysłowie, czyli:

„ciąg znaków"

jest traktowany jako łańcuch (tekst). Jeżeli umieścimy ten łańcuch po słowie PRINT, to po wykonaniu PRINT na ekranie ukaże się nam ten ciąg znaków w miejscu, które wyznacza jego położenie w PRINT lub znaki specjalne dozwolone w PRINT jak: , (przecinek) ; (średnik), ' (apostrof).

Znaki te napisane w PRINT powodują wypisywanie na ekranie tekstów lub liczb w następujący sposób:

; (średnik) — obok siebie
, (przecinek) — w dwóch kolumnach
' (apostrof) — w jednej kolumnie (w lewej części ekranu)

Więcej na ten temat napiszemy w innych wykładach, gdy będzie omawiana instrukcja PRINT w BASIC'u.

A teraz kilka przykładów. Wprowadźmy do komputera poniższe zapisy i wykonajmy je.

1. PRINT „C=";15, „D=";2
2. PRINT „TO JEST ZX SPECTRUM", „DZIEŃ DOBRY!"
3. PRINT „APOSTROF" ' „1 KOLUMNA" ' „LEWA CZĘŚĆ EKRANU"

Po wykonaniu na ekranie otrzymamy kolejne napisy postaci:

```
C = 15                          d = 2
TO JEST ZX SPECTRUM             DZIEŃ DOBRY!
APOSTROF
1 KOLUMNA
LEWA CZĘŚĆ EKRANU
```

Zadanie

1. Obliczyć wartości wyrażeń:

$$W_1 = \frac{-2 + 3 \cdot \frac{15}{5}(2 - 3^4)}{2 - 3^3} \qquad W_2 = \frac{0{,}25 - 4\,(16 - 2^3)}{-5}$$

dokończenie na stronie 28

dokończenie ze strony 27

$$W_3 = \frac{12 \cdot 13 \cdot 15}{16 \cdot 17} \qquad W_4 = \frac{17{,}25 - 16}{-(14 - 9)}(2^3 - 4)$$

używając dwa razy słowa PRINT. Otrzymane wyniki mają mieć postać:

W1 = W2 =
W3 =
W4 =

Uwagi: zamiast przecinka piszemy kropkę (.)
obliczając wartość ułamka należy wzór występujący w liczniku zamknąć w nawiasy, tak samo w mianowniku i dopiero dzielić: (licznik)/(mianownik)

Ekran telewizyjny zawiera 24 linie po 32 znaki i jest podzielony na dwie części w poziomie i w pionie. Linie i znaki numerowane są od zera. Górna część ekranu zawiera maksimum 22 linie (0÷21) i wyświetla program albo wyniki. Dolna część ekranu jest wykorzystywana do wprowadzania rozkazów, wierszy programu i danych wejściowych, a także do wyświetlania komunikatów. Zawiera ona dwie linie (linie 22 i 23), górna jest zawsze pusta, ale powiększa się w miarę zapisywania. Gdy dolna część tekstu dojdzie do tekstu zapisywanego w części górnej ekranu, powoduje to w miarę zapełniania, stopniowe usuwanie tekstów napisanych w pierwszych wierszach ekranu poza ekran.

Znaki w każdej linii podzielone są na dwie części:
pierwsza — znaki od 0 ÷ 15
druga — znaki od 16 ÷ 31.

Każda taka część nazywana jest strefą, można także powiedzieć, że każda z nich jest kolumną.

Jak zauważyliśmy , (przecinek) w PRINT powodował pisanie danej umieszczonej przed , w pierwszej części (kolumnie), a danej umieszczonej po , w drugiej części (kolumnie), natomiast ' (apostrof) pisanie w pierwszej części (w jednej kolumnie).

Otrzymywane przez nas wyniki pozostawały na ekranie. Naciśnijmy klawisz z symbolem CLS (klawisz V) na kursorze K i ENTER. Nastąpi „oczyszczenie" ekranu, na dole ekranu pojawi się napis ØOK, Ø : 1. Zapamiętajmy CLS — służy do „oczyszczania" ekranu telewizora.

W czasie naszej pracy mogły pozostać w naszym komputerze, w jego pamięci, niepotrzebne informacje, naciśnijmy klawisz z symbolem NEW (kursor K — klawisz A) i ENTER. Nastąpi „oczyszczenie" pamięci komputera (i ekranu), a na ekranie pojawi się napis:

© 1982 Sinclair Research Ltd

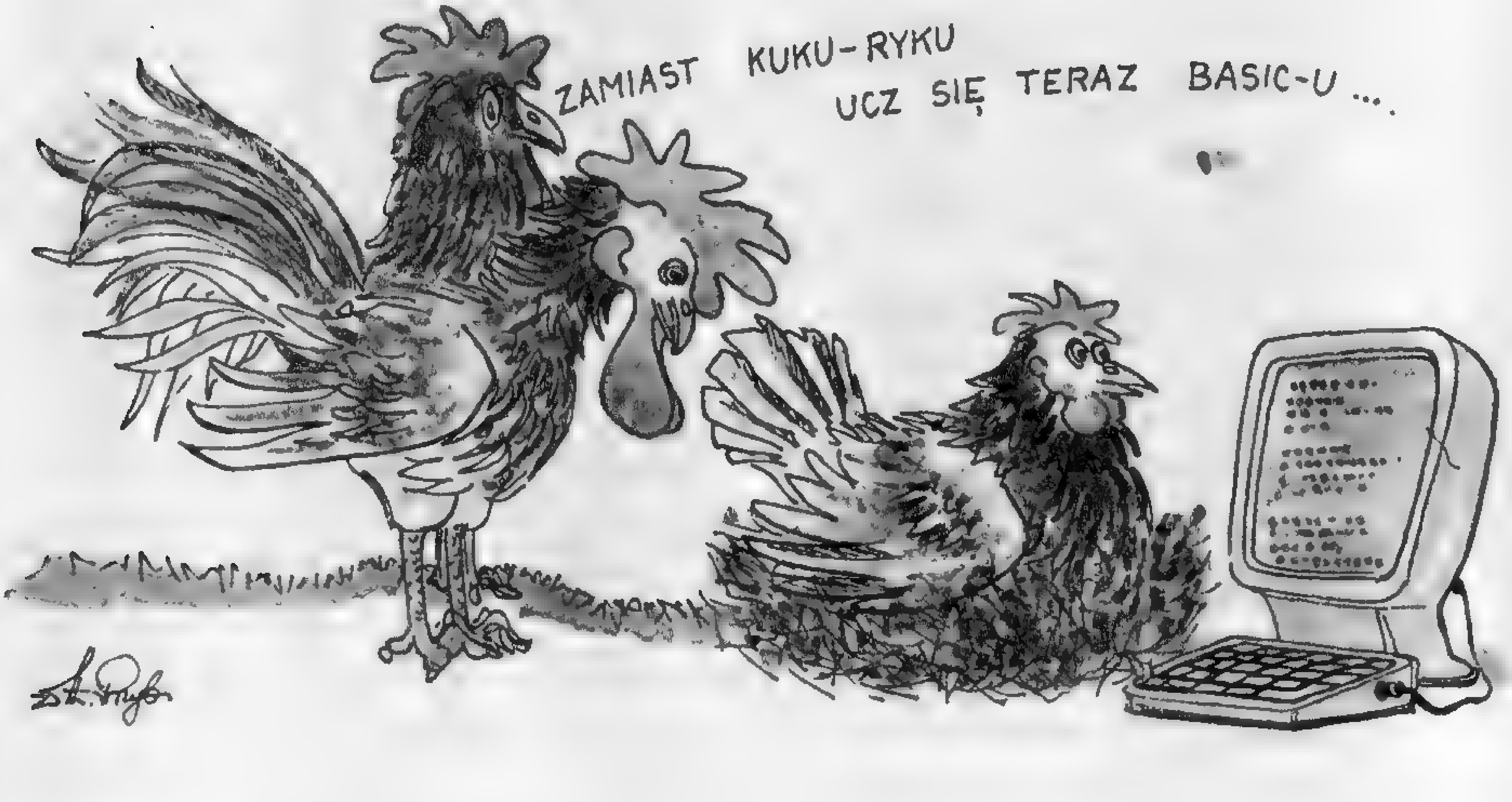
Zapamiętajmy, że przed rozpoczęciem nowej pracy należy „oczyścić" pamięć komputera i do tego służy komenda NEW. Weźmy teraz numer pierwszy „IKSa" i spróbujmy wprowadzić do komputera jeden z podanych tam programów np. Kalendarz. Pamiętajmy, że wprowadzając kolejne linie opatrzone numerami, po zakończeniu pisania każdej z nich, naciskamy ENTER.

Po wprowadzeniu ostatniej linii (linie te powinny po wykonaniu naszych czynności znajdować się w górnej części ekranu) naciśnijmy klawisz z symbolem RUN (kursor K — klawisz R) i ENTER.

Wprowadzone linie znikną, a na ekranie zaczną się pokazywać odpowiednie napisy wyprowadzane przez ten program.

Jeżeli udało się nam wprowadzić bezbłędnie ten program i wykonać go, możemy uznać, że umiemy już posługiwać się klawiaturą ZX Spectrum.

Z.W.



Przed nami piąta generacja

TRUDNO jest dziś określić, jakie będą komputery piątej generacji, bo chociaż prace nad nimi są zaawansowane, to żadna z firm nie zaprezentowała do tej pory takiego



Jaka będzie piąta generacja?

urządzenia. Pojawia się natomiast coraz więcej publikacji zapowiadających rychłe nadejście nowej generacji. Wynika z nich, że w komputerach tych będzie powszechnie stosowana sztuczna inteligencja, umożliwiająca między innymi porozumiewanie się z maszyną za pomocą języka naturalnego — takiego, jakim posługujemy się na co dzień. Znajdą się w nich również systemy ekspert, dające możliwość analizowania najbardziej skomplikowanych problemów dotyczących konkretnej dziedziny na podstawie własnej bazy programów i danych zawartych w pamięci komputera i możliwych do przechowywania na dyskach lub innych nośnikach informacji.

Jednym z głównych sponsorów badań w tej dziedzinie jest Pentagon, który w ramach programu SCI (Strategic Computing Initiative — Strategiczna Inicjatywa Obliczeniowa) przeznaczył do końca 1990 roku 600 mln dol. na rozwój najnowszych technologii komputerowych. Program ten, obejmujący między innymi mikroelektronikę, architekturę komputerową, systemy ekspert i sztuczną inteligencję, odegrać ma znaczącą rolę w rozwoju techniki komputerowej nie tylko dla celów wojskowych, ale również komercyjnych.

W programie tym przedstawiony jest następujący przykład wymagań stawianych komputerowi nowej generacji, zastosowanemu do sterowania wozem bojowym; pojazd ten musi mieć zdolność samodzielnego planowania trasy przejazdu z szybkością co najmniej 60 km/godz., wykorzystując do tego celu dane cyfrowe o terenie i otoczeniu, uwzględniając jednocześnie zmieniające się warunki pola walki. Aby spełnić te wymagania będzie wyposażony w zespół sprzężonych komputerów, przystosowanych do pracy w trudnych warunkach terenowych i bojowych, które będą sterować systemem wizyjnym i systemem ekspert dla celów nawigacji. Komputery te będą posiadały procesory równoległe, ponieważ żaden komputer wykonujący instrukcje szeregowo nie byłby w stanie spełnić narzuconych wymagań. System ekspert zastosowany do celów nawigacji będzie zawierał około 6500 podprogramów, które będą wykonywane z szybkością 7000 podprogramów na sekundę (dla porównania obecne systemy ekspert rzadko przekraczają 1000 podprogramów i mają szybkość nie większą niż 100 podprogramów na sekundę). System wizyjny ma przetwarzać dane w czasie rzeczywistym. Do tego celu będzie zastosowany komputer o możliwości wykonywania od 10 do 100 mld operacji na sekundę (dzisiejsze komputery uzyskują szybkość zaledwie kilkuset milionów operacji na sekundę). Kierujący pojazdem będzie miał do swojej dyspozycji komputer wspomagający, który odciąży go od wszelkich czynności związanych z obsługą pojazdu, tak aby cały swój wysiłek intelektualny mógł skupić na wykonywaniu zadań bojowych. System rozpoznawania mowy zapewni możliwość porozumiewania się z maszyną za pomocą komend słownych.

TAKA wizja rozbudza wyobraźnię. Komputery piątej generacji dokonają kolejnej rewolucji naukowo-technicznej, w fabrykach będą sterować superprecyzyjnymi robotami i całymi ciągami technologicznymi, w lotnictwie zastąpią pilotów, będą sterować okrętami, a prowadzenie samochodu to już drobnostka — zastąpią człowieka nie tylko tam, gdzie wymagana jest duża szybkość i precyzja działania, ale i w takich dziedzinach, w których potrzebna jest przede wszystkim ogromna wiedza i doświadczenie, jak np. medycyna, tłumaczenie z języków obcych czy finansowość.

PODSTAWOWĄ trudnością we wprowadzeniu nowej generacji jest brak elementów elektronicznych zapewniających odpowiednio dużą szybkość działania. Nadzieją są elementy wykonane z arsenku galu charakteryzujące się małym poborem energii, dużą szybkością działania i równie dużą odpornością na promieniowanie. Optymiści twierdzą, że komputery piątej generacji podbiją świat na początku lat dziewięćdziesiątych naszego stulecia. Jak będzie — zobaczymy. Na razie musimy jeszcze trochę poczekać.

W.O.



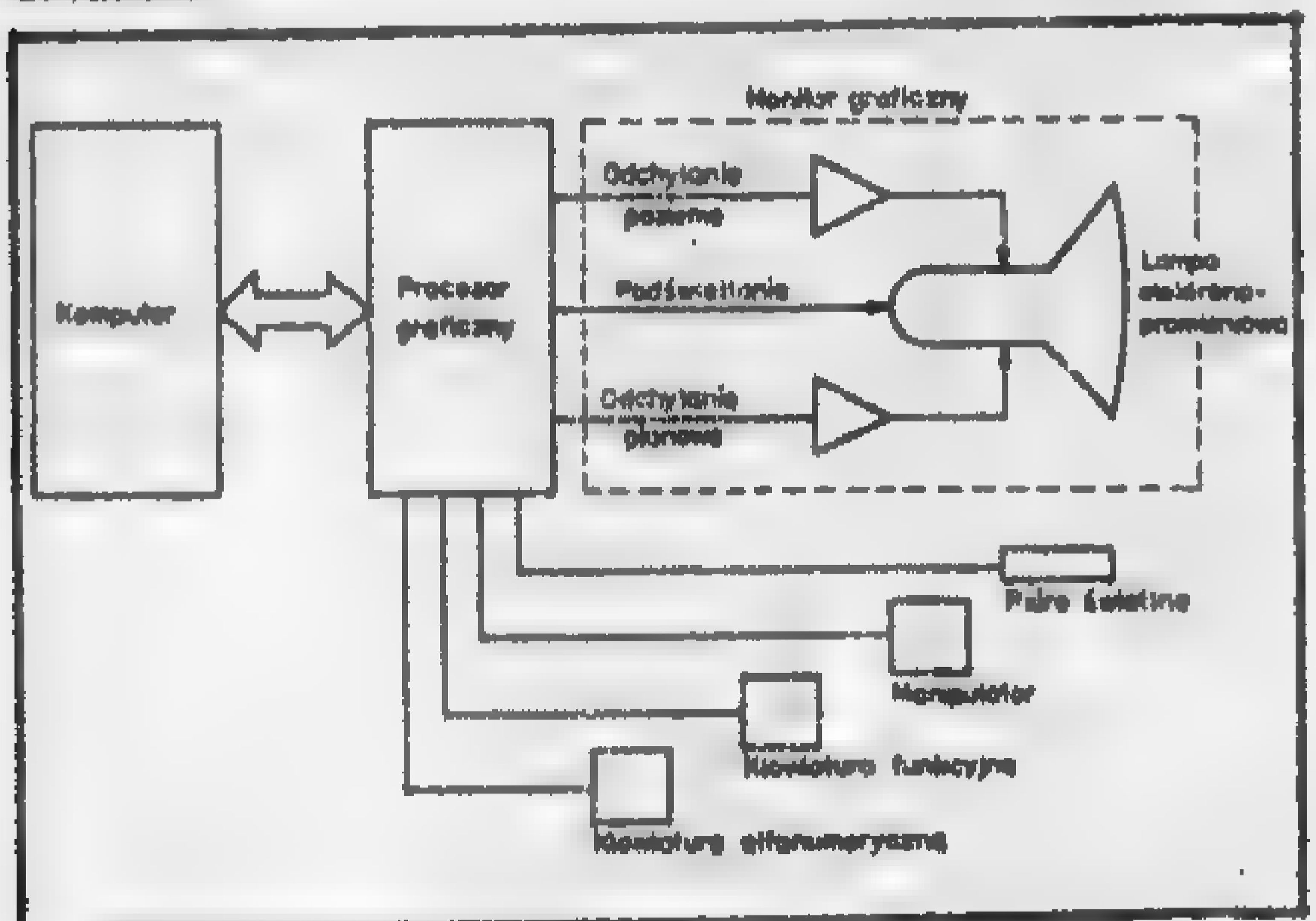
dokończenie ze strony 19

Procesor graficzny steruje wyświetlaniem obrazu na ekranie lampy elektronopromieniowej oraz przyjmuje sygnały pochodzące od urządzeń operatorskich. W prostszych rozwiązaniach niektóre urządzenia operatorskie dołączone są bezpośrednio do komputera.

Procesor graficzny wyposażony jest w pamięć buforową (pamięć obrazu). W pamięci obrazu znajdują się programy graficzne, opisujące obraz wyświetlany na ekranie lampy. Programy graficzne generowane są w komputerze i przesyłane do procesora graficznego.

Urządzenia operatorskie grafoskopu służą do komunikacji operatora z komputerem. Komunikacja ta może odbywać się z wykorzystaniem danych zobrazowanych na ekranie monitora.

Pióro świetlne jest urządzeniem reagującym na światło emitowane przez luminator lampy. Umożliwia to bezpośrednie sprzężenie pióra z obrazem wyświetlanym na ekranie. Operator, posługując się piórem świetlnym, ma możliwość wskazywania elementów obrazu lub też, poprzez pozycjonowanie piórem specjalnego znacznika świetlnego, wskazywania interesujących go miejsc na ekranie.



rys. 2

Manipulator kulowy służy do pozycjonowania znacznika świetlnego na powierzchni ekranu. Manipulator składa się z kuli opartej na rolkach i z dwóch przetworników analogowo-cyfrowych. Obracając kulą wywołuje się obrót przetworników, z których jeden odpowiada zmianom współrzędnej X, a drugi współrzędnej Y. Postać cyfrowa współrzędnych X, Y wykorzystywana jest do pozycjonowania znacznika świetlnego; współrzędne te mogą być ponadto wykorzystane do budowania obrazu kreślonego bezpośrednio na ekranie monitora.

Klawiatura funkcyjna umożliwia przesyłanie do komputera danych cyfrowych, odpowiadających numerom klawiszy funkcyjnych. Przesyłanie danych odbywa się w trybie przerwań. Poszczególnym klawiszom funkcyjnym można przyporządkować procedury użytkowe. Użytkownik ma w ten sposób bardzo dogodną możliwość aktywizowania wybranych fragmentów oprogramowania użytkowego.

Klawiatura alfanumeryczna służy do wprowadzania do komputera danych tekstowych. Tą drogą kierowane są przede wszystkim dyrektywy systemu operacyjnego, dyrektywy edycyjne itp.

Jerzy CHMURZYŃSKI

Jak budować program graficzny — napiszemy w numerze 3.

Liga Myślących

Zadanie nr 1

Boisko sportowe ma kształt prostokąta o bokach długości 80 m i 60 m. Dookoła boiska (w jego wnętrzu) biegnie droga o stałej szerokości. Jaka jest szerokość drogi, jeżeli jej powierzchnia równa się połowie powierzchni boiska?

Zadanie nr 2

Antykwariat zakupił dwa przedmioty za 225 zł, a sprzedał je o 40 % drożej. Ile zapłacono za każdy przedmiot, jeżeli cena pierwszego zwiększyła się o 25 %, a drugiego o 50 %.

Zadanie nr 3

Woda wlewa się do zbiornika przez 2 krany, z których jeden może napełnić zbiornik w 12 godzin, drugi w 8 godzin. Początkowo otwarty został kran pierwszy, a po upływie 2 godzin otworzono również i drugi. Po ilu godzinach od chwili otworzenia pierwszego kranu zbiornik będzie pełen?

Zadanie nr 4

Podaj algorytm rozwiązania równania kwadratowego

$$ax^2 + bx + c = 0$$

Zadanie nr 5

Pociąg odchodzi o godz. szóstej. Ile minut pozostało do jego odjazdu, jeżeli przed 50 minutami było, licząc od godziny trzeciej, cztery razy tyle czasu, ile pozostaje obecnie?

Zadanie nr 6

Zegar wybija godzinę trzecią, a czas trwania jego uderzeń wynosi 3 sekundy. Ile czasu zajmuje wybicie przez zegar godziny czternastej?

Rozwiązania zadań prosimy przysyłać pod adresem redakcji do końca czerwca br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Pocztowa giełda

Drodzy Czytelnicy! Ogromnie ucieszył nas „bukiet" kolorowych kartek i listów jako odpowiedź na ukazanie się pierwszego numeru „IKS-a". Tak jak zapowiadaliśmy wcześniej, wśród Czytelników, którzy nadesłali prawidłowe odpowiedzi, rozlosujemy nagrody. Podsumujemy również „Ligę Myślących". Wykorzystamy wszystkie Wasze uwagi zawarte w nadsyłanych ankietach.

Wszystko to uczynimy w 3 numerze.

Rozwiązania krzyżówek należy przysyłać tylko na kartkach pocztowych. Przypominamy nasz adres:

„IKS" — ul. Grzybowska 77,
00—950 Warszawa

Notowania na warszawskich „komputerowych targowiskach" w kwietniu przedstawiały się następująco:

ZX Spectrum 48K	— 80—90	tys.
ZX Spectrum Plus	— 125	tys.
Commodore VC 1531	— 90	tys.
Commodore C 64	— 180	tys.
Commodore VC-20	— 65	tys.
Atari 130 XE	— 160	tys.
Schnaider Amstrad CPC464	— 260	tys.
Meritum 1	— 160	tys.

Leszek Denkiewicz

UWAGA!

Przykłady programów będziemy również publikować w sobotnio-niedzielnych wydaniach „Żołnierza Wolności".

Przyjmujemy ogłoszenia reklamowe oraz ewentualne płatne ogłoszenia o kupnie lub sprzedaży sprzętu komputerowego. Chcesz kupić (sprzedać) mikrokomputer napisz do „IKS-a"!

W naszym komputerlandzie

W naszym Komputerlandzie także już wiosna. Zapracowany Spektruś nie spostrzegł jej nawet, bo w zimie tak wszystko zamarzło, że tylko dziełoroby, góralami zwane, i roboty z MPO pamiętały, że zima kiedyś się kończy.

Jednakże rozglądając się pilnie dookoła, Spektruś zauważył, że niemłoda już komputerzyca, Odra, jakby ożyła, poweselała i zalotnie jęła zerkać w jego kierunku wielkimi, wirującymi oczami.

Od kilkunastu już lat zajmowała pierwsze miejsce pośród największych i najpiękniejszych w komputerlandzkim fraucymerze. Jedną tylko wadę miała, o której mało kto wiedział, ale komputerlandzkie roboty nieraz się przez nią nacierpieć musiały. Grymaśna i kapryśna bowiem była okrutnie, zwłaszcza gdy o programy chodziło.

Pewnego razu przed wielkim balem, na który kwiat Komputerlandu sproszono, owa komputerowa dama specjalnie szykownego programu u Spektrusia zażądała, by gości oczarować. Pracował Spektruś dniem i nocą, by coraz to innym kaprysom komputerzycy sprostać. W przeddzień imprezy, gdy po napisaniu ostatniego algorytmu rozwiązania programowe zaproponować jej pragnął, dama kapryśna wydziwiać zaczęła. A to program za mało ambitny, a to niezbyt szeroki, wreszcie niekomunikatywny. Na koniec i to, że niestarannie zapisany. Rwać poczęła taśmy i niszczyć mozolne dzieło Spektrusia. Łatał, składał do rana...

Podglądał:
Eugeniusz MLECZAK

Lista „przebojów"

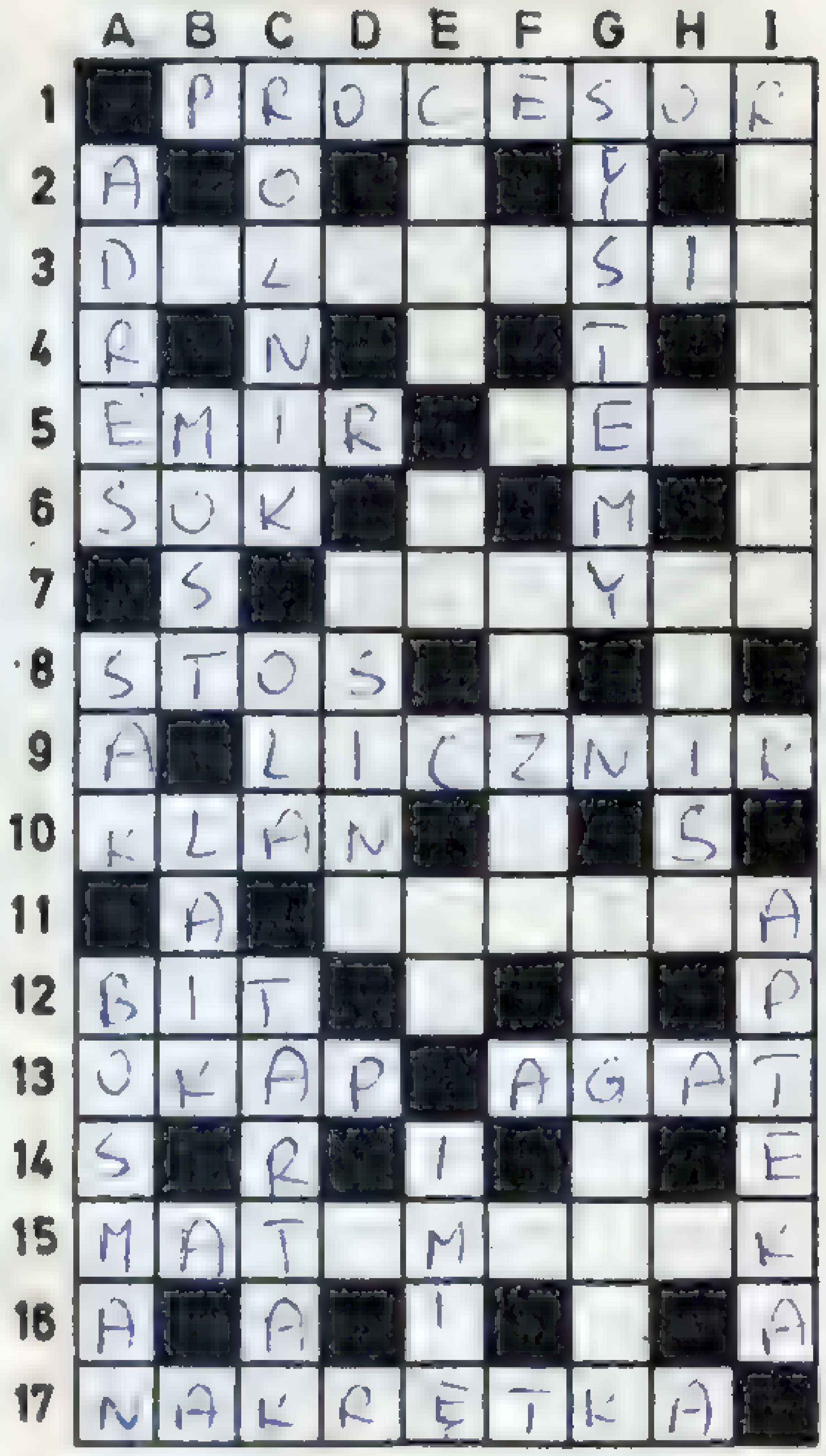
Różne są zamieszczone na naszych łamach programy; bywają dobre i atrakcyjne, a także odwrotnie, wzbudzają niewielkie zainteresowanie. Lista „przebojów" to aktualne notowania popularności naszych prezentacji.

Oto pierwsze pozycje na „Liście", które ustaliliśmy na podstawie wielu odebranych telefonów:

1. **Program 2 — „Wykresy"**
2. **Program 4 — „Wyścigi psów"**

Nasza „Lista" jest otwarta. Będziemy na niej starannie odnotowywać głosy Czytelników. Czekamy też na inne propozycje — na najciekawsze z nich, oprócz opublikowania i autorskiego honorarium, czekają nagrody.

Krzyżówka nr 2



POZIOMO: B-1) serce komputera, A-3) umożliwia porozumiewanie się z komputerem, A-5) dowódca arabski w XII w., F-5) płyta, A-6) owocowy lub trawienny, D-7) polowanie, A-8) zestaw danych, C-9) jeden z rejestrów mikroprocesora, A-10) grono, koło, D-11) klacz, A-12) 0 lub 1, A-13) daszek, F-13) kamień półszlachetny, A-15) myśliciel, A-17) zakrętka.

PIONOWO: A-2) ma każdy bajt pamięci, A-8) sieć, A-12) na statku, sierżant, B-5) nad rzeką, B-10) nie umie programować, C-8) mała Aleksandra, C-1) uprawia ziemię, C-12) zakład drzewny, D-7) rodzaj drewna, E-1) może być rozkazowy, E-6) rzeka w ZSRR, E-11) w gamie, E-14) nazwisko i ..., G-1) w „Iksie", G-11) chrząszcz, F-7) płynie po twarzy dziecka, H-7) wciśnięcie, I-1) skrawek, I-11) sklep z lekarstwami.

HASŁO: F-5, B-3, H-7; H-10, D-9, E-17; F-9; D-10, B-15, B-5, H-3.

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca czerwca br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Mikrociekawostki

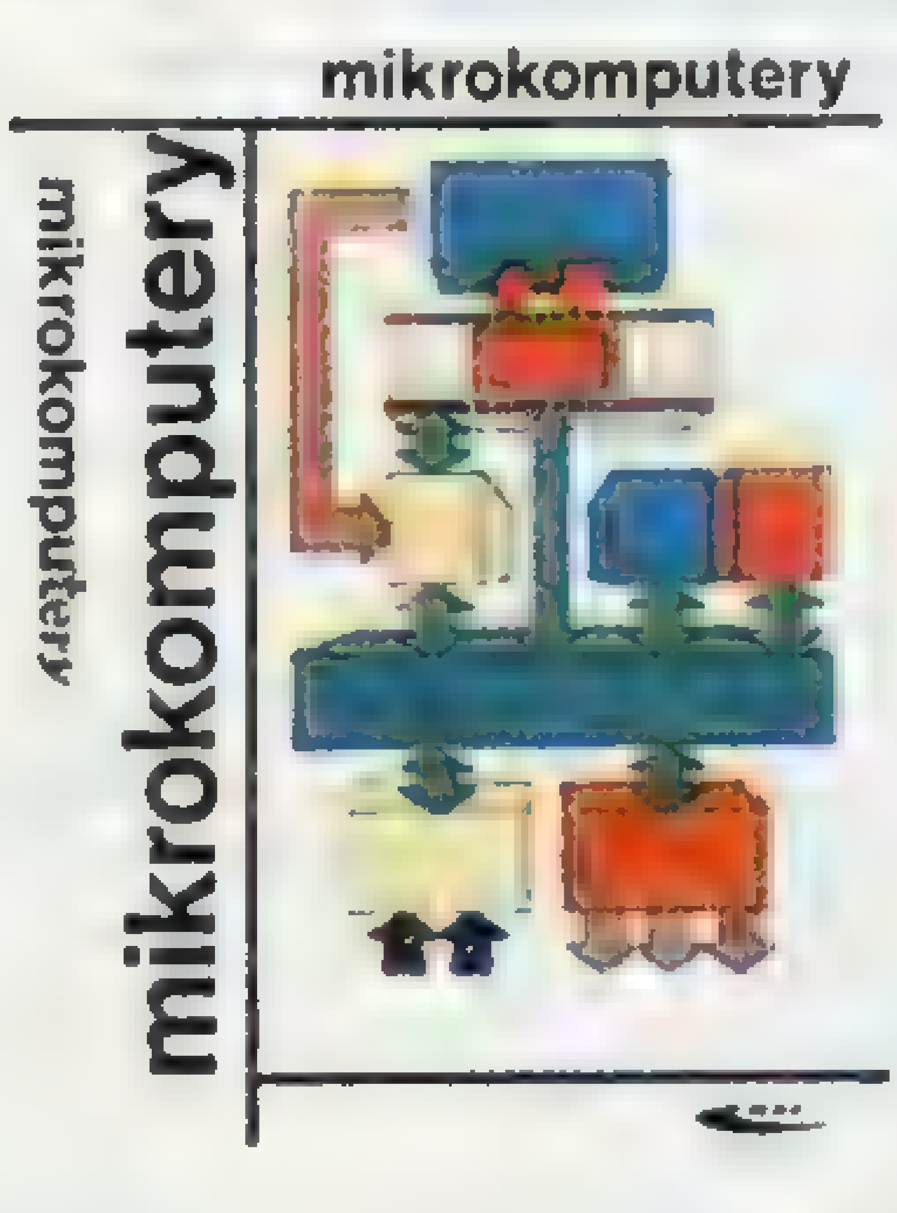


Wiele spośród produkowanych poza Europą systemów mikrokomputerowych nie zawsze w pełni odpowiada wymaganiom i właściwościom gospodarki europejskiej. Jedną z firm, która specjalizuje się w produkcji systemów mikrokomputerowych uwzględniających wymagania rynku europejskiego jest holenderska firma COMPUDATA. Dotychczas wypuściła ona na rynek europejski ponad 7500 różnych systemów mikrokomputerowych. Obecnie oferuje nowy, 16-bitowy system mikrokomputerowy do zastosowań komercyjnych.

System TULIP, bo o nim mowa, obejmuje 16-bitowy mikrokomputer mający magistralę o strukturze 16-bitowej (mikroprocesor Intel 8086) i bardzo dużej szybkości przesyłania informacji koprocesor arytmetyczny (Intel 8087), pamięć wewnętrzną o pojemności 128 KB, którą można rozszerzyć do 896 KB, pamięć zewnętrzną na dyskach twardych o pojemności do 10 MB plus 5 MB na dyskach wymiennych. W skład systemu wchodzi również barwny monitor o dużej zdolności rozdzielczej oraz klawiatura pozwalająca realizować 104 funkcje. Możliwość pracy w systemie operacyjnym o MS/DOS jak również CP/M86 umożliwia korzystanie z bogatego oprogramowania profesjonalnego, opracowanego przez inne firmy.

CS

Polecamy uwadze!



„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera, Ryszard Rogoń. Adres redakcji: 00—950 Warszawa ul. Grzybowska 77, telefon 20-21-27 i 359-34. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 7695. Nr ind. 382809 P-70

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 3/1986 ISSN 0860-2794



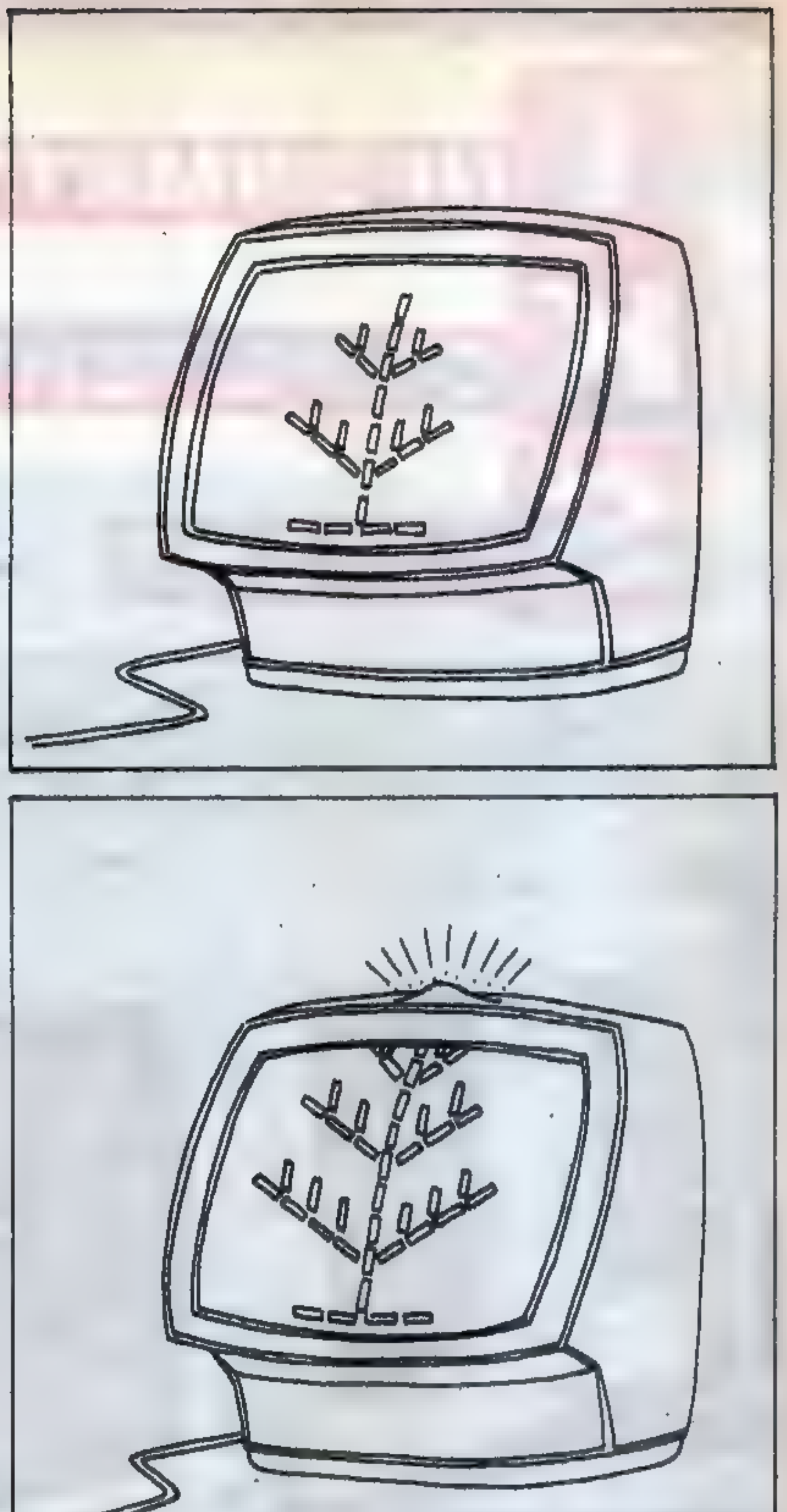
JUŻ wkrótce po ukazaniu się pierwszego numeru „IKS-a" dosłownie zasypani zostaliśmy listami. Ile ich przyszło? Nie liczyliśmy, ale codziennie poczta dostarczała nam ich ponad tysiąc — może jest ich 15, a może 20 tysięcy. Nie wszystkie możemy czytać natychmiast, ale z pewnością wszystkie przeczytamy. Dziękujemy za rady, uwagi i życzenia. Jeśli „IKS" ma być pismem dobrym, to każdy jego numer powstawać musi przy pomocy Czytelników. Stąd też tak niecierpliwie oczekujemy Waszego głosu.

Niezwykle ciekawa jest lektura nadesłanych ankiet. Prawie wszyscy domagacie się publikowania na naszych łamach programów na różne mikrokomputery, jedni proszą o programy gier, inni o programy użytkowe. Oczywiście będziemy je drukować. Ale prasa rządzi się własnymi prawami; nie możemy zamieszczać programów długich. To, co proponujemy, to krótkie konstrukcje programowe, sądzimy jednak, że okażą się ciekawe.

Ale prezentowanie programów nie może być naszym najważniejszym celem. Bo przecież bez względu na ich atrakcyjność wymagają od użytkowników jedynie biernych umiejętności : ciąg komend trzeba wczytać do komputera i potem jednym rozkazem uruchomić wprowadzony program. Nie potrzebna tu jest żadna wiedza — liczy się jedynie sprawność rąk. A gdy zdarza się pomyłka, bo przecież zdarzyć zawsze się może, okazuje się, że nie zawsze można sobie z jej usunięciem poradzić.

JEDNI uważają, że od użytkownika takich umiejętności wymagać nie można; wystarczy, że będzie znał programy, ich przeznaczenie i poradzi sobie z ich uruchomieniem. Zgoda. Dla wielu komputer pozostanie na zawsze czarną skrzynką, urządzeniem o nieznanej konstrukcji, ale poznanych możliwościach. Dociekliwi będą natomiast poszukiwać i im chcemy pomóc. Z „IKS-em" łatwiej będzie usiąść przed komputerem i rozpocząć z nim „rozmowę". Mamy nadzieję, że tak burzliwie rozwijające się kluby mikrokomputerowe sprawią, że dostęp do tych urządzeń będzie łatwiejszy, że komputery trafią „pod strzechy".

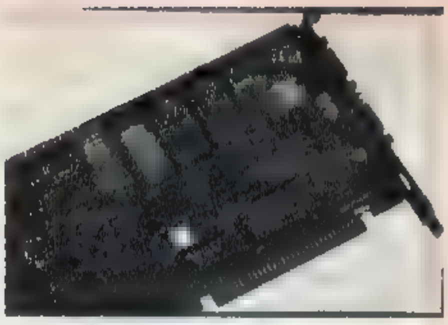
Redakcja



Uwaga! Błąd w programowaniu...

Rys. Michał Przybyłowski

Krótka historia mikroprocesorów



Niemal czterdzieści lat temu, bo w 1947 roku powstał pierwszy tranzystor. Po kilku latach doświadczeń, prawdopodobnie w celach oszczędnościowych, spróbowano umieścić kilka tych elementów w jednej obudowie. Rezultatem poszukiwań był (w 1959 roku) pierwszy układ scalony, tak zwanej *małej skali integracji* (SSI — small — scale integration). Z czasem w jednej obudowie „upakowywano" coraz więcej elementów a produkt finalny elektronicy nazwali „kością". Pojawiła się również *średnia skala integracji,* około 50 tranzystorów w jednej obudowie (MSI — medium — scale integration), następnie *duża skala integracji* — tysiące tranzystorów w jednej obudowie (LSI — large — scale integration). — Ostatnio króluje *bardzo duża skala integracji* — miliony tranzystorów w jednej obudowie (VLSI very large scale integration).

Istniejąca zaledwie od roku firma **Intel** na bazie osiągnięć „upakowania" zbudowała pierwszy układ pamięci półprzewodnikowej, o zawrotnej jak na ówczesne czasy (1969) pojemności — tysiąc (1K) bitów. Był to okres intensywnych prac nad kalkulatorami. Poświęcone im badania, w ciągu niecałych dwóch lat dają efekt w postaci pierwszego mikroprocesora 4004. Zgodnie z przyjętą nomenklaturą mikroprocesory zazwyczaj nie mają nazw literowych, jest to szereg cyfr (minimum czterech), czasami poprzedzonych symbolem literowym. Czterobitowa struktura nowego mikroprocesora wynikała z przeznaczenia — kalkulatory oraz obowiązujące standardy BCD (binary-coded decimal). Innymi słowy chodziło o to, by procesor przetwarzał cyfry dziesiętne — zapisanie ich wymaga właśnie czterech bitów. 4004 wykonuje sześćdziesiąt tysięcy operacji w czasie jednej sekundy.

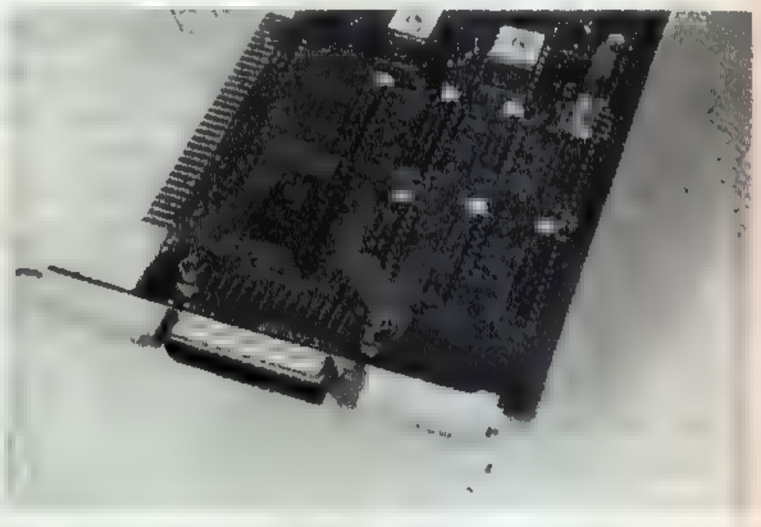
Między innymi konieczność dostosowania się do terminali wykorzystujących standard ASCII (American Standard Code for Information Interchange) przyczyniła się do rozpoczęcia badań nad mikroprocesorem ośmiobitowym. W kwietniu 1972 roku prace te zostają uwieńczone sukcesem — uruchomiono produkcję mikroprocesora 8008. Jego szybkość jest imponująca — ⅓ miliona instrukcji na jedną sekundę. Listę rozkazów tworzy aż 45 różnych instrukcji, które mogą operować na danych z pamięci o pojemności 16 000 (16K) słów.

Wydaje się, że osiągnięcie to stanowiło ostatni element fundamentów mikrokomputera, bowiem już na przełomie wiosny i lata 1974 roku pismo Radio — Elektronics prezentuje pierwszy mikrokomputer MARK-8. Jego twórcą jest Jonathan Titus.

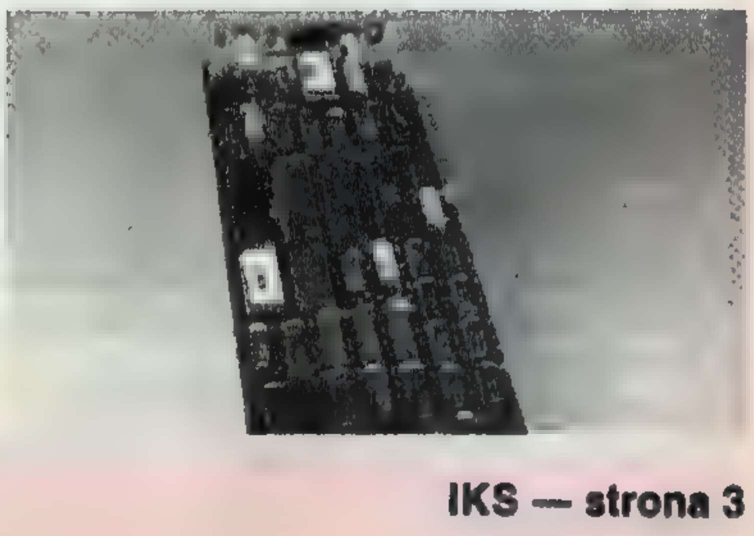
Kolejnym dzieckiem z rodziny mikroprocesorów jest 8080, który realizuje 75 instrukcji, a jego pamięć może mieć 64K słów.

W celach sondażowych firma MITS buduje mikrokomputer ALTAIR 8800 z pamięcią 256 słów (nie 256k!) za 395 $. Jest to cena stosunkowo niska, a producent, nie bez podstaw, zapewniał kupujących, że nabywają swój własny, prywatny, osobisty komputer. Firma MITS mogła sprzedać więcej w jeden dzień, niż planowała sprzedać przez okres całej produkcji ALTAIRa. Sukcesu należy upatrywać w tym, iż architektura tego komputera pozwalała dołączać urządzenia peryferyjne. Jednym z nich jest kontroler dysków zbudowany przez Digital Microsystems. W mikrokomputerze wykorzystano po raz pierwszy system operacyjny CP/M (Control Program for Microcomputers).

dokończenie na str. 4



Masowa produkcja układów scalonych — nowa perspektywa elektroniki



dokończenie ze str. 3

W odpowiedzi na 8080 firma Motorola rozpoczyna produkcję 6080 wraz z układami pozwalającymi łączyć mikroprocesor z peryferiami. Krótko potem rodzina tego procesora powiększa się o układy zapewniające kontakt z urządzeniami zewnętrznymi drogą transmisji równoległej — 6820 oraz szeregowej — 6850.

Jeden z filarów Intela — F. Faggin uznał, nie bez podstaw, iż bardziej kalkuluje się mu założyć własną firmę — Zilog. Wraz z Masatoshi Shima tworzą mikroprocesor „super 8080" — Z80 (4 MHz) 176 instrukcji! (Częstotliwość np.: 4 MHz, 2 MHz określa liczbę podstawowych operacji, jakie może wykonać procesor w czasie jednej sekundy). Był to bezwzględnie lepszy mikroprocesor od swego konkurenta 8080.

Intel „trzymający rękę na pulsie rozwoju mikroprocesorów" buduje w 1976 roku 8085 (3 MHz). Może ze względu na nieobecność F. Faggina, procesor te nie spełnił pokładanych w nim nadziei. Dopiero w 1978 roku prawie pełen sukces odniósł już szesnastobitowy 8086, ale tu wystąpiły kłopoty z oprogramowaniem.

Realizacja pomysłu tak zwanego „busa" dopełniła oczekiwanego sukcesu. Mikroprocesor z „busem" 8088 („młodszy brat" 8086) jest w swej istocie 16-bitowy, ale ma ośmiobitowe połączenie (bus) „do zewnątrz", tam gdzie niepodzielnie panuje 8-bitowy standard.

Podobnie Motorola w 1979 roku prezentuje „szesnastkę" — MC 68000. Mimo skorzystania z doświadczeń Intela w zakresie dobudowania „busa", czego wynikiem był 68008, mikroprocesor 8088 okazał się znacznie lepszy.

Warto wiedzieć, że 68000 to układ z możliwością mikroprogramowania (tak zwane programowanie „random — logic"). Ma to kapitalne znaczenie wszędzie tam, gdzie istotna jest szybkość wykonywania operacji.

Motorola 68020 pozwala na dynamiczną modyfikację rozmiaru „busa" 8, 16 lub 32 bity i jest kompatybilna z 68000. Zwiastuje to narodziny pierwszego mikroprocesora.

Wysiłki Intela skierowane zostają na budowę 80386 — 32-bitowej wersji 80286. Również wcześniej wspominany procesor ZX 80 jest doskonalony, powstaje 16-bitowy Zilog Z800 kompatybilny z Z80. Niestety prace te skończyły się niepowodzeniem. Jeden z głównych konstruktorów Z800 Masatoshi Shima odszedł, podobnie jak wcześniej jego szef z Intela, i nikt nie potrafił poprawnie dokończyć oprogramowania Z800. Zdarzenie to dowodzi, że i dzisiaj w dobie mikroprocesorów, praca i talent jednostki bywają niezastąpione.

Dotychczasowa praktyka konstruktorów wskazuje, iż jednym z podstawowych dylematów podczas budowy nowych mikroprocesorów jest liczba instrukcji. Wartość ta z jednej strony decyduje o szybkości, z drugiej natomiast o złożoności układu procesora; im krótszy czas wykonywania określonych czynności tym bardziej skomplikowana produkcja tego układu.

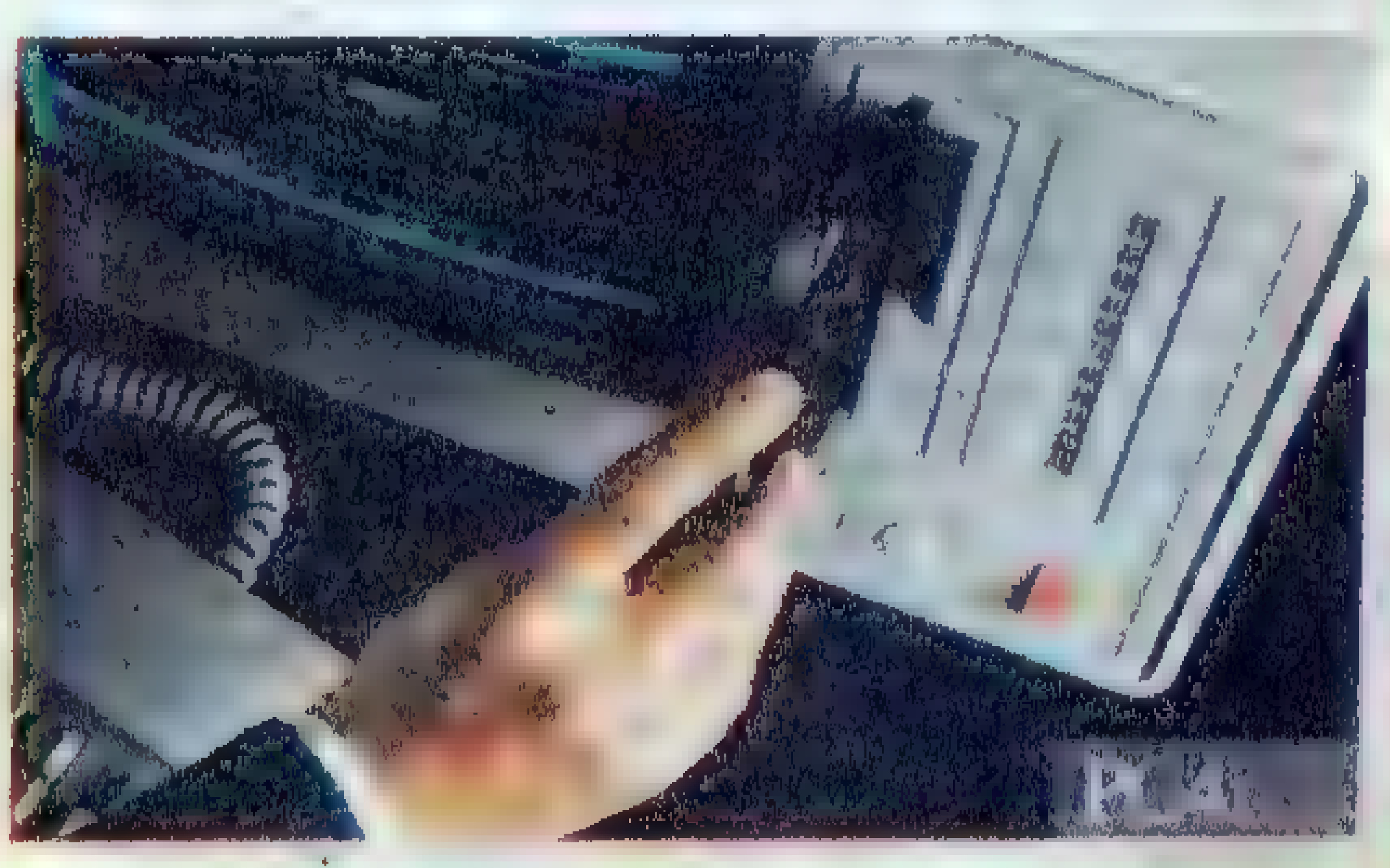
dokończenie na str. 5





Amerykańska firma NCR, której siedziba mieści się w Dayton, jest jeszcze mało znana na światowym rynku komputerowym. Oferowane przez nią komputery NCR PC4i, NCR PC6 i NCR PC8 są w pełni kompatybilne z komputerami firmy IBM a ponadto są od nich tańsze. Pracują w oparciu o 16-bitowy procesor 8088 (NCR PC6 i NCR PC4i) i 80286 (NCR PC8). Pojemność pamięci 256KB z możliwością rozszerzenia do 640 KB. Posiadają możliwość dołączenia stacji dysków miękkich i twardych co pozwala na zwiększenie pamięci do 20 MB.

O



dokończenie ze str. 4

Ostatnio podejmowane są starania, by procesory wykonywały bezpośrednio instrukcje języków wyższego poziomu. Przykładem jest procesor NOVIX, który wykonuje instrukcje języka FORTH.

Doskonalenie sprzętu mikro często jest źródłem nieoczekiwanych wytworów, jakim jest np. INMOS — transputer, układ pozwalający na równoległe przetwarzanie danych. Minie prawdopodobnie wiele czasu do jego użytkowego oprogramowania. Jednak perspektywa jest oszałamiająca. Ale już dzisiaj można kupić układ, RISC (reduced instruction set computer), „przyśpieszacz", który po dołączeniu do procesora znacznie skraca czas realizacji instrukcji.

Co dalej? Firma Hitachi proponuje pamięci, w niektórych tradycyjny bit stał się wielowartościowy. Ma cztery stany, a czas dostępu wynosi od 1 do 2 mikrosekund! Dla koneserów mikro Toschiba poleca pamięci RAM o pojemności 1 megabit każdy układ, z czasem dostępu 60 nanosekund.

Z historii mikroprocesorów

1948 Wynaleziono tranzystor.
1959 W Texas Instruments powstaje pierwszy zintegrowany obwód.
1964 J. G. Kemeny, T. E. Kurtz w Dortmouth College stworzyli BASIC
1971 Intel Corp. rozpoczyna sprzedaż 4-bitowego mikroprocesora 4004 za 200 $.
1972 W grudniu Intel proponuje 8-bitowy mikroprocesor 8008.
1973 Początek produkcji 8080.
1974 Gary Kidall opracowuje system operacyjny CP/M.
Zimą Zilog wprowadza do handlu mikroprocesor Z80.
1975 W październiku firma IBM oferuje mikrokomputer w teczce — IBM 5100, Basic, 16K pamięci, cena 9000 $.
1976 Pierwszy 16-bitowy mikroprocesor TM S9000 produkcji Texas Instruments.
W sierpniu pierwszy floppy-disk-drive, 8 cali (1195 $).
1977 Kwiecień — Commodore Business Machines Inc. produkuje komputer PET (procesor 6502, 4K RAM, 14K ROM, klawiatura, monitor, magnetofon.
Lipiec — Apple Computer (6502, 4K RAM, 16K ROM, klawiatura, interfejs do kaset, specjalizowane gniazda, kolorowy monitor.
1978 Grudzień — Zaprezentowana zostaje drukarka mozaikowa MX-80. Powstaje Atari 400 i 800, 8K RAM (do 48K), dwa gniazda do ROM.
1980 Powstaje do dzisiaj popularny na bazie Z80A, 1K RAM, 4K ROM.
Czerwiec — Shugart sprzedaje pierwsze „twarde dyski" typu Winchester — 5¼ cala.
1981 Wrzesień — dostępny w sklepach IBM Personal Computer (PC), mikroprocesor 8088, 64K RAM, 40K ROM, 5¼ cala dyski.
1982 Lato — Commodore 64, procesor 6510, 64K RAM, 20K ROM, dobrej jakości generator dźwięku, kolory, szeregowy interfejs.
LOGO — język programowania jest dostępny w większości komputerów.
Lipiec — Intel wprowadza na rynek mikroprocesory 80186, 80286.
1983 Styczeń — pojawia się doskonalsza wersja Atari — ATARI 1200 XL.
Luty — IBM PC XT, twardy dysk 10 MB, 128K RAM.
Wrzesień — dostępne są ekrany dotykowe — Touchsreen produkcji Hewlett--Packard Company — HP 150.
Październik — Shugart Comp. rozpoczyna produkcję optycznych dysków, tylko do odczytu, pojemność jeden gigabajt.
Dostępne są drukarki laserowe o zdolności nanoszenia 300 punktów na 2,5 cm.
1984 Styczeń — Apple Computer Inc. przedstawia mikrokomputer Macintosh.
Pierwszy zegarkowy komputer (produkcji Seiko Instruments), 10 znaków w każdej z 4 linii na mini-ekranie LCD, 2K CMOS RAM, 6K ROM.
Sinclair QL (procesor 68008, 128K RAM, 2 gniazda na mikrokasety).
Czerwiec — Motorola rozpoczyna produkcję 68020 — 32-bitowy procesor.
Sierpień — IBM PC AT, 80286, 256K RAM, 1,2 MB floppy-disk drive i szereg innych udoskonaleń, w cenie 5469 $.

cdn.

oprac. **W. GOGOŁEK**

PROGRAM 11

ATARI

Plansza gry, przedstawiona na ekranie, składa się z dziewięciu pól. Część z nich, wybrana w sposób losowy jest jasna, pozostałe są ciemne. Zadaniem gracza jest zapalenie 8 pól brzegowych i wygaszenie środkowego. W celu przełączenia wybranego pola należy przesunąć w jego kierunku drążek manipulatora i nacisnąć czerwony przycisk (FIRE). Można wybierać tylko zapalone pola. Zmieniając wybrane pole zmienia się również stan jego sąsiadów stykających się z nim brzegami.

```
   1 REM « LAMIGLOWKA »
   5 GR.2:GOSUB 20000
  12 DIM A$(1),PZ(10)
 100 MV=1:SET.4,0,0
 110 GOSUB 6000:REM KRESLENIE PLAN
     SZY
 115 GOSUB 7000:REM LOSOWANIE POL
 116 GOSUB 7100:REM KRESLENIE POL
 117 ?"CZY CHCESZ INSTRUKCJE GRY";
     :I.A$
 118 IF A$="T"THEN GOSUB 3000:REM
     INSTRUKCJA
 120 TRAP 120
 121 ?"[CLEAR] RUCH NR "; MV;"?"
 130 IF STRIG (JS)=0 THEN 130:REM
     OCZEKIWANIE NA NACISNIECIE
     PRZYCISKU
 131 ST=STICK (JS):IF STRIG (JS)=1
     THEN 131
 138 GOSUB 140+(ST-5)
 139 GOTO 155
 140 M=3:RETURN
 141 M=9:RETURN
 142 M=6:RETURN
 144 M=1:RETURN
 145 M=7:RETURN
 146 M=4:RETURN
 148 M=2:RETURN
 149 M=8:RETURN
 150 M=5:RETURN
 155 IF PZ(M)=0 THEN ?"[CLEAR] WY
     BIERAJ TYLKO ZAPALONE POLA
     -TWOJ RUCH": GOTO 130
 160 MV=MV+1
 190 PZ(M)=0:REM WYLACZENIE WYBRA
     NEGO POLA
 191 GOSUB 7100:REM AKTUALIZACJA
     PLANSZY
 250 GOSUB 300+M*10
 260 MN=0:GOSUB 7100
 270 FOR Q=1 TO 9:WN=WN+PZ(Q):
     NEXT Q
 280 IF WN=8 AND PZ(5)=0 THEN 7500:
     REM WYGRANA
 290 IF WN=0 THEN 7600:REM PRZEGRANA
 300 GOTO 120:REM NASTEPNY RUCH
 310 PZ(2)=1-PZ(2):PZ(4)=1-PZ(4)
 312 PZ(5)=1-PZ(5):RETURN
 320 PZ(1)=1-PZ(1):PZ(3)=1-PZ(3)
 322 RETURN
 330 PZ(2)=1-PZ(2):PZ(5)=1-PZ(5)
 332 PZ(6)=1-PZ(6):RETURN
 340 PZ(7)=1-PZ(7):PZ(1)=1-PZ(1)
 342 RETURN
 350 PZ(8)=1-PZ(8):PZ(4)=1-PZ(4)
 352 PZ(6)=1-PZ(6):PZ(2)=1-PZ(2)
 354 RETURN
 360 PZ(9)=1-PZ(9):PZ(3)=1-PZ(3)
 362 RETURN
 370 PZ(8)=1-PZ(8):PZ(4)=1-PZ(4)
 372 PZ(5)=1-PZ(5):RETURN
 380 PZ(7)=1-PZ(7):PZ(9)=1-PZ(9)
 382 RETURN
 390 PZ(8)=1-PZ(8):PZ(5)=1-PZ(5)
 392 PZ(6)=1-PZ(6):RETURN
 999 END
1000 JS=99
1010 FOR X=0 TO 3
1020 IF STRIG(X)=0 THEN JS=X:GOSUB
     1100
1030 NEXT X
1040 IF JS>3 THEN 1010
1050 RETURN
1100 IF STRIG(JS)=0 THEN 1100:REM
     CZEKANIE NA ZWOLNIENIE PRZYCI
     SKU
1199 RETURN
```

```
6000 COLOR 3
6010 FOR Z=0 TO 12 STEP 6
6015 PLOT 10,Z:DR.28,Z
6020 FOR X=10 TO 28 STEP 6
6030 FOR Y=1 TO 5
6040 PLOT X,Y+Z
6050 NEXT Y:NEXT X:NEXT Z
6080 PLOT 10,18:DR.28,18
6099 RETURN
7000 FOR R=0 TO 10
7010 PZ(R)=INT(RND(0)+0.5)
7020 NEXT R
7099 RETURN
7100 PN=0
7110 FOR Z=12 TO 0 STEP -6
7120 FOR X=1 TO 13 STEP 6
7122 PN=PN+1:COL.PZ(PN)
7125 FOR Y=1 TO 5
7130 PLOT X+10,Y+Z:DR.X+14,Y+Z
7140 NEXT Y:NEXT X:NEXT Z
7190 RETURN
7500 GOSUB 8000:REM AKTUALIZACJA
7570 ?"[CLEAR] WYGRALES W ";MV;" RU
     CHACH "
7571 SET.4,4,4
7575 ?"NOWA GRA ";:I.A$
7590 IF A$="T" THEN 100
7595 GR.0:END
7600 GOSUB 8000:REM AKTUALIZACJA
7605 ?"[CLEAR] NIE MOZESZ TERAZ WY
     GRAC."
7606 SET.4,12,4
7610 GOTO 7575
8000 NG=NG+1
8010 AV=INT((AV*(NG-1)+MV)/NG)
8099 RETURN
20000 REM STRONA TYTULOWA
20010 ?"[CLEAR]":GR.2:POS.5,4
20020 ?#6;"LAMIGLOWKA"
20030 ?"  GRA WYMAGA MANIPULATORA!  "
20040 ?" nacisnij czerwony przycisk w celu "
20050 ?"      rozpoczecia gry       "
20060 JS=99
20070 FOR X=0 TO 3
20080 IF STRIG(X)=0 THEN JS=X
20090 NEXT X
20100 IF JS>3 THEN 20070
20110 GR.3:RETURN
```

```
30000 ?"[CLEAR] CELEM GRY JEST ZAPA
      LENIE WSZYSTKICH"
30010 ?"POL PLANSZY OPROCZ SRODKO
      WEGO."
30020 ?"(C.D. PO NACISNIECIU <FIRE>)...
      ":GOSUB 1000
30030 ?"[CLEAR] PRZELACZASZ POLE PO
      PRZESUNIECIU"
30040 ?"DRAZKA MANIPULATORA W JEGO
      KIERUNKU"
30050 ?"I NACISNIECIU CZERWONEGO
      PRZYCISKU":GOSUB 1000
30060 ?"[CLEAR] MOŻESZ WYBIERAC TYL
      KO ZAPALONE POLA."
30070 ?"ZMIENIAJAC WYBRANE POLE
      ZMIENIASZ"
30080 ?"STAN JEGO SASIADOW.":GOSUB
      1000
30090 ?"[CLEAR] PAMIETAJ: RUCH MANI
      PULATORA W GORE"
30100 ?"WYBIERA GORNY SRODEK, W
      DOL — DOLNY"
30110 ?"SRODEK; PODOBNIE INNE KIERU
      NKI:"GOSUB 1000
30120 ?"[CLEAR] PO DOKONANIU WYBO
      RU NACISNIJ"
30130 ?"CZERWONY PRZYCISK (TRZYMA
      JAC JEDNA"
30140 ?"REKA DRAZEK MANIPULATORA)."
      :GOSUB 1000
30150 ?"[CLEAR] JESLI WYLACZYSZ NARO
      ZNIK, TO"
30160 ?"ZMIENISZ TAKZE JEGO TRZECH
      SASIADOW.":GOSUB 1000
30170 ?"[CLEAR] JESLI WYLACZYSZ SRO
      DEK BOKU, TO"
30180 ?"ZMIENISZ TAKZE JEGO DWOCH
      SASIADOW NA TYM"
30190 ?"SAMYM BOKU.":GOSUB 1000
30200 ?"[CLEAR] JESLI WYLACZYSZ SRO
      DEK PLANSZY, TO"
30210 ?"ZMIENISZ TAKZE SRODKOWE
      KWADRATY"
30220 ?"NA KAZDYM BOKU.":GOSUB 1000
30230 RETURN
```

W następnym numerze gra losowa: „Pole minowe"

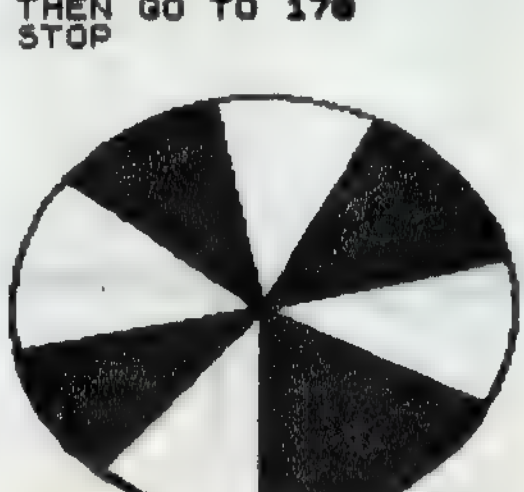
PROGRAM 12

„Histogram" i „Wykres kołowy" są programami napisanymi na „Spectrum". Przeznaczone są do graficznego zobrazowania danych — dzięki nim wielkości liczbowe z łatwością przenoszone są na odpowiednie wykresy (mogą przydać się w szkole).

```
  10 REM WYKRES KOLOWY
  20 REM
  30 BORDER 1: PAPER 7:
     INK 1: CLS
  40 PRINT AT 9,8;
     "WYKRES KOLOWY"
  50 PRINT AT 16,4; FLASH 1;
     "nacisnij dowolny klawisz":
     PAUSE 0: CLS
  60 INPUT "Ile liczb wprowadzis
z? ";n
  70 DIM t(n)
  80 LET suma=0
  90 REM

 100 REM Wczytanie kolejnych
         liczb
 110 FOR I=1 TO n
 120 PRINT AT 16,5;"Liczba nr ";
     I
 130 BEEP .1,5
 140 INPUT liczba
 150 LET suma=suma+liczba:
     LET t(I)=suma
 160 NEXT I
 170 BORDER 7: CLS
 180 LET a=-0.01
 190 REM

 200 REM Rysowanie wykresu
 210 FOR I=1 TO n
 220 INK (I-(INT (I/6))*6)/2
 230 LET a=a+0.01
 240 PLOT 127,87
 250 IF INT (I/2)=(INT I)/2
     THEN PLOT 127+70*COS (a),
         87+70*SIN (a):
     GO TO 270
 260 DRAW 70*COS (a),70*SIN (a)
 270 IF a<t(I)*2*PI/suma-0.01
        THEN GO TO 230
 280 NEXT I
 290 INPUT "Nastepny wykres (Tak
,Nie)?";r$
 300 IF r$(1)="T" OR r$(1)="t"
     THEN GO TO 170
 310 STOP
```



SPECTRUM

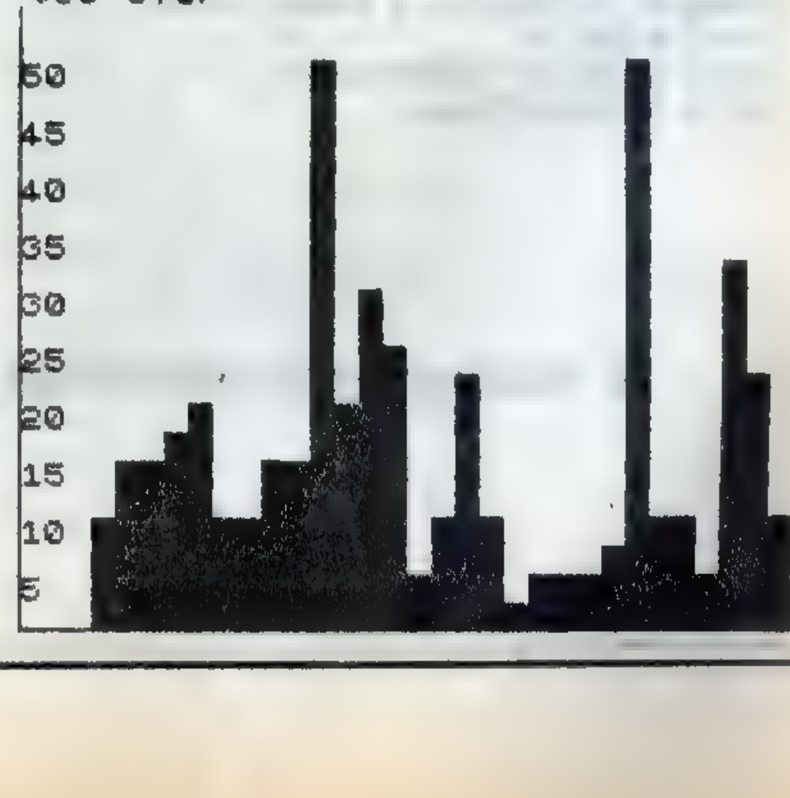
```
  10 REM HISTOGRAM

  20 REM
  30 BORDER 7: PAPER 7:
     INK 1: CLS
  40 PRINT AT 3,10;"HISTOGRAM"
  50 PRINT AT 6,1;"Program wyswi
etlania danych w    postaci wykr
esu kolumnowego"
  60 PRINT AT 17,4; FLASH 1;
     "nacisnij dowolny klawisz"
  70 PAUSE 0: CLS
  80 INPUT "Ile liczb wprowadzis
z(max 29)? ";illiczb
  90 IF illiczb>=30 THEN
        GO TO 80
 100 IF illiczb<1 THEN GO TO 80
 110 LET illiczb=INT (illiczb)
 120 DIM v(illiczb)
 130 LET max=0
 140 REM
 150 REM Wczytanie kolejnych
         liczb
 160 FOR I=1 TO illiczb
 170 PRINT AT 18,5; INK 2;
     "Liczba nr "
 180 BEEP .1,5:
     PRINT AT 18,15;I
 190 INPUT v(I)
 200 IF v(I)>max THEN
        LET max=v(I)
 210 NEXT I
 220 CLS
 230 REM

 240 REM Przeskalowanie
 250 LET skala=1
 260 IF max>20 THEN
        LET skala=max/20
 270 FOR I=1 TO illiczb
 280 LET v(I)=INT (v(I)/skala)
 290 NEXT I
 300 REM

 310 REM Rysowanie osi liczbowej
 320 FOR I=1 TO 20 STEP 2
 330 PRINT AT 21-I,0;
   INT (I*skala+.5)+INT (skala)
 340 NEXT i
 350 PLOT 0,0
 360 DRAW 0,175
 370 PLOT 255,0
 380 DRAW -255,0
 390 REM

 400 REM Rysowanie histogramu
 410 FOR I=1 TO illiczb
 420 FOR J=1 TO v(I)
 430 PRINT AT 22-J,2+I;"█"
 440 NEXT J
 450 NEXT I
 460 STOP
```



PROGRAM 13

SPECTRUM

PROGRAM KORELACJE

UWAGA STATYSTYCY! Program ten pozwala na obliczenie wielkości statystycznej zależności między szeregami dwóch kolumn liczb. Wynikiem są wartości współczynników Spearmana i Pearsona.

Ze względów praktycznych adresy instrukcji wydzielają moduły programu. Ma to na celu ułatwienie ewentualnych udoskonaleń: kontrola zakresów danych wejściowych, zamierzone ich przekształcanie np.: $a=a^{-1}$, $a=a^2$, $a=\ln a$. Przekształcenia danych wejściowych pozwalają na stwierdzenie innych, poza liniowymi, zależności; wykładniczych, logarytmicznych itp.

```
  10 REM Wspolczynniki korelacji
         PEARSONA i SPEARMANA
  20 GO SUB 4000
  30 CLS : INPUT "Podaj liczbe d
anych"'k
  40 DIM a(k)
  50 FOR n=1 TO k
  60 INPUT "Napisz wartosc a";VA
L "n";"=",a(n)
  70 NEXT n
  80 PRINT '''Koniec wprowadzani
a danych typu a"''
  85 PRINT '';"Kontrolny list dan
ych a(n)"
  90 FOR n=1 TO k
 100 PRINT "a";n;"=";a(n)
 110 NEXT n
 120 LET t$="t"
 130 INPUT "Czy  chcesz poprawic
 dane wejs- ciowe?, jesli TAK na
cisnij kla- wisz litery t, jesli
 NIE - kla- wisz n",u$
 140 IF t$=u$ THEN GO TO 200
 150 GO TO 1030
 200 INPUT "Podaj numer danej kt
ora chcesz   zmienic"'';n
 210 INPUT "Napisz poprawna wart
osc a",VAL "n",a(n)
 220 GO TO 85
1040 DIM b(k)
1050 FOR n=1 TO k
1060 INPUT "Napisz wartosc b";VA
L "n";"=",b(n)
1070 NEXT n
1080 PRINT "Koniec wprowadzania
b"''
1085 PRINT '';"Kontrolny list b(n
)"''
1090 FOR n=1 TO k
1100 PRINT "b";n;"=";b(n)
1110 NEXT n
1120 INPUT "Czy chcesz poprawiac
 dane wejs- ciowe typu b(n)?",u$
1130 IF t$=u$ THEN GO TO 1200
1140 GO TO 2000
1200 INPUT "Podaj numer danej kt
ora chcesz   zmienic",n
1210 INPUT "Napisz poprawna wart
osc b";VAL "n",b(n)
1220 GO TO 1085
2040 LET Nab=0: LET Sa=0: LET Sb
=0: LET ka=0: LET ksa=0: LET kb=
0: LET Ksb=0: LET r=0
2050 FOR n=1 TO k
2100 LET Nab=a(n)*b(n)+Nab
2110 LET Sa=a(n)+Sa
2120 LET Sb=b(n)+Sb
2130 LET ka=a(n)*a(n)+ka: LET ks

a=Sa*Sa: LET kb=b(n)*b(n)+kb: LE
T Ksb=Sb*Sb
2135 NEXT n
2140 REM  r  - wsp. korelacji
2150 LET r=(k*Nab-Sa*Sb)/SQR ((k
*ka-ksa)*(k*kb-Ksb))
2160 PRINT '''"Wspolczynnik kore
acji Pearsona r wynosi:",''....r
'''''
2165 GO SUB 4000
2167 PRINT ''''"Obliczanie wspol
czynnika        SPEARMANA po nac
isnieciu        dowolnego klawis
za!"; PAUSE 0: CLS
2170 PRINT "Ilustracja rang dla
korelacji  kolejnosci  Spearman
a:"'''
2900 LET D=0
3000 FOR n=1 TO k
3002 LET l=k
3004 LET m=k
3010 FOR s=1 TO k
3020 IF a(n)<=a(s) THEN GO TO 30
40
3030 LET l=l-1
3040 IF b(n)<=b(s) THEN GO TO 30
60
3050 LET m=m-1
3060 NEXT s
3080 LET D=(l-m)*(l-m)+D
3085 PRINT "Ranga a(";n;")=";l,"
b(";n;")=";m
3090 NEXT n
3106 LET Sp=0
3110 LET Sp=1-((6*D)/(k*(k↑2-1))
)
3120 PRINT '"Wartosc wspolczynni
ka Spearmana wynosi:"',Sp
3125 GO SUB 4000
3130 PRINT '''"Po nacisnieciu do
wolnego         klawisza program
rozpocznie      prace od nowa!"
3140 PAUSE 0: GO TO 20
4000 BEEP .1,14: BEEP .1,14: BEE
P .2,5: BEEP .1,14: BEEP .1,14:
BEEP .2,5: BEEP .1,14: BEEP .1,1
4: BEEP .1,5: BEEP .1,6: BEEP .2
,7: BEEP .2,5: BEEP .1,15: BEEP
.2,6: BEEP .1,15: BEEP .1,14: BE
EP .2,5: BEEP .1,14: BEEP .1,4:
BEEP .1,15: BEEP .1,14: BEEP .1,
6: BEEP .2,5: BEEP .2,5
6010 RETURN
```

PROGRAM 14

COMMODORE C-64

```
1000 POKE53280,0 POKE53281,0
1001 PRINT"                PROJEKTANT-SPRIT
ES" FORI=0TO39 PRINT"-". NEXT
1002 PRINT"            M          O        "
1003 FORI=1TO21 PRINT"IIIIIIIIIIIIIIIIIIIIIIII
IIIIIIIIII" NEXT
1004 POKE2040,14 V=53248 POKE650,255 FOR
I=896TO960 POKEI,0 NEXT POKEV+21,1 POKEV
+16,1
1005 POKEV,60 POKEV+1,220 X=0 Y=0
1006 PRINT"    "TAB(25)"  OPCJE  " PRIN
T
1007 PRINTTAB(25)"  W PRAWO" PRINTTAB(
25)"  W LEWO" PRINTTAB(25)"  GORA"
1008 PRINTTAB(25)"  DOL"
1009 PRINTTAB(25)"  ZMIANA PUNKTU"
1010 PRINTTAB(25)"  MULTI" PRINTTAB(25)"
  KOLOR 01" PRINTTAB(25)"  KOLOR 11"
1011 PRINTTAB(25)"  KOLOR 10" PRINTTAB
(25)"  SPRITE-ZAPIS"
1012 PRINTTAB(25)"  NORMALNY STOP" PRINT
TAB(25)"  KASOWANIE"
1013 GETA$ IFA$=""THENGOSUB1049 GOTO1013
1014 IFA$="T"ORA$="B"ORA$="F"ORA$="H"ORA
$="G"THEN1020
1015 IFA$="S"THEN1028
1016 IFA$="M"ORA$="1"ORA$="2"ORA$="3"THE
N1044
1017 IFA$="K"THEN1054
1018 IFA$="N"THENPRINT"  " POKEV+21,0 END
1019 GOTO1013
1020 X=X+(A$="F")-(A$="H") Y=Y+(A$="T")-
(A$="B")
1021 IFX>23THENX=23
1022 IFY>20THENY=20
1023 IFX<0THENX=0
1024 IFY<0THENY=0
1025 Z=1144+X+Y*40 ZZ=PEEK(Z) YY=Y*3+INT
(X/8)+896 IFA$<>"G"THEN1013
1026 IFZZ=160THENPOKEZ,76 POKEYY,PEEK(YY
)AND255-(2↑(7-(X-INT(X/8)*8))) GOTO1013
1027 POKEZ,160 POKEYY,PEEK(YY)OR2↑(7-(X-
INT(X/8)*8)) GOTO1013
1028 PRINT"                       "TAB(25)"
NR WIERSZA" PRINTTAB(25) INPUTNR$
1029 PRINT"   " PRINTTAB(25)"
" PRINTTAB(25)"
1030 L=VAL(NR$) IFL<0ORL>999THEN1028
1031 POKEV+28,0 POKEV,255 K=1020 GOSUB10
51 L=895 K=1022 GOSUB1051
1032 DEF FNLI(K)=PEEK(K)*256+PEEK(K+1) P
RINT" "FNLI(1020)"DA", K=1022 FORI=0TO5
1033 GOSUB1052 PRINTPEEK(FNLI(1022))",".
 NEXT GOSUB1052 PRINTPEEK(FNLI(1022))
1034 POKE198,9 POKE631,19 POKE632,13 POK
E633,71 POKE634,111 POKE635,49 POKE636,4
8
1035 POKE637,51 POKE638,54 POKE639,13 EN
D
1036 DEF FNLI(K)=PEEK(K)*256+PEEK(K+1) K
=1020 GOSUB1052
1037 IFFNLI(1020)=1000THENPRINT" PRZEKRO
CZONO ZAKRES NUMERACJI WIERSZY" END
1038 IFFNLI(1022)<949THEN1032
1039 PRINT" "FNLI(1020)"DA", K=1022 FORI
=0TO7
1040 GOSUB1052 PRINTPEEK(FNLI(1022))","
 NEXT GOSUB1052 PRINTPEEK(FNLI(1022))
1041 POKE198,9 POKE631,19 POKE632,13 POK
E633,71 POKE634,111 POKE635,49 POKE636,4
8
1042 POKE637,52 POKE638,51 POKE639,13 EN
D
1043 GOTO1000
1044 IFA$<>"M"THEN1047
1045 IFPEEK(V+28)=1THENPOKEV+28,0 GOTO10
13
1046 POKEV+28,1 GOTO1013
1047 A=VAL(A$) I=PEEK(V+36+A)AND15 I=I+1
 IFI<16THENPOKEV+36+A,IOR240 GOTO1013
1048 POKEV+36+A,IOR240 GOTO1013
1049 Z=1144+X+Y*40 IFPEEK(Z)=160THENPOKE
Z,76 POKEZ,160 POKEZ,76 POKEZ,160 RETURN
1050 POKEZ,160 POKEZ,76 POKEZ,160 POKEZ,
76 RETURN
1051 POKEK,L/256 POKEK+1,L-(INT(L/256)*2
56) RETURN
1052 L=PEEK(K+1)+1 IFL>255THENPOKEK,PEEK
(K)+1 L=0
1053 POKEK+1,L RETURN
```

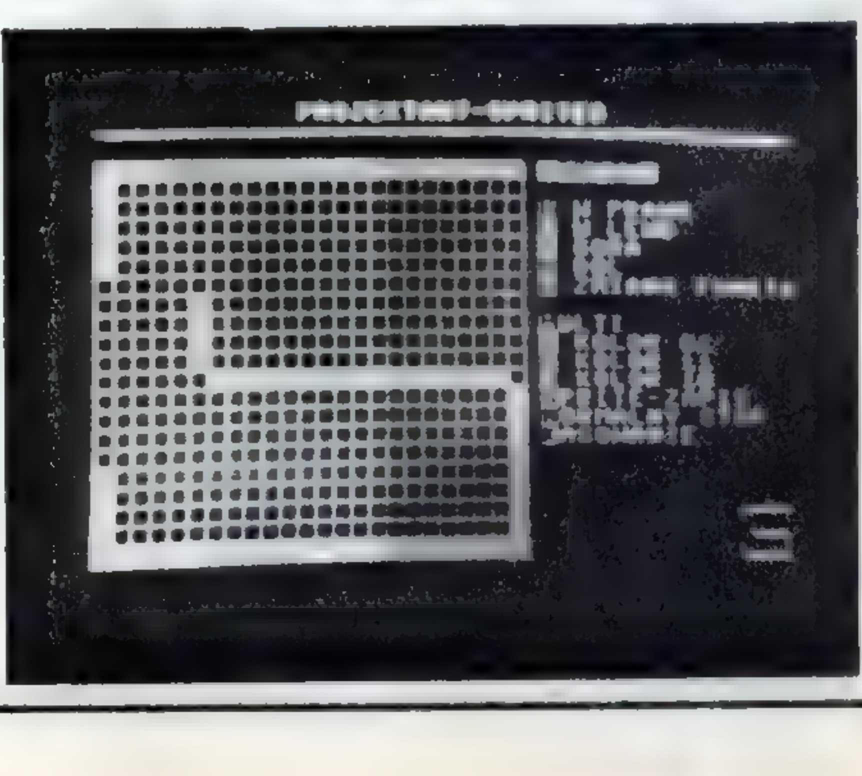
Jest to propozycja na C-64 dla wszystkich, którzy chcą wykorzystać w swoich programach sprite, czyli ruchomy obiekt. Program ten pozwala opisać postać sprite-a na siatce projektowej, równocześnie wyświetlając jego obraz w prawym dolnym rogu ekranu; pozwala również na podstawie utworzonego obrazu sprite wygenerować wiersze programu w Basic-u zawierające jego zapis liczbowy zgodnie z wymaganiami mikrokomputera C-64.

Szczegółowy opis niektórych opcji (pozostałe wyczerpująco wyjaśnione są w programie):

- **M** — wybór rodzaju grafiki sprite-a. Możliwe są dwa rodzaje grafiki: dwukolorowa i wielobarwna. W grafice dwukolorowej jedno okno siatki projektowej odpowiada jednemu punktowi świetlnemu w obrazie sprite-a. W grafice wielobarwnej para okien w poziomie interpretowana jest jako jeden punkt obrazu sprite. Naciskanie klawisza M powoduje przechodzenie od jednego rodzaju grafiki do drugiego.
- **1, 2, 3** — wybór kolorów obrazu sprite. Przy pracy w trybie dwukolorowym naciskając klawisz 2 możemy zmieniać kolor sprite, pozostałe klawisze (1 i 3) nie działają. Przy pracy w trybie wielokolorowym różnym parom okien siatki projektowej odpowiadają różne klawisze i naciskając klawisze 1, 2 lub 3 można zmieniać kolory odpowiednich punktów obrazu sprite.
- **Z** — zapis interpretacji liczbowej zaprojektowanego sprite do pamięci mikrokomputera, jako ciąg kolejnych wierszy programu. Numer pierwszego wiersza jest wprowadzany z klawiatury i nie może być większy od 994. Postać pojedynczego wiersza:
  nr wiersza DATA liczba, liczby, ..., liczba
- **K** — zakończenie pracy programu poprzez usunięcie z pamięci wszystkich jego wierszy w Basic-u. Pozostają tylko wiersze utworzone opcją Z.
- **N** — normalne zakończenie pracy programu (program pozostaje w pamięci).

UWAGA: Podczas uruchamiania programu należy posiadać jego kopię na taśmie lub dysku. Wersja rezydująca w pamięci mikrokomputera w przypadku wystąpienia błędów (powstałych w trakcie przepisywania kodu programu z „IKS"-a) może ulec uszkodzeniu.

```
1054 POKE1020,0
1055 PRINT" "PEEK(1020)+1000; POKE198,9
POKE631,19 POKE632,13 POKE633,71
1056 POKE634,111 POKE635,49 POKE636,48 P
OKE637,53 POKE638,55 POKE639,13 END
1057 POKE1020,PEEK(1020)+1 IFPEEK(1020)<
54THEN1055
1058 PRINT" 1056     1057     1058" POKE
198,4 POKE631,19 POKE632,13 POKE633,13 P
OKE634,13 END
```



PROGRAM 15

COMMODORE C-64

Projektant — znaki

Jest to program użytkowy na C-64 wspomagający projektowanie własnych znaków. Tworzony znak na siatce projektowej jest widoczny w środkowej części ekranu. Znaczenie opcji jest takie jak w programie „Projektant — sprites".

```
1000 POKE53280,0:POKE53281,0
1001 PRINT"     PROJEKTANT-ZNAKI"
 FORI=0TO39:PRINT"-", NEXT
1002 PRINT"    "
1003 FORI=1TO8 PRINT"         " NEXT
1004 POKE2040,14 V=53248 POKE650,255 FOR
I=896TO960 POKEI,0 NEXT POKEV+21,1
1005 POKEV,160 POKEV+1,100 X=0 Y=0
1006 PRINT"    "TAB(25)"  OPCJE  " PRIN
TTAB(15)"ZNAK"
1007 PRINTTAB(25)"  H W PRAWO" PRINTTAB(
25)"  F W LEWO" PRINTTAB(25)"  T GORA"
1008 PRINTTAB(25)"  B DOL"
1009 PRINTTAB(25)"  G ZMIANA PUNKTU"
1010 PRINTTAB(25)"  Z NAK-ZAPIS"
1011 PRINTTAB(25)"  N ORMALNY STOP" PRINT
TAB(25)"  K ASOWANIE"
1012 GETA$ IFA$=""THENGOSUB1036 GOTO1012

1013 IFA$="T"ORA$="B"ORA$="F"ORA$="H"ORA
$="G"THEN1018
1014 IFA$="Z"THEN1027
1015 IFA$="K"THEN1038
1016 IFA$="N"THENPRINT" " POKEV+21,0:END

1017 GOTO1012
1018 X=X+(A$="F")-(A$="H") Y=Y+(A$="T")-
(A$="B")
1020 IFX>7THENX=7
1021 IFY>7THENY=7
1022 IFX<0THENX=0
1023 IFY<0THENY=0
1024 Z=1224+X+Y*40 ZZ=PEEK(Z):VV=Y*3+INT
(X/8)+896 IFA$<>"G"THEN1012
1025 IFZZ=160THENPOKEZ,76 POKEVV,PEEK(VV
)AND255-(2↑(7-(X-INT(X/8)*8))):GOTO1012
1026 POKEZ,160 POKEVV,PEEK(VV)OR2↑(7-(X-
INT(X/8)*8)) GOTO1012

1027 PRINT"                    "TAB(25)"NR
WIERSZA" PRINTTAB(25) INPUTNR$
1028 PRINT"   " PRINTTAB(25)"
 " PRINTTAB(25)"          "
1029 L=VAL(NR$) IFL<0ORL>999THEN1027
1030 POKEV+28,0 POKEV,255
1031 PRINT" "L"D◆", FORI=896TO914STEP3
1032 PRINTPEEK(I)" ,", NEXT PRINTPEEK(91
7)
1033 POKE198,9 POKE631,19 POKE632,13 POK
E633,71 POKE634,111 POKE635,49 POKE636,4
8
1034 POKE637,51 POKE638,53 POKE639,13 EN
D
1035 GOTO1000
1036 Z=1224+X+Y*40 IFPEEK(Z)=160THENPOKE
Z,76 POKEZ,160 POKEZ,76 POKEZ,160 RETURN

1037 POKEZ,160 POKEZ,76 POKEZ,160:POKEZ,
76 RETURN

1038 POKE1020,0
1039 PRINT" "PEEK(1020)+1000; POKE198,9
POKE631,19 POKE632,13 POKE633,71
1040 POKE634,111 POKE635,49 POKE636,48 P
OKE637,52 POKE638,49 POKE639,13 END
1041 POKE1020,PEEK(1020)+1 IFPEEK(1020)<
40THEN1039
1042 PRINT" 1040     1041     1042" POKE
198,4 POKE631,19 POKE632,13 POKE633,13 P
OKE634,13 END
```



LOGO

LOGO to język programowania, zasłużenie zdobywający sobie coraz większą popularność wśród użytkowników mikrokomputerów. Wszyscy interesujący się programowaniem słyszeli z pewnością o LOGO, a wielu podejmuje próby pisania programów w tym języku. LOGO został stworzony w latach siedemdziesiątych przez zespół kierowany przez Seymoura Paperta w Massachusetts Institute of Technology. Zaprojektowano go z myślą o początkujących programistach. LOGO doskonale nadaje się do celów edukacyjnych, a zarazem jest językiem o dużej sile (powstał na bazie LISP-u) i można w nim łatwo programować naprawdę ciekawe i skomplikowane problemy. Pozwala również na uzyskanie ciekawych efektów programistycznych już po krótkiej nauce. Wyrabia też właściwe nawyki na przyszłość.

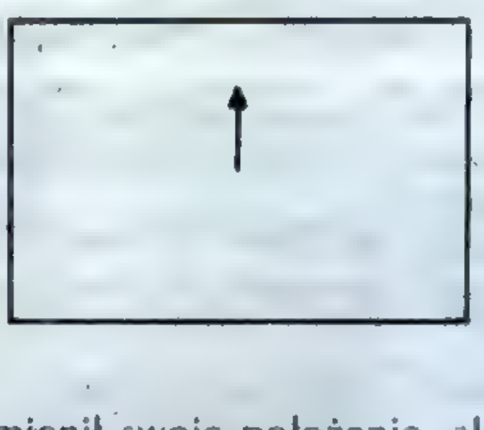
Można w nim w zwięzły i naturalny sposób zaprogramować wiele problemów, a jasna diagnostyka błędów i wbudowane w język komendy edytora pozwalają szybko znaleźć i usunąć pomyłki. Po prostu większość żmudnej pracy wykonuje za nas komputer. Człowiek koncentruje się tylko na celowym wysiłku intelektualnym.

Znane dotychczas w Polsce wersje języka LOGO na różne mikrokomputery są wersjami angielskimi — to znaczy słowa kluczowe języka i komunikaty o błędach mają brzmienie angielskie. Jest to nie lada przeszkoda dla osób, które nie znają angielskiego, a głównie dla dzieci i młodzieży. Dopiero LOGO w wersji narodowej nabiera pełnej wartości. Opracowano już standard polskiego LOGO, a niebawem będzie rozpowszechniany jego translator na ZX Spectrum — najpopularniejszy obecnie w Polsce komputer osobisty. Dlatego też, rozpoczynając w naszym piśmie kurs LOGO, chcemy od razu operować polskim słownictwem. Naturalnie obok polskich terminów będziemy podawali ich angielskie odpowiedniki, by ci, którzy dysponują angielskim translatorem, mogli sprawdzać na komputerze swoje programy. W szczegółach będziemy opierać się na wersji języka dla mikrokomputera ZX Spectrum.

LOGO zawiera pewien zasób słów, które są dla niego zrozumiałe. Są to tak zwane słowa kluczowe. Posługując się tym zbiorem słów można pisać programy, które rysują na ekranie, manipulują słowami i listami, dokonują obliczeń. Liczbę słów rozumianych przez LOGO można zwiększyć, definiując nowe pojęcia przy użyciu już znanych — pisząc procedury o określonych nazwach.

Rysunki wykonuje w LOGO żółw. Jest to abstrakcyjny obiekt, wyobrażony na ekranie komputera w postaci małego trójkąta. Żółwiowi można wydawać różne rozkazy, każąc mu poruszać się po ekranie. Żółw rysuje ciągnąc za sobą kreskę. Aktualny stan żółwia opisują: jego pozycja na ekranie określona w układzie współrzędnych, którego środek pokrywa się ze środkiem ekranu, i kierunek, w którym jest zwrócona głowa żółwia. Żółw może być widoczny albo nie. Gdy napiszemy PŻ (od pokaż żółwia) ang. SHOWTURTLE i odpowiednio skrót ST — trójkącik pojawi się na ekranie. Komenda SŻ „schowaj żółwia", ang. HIDETURTLE HT, sprawi, że zniknie. Na początku pracy z LOGO komenda PŻ spowoduje, że trójkąt pokaże się na środku ekranu i będzie skierowany do góry. Jest to początkowe położenie żółwia. Ma on wtedy pozycję (0, 0) w układzie współrzędnych i przypisany kąt 0 stopni. Jeśli kiedykolwiek później będziemy chcieli, by żółw znalazł się znów w tym miejscu, wystarczy podać komendę WRÓĆ (HOME).

NAPRZÓD (NP FORWARD FD) to komenda przesuwająca żółwia do przodu. Trzeba określić jak daleko ma on się posunąć. Dlatego też, po słowie NAPRZÓD, podajemy parametr. Na przykład NAPRZÓD 30 przesunie żółwia o 30 kroków w kierunku, w jakim jest on ustawiony:



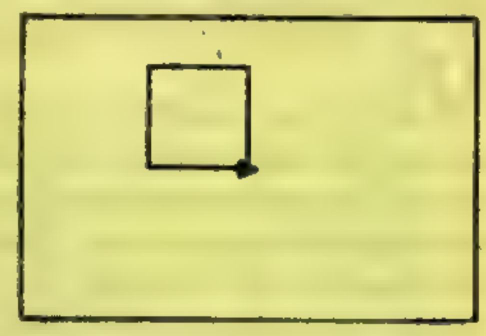
Żółw zmienił swoje położenie, ale w dalszym ciągu „patrzy" w tę samą stronę — w górę. Gdy chcemy, aby żółw skręcił w bok, musimy kazać mu się obrócić w lewo bądź w prawo. Robimy to przy pomocy komend PRAWO (PW RIGHT RT) i LEWO (LW LEFT LT), których parametrami są liczby stopni, o jakie żółw powinien się obrócić. LEWO 90 zmieni kierunek żółwia tak:



Posługując się poznanymi komendami, możemy już rysować różne figury geometryczne. Zanim to zrobimy, warto przygotować sobie ekran do rysowania — zmazać to, co było na nim poprzednio narysowane. Robimy to podając komendę CZYŚĆ (CS CLEARSCREEN CS). Po wykonaniu tej komendy żółw wraca do położenia początkowego, pośrodku ekranu. Gdybyśmy chcieli wyczyścić ekran, nie zmieniając pozycji żółwia, musimy podać komendę ZMAŻ (CLEAN). Jak łatwo zauważyć, napisanie kolejno WRÓĆ i ZMAŻ daje ten sam efekt co pojedyncza komenda CZYŚĆ.

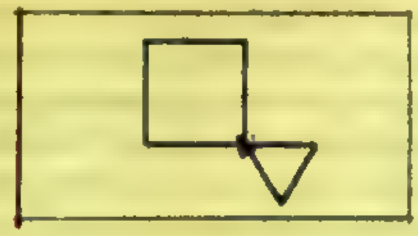
Na czystym ekranie narysujmy kwadrat:

NAPRZÓD 50
LEWO 90
NAPRZÓD 50
LEWO 90
NAPRZÓD 50
LEWO 90
NAPRZÓD 50



Jeśli teraz narysujemy trójkąt równoboczny, pisząc kolejno komendy:
NAPRZÓD 30
PRAWO 120
NAPRZÓD 30
PRAWO 120
NAPRZÓD 30

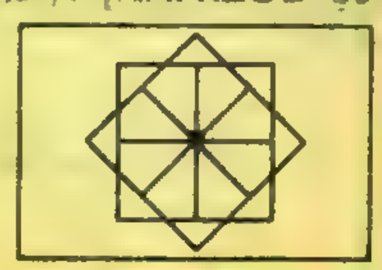
uzyskamy taki efekt na ekranie:



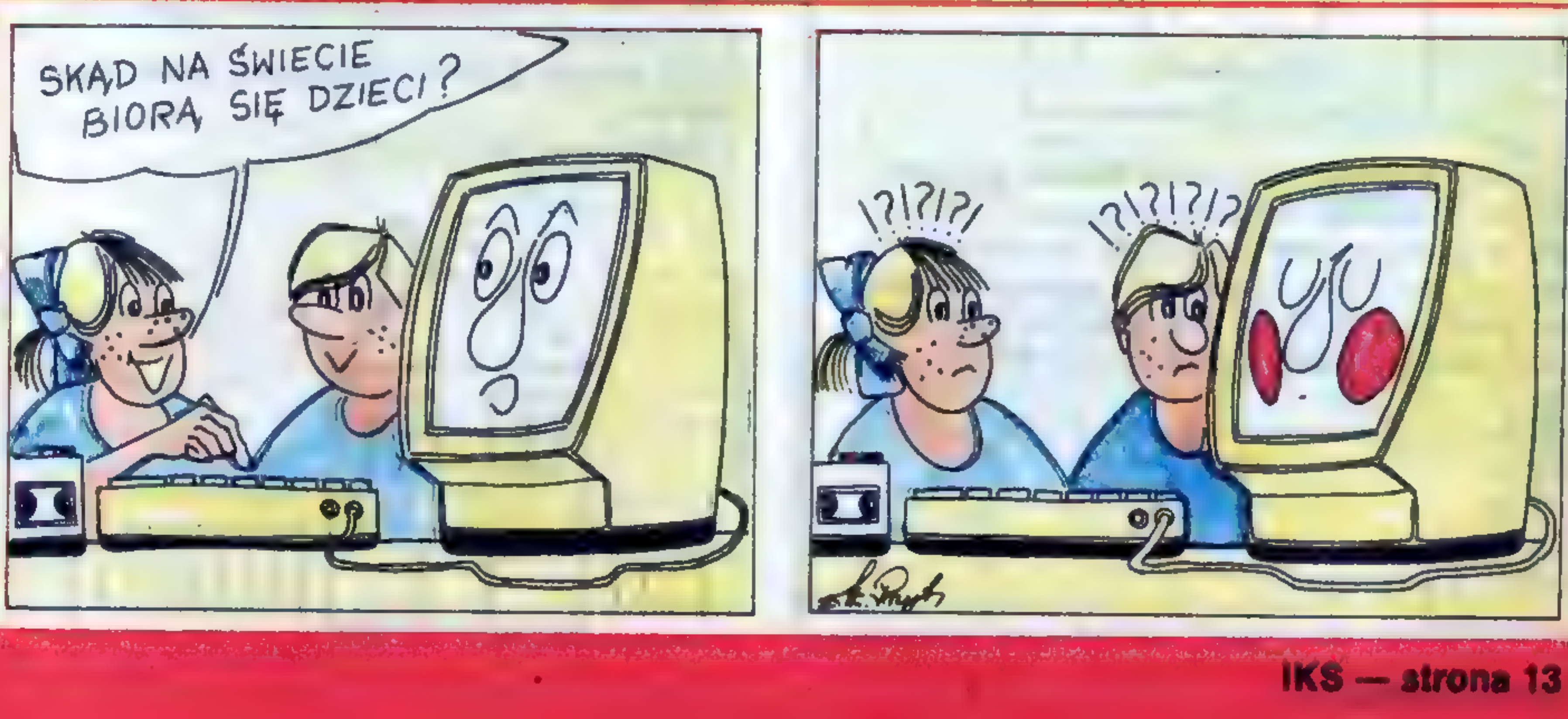
Przy rysowaniu kwadratu i trójkąta powtarzaliśmy kilkakrotnie te same komendy z tymi samymi parametrami. Można zapisać to krócej, używając komendy POWTÓRZ (REPEAT). Po słowie POWTÓRZ podajemy, ile razy ma nastąpić powtórzenie, a dalej listę instrukcji do wielokrotnego wykonania, ujętą w nawiasy kwadratowe. Kwadrat możemy zatem narysować w ten sposób (posługując się dla jeszcze większego skrótu skróconymi nazwami komend):
POWTÓRZ 4 [NP 50 LW 90]

Zauważmy, że po wykonaniu tej instrukcji żółw będzie skierowany do góry, a nie w prawo, tak jak przy poprzednim rysowaniu kwadratu, gdzie po ostatnim NAPRZÓD nie podaliśmy już komendy LEWO 90. Najlepiej jest przyjąć jako zasadę, że po narysowaniu figury żółw znajduje się w takiej pozycji, jak przed rozpoczęciem rysowania. Zaoszczędzi nam to czasu na zastanawianie się, jak jest ustawiony żółw i błędów wynikających z niewłaściwego określenia jego pozycji.

Spróbujmy teraz wykonać nieco bardziej skomplikowany rysunek: rozetę złożoną z kwadratów, której środek będzie się znajdował pośrodku ekranu.
POWTÓRZ 8 [POWTÓRZ 4 [NAPRZÓD 50 LEWO 90] LEWO 45]
Uzyskamy taki rysunek:



dokończenie na str. 14



dokończenie ze str. 13

Chcemy teraz z rozety zrobić gwiazdkę — usunąć linie rozchodzące się promieniście ze środka ekranu. Ale jak zmazywć kreski?

Otóż żółw może chodzić po ekranie, rysując kreski, może zmieniać swoją pozycję, nie pozostawiając żadnego śladu i może też idąc po kresce ścierać ją, czyli zachowywać się jak gumka. Możemy sobie wyobrazić, że żółw ma pióro, którym rysuje po ekranie. Jeśli trzyma to pióro uniesione w górę, idzie, ale nie zostawia śladu. Jeśli je opuści, rysuje kreski, a jeśli zamieni pióro na gumkę,ściera je. Odpowiednie komendy określają, co ma robić żółw ze swoim piórem.

POD (od podnieś, ang. PENUP i skrót PU) podnosi pióro. Od chwili wydania tej komendy żółw przemieszczając się nic nie rysuje.

OPU (od opuść ang. PENDOWN skrót PD) opuszcza pióro — żółw zaczyna rysować.

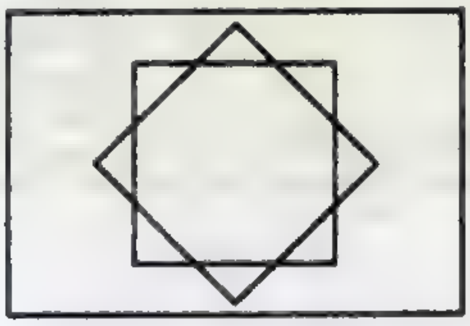
ŚCIERANIE (ang. PENERASE skrót PE) — sprawia, że żółw zmazuje kreski, które napotyka na swej drodze.

Dla narysowania gwiazdki przyda się także komenda cofająca żółwia, tak jak gdyby szedł on tyłem. Rozkaz WSTECZ (skrót WS, ang. BACK i skrót BK) cofa żółwia z powrotem, nie zmieniając kierunku, w jakim jest on ustawiony. Żółw wraca tyle kroków, ile podaje parametr (tak jak w komendzie NAPRZÓD).

Następująca sekwencja instrukcji przekształci naszą rozetkę w gwiazdkę:

ŚCIERANIE
POWTÓRZ 8 [NAPRZÓD 50 WSTECZ 50 LEWO 45]
OPU

Ostatnia instrukcja przygotowuje żółwia do dalszego rysowania. Na ekranie uzyskamy taki efekt:



Łatwo możemy narysować kontur gwiazdki od razu bez żadnych linii wewnątrz. Wystarczy tylko zastanowić się, o jakie kąty powinien obracać się żółw. Ustawmy najpierw żółwia we właściwej pozycji — rysowanie gwiazdki zaczniemy od lewego górnego ramienia.

CS
POD
NP 50
LW 90
NP 20
OPU

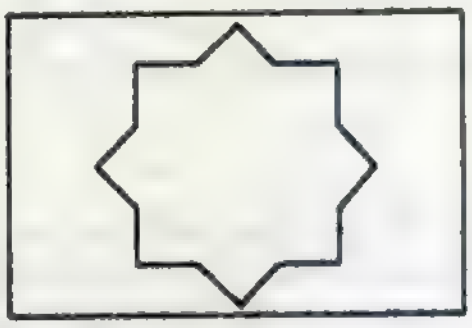
i rysujemy:

POWTÓRZ 8 [NP 30 LW 90 NP 30 PW 45]

Wracamy żółwiem na środek ekranu:

POD
WRÓĆ
OPU

Rysunek wygląda tak:



Bardzo ciekawe efekty można uzyskiwać, rysując proste figury wielokrotnie w różnych miejscach ekranu. Na przykład naszą ośmioramienną gwiazdkę można powtórzyć 36 razy, obracając ją za każdym razem o 10 stopni. Zaczynamy od środka ekranu i rysujemy przy użyciu takich instrukcji:

POWTÓRZ 36 [POWTÓRZ 8 [NP 20 LW 90 NP 20 PW 45] LW 10]

Czytelnicy, którzy mają komputer i translator LOGO, niech sprawdzą ten program w angielskiej wersji:

REPEAT 36 [REPEAT 8 [FD 20 LT 90 FD 20 RT 45] LT 10]

Dotychczas poznane instrukcje graficzne pozwalają narysować dowolną figurę. Naturalnie zaprogramowanie złożonych rysunków wymaga dużego nakładu pracy. Często można zmniejszyć ten wysiłek korzystając z innych instrukcji graficznych LOGO, które wykorzystują współrzędne punktów w układzie o środku umieszczonym pośrodku ekranu. Podziałkę na osiach układu można ustalać dowolnie — pozwala to na łatwe rysowanie spłaszczonych lub wydłużonych figur. O tym wszystkim będzie mowa w następnych odcinkach.

A teraz zadania do samodzielnego zaprogramowania:

1) Narysować gwiazdy sześcio- i dwunastoramienną

2) Narysować taki domek:



Nie pisaliśmy nic na temat błędów, jakie możemy popełniać przy naszej konwersacji z LOGO. Każdy kto ma w tej chwili komputer przed sobą (a tak by było najlepiej) widzi znak „?" i migający kursor. Znaczy to, że LOGO gotowe jest wykonywać nasze rozkazy (komendy lub procedury). Prawie na pewno przynajmniej raz pomylimy się i napiszemy np. „LLW" zamiast „LW". LOGO reaguje wtedy zupełnie naturalnie — pisze na ekranie „Nie wiem, jak zrobić LLW". Informuje nas, że nie rozumie komendy (popełniliśmy błąd) i czyni to w sposób najzupełniej dla nas zrozumiały (angielskie LOGO sygnalizuje — „I dońt know how to..."). Drugi rodzaj błędu, jaki mogliśmy do tej pory popełnić, to napisanie dla przykładu „NP 30 20...". W takim wypadku LOGO oczywiście wykonuje komendę NAPRZÓD 30 i dalej pisze: „Nie powiedziałeś co zrobić z 20". Ta reakcja LOGO nie wymaga chyba komentarza. Wreszcie mogliśmy napisać „NP LW 30", co nie ma sensu, gdyż wprawdzie wiadomo, że trzeba obrócić się o 30 stopni w lewo ale nie wiadomo ile kroków iść do przodu. No i LOGO zasygnalizuje: „LW nie daje wyniku potrzebnego NP". Gdy natomiast napiszemy: „NP" i wciśniemy ENTER, zobaczymy: „za mało danych dla NP". Właściwie wszystkie komunikaty o błędach są tak oczywiste, że nie ma sensu ich wyjaśniać — wiadomo, że masło jest maślane. Odstępując od tej zasady dodajmy, że po napisaniu np. „POWTÓRZ 4 5" będzie: „POWTÓRZ nie chce 5 jako danej" (bo drugim parametrem komendy POWTÓRZ musi być lista ujęta w nawiasy kwadratowe), a po napisaniu „POWTÓRZ 10 NP 20" zobaczymy „NP nie daje wyniku potrzebnego powtórz" (bo nie daje listy — powód ten sam). Na koniec zauważmy jeszcze, że jeśli coś rysujemy, na napisy (konwersacja z komputerem) przeznaczone są tylko dwie linie u dołu ekranu. Zdarza się wtedy czasem, że komputerowi brakuje miejsca na tekst i wówczas w prawym dolnym rogu miga pionowa strzałka, co oznacza: przeczytaj to, co napisałem, ale to jeszcze nie koniec — wciśnij coś, to zobaczysz resztę. Wszystkie te praktyczne wskazówki są tak naprawdę w tym miejscu niepotrzebne. Wystarczy dłuższa chwila samodzielnej zabawy (na razie zabawy) by wszystko stało się oczywiste. LOGO zostało zaprojektowane tak, by samodzielnie uczyć się i ograniczyć do minimum kontakt z nauczycielem. Dlatego też nauka LOGO ma sens, kiedy mamy rzeczywisty kontakt z komputerem, nie tylko kartkę i ołówek. Optymistyczny jest fakt, że jak na razie (wynika to ze statystyk) liczba tych drugich przypadków systematycznie maleje.

Uwagi o polskim translatorze LOGO na ZX Spectrum

Standard polskiego LOGO (brzmienie słów kluczowych i komunikatów) został opracowany przez zespół pracujący pod egidą Polskiego Towarzystwa Informatycznego. Zgodnie z tym standardem przygotowano translator LOGO na ZX Spectrum. Polski translator (jest to tzw. interpreter) jest już gotowy i będzie niebawem (albo już jest, gdy niniejszy tekst ukaże się w druku) rozpowszechniany w szkołach oraz wśród prywatnych użytkowników mikrokomputerów. Jest on obiektywnie lepszy niż angielski — na którego bazie powstał (firmy software'owej SOLI—LCSI), gdyż jego autorzy oprócz spolszczenia słów kluczowych i komunikatów usunęli kilka poważnych błędów, które powodowały nieprawidłowe działanie paru instrukcji. Wprowadzono możliwość posługiwania się literami charakterystycznymi dla języka polskiego (ą, ę, ł itp.), które mogą być używane w nazwach określanych przez użytkownika, a w słowach kluczowych np. powtórz, są wręcz niezbędne (wszak ma to być po polsku). Litery te otrzymuje się na ekranie po wciśnięciu klawisza „graph" i odpowiedniego klawisza litery: np. ą to „graph" i a, ł to „grahp" i l itd. Ponadto polskie LOGO ma nowe (nie istniejące w angielskiej wersji) atrakcyjne komendy i operacje. Są to m.in. funkcje rzeczywiste ln i exp oraz np. komenda graficzna ZAMALUJ (ZAM) pozwalająca kolorować powierzchnie rysunków. O tej ostatniej komendzie — w następnym odcinku, gdzie również spróbujemy napisać pierwsze programy — czyli nauczymy LOGO rozumieć nowe słowa — zdefiniujemy procedury.

SIS



Rys. Michał Przybyłowski





Mikroprocesory odmierzyły kolejny krok w rozwoju techniki. W chwili obecnej nie ma już praktycznie takiej dziedziny życia, w której wprost lub pośrednio nie byłyby one stosowane. Są wszędzie tam, gdzie mowa jest o elektronicznych, twórczych rozwiązaniach.

Wszystko, co nowe, co stwarza szansę rozwoju — w wojsku preferuje się szczególnie. W nim, jak w soczewce zbiegają się wszystkie nici postępu. Każda nowatorska myśl jest starannie analizowana, rozwijana, wartościowe projekty są szybko wdrażane — przede wszystkim z myślą o sprawności, skuteczności i niezawodności sił zbrojnych.

Współczesne konstrukcje sprzętu bojowego, wykorzystywanego na lądzie, morzu i w powietrzu oparte są na najnowszych zdobyczach techniki. Prym wiedzie — co zrozumiałe — elektronika, wymagająca od żołnierzy obsługujących ów sprzęt, bezpośrednich użytkowników, odpowiedniego stopnia wtajemniczenia. Budowy i zasad działania tych skomplikowanych urządzeń uczyć ich musi kadra zawodowa, wysokiej klasy specjaliści. Dziś przyszli dowódcy zdobywają wiedzę w nowoczesnych, doskonale wyposażonych pracowniach i labolatoriach. A potem praktyka: intensywne poligonowe ćwiczenia.



















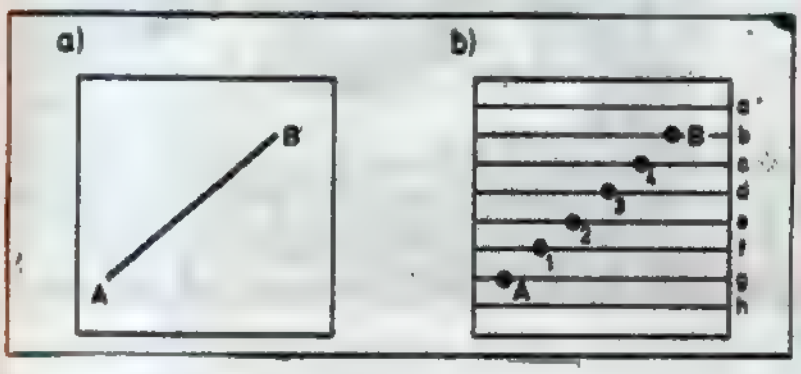
GRAFIKA KOMPUTEROWA

Jerzy Chmurzyński

Program graficzny

Program graficzny jest to ciąg kodów sterujących pracą procesora graficznego w zakresie kreślenia obrazu. Program graficzny generowany jest w komputerze i przesyłany do pamięci obrazu grafoskopu. W zależności od przyjętej metody kreślenia obrazu na ekranie lampy elektronopromieniowej różne są postacie programów graficznych.

W przypadku metody kreślenia konturowego program graficzny złożony jest z ciągu rozkazów opisujących pozycjonowanie i przemieszczanie strumienia elektronów oraz sterujących jasnością i kolorem świecenia plamki świetlnej na ekranie lampy. Rozkazy te są w procesorze graficznym dekodowane i wykorzystane w układach odchylania i podświetlania do bezpośredniego sterowania strumieniem elektronów. Przykładowy program graficzny kreślenia odcinka AB z rys. 1a przedstawiono poniżej:




```
POSITION (XA, YA)
VECTOR (XB,YB)
```

Rozmieszczenie programu graficznego w pamięci obrazu grafoskopów kreślących metodą konturową nie jest orientowane geometrycznie względem wymiaru rastrowego ekranu. Orientacja geometryczna następuje w końcowej fazie kreślenia obrazu, to znaczy w układach odchylania pionowego i poziomego, gdzie wykorzystuje się przechowywane w pamięci obrazu argumenty rozkazów programu graficznego, np. współrzędne XA, YA, XB, YB. Inaczej zagadnienie to przedstawia się w grafoskopach kreślących metodą rastrową z wybieraniem liniowym. W tym przypadku rozmieszczenie programu graficznego w pamięci obrazu jest jednoznacznie zorientowane geometrycznie względem wymiaru rastrowego ekranu. Przedstawia to rys. 3.

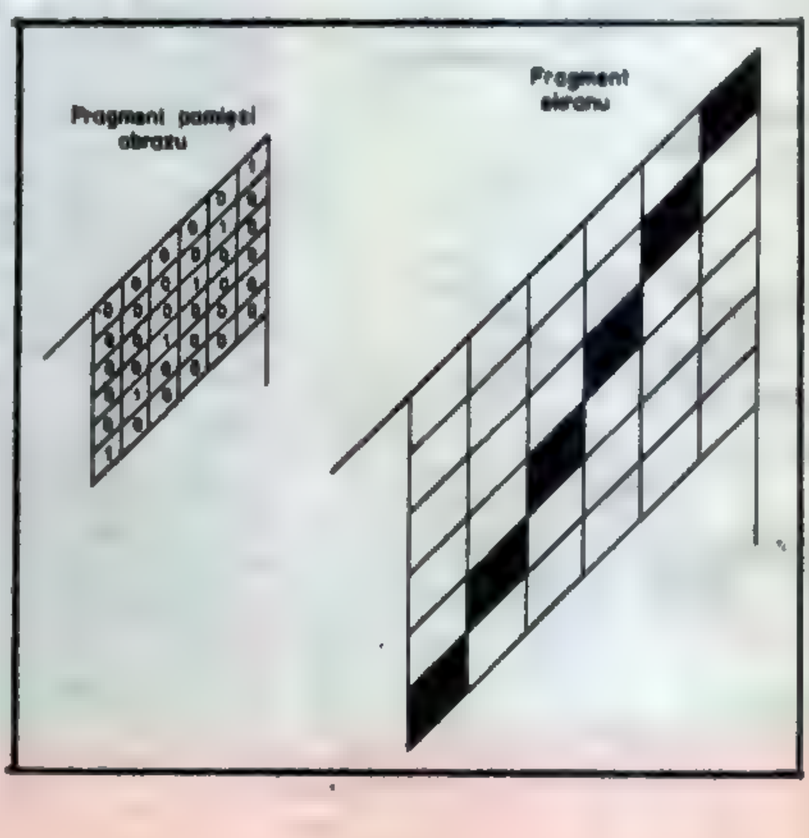
Program graficzny złożony jest z kodów określających rodzaj wyświetlanej informacji dla każdego adresowalnego punktu ekranu. Kody te uporządkowane są w kolejności linii wybieranych przez układ odchylania pionowego strumienia elektronów. W ramach każdej linii uporządkowanie kodów jest zgodne z kolejnością punktów wybieranych przez układ odchylania poziomego. Kod wyświetlanej informacji w najprostszym przypadku wartością 1 oznacza żądanie wyświetlenia punktu na ekranie, natomiast wartością 0 — zakaz wyświetlania. Kody zero — jedynkowe przesyłane są wierszami do układu sterującego podświetlaniem plamki świetlnej, synchronicznie z przemieszczaniem się strumienia elektronów wzdłuż wybranej linii poziomej. W punktach oznaczonych kodem 1 nastąpi rozjaśnienie plamki, natomiast w punktach oznaczonych kodem 0 — obraz nie będzie wyświetlany. Pamięć obrazu zawiera zatem dokładne odwzorowanie obrazu wyświetlanego na ekranie monitora. Pojedynczy płat pamięci może odwzorowywać obraz kreślony w jednym kolorze. Do wysterowania monitora kolorowego, dla każdego z trzech podstawowych kolorów potrzebne są oddzielne płaty pamięci. Kody z każdego płatu pamięci kierowane są synchronicznie do oddzielnych układów sterujących podświetlaniem plamek świetlnych w kolorze czerwonym, zielonym i niebieskim.

Wyświetlanie obrazu na ekranie monitora z wybieraniem liniowym wymaga wprowadzenia z komputera do pamięci obrazu grafoskopu programu graficznego w przedstawionej powyżej postaci. Zadaniem procedury generującej programy graficzne jest wyznaczanie współrzędnych punktów aproksymujących elementy składowe obrazu, tj. odcinki linii prostych, okręgi, łuki itp. Przykład procedury generującej program graficzny odcinka AB z rys. 1b przedstawiono poniżej.

```
100 SET (XA, YA)
110 SET (X1, Y1)
120 SET (X2, Y2)
130 SET (X3, Y3)
140 SET (X4, Y4)
150 SET (XB, YB)
160 RETURN
```

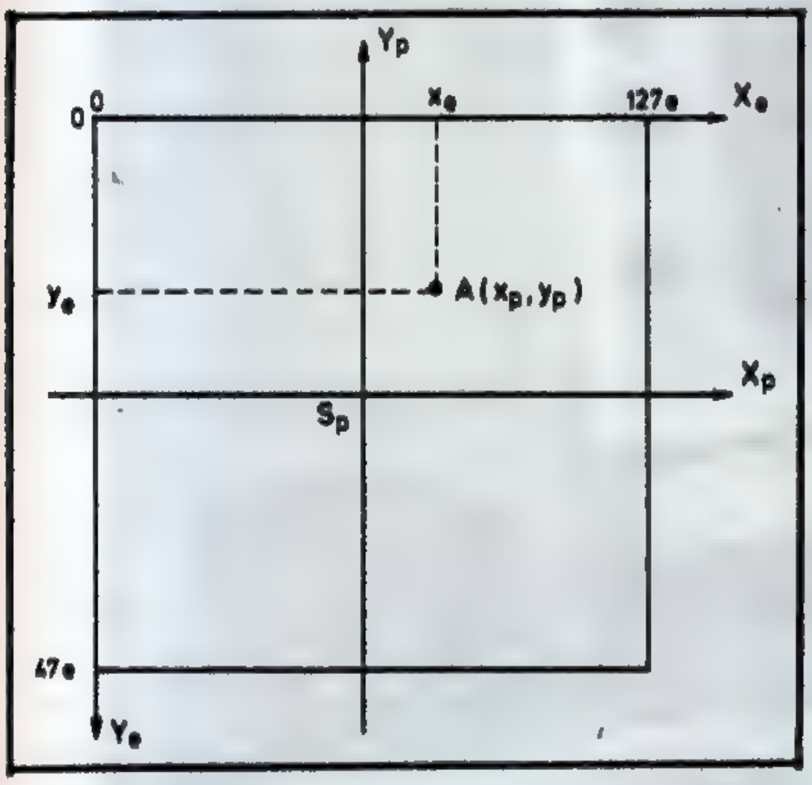
Procedura ta zapisana została w języku BASIC dla mikrokomputera MERITUM I. Instrukcja SET (X, Y) wyświetla punkt na ekranie w miejscu określonym współrzędnymi X, Y. Początek układu współrzędnych tego mikrokomputera znajduje się w lewym górnym rogu ekranu.

Rys. 3



Zmiana układu współrzędnych

Położenie środka układu współrzędnych prostokątnych dla danego monitora ekranowego określane jest na poziomie oprogramowania systemowego. Położenie to może być zatem zmieniane programowo na poziomie oprogramowania systemowego lub użytkowego. Na rys. 4 przedstawiono ekranowy układ współrzędnych prostokątnych X_e, Y_e.



Rys. 4

Środek S_p przykładowego, programowo zdefiniowanego układu współrzędnych X_p, Y_p znajduje się w punkcie o współrzędnych (64e, 24e) ekranowego układu współrzędnych. Aby wyświetlić na ekranie monitora dowolny punkt A o współrzędnych (x_p, y_p) należy dokonać następującego prostego przeliczenia współrzędnych prostokątnych tego punktu z układu X_p, Y_p na układ X_e, Y_e:

$$x = 64 + x_p \qquad y_e = 24 - y_p$$

Stąd wyświetlenie punktu $A(x_p,y_p)$ następuje zgodnie z instrukcją SET(64 + XP, 24 — YP).

Zobrazowanie odcinków

Odcinki linii prostych należą do podstawowych składników obrazów. Zobrazowanie odcinka metodą rastrową z wybieraniem liniowym wymaga wyznaczenia współrzędnych zbioru punktów aproksymujących dany odcinek. W grafoskopach, które nie mają procesów graficznych z możliwościami generowania punktów aproksymujących, należy to wykonywać programowo. Poniżej przedstawiona jest procedura zobrazowania odcinka ukośnego AB z rys. 1b.

```
300 D = (YB — YA) / (XB — XA)
310 FOR X = XA TO XB
320 Y = YA + D * (X — XA)
330 SET (X, Y)
340 NEXT X
350 RETURN
```

Transformacje obrazów na płaszczyźnie

Położenie punktu P na płaszczyźnie określamy za pomocą współrzędnych (x,y). Współrzędne te w komputerze mogą być pamiętane jako elementy macierzy [x,y]. Podstawowe transformacje obrazów na płaszczyźnie obejmują następujące operacje:

— zmiana skali obrazu,
— przesunięcie (translacja) obrazu,
— obrót obrazu wokół środka układu współrzędnych,
— obrót obrazu wokół dowolnego punktu.

W wyniku dokonania transformacji punktu o współrzędnych [x,y] otrzymujemy współrzędne transformowane [x',y'] tego punktu. Transformacje odcinków sprowadzają się do transformacji punktów początkowych i końcowych tych odcinków.

Transformacje obrazów na płaszczyźnie dokonywane są w przestrzeni trójwymiarowej, z wykorzystaniem współrzędnych jednorodnych. Oznacza to przedstawienie macierzy współrzędnych punktu [x,y] za pomocą macierzy [kx, ky, k] dla $k \neq 0$. Współczynniki transformacji zebrane są w macierzy transformacji T o wymiarach 3 x 3:

$$T = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & s \end{bmatrix}$$

Poszczególne elementy macierzy T dokonują następujących transformacji składowych:

a — zmiana skali współrzędnej X,
b, c — obrót i skręcenie względem środka układu,
d — zmiana skali współrzędnej Y,
e — przesunięcie względem osi X,
f — przesunięcie względem osi Y,
s — zmiana skali współrzędnej X i Y

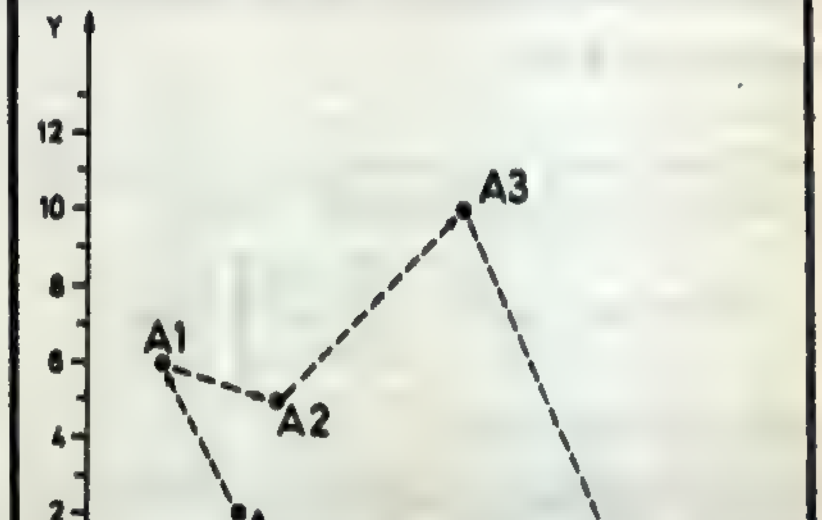
Współrzędne transformowane [x', y']punktu [x,y] otrzymuje się w wyniku wymnożenia współrzędnych jednorodnych punktu [x,y,1] przez macierz transformacji T:

$$[X, Y, K] = [x, y, 1] \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & s \end{bmatrix}$$

i przekształcenie transformowanych współrzędnych jednorodnych do postaci:

$$x' = \frac{X}{K} \qquad y' = \frac{Y}{K}$$

Rozważmy następujący ciąg transformacji punktu A (4,2) (rys. 5)



Rys. 5

dokończenie na str. 20



dokończenie ze str. 19

W wyniku zmiany skali współrzędnych X (dwukrotne zmniejszenie) i Y (trzykrotne zwiększenie)

$$[X1, Y1, K1] = [4\ 2\ 1] \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [2\ 6\ 1]$$

otrzymamy punkt A1 (2,6).

W wyniku przesunięcia punktu A1 wzdłuż osi X o 3 jednostki, a wzdłuż osi Y o — 1 jednostkę:

$$[X2, Y2, K2] = [2\ 6\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & -1 & 1 \end{bmatrix} = [5\ 5\ 1]$$

otrzymamy punkt A2 (5,5).

Po dokonaniu dwukrotnego zwiększenia skali współrzędnych X i Y punktu A2 (5,5):

$$[X3, Y3, K3] = [5\ 5\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} = [5\ 5\ \tfrac{1}{2}]$$

otrzymamy punkt A3 (10,10).

Aby dokonać obrotu punktu A3 wokół środka układu o kąt 45°, zgodnie z ruchem wskazówek zegara, należy dokonać następującej transformacji:

$$[X4, Y4, K4] = [10\ 10\ 1]$$

$$\begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} = [14.1\ 0\ 1]$$

W wyniku otrzymamy punkt A4 (14.1,0).

Poniżej przedstawiono w języku BASIC procedurę wyznaczania współrzędnych transformowanych punktu na płaszczyźnie. Założono, że współrzędne punktu transformowanego zawarte są w tablicy A, współrzędne transformowane w tablicy A1, zaś współczynniki macierzy transformacji zawarte są w tablicy T.

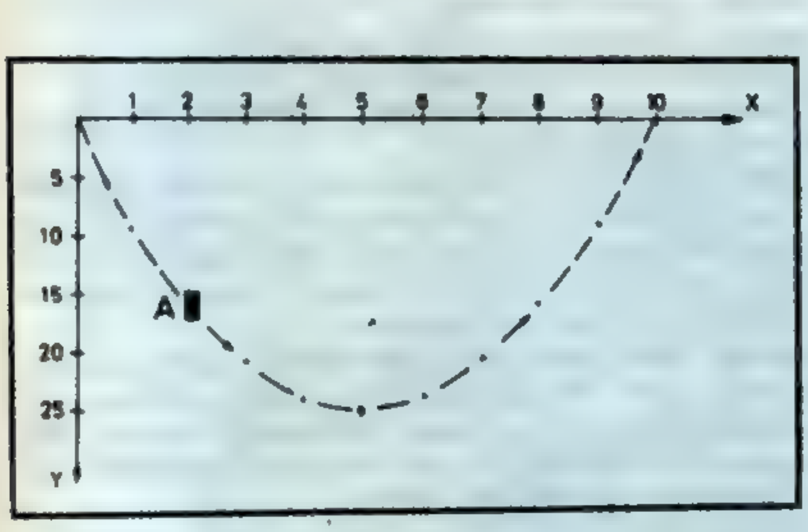
```
 10 DIM A (2)
 20 DIM A1 (2)
 30 DIM T (2,2)
370 A1 (0) = 0
380 A1 (1) = 0
390 A1 (2) = 0
400 FOR I = 0 TO 2
410 FOR J = 0 TO 2
420 A1 (I) = A1 (I) + A (J) * T (J,I)
430 NEXT J
440 NEXT I
450 A1 (0) = A1 (0) / A1 (2)
460 A1 (1) = A1 (1) / A1 (2)
470 RETURN
```

Obszerne informacje na temat transformowania obrazów Czytelnik znajdzie w pracy T. Parlielisa: Algorytmy dla grafiki i przetwarzania obrazów WNT. Warszawa 1986 r.

Animacja komputerowa

Transformacje obrazów stają się bezpośrednio użyteczne podczas badań symulacyjnych projektowanych obiektów. Animacja komputerowa pozwala śledzić na ekranie monitora przebieg określonego procesu w różnej skali czasu. Badania takie mogą być poświęcone np. ocenię skutków zderzenia samolotu z przeszkodą, ocenie metody lądowania pojazdu na Księżycu, szacowaniu skutków wybuchu jądrowego itp. Ruch obrazu na ekranie monitora można uzyskać poprzez wygaszenie i wyświetlenie kadrów kolejnych faz ruchu z częstotliwością ok. 25 Hz. Kadry wyświetlane na ekranie mogą być filmowane i wykorzystane następnie do produkcji filmów animowanych. Na ekranie monitora filmy takie mogą być wielokrotnie powtarzane dla różnych parametrów badanego procesu. Animacja komputerowa w istotny sposób przyspiesza proces projektowania. Ruchomy obraz modelujący badany proces ułatwia projektantom wybór parametrów optymalnych dla danej konstrukcji czy procesu.

Poniżej przedstawiono procedurę przemieszczania punktu A (rys. 6) po paraboli $y = -x^2 + 10 \cdot x$ w przedziale $x = [0,10]$.



Rys. 6

```
500 FOR X = 0 TO 10
510 Y = 10 * X — X * X
520 SET (X,Y)
530 GOSUB 600
540 RESET (X,Y)
550 NEXT X
560 RETURN
600 P = 2
610 FOR I = O TO P * X
620 FOR J = 1 TO 1000
630 Z = 0
640 NEXT J
650 NEXT I
660 RETURN.
```

Podprogram 600 — 660 symuluje zmianę parametru P w funkcji drogi przebytej wzdłuż osi X. Procedura RESET (X,Y) wygasza punkt świecący na ekranie o współrzędnych X,Y.

Zastosowania

Zakres zastosowań grafiki komputerowej jest obszerny. Głównym obszarem zastosowań jest projektowanie wspomagane komputerem (CAD). Nowoczesne systemy komputerowe dzięki grafice dostarczają wyniki prac projektowych w postaci wykresów lub rysunków technicznych projektowanej konstrukcji. W zakresie projektowania wspomaganego komputerem główne dziedziny zastosowań grafiki komputerowej to:

— przemysł elektroniczny — 55 %
— kreślenia — 15 %
— kartografia — 10 %
— architektura i inne dziedziny techniki i nauki — 20 %

W przemyśle elektronicznym komputerowe systemy graficzne stosowane są głównie do projektowania układów elektronicznych, płytek drukowanych i układów scalonych. W dziedzinie budownictwa grafika komputerowa znalazła zastosowanie w projektowaniu autostrad i konstrukcji architektonicznych. Stosowane systemy graficzne umożliwiają prowadzenie symulacyjnych badań bezpieczeństwa jazdy na projektowanych fragmentach dróg.

W przemysłach lotniczym i motoryzacyjnym oraz w badaniach kosmicznych systemy graficzne stosowane są na wszystkich etapach projektowania. Systemy graficzne umożliwiają zwiększenie efektywności procesu projektowania i produkcji oraz dokonywanie symulacyjnych badań obiektu bez konieczności budowania wielu kosztownych modeli. W wypadku badań kosmicznych wiele badań eksploatacyjnych bez systemów graficznych byłoby niemożliwych.

Grafika komputerowa znalazła również wielostronne zastosowanie w wojsku. Komputerowe urządzenia zobrazowania danych w postaci graficznej znajdują się w wyposażeniu stanowisk dowodzenia różnych rodzajów wojsk. Przenośne mapniki elektroniczne wspomagają dowódców niższych szczebli bieżąco aktualizowaną drogą radiową informacją o sytuacji i zadaniach w danym rejonie. Komputerowe systemy graficzne są coraz częściej podstawą trenażerów różnych pojazdów, głównie samolotów i śmigłowców.

Zakres zastosowań grafiki komputerowej stale się zwiększa. Wynika to ze znacznych korzyści ekonomicznych, które są z tym związane. Systemy graficzne umożliwiają skrócenie czasu wykonywania projektu lub przeprowadzenia badań, zaoszczędzenie materiałów i robocizny, automatyczne wykreślanie rysunków i opracowanie dokumentacji technicznej wyrobu oraz opracowanie programów sterujących robotami wykonującymi dany wyrób w fabryce. Obecnie uważa się, że dzięki systemom graficznym jest możliwe skrócenie 5—8-krotne czasu prac projektowo-wdrożeniowych. Można zatem przyjąć, że grafika komputerowa będzie jedną z najbardziej dynamicznie rozwijających się dziedzin informatyki.

Jerzy CHMURZYŃSKI

2

BASIC

(Podstawy programowania)

W poprzednim numerze „IKS-a" rozpoczęliśmy wykład z programowania w języku BASIC na mikrokomputer ZX Spectrum. Kontynuujemy więc pracę, włączamy komputer i zaczynamy.

Wykład drugi: program w BASIC-u edytor na ZX Spectrum
elementy BASIC-u: stałe zmienne nazwy,
instrukcje: REM, LET, PRINT, INPUT
komendy: RUN, LIST, EDIT, NEW

Język BASIC (**B**eginners **A**ll Purpose **S**ymbolic **I**nstruction **C**ode) jest najpopularniejszym uniwersalnym językiem programowania wyższego rzędu. Stosują go zarówno użytkownicy posługujący się komputerami do przeprowadzania dość prostych obliczeń, jak i ci, którzy posługują się nimi przy rozwiązywaniu bardzo skomplikowanych problemów naukowo-technicznych, organizacyjnych lub ekonomicznych. Zaletą BASIC-u jest z jednej strony prostota, umożliwiająca szybkie opanowanie podstaw programowania, z drugiej strony istnienie w nim elementów pozwalających na efektywne zaprogramowanie wielu, nawet skomplikowanych problemów.

BASIC jest językiem konwersacyjnym, co pozwala programującemu na nadzorowanie i bieżącą korektę programu podczas jego uruchamiania.

Program w BASIC-u. Edytor na ZS Spectrum

Program w języku BASIC składa się z wierszy (linii), które zawierają instrukcje opisujące wykonanie poszczególnych kroków algorytmu. Włączamy komputer i zaczynamy pracę od przykładu.

Przykład 2.1

Obliczyć wartość sumy n pierwszych wyrazów postępu arytmetycznego według wzoru:

$$s_n = \left[a_1 + \frac{(n-1)\,d}{2} \right] n$$

gdzie: a_1 — pierwszy wyraz, d — różnica
dla $a_1 = 5$, $n = 11$, $d = 1$

Program realizujący te obliczenia ma postać:

```
10 REM Przyklad 2.1
30 LET d=1
40 LET n=11
20 LET a1=5
50 LET sn=(a1+(n-1)*d/2)*n
60 PRINT "SUMA =";sn
```

Wprowadźmy ten program. Załóżmy, że wprowadziliśmy 3 wiersze, na ekranie telewizora otrzymamy:

```
10 REM Przyklad 2.1
30 LET d=1
40 LET n=11
```

Wprowadźmy następny wiersz, a teraz ekran wygląda tak:

```
10 REM Przyklad 2.1
20 LET a1=5
30 LET d=1
40 LET n=11
```

Wprowadzając zauważamy, że wiersze programu zaczynają się od numerów i nie pojawiają się od razu, lecz zostają „odłożone" na później jako program, ponadto numery wierszy decydują o ich kolejności w programie (jest to także istotne przy wykonywaniu programu). Numer wiersza jest liczbą całkowitą, maksymalnie czterocyfrową ($1 \div 9999$). Wiersz w ZX Spectrum może składać się z kilku linii opatrzonych jedynie numerem w linii pierwszej.

Wprowadzając następny wiersz, na ekranie otrzymamy:

```
10 REM Przyklad 2.1
20 LET a1=5
30 LET d=1
40 LET n=11
50 PRINT "SUMA =";sn
```

Sprawdzamy program, okazuje się, że popełniliśmy błąd. Z numerem 50 został wprowadzony wiersz 60 naszego programu, tzn. chcemy wydrukować wartość S_n, która nie została obliczona. Wobec tego wprowadzamy wiersz:

45 **LET** $s_n = (a1 + (n - 1) * d/2) * n$

i na ekranie otrzymamy:

```
10 REM Przyklad 2.1
20 LET a1=5
30 LET d=1
40 LET n=11
45 LET sn=(a1+(n-1)*d/2)*n
50 PRINT "SUMA =";sn
```

Niemożliwe byłoby wstawienie wiersza między poprzednie, gdyby wiersze były numerowane kolejno np: 41, 42, 43 itd. Przy pisaniu programów i ich konwersacyjnym wprowadzaniu do komputera zaleca się numerować wiersze np. co 5 lub 10.

Naciskamy teraz klawisz **RUN** (i **ENTER**), w lewym górnym rogu ekranu pokaże się nam napis:

SUMA = 110

jest on wynikiem wykonania tego programu. Program „schował" się w pamięci komputera.

RUN — *jest komendą, która służy do uruchomienia i wykonania danego programu.*

Wykonujemy te same obliczenia, ale dla d = 4.

Naciskamy teraz klawisz **LIST** (i **ENTER**). Program ukazał się na ekranie.

LIST — *jest komendą, która napisana bez numeru wyświetla na ekranie tekst, poczynając od początku programu (do końca programu), natomiast LIST nr wiersza — od podanego numeru wiersza programu do końca programu.*

Ponadto tekst programu jest wyświetlany automatycznie, gdy wprowadzamy nowy wiersz do „schowanego" programu. Naciśnijmy teraz **CLS** (klawisz V) i **ENTER** — ekran został oczyszczony, ale program pozostał w pamięci.

Mamy wykonać obliczenia dla d = 4, w tym celu wprowadzamy

30 **LET** d = 4 (i **ENTER**)

Program ukazał się na ekranie, poprzedni wiersz o numerze 30 został zastąpiony nowo wprowadzonym.

Błędny wiersz z danym numerem można usunąć z programu przez wprowadzenie nowego wiersza z tym samym numerem. Takie poprawianie jest nie-

wygodne, gdy liczba znaków w wierszu jest duża. Usunięcie wiersza z programu wykonujemy przez napisanie jego numeru i naciśnięcie klawisza ENTER. Napiszmy teraz 29 (i **ENTER**), *w tym wypadku zniknął kursor wiersza* ⊠.

„Schował" się on między linie w numerach 20 i 30. Naciśnijmy teraz jednocześnie klawisze **CAPS SHIFT** i **EDIT** *(klawisz z cyfrą 1). W lewym dolnym rogu ekranu pokazał się nam wiersz o numerze 30. (30* **LET** *d = 4). Naciśnijmy* **ENTER** *— wiersz 30 powrócił na swoje miejsce w programie. Wykonajmy ten program* **RUN** *(i* **ENTER**).

Otrzymamy wynik:

SUMA = 275

Wykonajmy teraz kolejno następujące czynności:

LIST (klawisz K i ENTER)

EDIT (klawisze CAPS SHIFT i 1)

Na ekranie pokazał się program, a w lewym dolnym rogu wiersz o numerze 10. Naciśnijmy klawisz **ENTER**, wiersz o numerze 10 powrócił na swoje miejsce i kursor wiersza znajduje się w tym wierszu. (10 ⊠REM Przykład 2.1) Kursor wiersza można przenosić z wiersza do wiersza, w dół naciskając ↓ (klawisz **CAPS SHIFT** i 6), a w górę ↑ (klawisz **CAPS SHIFT** i 7). Przesuńmy kursor (wiersza) ⊠ do wiersza o numerze 40, następnie naciśnijmy **EDIT**. Wiersz 40 pojawił się w lewym dolnym rogu ekranu. Wykorzystując ←, → lub **DELETE** wprowadzamy na n liczbę 21. Poprawiać w wierszu (gdy jest on na dole ekranu) uczyliśmy się w pierwszym wykładzie. Po poprawkach wiersz powinien wyglądać tak:

40 **LET** n = 21

Naciskamy teraz kolejno: **ENTER**, **RUN** i **ENTER**. W lewym górnym rogu otrzymamy wynik:

SUMA = 945

Naciśnijmy teraz klawisz **NEW** (i **ENTER**)

NEW *— jest komendą, która usuwa z pamięci komputera program i czyści ekran.*

Po jej wykonaniu możemy wprowadzać nowy program.

Stałe, zmienne, nazwy, instrukcje: REM, LET, PRINT, INPUT

Program podany w przykładzie 2.1 zapiszmy w następującej postaci:

```
10 REM Przyklad 2.1.1
15 REM to co jest napisane po
  slowie REM to jest komentarz
20 LET a1=5
30 LET d=1
40 LET n=11
45 REM liczby 5,1,11 sa staly-
   mi numerycznymi
50 LET sn=(a1+(n-1)*d/2)*n
52 REM napisy a1,d,n,sn sa
   nazwami zmiennych prostych
54 REM napis (a1+(n-1)*d/2)*n
  jest wyrazeniem arytmetycz-
  nym
56 REM LET nazwa = wyrazenie
   arytmetyczne jest to aryt-
   metyczna instrukcja podsta-
   wienia
60 PRINT "a1=";a1;"    ";"d=";
   d;"    ";"n=";n
65 PRINT
70 PRINT "SUMA =";sn
75 REM kazdy ciag znakow napi-
   sany w cudzyslowie nazywa
   sie stala alfanumeryczna
  (lub lancuchem)
80 REM dzialanie instrukcji
   PRINT bylo omawiane, gdy
   traktowalismy komputer
   jako kalkulator
```

wykonajmy ten program, tzn. naciśnijmy **RUN** (i **ENTER**). Na ekranie telewizora w lewym górnym rogu otrzymamy:

```
a1=5    d=1    n=11

SUMA =110
```

Komputer zignorował wszystkie teksty napisane po

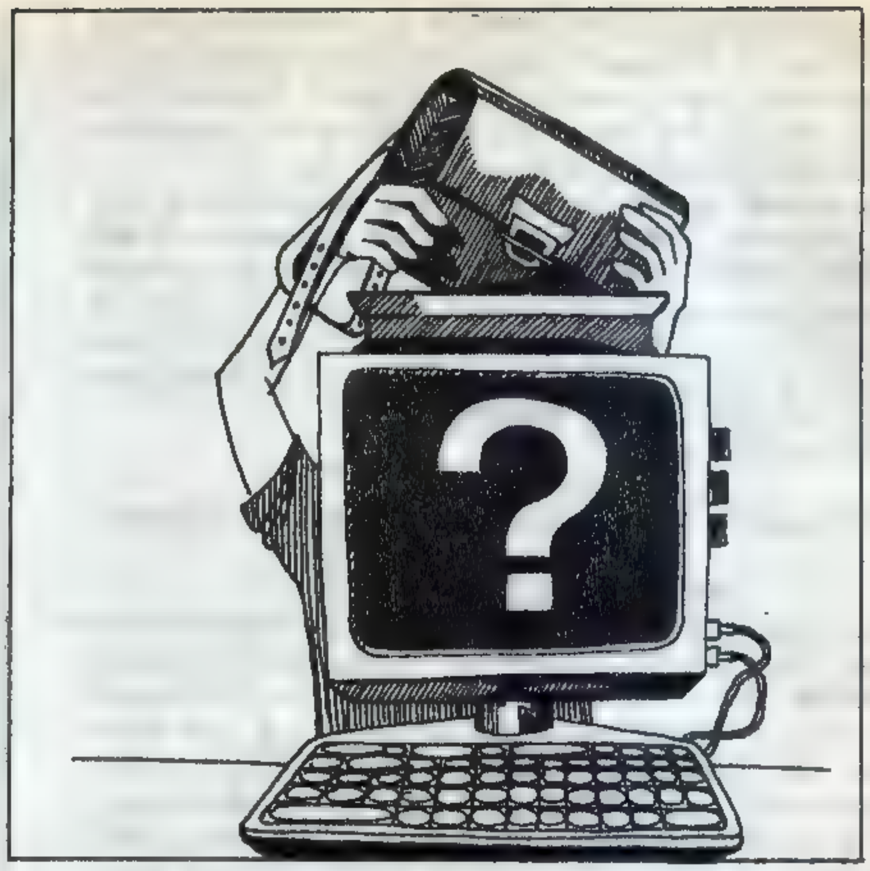
dokończenie na str. 24



Komputery... fascynacja dla wszystkich

Foto. S. Iwan





dokończenie ze str. 23

słowie kluczowym **REM**. Instrukcja **REM** ma postać:

n **REM** dowolny tekst

n — numer wiersza

REM — *napisane w dowolnym wierszu jest komentarzem, który został umieszczony w celu opisania naszego programu.*

W ZX Spectrum po słowie **REM** napisany tekst może zajmować kilka linii. W trakcie wykonywania programu komputer pominie tekst aż do końca ostatniej linii.

Aby otrzymać na ekranie (w wyniku wykonania programu) wszystkie teksty, które zostały napisane po słowie REM, należy zamienić słowo **REM** — słowem kluczowym **PRINT** i każdy tekst zamknąć w cudzysłów.

Wykonajmy te poprawki, będzie to dobrym sprawdzianem omówionych wcześniej czynności wykonywanych przy poprawianiu wierszy. Po poprawkach program powinien wyglądać tak:

```
  10 REM Przyklad 2.1.2
  15 PRINT "to co jest napisane
po slowie   REM to jest komentar
z"
  20 LET a1=5
  30 LET d=1
  40 LET n=11
  45 PRINT '"Liczby 5,1,11 sa st
alymi         numerycznymi"
  50 LET sn=(a1+(n-1)*d/2)*n
  52 PRINT "napisy a1,d,n,sn sa
 nazwami      zmiennych prostych"
  54 PRINT '"napis (a1+(n-1)*d/2
)*n jest      wyrazeniem arytmety
cznym"
  56 PRINT '"LET nazwa = wyrazen
ie arytme-    tyczne jest to aryt
metyczna      instrukcja podstawi
enia"
  60 PRINT '"a1=";a1;"      ";"d="
;   d;"    ";"n=";n
  65 PRINT
  70 PRINT "SUMA =";sn
  75 PRINT '"kazdy ciag znakow n
apisany       w cudzyslowie nazyw
a sie stala   alfanumeryczna (lub
 lancuchem)"
  80 REM dzialanie instrukcji
     PRINT bylo omawiane, gdy
     traktowalismy komputer
     jako kalkulator
```

a po wykonaniu na ekranie otrzymamy:

```
to co jest napisane po slowie
REM to jest komentarz

liczby 5,1,11 sa stalymi
numerycznymi
napisy a1,d,n,sn sa  nazwami
zmiennych prostych

napis (a1+(n-1)*d/2)*n jest
wyrazeniem arytmetycznym

LET nazwa = wyrazenie arytme-
tyczne jest to arytmetyczna
instrukcja podstawienia

a1=5      d=1      n=11

SUMA =110

kazdy ciag znakow napisany
w cudzyslowie nazywa sie stala
alfanumeryczna (lub lancuchem)
```

Instrukcja **PRINT** ma postać:

n **PRINT** lista lub n **PRINT**

n — numer wiersza

lista — może składać się z nazw zmiennych, wyrażeń arytmetycznych, stałych numerycznych, stałych alfanumerycznych i znaków specjalnych (' , ;)

Znaki specjalne sterują postacią wyprowadzanych wyników, zapoznaliśmy się z ich działaniem, gdy traktowaliśmy komputer jako kalkulator. W wyniku wykonania drugiej postaci instrukcji **PRINT** na ekranie otrzymujemy pustą linię.

Omawialiśmy też wyrażenie arytmetyczne i kolejność wykonywania działań w celu obliczenia wartości wyrażenia, w którym występowały stałe numeryczne (liczby) połączone operatorami arytmetycznymi. *W naszym przykładzie napis:*

$$(a1 + (n - 1) * d / 2) * n$$

jest wyrażeniem arytmetycznym, w którym występują proste zmienne numeryczne, stała numeryczna (liczba 2), operatory arytmetyczne (+, −, /, *) oraz nawiasy ( ). W trakcie obliczania wartości takiego wyrażenia według obowiązujących reguł (podaliśmy je w przykładzie 1) komputer pobiera





Rys. Michał Przybyłowski

wartości odpowiednich zmiennych i na tych wartościach wykonuje obliczenia.

Zmienna prosta numeryczna przyjmuje wartości będące liczbami rzeczywistymi. Nazwa tej zmiennej może zawierać dowolne litery lub cyfry, ale musi zaczynać się od litery. Stała alfanumeryczna (tekstowa) ma postać:

"ciąg znaków"

— ciąg znaków: dowolny ciąg znaków
— znaki cudzysłowu ograniczają jej początek i koniec
— tekst "" bez żadnego znaku zwany jest tekstem pustym
— tekst " " nie jest tekstem pustym, oznacza jedną spację, spacje wewnątrz cudzysłowu są znakami tekstu

Instrukcja **LET** wyznacza wartość zmiennej podczas działania programu i ma postać:

n **LET** V = e

n — numer wiersza
V — nazwa zmiennej
e — wyrażenie arytmetyczne

Wykonanie tej instrukcji polega na obliczeniu wartości wyrażenia arytmetycznego stojącego po prawej stronie znaku = (równa się) i podstawienie obliczonej wartości na miejsce zmiennej występującej po lewej stronie znaku = (równa się).

Przykład 2.2.

Dany jest program:

```
 10 REM Przyklad 2.2
 20 PRINT "cwiczenia"
 25 PRINT
 30 LET a=2.718281
 40 PRINT "a=";a
 45 PRINT
 50 LET b=2.71828182
 60 PRINT "b=";b
 70 LET c=2.7182818285
 75 PRINT
 80 PRINT "To jest przyblizona
wartosc        stalej e :";c
```

Wprowadźmy i wykonajmy ten program. Na ekranie otrzymamy:

```
cwiczenia

a=2.718281

b=2.7182818

To jest przyblizona  wartosc
     stalej e :2.7182818
```

Napiszmy teraz:

70 **LET** c = 0.27182818285 e^{-1}

i wykonajmy program, następnie napiszmy:

70 **LET** c = 27182818285 e^{-10}

i wykonajmy program. Kolejne realizacje programu dla różnych postaci zapisu stałej w wierszu 70 dają ten sam wynik:

2.2182818

Zapamiętajmy:
— litera e oznacza mnożnik dziesiętny np. $2.5e^2$ czytamy jako $2{,}5 \cdot 10^2 = 250$
— zapis liczby postaci $2{,}5e^2$ nazywa się zapisem wykładniczym,
— kropka jest używana w miejsce przecinka oddzielającego część całkowitą od części ułamkowej,
— PRINT drukuje najwyżej 8 cyfr znaczących,
— do zapisania liczby nie może być użytych więcej niż 14 znaków,
— liczby są pamiętane z dokładnością do około dziewięciu i pół cyfry,
— największa liczba całkowita, która może być zapamiętana dokładnie, to $2^{32} - 1 = 4294967295$

Analogicznie do pojęcia zmiennej prostej numerycznej istnieje pojęcie zmiennej prostej alfanumerycznej, której „wartością" jest tekst. Nazwa takiej zmiennej składa się z litery (tylko jednej), po której musi następować znak dolara (**$**). Nazwom zmiennych alfanumerycznych można nadawać „wartości", używając instrukcji LET, np:

20 **LET** p **$** = „zadanie"

dokończenie na str. 26

dokończenie ze str. 25

Napisanie w programie instrukcji postaci:

30 **PRINT** p $

spowoduje w trakcie jej wykonania wypisanie na ekranie słowa: zadanie.

Przykład 2.3.

Napisać program w BASIC-u na obliczenie pola powierzchni całkowitej stożka obrotowego ściętego według wzoru:

$$P = \pi * R * (R + l) + \pi * r * (r + l)$$

gdy: tworząca l = 30,25
promień duży R = 12,75
promień mały r = 7,25
W programie należy zastosować zmienne alfanumeryczne.

Rozwiązanie:
BASIC (ZX Spectrum) nie rozróżnia liter małych i dużych, tzn. nazwy R i r oznaczają tę samą nazwę, przyjmujemy więc:
rd — promień duży
rm — promień mały

```
  10 REM Przyklad 2.3
  20 PRINT "Jest to przykladowy
program,    jezeli chcesz to nap
isz go sam   w inny sposob"
  25 PRINT
  30 LET rd=12.75
  35 LET rm= 7.25
  40 LET l =20.75
  45 LET r$="promien"
  50 LET d$=" duzy"
  55 LET m$=" maly"
  60 PRINT "obliczanie pola powi
erzchni      stozka scietego,gdy
dane sa :"
  65 PRINT
  70 PRINT "tworzaca =";l'r$;d$;
" =";rd'r$;m$;" =";rm
  75 PRINT
  77 REM PI znajduje sie nad kla
wiszem M
  80 LET P=PI*rd*(rd+l)+PI*rm*(r
m+l)
  90 PRINT '"pole pow.calk. stoz
ka scietego"'"          P =";P
```

Wprowadźmy i wykonajmy ten program. Sprawdźmy otrzymany wynik z wynikiem podanym niżej:

```
Jest to przykladowy program,
jezeli chcesz to napisz go sam
w inny sposob

obliczanie pola powierzchni
stozka scietego,gdy dane sa :

tworzaca =20.75
promien duzy =12.75
promien maly =7.25


pole pow.calk. stozka scietego
        P =1979.5961
```

Przerabiając podane przykłady programów zauważyliśmy, że zmieniając jakąś wartość danej, należało wprowadzić całą instrukcję od nowa, np. gdy w programie było 30 **LET** d = 1, to zmieniając wartość d na 4, wprowadziliśmy 30 **LET** d = 4.

Napisane dotychczas programy wykonywały obliczenia na tych samych danych. Każda zmiana wartości dowolnej danej wymagała jej wprowadzenia do programu przez zmianę odpowiedniej instrukcji.

Napiszemy teraz program z Przykładu 2.3, tak aby mógł być wykonywany dla dowolnych wartości l, R i r.

```
  10 REM Przyklad 2.3.1
  20 PRINT "Jest to przykladowy
program,    jezeli chcesz to nap
isz go sam   w inny sposob"
  25 PRINT
  30 INPUT "rd=";rd,"rm=";rm
  40 INPUT "l=";l,"r$=";r$
  50 INPUT "d$=";d$,"m$=";m$
  60 PRINT "obliczanie pola powi
erzchni      stozka scietego,gdy
dane sa :"
  65 PRINT
  70 PRINT "tworzaca =";l'r$;d$;
" =";rd'r$;m$;" =";rm
  75 PRINT
  77 REM PI znajduje sie nad kla
wiszem M
  80 LET P=PI*rd*(rd+l)+PI*rm*(r
m+l)
  90 PRINT '"pole pow.calk. stoz
ka scietego"'"          P =";P
```

Wprowadźmy i wykonajmy ten program.

Omówimy teraz kolejność działań wykonywanych przez komputer i wyświetlanych na ekranie.

W górnej części ekranu ukaże się napis:
Jest to przykładowy program, jeżeli chcesz to napisz go sam w inny sposób.

Następnie w lewym dolnym rogu wyświetlone zostaną kolejno odpowiednie teksty i kursor L, należy wtedy wprowadzić dane. A więc:

rd = L

Komputer czeka na wprowadzenie liczby — wprowadzamy 12.75 (i **ENTER**), a komputer w tym samym wierszu w połowie ekranu wypisze:

rm = L

wprowadzamy 7.25 (i **ENTER**). Oba napisy zniknęły, a komputer wypisał (w lewym dolnym rogu ekranu):

l = L

wprowadzamy 20.75 (i **ENTER**), a komputer w tym samym wierszu w połowie ekranu wypisze:

r $ = "L"

wprowadzamy słowo promień (i **ENTER**). Oba napisy zniknęły, a komputer wypisał:

d $ = "L"

wprowadzamy słowo duży (i **ENTER**), a komputer w tym samym wierszu w połowie ekranu wypisze:

m $ = ["L"]

wprowadzamy słowo mały (i **ENTER**). Oba napisy zniknęły, a komputer wypisał w górnej części ekranu wyniki, które już znamy.

Instrukcja **INPUT** *umożliwia nadawanie wartości zmiennym w czasie działania programu.* Możliwość ta jest szczególnie ważna przy pracy konwersacyjnej.

Instrukcja **INPUT** ma postać:

n **INPUT** lista

n — numer wiersza

lista — może składać się z nazw zmiennych, stałych alfanumerycznych i znaków specjalnych.

Działanie znaków specjalnych jest takie samo jak w instrukcji **PRINT**.

W czasie wykonywania instrukcji **INPUT** na jej zlecenie wprowadzamy z klawiatury wartości stałych numerycznych i alfanumerycznych, które są następnie przenoszone pod kolejne zmienne. Stałe te nazywają się danymi.

Stałe alfanumeryczne (tekstowe) wprowadzamy w cudzysłowie, komputer sam to sygnalizuje podając: ["L"] (kursor L). Umieszczanie stałych tekstowych opisujących nazwy zmiennych (na liście INPUT) którym mają być nadane wartości, ułatwia wprowadzanie tych wartości, a tym samym zmniejsza liczbę pomyłek.

Instrukcja (na ZX Spectrum) postaci:

```
30 INPUT "rd = "; rd
```

w wyniku wykonania daje to samo, co poniższe instrukcje:

```
30 PRINT "rd =";
35 INPUT rd
```

A teraz wykonajmy jeszcze raz ostatni program dla danych np. rd = 75,2, rm = 55, l = 100,25

Zadania

1. Napisać w BASIC-u program obliczania wartości y danej wzorem:

$$y = \frac{a^4 - b^4}{2(a^2 + b^2)} + \frac{a^2 - b^2}{a^2 + b^2} - \frac{a^2 + b^2}{2}$$

dla dowolnych dodatnich wartości a i b.

Uwaga: $a^4 - b^4 = (a^2 - b^2) * (a^2 + b^2)$, wprowadzić zmienne pomocnicze: $c = a^2$, $d = b^2$, $w = c + d$, $z = c - d$ np. dla a = 2 i b = 3 y = − 9.3846154

2. Napisać w BASIC-u program obliczania wartości z danej wzorem:

$$z = \frac{x^2 + y^2}{2xy} - 6(x^2 - 2xy + y^2)$$

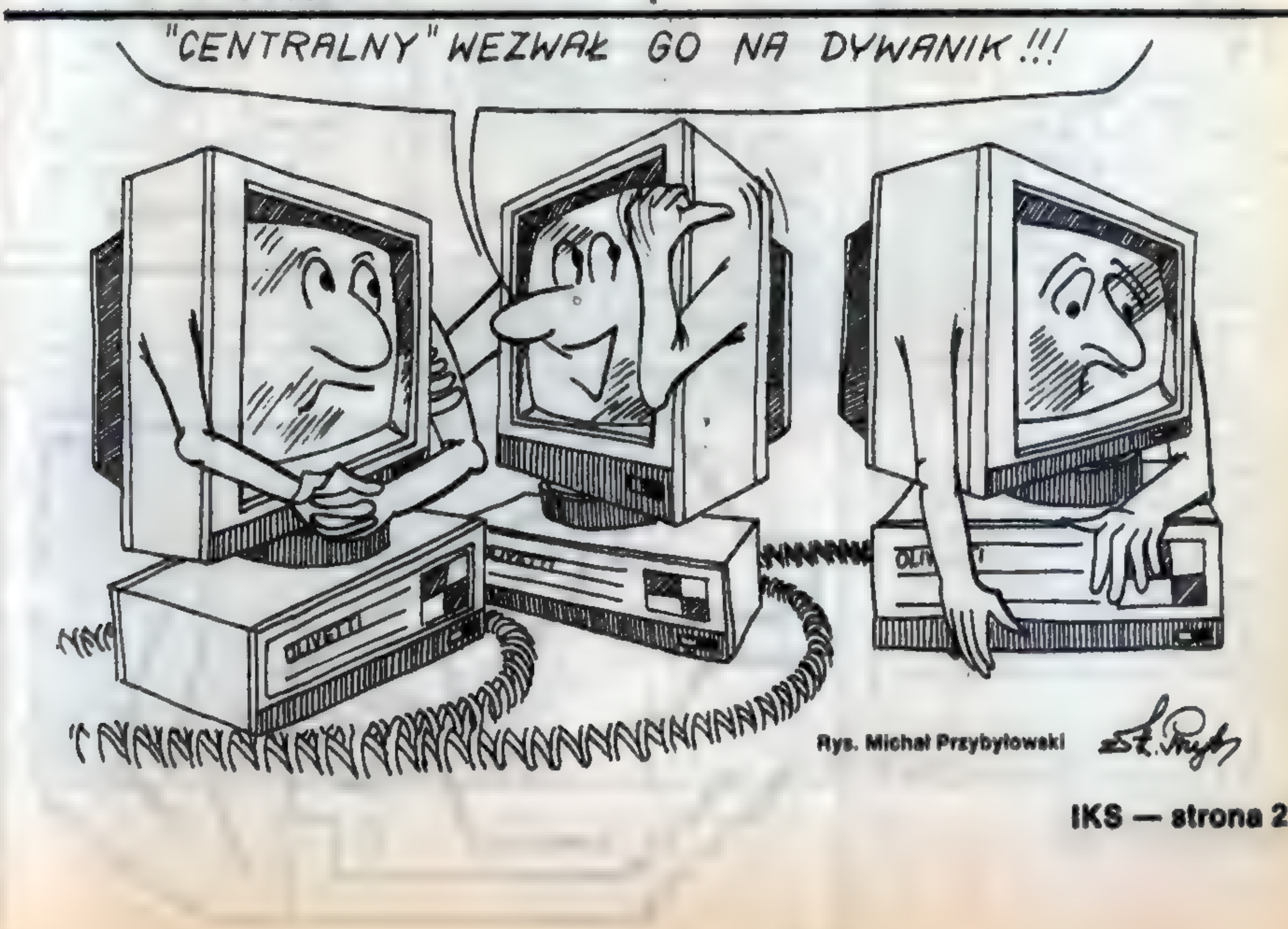
gdy: $x = 5u^2 + 2v$, $y = 3{,}7 + v^2$
u, v dowolne wartości, ale $u > 0$ i $v > 0$.

3. Napisać w BASIC-u program, który dla dowolnej osoby wyprowadza na ekranie:

pierwsza strefa	druga strefa
NAZWISKO I IMIĘ ADRES: KOD MIEJSCOWOŚĆ ULICA NR DOMU NR MIESZK. NR TELEFONU ZAWÓD	w tej części dane dotyczące dowolnej osoby

ekranu

Z.W.



Rys. Michał Przybyłowski

PIÓRO ŚWIETLNE DO ZX SPECTRUM

Jednym z ciekawszych urządzeń zewnętrznych do mikrokomputera ZX Spectrum jest pióro świetlne. Jest ono urządzeniem, które samodzielnie może zbudować każdy posiadacz mikrokomputera. Wymagana jest jedynie umiejętność lutowania, a do uruchomienia pióra nie jest potrzebny żaden przyrząd pomiarowy. Do budowy wykorzystano części produkcji krajowej, których koszt nie przekracza 600 zł!

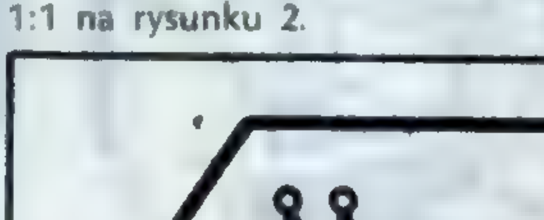
Zastosowanie odpowiedniego oprogramowania umożliwia rysowanie piórem na ekranie telewizora; kreślenie linii, prostokątów, okręgów, kolorowanie wskazanego obszaru, umieszczanie napisów, wybór koloru atramentu i papieru. Istnieją również programy wykorzystujące pióro przy wykonywaniu całego szeregu obliczeń matematycznych, np. pakiet programów „Library" umożliwia wykonywanie rachunku macierzowego, całkowania funkcji, obliczeń statystycznych.

Posługiwanie się piórem polega na dotknięciu wybranego miejsca ekranu (np. wskazanie opcji programu) i naciśnięciu dowolnego klawisza na pulpicie mikrokomputera.

Pióro świetlne, wbrew swojej nazwie, nie jest urządzeniem emitującym lecz wykrywającym światło. Każdy element obrazu widocznego na ekranie TV posiada swój adres w pamięci mikrokomputera. Zawartość pamięci obrazu wyświetlana jest 50 razy na sekundę, a poszczególne elementy obrazu wyświetlane są kolejno. Pióro wykrywa moment zaświecenia się wskazywanego miejsca i informację o tym wysyła do ZX Spectrum. Na podstawie tej informacji mikrokomputer „wie", który element obrazu został wskazany. Lokalizacja odbywa się z dokładnością do najmniejszego elementu obrazu — piksela.

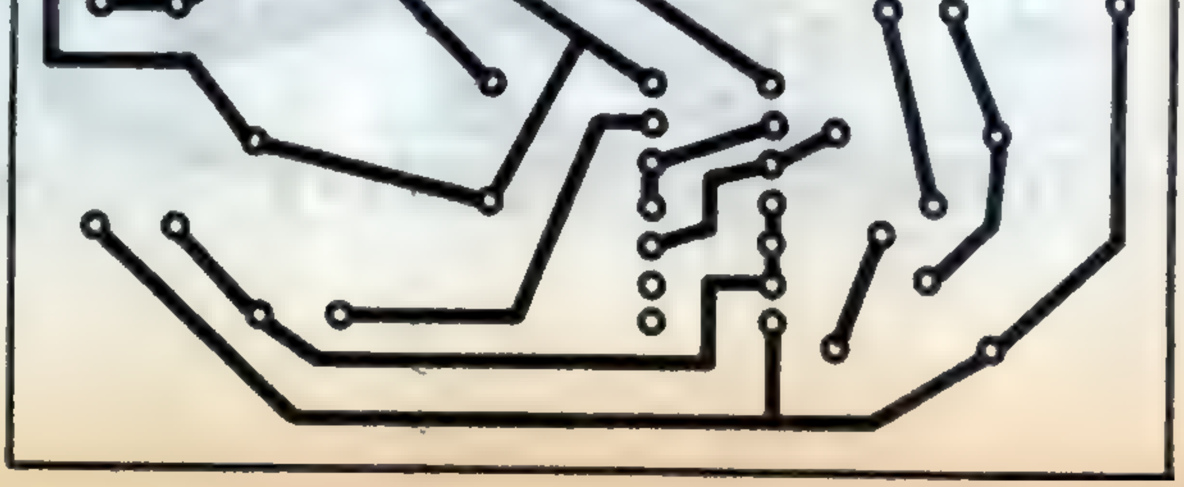
Schemat elektroniczny pióra świetlnego przedstawiono na rysunku 1.

Czujnikiem wykrywającym moment zaświecenia się wskazywanego przez pióro miejsca jest fototranzystor T1. W wyniku oświetlenia fototranzystora zostaje wyzwolony przerzutnik monostabilny, który generuje krótki dodatni impuls. Impuls ten jest potwierdzeniem wykrycia momentu zaświecenia się wskazywanego punktu ekranu. Układ przerzutnika tworzą bramki NOR i NOT (ukł. scalony US1) wraz z rezystorem R2 i kondensatorem C2. ZX Spectrum wymaga jednak sygnału o polaryzacji ujemnej. Zmianę fazy sygnału z przerzutnika realizuje tranzystor T2. Zastosowanie w tej roli tranzystora zwiększa odporność wyjścia układu (gniazdo EAR) na zwarcia.



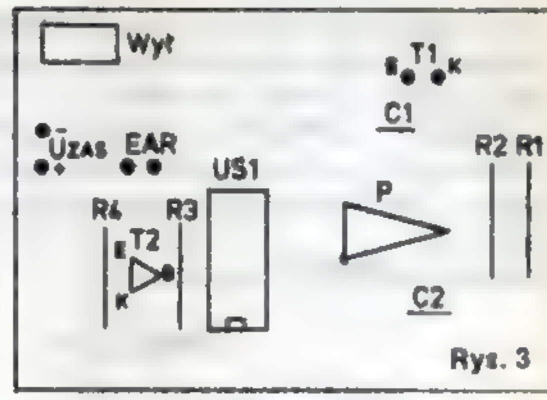
Rys. 1

Elementy układu (za wyjątkiem T1) zmontowano na płytce drukowanej przedstawionej w skali 1:1 na rysunku 2.



Rys. 2

Na rysunku 3 przedstawiono rozmieszczenie elementów na płytce.

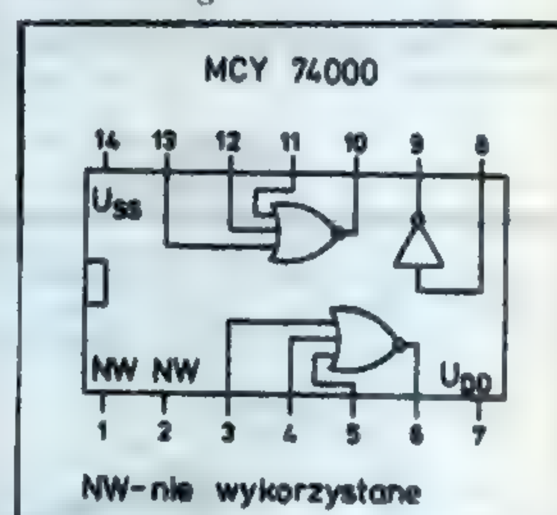


Rys. 3

W pierwszej kolejności lutujemy elementy dyskretne, natomiast układ scalony US1 montujemy na końcu. Zastosowany układ scalony jest wykonany w technologii CMOS i należy obchodzić się z nim ostrożnie przestrzegając następujących zasad:

— nie dotykać palcami końcówek układu,
— lutować uziemioną lutownicą.

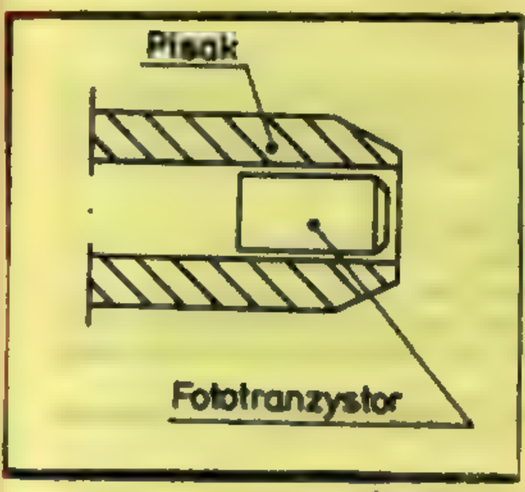
Nieprzestrzeganie zaleceń może spowodować nieodwracalne uszkodzenie układu. Na rysunku 4 pokazano rozmieszczenie wyprowadzeń zastosowanego układu scalonego.



Rys. 4

Fototranzystor T1 umieszczono w obudowie pisaka i połączono z

resztą układu za pośrednictwem długiego przewodu zakończonego wtyczką „mini Jack”. Należy zwrócić uwagę na sposób umieszczenia tego tranzystora — jego koniec powinien być osłonięty (rys. 5).



Rys. 5

Zmontowaną płytkę drukowaną umieszczono w obudowie wspólnie z baterią 6F22. Na jednej z bocznych ścianek obudowy umocowano dwa gniazdka. Gniazdko oznaczone EAR służy do połączenia pióra z gniazdkiem EAR ZX Spectrum. Gniazdko oznaczone PEN służy do podłączenia czujnika pióra (fototranzystor T1).

Tak wykonany i sprawdzony pod względem poprawności montażu układ możemy podłączyć do mikrokomputera.

UWAGA

Podłączenie do ZX Spectrum należy wykonywać przy odłączonym zasilaniu pióra. Najpierw łączymy fototranzystor z gniazdkiem PEN, następnie łączymy gniazdka EAR pióra i mikrokomputera (firmowym kabelkiem). Dopiero teraz włączamy zasilanie pióra.

W czasie tych czynności ZX Spectrum jest włączony i ma wczytany program obsługi pióra. Niektóre egzemplarze ZX Spectrum lepiej pracują, gdy podłączymy pióro do gniazdka MIC.

Jedyną regulacją, jakiej wymaga pióro jest kalibracja, do przeprowadzenia której służy potencjometr montażowy P (rys. 1). Należy zaznaczyć, że kalibracji dokonuje się w czasie pracy z programem obsługi pióra. Ustawiamy suwak potencjometra w położeniu środkowym i postępując według instrukcji zawartej w programie próbujemy pióro skalibrować. Jeżeli próba nie powiodła się, zmieniamy położenie suwaka (w górę lub w dół) i próbujemy skalibrować pióro ponownie. Czynności te powtarzamy aż do skutku.

Raz skalibrowane pióro praktycznie nie wymaga już regulacji.

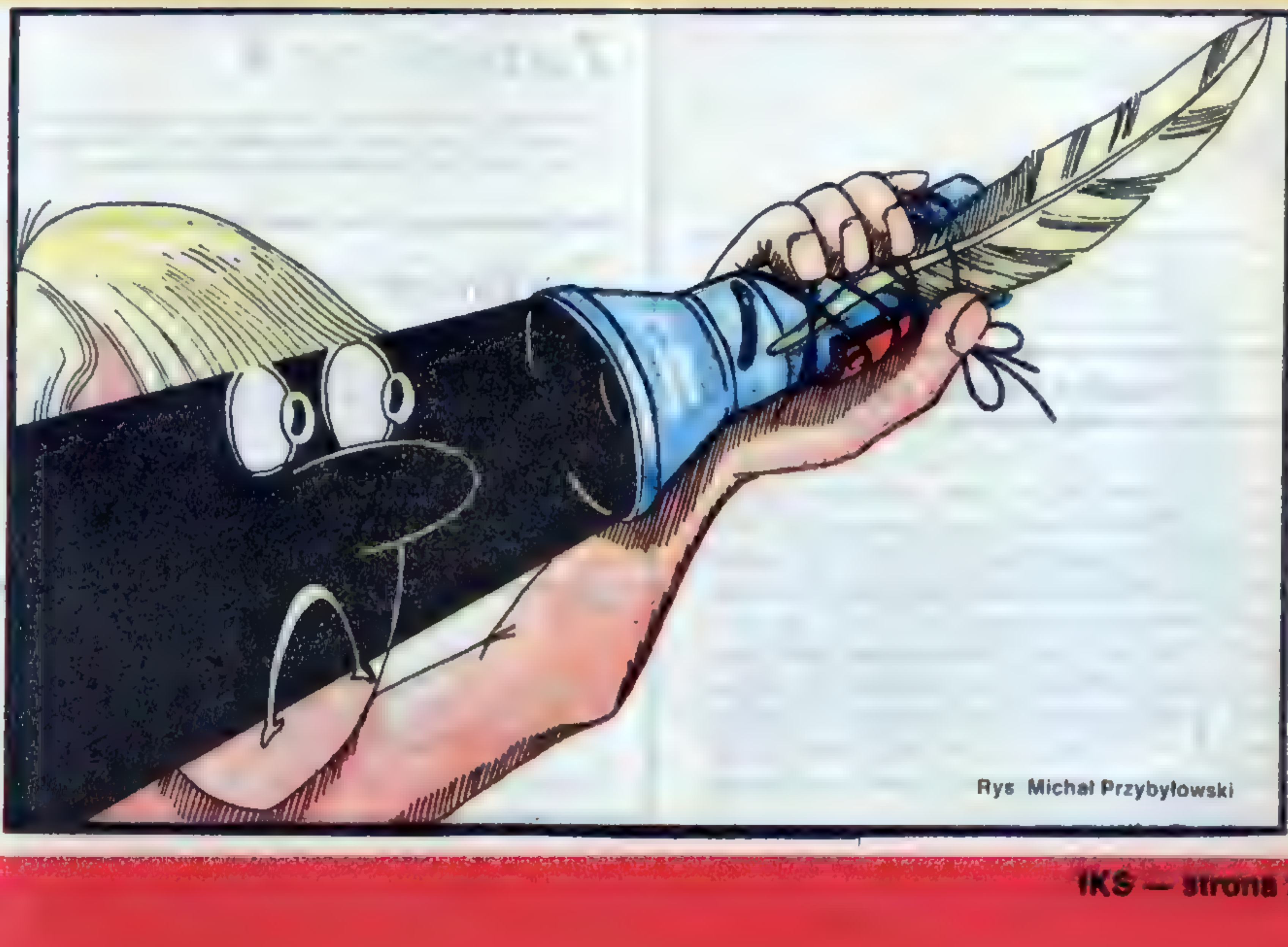
Spis części

Kondensatory (dowolnego typu)
- C1 — 22 nF
- C2 — 1,5 nF

Rezystory (najlepiej typu MŁT 0,125 W lub 0,25 W)
- R1 — 1 kΩ
- R2 — 2,7 MΩ
- R3 — 47 kΩ
- R4 — 2,7 kΩ

Półprzewodniki
- US1 — MCX74000 N
- T1 — BPYP 22
- T2 — BC 238, BC 237

Inne
- P — miniaturowy potencjometr montażowy 220 kΩ
- 2 gniazdka typu „mini Jack”
- 2 wtyczki typu „mini Jack”
- Miniaturowy wyłącznik zasilania



Rys. Michał Przybyłowski

Rozwiązania zadań z nr. 1

Zadanie 1: Aby dowiedzieć się, jakie kule są w poszczególnych koszach wystarczy tylko jedno ciągnienie. Jeżeli np. z kosza oznaczonego kartką czarno-białą (o którym wiemy, że nie ma w nim kul białych i czarnych) wyciągniemy kulę białą, możemy powiedzieć, że są tam 2 kule białe, w konsekwencji — w tym, na którym jest kartka biała-biała, muszą być 2 kule czarne, a w trzecim koszu: biała-czarna, analogiczne rozumowanie trzeba przeprowadzić gdy wyciągniemy kulę czarną.

Zadanie 2: Jest to najbardziej znane zadanie w teorii grafów nazywane „problemem mostów królewskich". Został on rozwiązany przez Leonarda Eulera (1707—1783) w roku 1736. Dwie wyspy, A i B, na rzece Pregole w Królewcu (ówczesna stolica Prus Wschodnich, obecnie Kaliningrad w zachodniej części ZSRR) były połączone ze sobą i z brzegami C i D siedmioma mostami, tak jak to pokazano na rys. 1.1. Euler przedstawił tę sytuację za pomocą grafu (rys. 1.2.).

Punkty A, B, C, D, zwane w teorii grafów wierzchołkami, reprezentują części lądowe, a linie łączące te punkty, w teorii grafów — krawędzie reprezentują mosty. Euler udowodnił, że rozwiązanie tego zagadnienia nie istnieje, a więc nie jest możliwe przejście przez wszystkie mosty dokładnie jeden raz, tak, aby powrócić do punktu wyjścia.

Zadanie 3: To typowe zadanie z kombinatoryki. Jest to 5-elementowa permutacja określona na 5-elementowym zbiorze i wyraża się wzorem $5! = 1 \times 2 \times 3 \times 4 \times 5 = 125$, a więc na pięciu krzesłach pięcioosobowa komisja może usiąść na 125 różnych sposobów.

Zadanie 4: W pudełku było x sztuk amunicji. Wiedząc, że liczba amunicji na wszystkich stanowiskach jest równa, możemy ułożyć równanie:

$1 + \frac{x-1}{6} = 2 + (x - \frac{x-1}{6} - 1 - 2) : 6$; po uproszczeniu

$\frac{x+5}{6} = 2 + \frac{5x-17}{36}$, skąd

$6x + 30 = 72 + 5x - 17$

$x = 25$

Na każde stanowisko ogniowe było przeznaczone $1 + \frac{x-1}{6} = 5$ sztuk amunicji, a stąd otrzymujemy, że było 5 stanowisk ogniowych.

Przedstawcie się...

Ile jest w kraju mikrokomputerów i jakich? Pomóżcie nam zbadać „rozkład" sprzętu komputerowego w Polsce. Prosimy przysyłajcie do nas informacje o tym, czy macie mikrokomputer bądź dostęp do niego (u kolegi, w klubie) i koniecznie zaznaczcie, jaki jest jego typ. Czekamy na informacje. Prosimy o odpowiedzi na kartkach pocztowych — ułatwi to nam znacznie pracę redakcyjną, a wyniki tej sondy pomogą w redagowaniu „IKS-a".

LIGA MYŚLĄCYCH

Zadanie nr 1

— Roman dojeżdża do szkoły autobusami linii miejskiej, autobusy te w godzinach szczytu jeżdżą w równych od siebie odstępach czasowych, co 6 minut. Czas postoju na przystanku zależy od trzech czynników:
— liczby pasażerów wysiadających,
— liczby pasażerów wsiadających,
— stopnia napełnienia autobusu.
Jeden z autobusów pod wpływem przypadkowych zakłóceń opóźnił się. Co będzie się działo dalej na tej linii autobusowej?

Zadanie nr 2

— Mapa Polski jest podzielona na 49 województw. Na mapie administracyjnej każda para sąsiadujących ze sobą województw powinna mieć inne barwy. Ile najmniej barw trzeba użyć do prawidłowego sporządzenia takiej mapy Polski?

Zadanie nr 3

— Proszę podać czynności Andrzeja przy próbie telefonowania do Piotra w kolejności ich wykonania. Czynności te proszę narysować w dowolnym schemacie rysunkowym, uwzględniając wszystkie okoliczności towarzyszące próbie telefonowania, przy czym schemat rysunkowy powinien zaczynać się jedną czynnością — kończyć też jedną czynnością, a wszystkie one powinny być połączone ze sobą w sposób logiczny.

Zadanie nr 4

— Pociąg pośpieszny mija zawiadowcę stojącego na początku peronu w ciągu 11 sek., a cały peron mający 120 m długości w ciągu 26 sek. Jaka jest długość tego składu pociągu?

Zadanie nr 5

— Aby rozstrzygnąć, która z dwóch drużyn piłkarskich jest lepsza, należy rozegrać jeden mecz. Podaj ile meczy musi rozegrać dwanaście drużyn, grając systemem każdy z każdym.

Rozwiązania zadań prosimy przysyłać pod adresem redakcji do końca czerwca br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca lipca br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Pocztowa giełda

Rozwiązanie krzyżówki nr 1. Hasło brzmi: **„IKS MOIM PISMEM"** Bony pieniężne 1000 zł wylosowali: **Przemysław Murawski** Radom, **Elżbieta Bieńkowska** Szczecin, **Ewa Chruściel** Rzeszów, **Zenon Świerczyński** Ogrodzieniec, **Adam Mazurczak** Sokołów Podlaski.

Nagrody książkowe otrzymują: **Zdzisław Woźniczko** Tarnowskie Góry, **Romuald Sztukiewicz** Poznań, **Praveen kant Sharma** Wrocław, **Witold Turkiewicz** Kołobrzeg, **Mariusz Jaworski** Pabianice, **Leszek Szydełko** Kalisz, **Dariusz Rulski** Piekary Śląskie, **K. Kulikiewicz** Wrocław, **Mikołaj Kmita** Gdynia Karwiny I, **Jerzy Pyszny** Rybnik 12

W **„Lidze Myślących"** wysoki poziom. Kilkuset uczestników przysłało prawidłowe odpowiedzi. „Liderów" ogłosimy w następnym numerze.

Warszawskie notowania „komputerowe":

ZX Spectrum 48K + magnetofon — 100 tys.
ZX Spectrum 128K — 200 tys.

Atari 600XL + magnetofon — 120 tys.
Atari 800XL + magnetofon — 120 tys.
Atari 130XE — 130 tys.

Commodore C 64 + magnetofon + 2 joystiki — 300 tys
Commodore VC 20 — 55 tys.
Commodore 16 + magnetofon — 65 tys.

Sharp 700 — 190 tys.
Schneider Amstrad CPC 464 — 250 tys.

Lista „przebojów"

Notowań część druga. Największym powodzeniem cieszył się program **„Odwracanka"** — tak zadecydowali nasi Czytelnicy. „Tu trzeba myśleć" — słyszeliśmy najczęściej w uzasadnieniu. W naszej tabeli na drugie miejsce wysunął się **„Compact"** — program na Spectrum.

Prowadzący w naszych notowaniach do tej pory program **„Wykresy"** spadł na trzecią pozycję, a jego użytkownicy wskazują na niedoskonałości — program zmodyfikujemy i zgodnie z życzeniem posiadaczy innych mikrokomputerów przedstawimy wersję na Commodore i Atari

Nadal czekamy na Wasze propozycje i opinie.

Ogłoszenia

W naszym komputerlandzie

Tak się jakoś złożyło, że w naszym komputerlandzie wypoczywały ostatnio prawie wszystkie roboty i jeśli nawet któryś miał ochotę, choćby dla draki, od rana coś zrobić, to i tak okazywało się, że jest kolejne święto, dzień wolny, dzień już odpracowany, dzień, który będzie odpracowany, wolna sobota, niedziela itp. Ponieważ zespół robotów w świętowaniu i odpoczynku osiągnął już prawdziwe mistrzostwo, postanowił przetestować następujące sytuacje:

Teza: Robot nie jest stworzony do pracy.

Analiza możliwości: Jak wiadomo z ogólnych doświadczeń praca robota męczy. Po pracy, podobnie zresztą jak przed pracą, robot czuje się zmęczony, wypluty, skołowany oraz oczekuje zbawczego dzwonka na fajrant.

Wniosek: Gdyby robot był stworzony do pracy, to by go praca nie męczyła c.b.d.u.

Wnioski praktyczne: Należy rozważyć pomysł przydatności wolnych poniedziałków, rozważyć koncepcję stopniowego wprowadzania wolnych wtorków, skupić uwagę nad projektem zarządzeń w sprawie wolnych śród, przeanalizować ideę ustalenia wolnych czwartków i dokonać wstępnych przymiarek do projektu wolnych piątków.

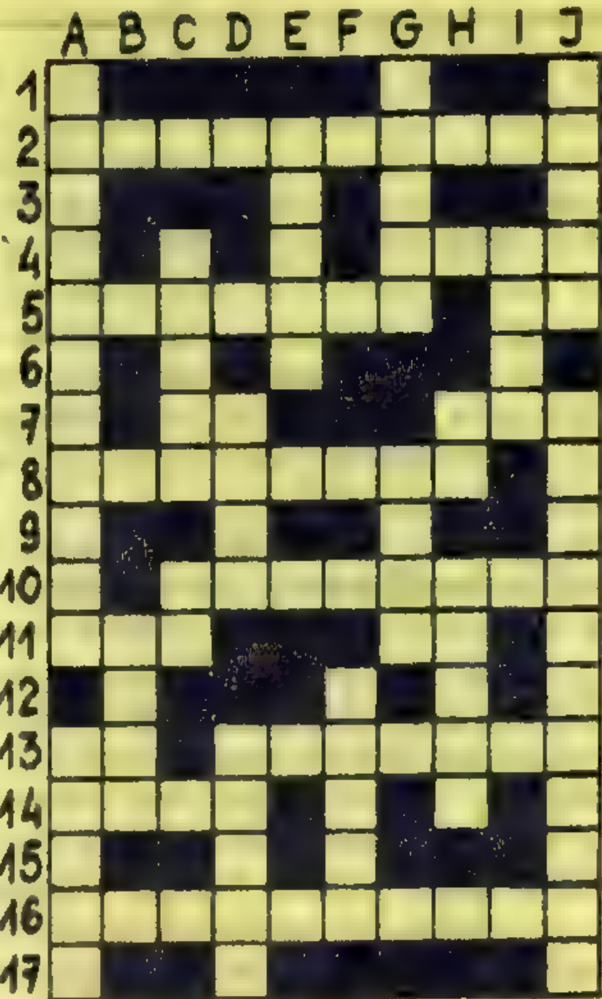
Gdyby jednak, wśród ogólnego luzu i bimbania znaleźli się ekstremiści i zaprzysięgli malkontenci (jak — nie wytykając — Spektruś), takim należy w nagrodę za dotychczasowe zasługi udzielić bezterminowego urlopu wypoczynkowego i ogłosić jeden wielki CZAS RELAKSU...

Podglądał:
Eugeniusz MLECZAK



Ryt. Michał Przybyłowski

Krzyżówka Nr 3



POZIOMO: 2-A zajmuje się elektroniką, 4-G napój alkoholowy, 5-A bizantyjski okręt wiosłowy, 5-I 100 m², 7-H popularny kwiat rosnący w zbożu, 8-A reda, 10-C 220 V, 11-A władca wiatrów, 13-A posiada, 13-D bieg na..., 14-A mała Urszula, 16-A np. odnoszenie się referatu lub ustawy do czegoś.

PIONOWO: A-1 przeciwieństwo kodowania, G-1 starsza siostra pompki, I-1 występuje na scenie, E-2 sakwa, kiesa, C-4 atomowa lub sezonu, I-4 może być cukrowa, D-7 imię żeńskie, J-7 zbieracz, G-8 Hg, H-10 wóz pancerny, B-11 kamień szlachetny, F-12 przedmiot, A-13 rzeka przepływająca przez Dessan, D-13 kijek.

HASŁO: 2-I, 14-I, 7-H, 8-E, 14-A, 2-E, 16-G, 13-E, 6-I, 4-I,
8-F, 10-A, 5-D, 10-E, 8-G, 16-F, 16-F, 16-D, 13-I, 5-G
10-H, 2-I, 2-A, 11-C, 11-A, 6-C.

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca lipca br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Mikrociekawostki



Komputery serii GRiDCase wyprodukowane przez firmę GRiD Systems, Co. są utrzymane w standardzie IBM. Ich wielkość odpowiada rozmiarom małej teczki. Poszczególne typy różnią się między sobą zastosowanymi monitorami od zwykłych, ciekłokrystalicznych w cenie 2975 $ do gazowo-plazmowych za 4350 $. Pracują opierając się na 16-bitowym mikroprocesorze 80C86. Standardowy akumulator umożliwia ciągłą pracę przez 5 godzin dla komputera z ciekłokrystalicznym monitorem i 1 godzinę dla komputera z monitorem gazowo-plazmowym. GRiDCase wyposażony jest w stacje dysków dla 3,5-calowych dyskietek, z których każda może zawierać 720 KB. (c.)



Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera, Ryszard Rogoń. Adres redakcji: 00—950 Warszawa ul. Grzybowska 77, telefon 20-21-27 i 359-34. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 7728. Nr ind. 382809 P-77

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 4/1986 ISSN 0860-27 94



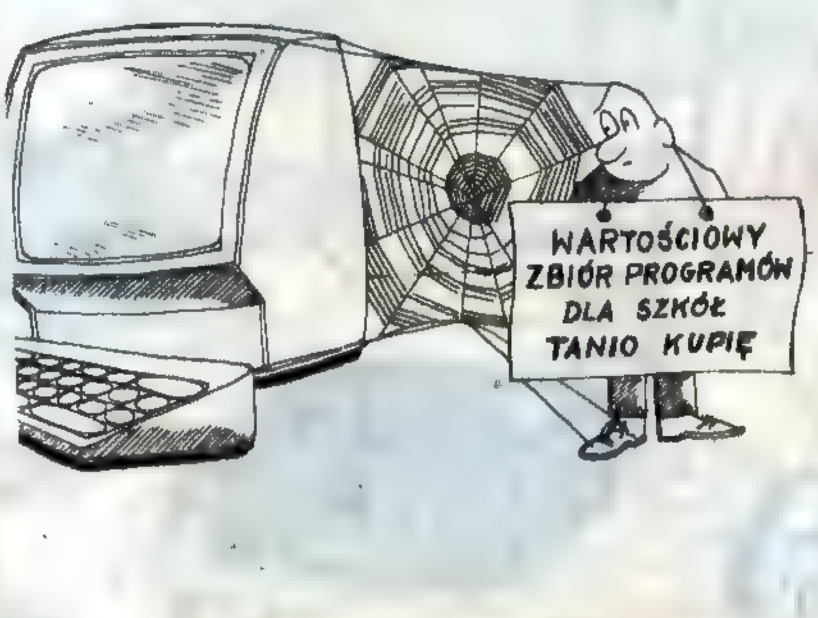
Trwają wakacje. Do tradycyjnych form letniego wypoczynku dołączyła jeszcze jedna — wakacje z komputerem. To frajda ogromna, a dla wielu jedyna okazja do pierwszego kontaktu z tym urządzeniem. Mikrokomputer pod namiotem, na leśnym biwaku, wszędzie, to też osiągnięcie techniki. Tacy już widać jesteśmy: jeśli nowość nam się podoba, ulegamy jej całkowicie: wszystko dla komputerów. W tym szaleńczym pędzie za nowoczesnością młodzi miłośnicy informatyki jakby zapomnieli, że to nie oni jej, ale to właśnie ona im służyć powinna.

Od września informatyka znajdzie się w programie nauczania szkół średnich — nie wszystkich, co prawda, ale pierwszy krok już został zrobiony. Teraz nowoczesną technikę trzeba poznać, zrozumieć i nauczyć się ją wykorzystywać. Wiadomo, jak jest ze szkolnym przedmiotem. Wszystko zależy od nauczycieli, a tych niestety nie mamy tylu, ilu do nauczania informatyki potrzeba. Wystarczy też chwila nieuwagi pedagoga i z dotychczasowej przyjemności zrobi się smutny obowiązek, a w konsekwencji przybędzie jeszcze jeden trudny przedmiot, męczący po nocach koszmarnymi snami, w których krwiożercze bajty pożerać będą niewinne bity.

Ale informatyka to krok pierwszy. Krok drugi, to mikrokomputery w szkole nie jako przedmiot, ale pomoc dydaktyczna, środek umożliwiający sprawniejszą naukę. Uczeń nie musi przecież znać tajników komputerowej techniki. Jego wiedza to sprawne posługiwanie się kilkoma klawiszami. Wszystko powinno zależeć od jakości programów edukacyjnych, które oprócz wartości merytorycznych przekazywanych treści muszą być konstrukcjami przejrzystymi, umożliwiającymi łatwą obsługę, dokładnie instruującymi użytkownika o tym, jak powinien zachować się przed komputerem.

W myślach o tej przyszłości umiaru nie ma. Trwają seminaria, sesje, konferencje, o tym, czym elektronika powinna być w szkole. Dobrze, że teoria wyprzedza praktykę. Jeszcze lepiej, kiedy udaje się ją wprowadzić w życie, a taką szansę daje wyprodukowanie, być może taniego, mikrokomputera we wrocławskiej ELWRO i niesłabnące zainteresowanie informatyką wśród młodzieży i dzieci.

REDAKCJA

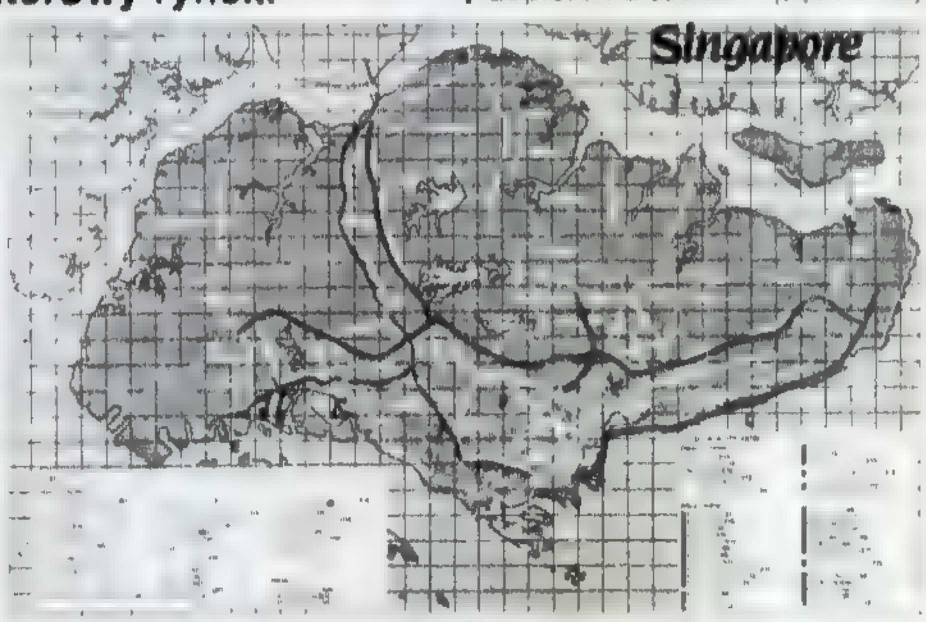




W komputerowym raju?

(Korespondencja własna z Singapuru)

Co, nie przywiozłeś stamtąd komputera??? Frajerze, przecież one są tam bajecznie tanie! Jak można zaprzepaścić taką wspaniałą okazję? Czy ty w ogóle potrafisz myśleć i liczyć? Jeśli tobie nie zależy, to przynajmniej pomyślałbyś o kolegach. **Tak mniej więcej przed rokiem powitał mnie jeden z kolegów, ogarnięty mikrokomputerową pasją. Kiedy więc niedawno nadarzyła mi się ponownie okazja podróży do Singapuru, postanowiłem zarówno pomyśleć, jak i policzyć, pomny tych gromkich reprymend. Najlepiej jednak „liczy" się na miejscu, „zanurzając" się w komputerowy rynek.**



Pierwsze zaskoczenie wiąże się z tym, że owo „zanurzenie się" nie jest wcale takie proste. Bo choć miasto-państwo, jakim jest Singapur wygląda od pierwszego wejrzenia na jeden olbrzymi — i trzeba to również przyznać — olśniewający supermarket, gdzie sprzedaje się dosłownie wszystko, to znalezienie się w komputerowym raju nastręcza pewne trudności. Trzeba po prostu wiedzieć, gdzie się udać. Nie chcąc nadwerężać cierpliwości Czytelników „IKS-a", powiem więc od razu, że składa się on z trzech miejsc godnych uwagi i odwiedzenia: FUNAN CENTRE mieszczącego się przy North Bridge Road 109, PEOPLE'S PARK CENTRE przy Upper Cross Street 101 i SIM LIM TOWER, znajdującego się przy Jalan Besar 10. I to właśnie w takiej kolejności.

Gdy już wiemy gdzie, wybierzmy się zatem do jednego z nich, najokazalszego i najbardziej renomowanego wśród polskich „turystów", a za taki uchodzi Funan Centre. Zresztą jest to bez znaczenia również z innego powodu. Jak bowiem nietrudno było się później zorientować, dystrybucją sprzętu komputerowego wśród pokaźnej liczby firm handlowych i salonów sprzedaży zajmuje się kilku głównych jego dostawców. Oni też określają jego podstawową cenę, poniżej której nie zejdzie żaden sprzedawca, mimo niezaprzeczalnych walorów osobistych i zdolności kupieckich nabywcy. To, co sprzedający opuści z ceny wywoławczej oznacza dokładnie pomniejszenie o tyle jego zysku. Często bywa i tak, że gdy klient jest bardzo cierpliwy i namolny w targowaniu się o cenę, a targowanie się jest w Singapurze na porządku dziennym przy wszelkich zakupach i nawet wskazane ze względów obyczajowych, wówczas sprzedawca sięga do specjalnej książki z cenami lub też telefonicznie kontaktuje się ze swoim dostawcą, czy aby nie przeholował przy opuszczaniu. Wynika więc z tego generalny wniosek, że cena podstawowa sprzętu komputerowego w różnych punktach jest stała, różni się natomiast zyskiem, jaki chce osiągnąć sprzedający. Tak więc identyczna drukarka, monitor czy klawiatura ze stacją dysków mogą się różnić ceną w zależności od miejsca kupna o 20, 50 a czasem i nawet 100 dolarów singapurskich. Ten ostatni przypadek należy jednak do rzadkości. Niezorientowanym i dla lepszego zrozumienia tego, o czym będzie później, winien jestem informację, że za jednego dolara amerykańskiego płacono w banku 2,17 dolara singapurskiego.

Uzbrojeni w te podstawowe dane możemy wreszcie przekroczyć progi Funan Centre. Właściwie szklano-betonowy wieżowiec o nowoczesnej sylwetce nie wyróżnia się niczym szczególnym wśród budowli, jakich tu w Singapurze jest wiele. Dokładnie rzecz biorąc nie ma także progów tylko lśniący marmurem, gładki jak stół hall, ale już podmuch zimnego klimatyzowanego powietrza wciąga nas do wnętrza. I tu czeka na nas pierwsze rozczarowanie. W środku fontanna, rozstawione kawiarniane stoliki, rozświetlone witryny sklepów ze sprzętem fotograficznym, radiotechnicznym, ciuchami wszelkiej maści, za to żadnych komputerów. Ktoś mało odporny psychicznie gotów więc natychmiast zrezygnować. Nic bardziej błędnego. Trzeba bowiem wiedzieć, że komputery ulokowały się dopiero na szóstym piętrze tego wielobranżowego supermarketu. A więc winda, wciśnięty guziczek z cyfrą 6, i jedziemy. Tak oto dostajemy się do krainy marzeń niejednego młodego człowieka.

Bo też rzeczywiście na tym jednym piętrze jest tyle sklepów z komputerowymi cudeńkami, ile bez mała wszystkich stoisk w warszawskich domach towarowych „Centrum". Pierwsze wrażenie — dominacja, i to zdecydowana, produktów japońskich. Potentata takiego jak amerykański IBM, kontrolujący do niedawna 80 proc. komputerowego rynku, niemal nie widać. Nie znaczy to wcale, by wyrobów tego renomowanego koncernu nie było. Nic podobnego, są.

Ale w swej masie przytłaczają je urządzenia z magiczną tabliczką Made in Japan oraz równie często sygnowane przez Tajwan, Hongkong czy Koreę Południową. Wystarczy bowiem powiedzieć, że obroty japońskiego giganta komputerowego FUJITSU LIMITED w sprzedaży w roku ubiegłym sięgnęły sumy prawie 7 miliardów dolarów. Tak-

dokończenie na str. 4

dokończenie ze str. 3

ze Tajwan, produkujący głównie na japońskiej licencji, zwiększył w ostatnich 3 latach swój eksport sprzętu komputerowego 4-krotnie, do około 2 miliardów dolarów rocznie, stając się czołowym eksporterem w tej branży. Takie firmy jak Compaq, Leading Edge, Epson, Panasonic, Kaypro czy Tandy są dziś w stanie zaopatrywać rynki w komputery osobiste tańsze i porównywalne parametrami z produktami IBM. Na przykład wyprodukowany w Korei Płd komputer Leading Edge wyposażony jest w dwie stacje dysków, większą pamięć operacyjną niż standardowy IBM PC, edytor tekstu oraz 15-miesięczną gwarancję. Te właśnie tendencje, jak w soczewce odbijają się na komputerowym rynku singapurskim.

Przejdźmy zatem do konkretów najbardziej interesujących, a więc cen. W salonach sprzedaży różnych firm w Funan. Nic bowiem bardziej frapującego, jak to za ile można nabyć komputer domowy właśnie tam. Poczynić jednak wypada istotną uwagę, iż wszystkie ceny zanotowane dla Czytelników „IKS-a" są cenami wywoławczymi, a więc bez targowania się, tudzież wyrażenia chęci nabycia kilku jednostek, co także skutecznie obniża koszty. Przedmiot marzeń niejednego z komputerowych zapaleńców, czyli kompletny zestaw IBM PC XT ze stacją dysków, monitorem, twardym dyskiem 640 KB, drukarką itd. można nabyć już za ok. 1800 dol. USA. Bez drukarki ten sam komputer kosztuje o 200—250 dol. mniej. Tych, którym i tak suma ta wydaje się zawrotna, pragnę pocieszyć, że kompatybilne hardware'owo i software'owo komputery np. Macintosh model Mac Charlie można nabyć o 100 - 200 dol. taniej. Słaba to co prawda pociecha, zwłaszcza dla mniej zasobnych finansowo kieszeni, ale jednak. Wszyscy bowiem zdajemy sobie sprawę, że twardy dysk jest tym elementem, który w decydujący sposób kształtuje ceny komputerów tej klasy. Niemniej specjaliści przewidują, że pamięci masowe będą taniały, a tym samym i ogólny koszt komputerów. W tym roku w Singapurze dodatkowy wpływ na wyższe ceny sprzętu komputerowego ma niewątpliwie także drastyczny wzrost kursu japońskiego jena (o 41 proc.) w stosunku do amerykańskiego dolara. A przecież reguły cen dyktuje w tym regionie Japonia.

Są oczywiście i komputery znacznie tańsze. Znany i u nas popularny Commodore C-64 ze stacją dysków i zielonym monitorem można kupić już za 630—650 dol. singapurskich. Posiadaczem innego modelu C1 300 tej samej firmy o pojemności pamięci 128 KB można stać się za 1100 dol. sing. Dla porównania mikrokomputer Sinclaira w wersji Spectrum ZX z pamięcią 64 KB kosztuje zaledwie 260 dol. sing. Wśród niezbyt drogich komputerów domowych obecna jest także na tutejszym rynku firma Atari ze swym modelem 800 XL, wycenionym wraz ze stacją dysków na 500 dol. sing. Bardzo ekspansywny także cenowo wydaje się w Singapurze Schneider CPC 6128, który konkuruje skutecznie z innymi firmami. Za jego wersję z zielonym monitorem trzeba zapłacić 1000, a z kolorowym 11[illegible] dol. sing. Prawdziwą furorę na tutejszym rynku robią komputery firmy Epson Corporation rodem z Nagano, zwłaszcza zaś Epson QX-16 oraz produkty innej renomowanej firmy japońskiej Seikosha Co., Ltd oznaczone jako Seiko seria 8600. Te ostatnie mają już wbudowane twarde dyski o pojemności 10 MB, 20 MB lub 50 MB. Większość komputerów osobistych wygląda tak samo, ma zbliżone parametry, pracuje ze standardowym oprogramowaniem i niewiele różni się w cenie. Do tego można się przyzwyczaić także w Singapurze.

Znacznie trudniej natomiast przyzwyczaić się do obfitości urządzeń peryferyjnych. Dosłownie każda firma handlująca sprzętem komputerowym jest nimi zawalona. Nic dziwnego, skoro Japończycy skierowali swe eksportowe wysiłki właśnie w tym kierunku. Drukarki, stacje dysków, monitory, dyskietki dostępne na singapurskim rynku a produkowane przez Japończyków są najwyższej klasy. A ceny, gdyż o nich mówimy od dłuższego czasu, wydają się interesujące. Wymieńmy kilka przykładowo. Drukarka do Schneidera, ciesząca się w naszym kraju niesłabnącym powodzeniem, Seikosha SP-1000 w wersjach A lub AP kosztuje 440-450 dol. sing., a wyprodukowana w Hongkongu czy Korei Płd drukarka do tego samego typu mikrokomputera Amstrad DMP-2000 jest do nabycia w cenie 450-460 dol. sing. Z kolei taśmy wymienne do Seikoshy SP-1000 kosztują 15 dol. sing., zaś do SP-500 są o 2 dol. sing. tańsze. A dyskietki do Amstrada, by pozostać przy tym samym typie komputera CF-2 wycenione są na 7 dol. sing. Inna drukarka, współpracująca z Macintoshem czy Epsonem, legitymująca się równoległym wejściem i prędkością druku 120 znaków na sek., oznaczona symbolem National Super 5, ceniona jest na 480-500 dol. sing. Zbliżoną cenę wykazują się też drukarki firmy Brother Industries Ltd, które w dużym wyborze typów i pokaźnych ilościach znajdują się na półkach singapurskich firm handlowych.

Nie wspominam już w tym miejscu o okablowaniu, bo w Funan Center są punkty wykonujące na poczekaniu wszystkie możliwe połączenia do dowolnego typu komputera za naprawdę niską opłatą. Obfitość programów, książek, instrukcji, fachowych periodyków z dziedziny informatyki jest doprawdy oszałamiająca. Specjalistyczna obsługa sprawia częściej wrażenie twego doradcy niż sprzedawcy, który stara się wyłącznie „upłynnić" posiadany towar. Wszędzie podłączony jest sprzęt do demonstracji, przy którym potencjalny klient może spędzić i cały dzień, wychodząc w końcu bez dokonania zakupu. Przy wyjściu usłyszy wtedy grzeczne zaproszenie, by odwiedził firmę w dniu następnym, gdyż może łatwiej będzie mu się zdecydować. Bo każdy tam zdaje sobie sprawę, że kupno komputera jest zawsze mniejszą lub większą inwestycją i pośpiech nie jest przy tym wskazany. Tym bardziej że zakupiony komputer winien spełniać określone zadania funkcjonalne nie zaś służyć wyłącznie dziecięcej zabawie. Mimo to miejscowych i zagranicznych klientów nie brakuje w Funan Centre. W sklepie firmy Syntax Computer Pte. Ltd. mogłem na przykład bez trudu zauważyć przygotowany do wysyłki kompletny zestaw Amstrada CPC 6128. Z nalepek adresowych wynikało, że zawędruje do dalekiego Madrasu w Indiach. Jeśli zaś o mnie chodzi, i w tym roku wróciłem bez komputera.

Jerzy MARKOWSKI



Rekordy o zmiennej długości na ZX Spectrum

Współczesne mikrokomputery domowe nie oferują użytkownikowi zbyt dużo pamięci RAM. W najpopularniejszych z nich przy pracy w BASIC-u dostępne jest — w zależności od komputera — około 39-43 KB. Jest to ilość niewystarczająca, szczególnie gdy chcemy operować większymi zbiorami danych.

Istniejące ograniczenie zmusza użytkownika do racjonalnego i oszczędnego gospodarowania pamięcią. W mikrokomputerze ZX Spectrum jedną z dróg prowadzących do przedstawionego wyżej celu jest wykorzystanie nieograniczonej długości zmiennej znakowej do operacji na rekordach o zmiennej długości.

Wyobraźmy sobie, że chcemy napisać program katalogujący nasz księgozbiór (lub zbiór posiadanych płyt, kaset itp.). Dla każdej pozycji należy zapamiętać nazwisko i imię autora oraz tytuł książki. Nazwiska autorów i tytuły mogą mieć różnorodną długość. Gdybyśmy chcieli przechowywać nazwiska w tablicy znakowej, trzeba by zadeklarować tablicę o rozmiarze (n, m), gdzie:
n — sumaryczna liczba pozycji do skatalogowania,
m — liczba znaków w najdłuższym nazwisku (około 25 znaków).

Podobne rozumowanie można przeprowadzić dla tytułów. Przechowywanie nazwisk i tytułów w tablicach znakowych okazuje się więc bardzo nieefektywne, gdyż duża część tych tablic pozostaje niewykorzystana.

Nazwiska i tytuły tworzą rekordy o zmiennej długości. Ekonomicznym — z punktu widzenia pamięciochłonności — rozwiązaniem jest przechowywanie nazwisk w jednej zmiennej znakowej, np. n$, a tytułów w drugiej, np. t$. Wykorzystujemy tu pewną osobliwość ZX Spectrum, polegającą na tym, że długość zmiennej znakowej jest nieograniczona (ściślej mówiąc jest ograniczona tylko pojemnością pamięci). Nazwiska i tytuły dopisywane są do zmiennych n$ i t$ w kolejności ich wprowadzania przy użyciu instrukcji

```
LET n$ = n$ + u$ + ";"
LET t$ = t$ + v$ + ";"
```

gdzie: u$ — kolejne nazwisko,
v$ — kolejny tytuł.

Znak średnika jest separatorem, oddzielającym kolejne nazwiska lub tytuły od siebie.

Dostęp do tak zapisanych rekordów możliwy jest dzięki zapamiętaniu początków wszystkich rekordów. Najlepiej użyć do tego dwóch tablic znakowych (odrębne tablice dla nazwisk i tytułów), np. a$ (n, 3) i b$ (n, 3). Zakładam tu, że z każdym tytułem skojarzone jest tylko jedno nazwisko, pozwalające dotrzeć do danego tytułu (fizycznie zapisanych może być więcej nazwisk, ale pamiętany jest tylko początek pierwszego). Aby uniknąć tego ograniczenia, należy zwiększyć wymiar tablicy a$, np. a$ (n + 1, 5 n, 3).

Takie rozwiązanie pozwala na dostęp do rekordów zarówno wg nazwisk, jak i tytułów. Aby dostęp do rekordów był szybki (np. metodą podziałów połówkowych) tablice a$ i b$ muszą być posortowane w kolejności alfabetycznej. Jeżeli zależy nam na szybkim dostępie tylko wg nazwisk, wystarczy aby posortowana była tylko tablica a$. Dwa pierwsze bajty w każdym wierszu tablic a$ i b$ użyte będą do przechowywania adresu początku rekordu w zmiennej znakowej odpowiednio n$ i t$, bajt trzeci jest odsyłaczem do tablicy organizacyjnej r$ (n, 4). Tablica ta wiąże nazwisko z odpowiednim tytułem i odwrotnie. Zawiera ona w każdym wierszu adres początku nazwiska w zmiennej n$ i adres początku tytułu w zmiennej t$ skojarzonego z danym nazwiskiem. Adresy początków rekordów przechowywane są w postaci znakowej na dwóch bajtach (jest to oszczędniejsze, gdyż liczby zajmują 5 bajtów):

adres = 2ØØ bajt 1. + bajt 2.

Znając adres początku rekordu (x) możemy go zapisać w postaci znakowej przykładowo dla i — tego wiersza tablicy a$:

```
LET a$ (i, 1) = CHR$ (INT (x) 2ØØ) — 32
LET a$ (i, 2) = CHR$ (x—2ØØ (CODE a$ (i, 1) — 32) + 32)
```

Adresy można łatwo rozkodować wg wzoru:

```
LET adres = 2ØØ (CODE a$ (i, 1) — 32) * (CODE a$ (i, 2) — 32)
```

Dodanie stałej 32 przy kodowaniu adresów pozwala na przesunięcie kodu znaku do przedziału 32, 232. Znaki o kodach mniejszych od 32 nie dają się bowiem wyświetlić, co utrudnia testowanie programu. Również mnożnik 2ØØ został przyjęty arbitralnie z myślą o ułatwieniu testowania programu (można przyjąć dowolny inny mnożnik = 223). Zapewnia on przestrzeń adresową w przedziale (Ø, 44823) co w praktyce i tak znacznie przekracza potrzeby. Odsyłacze do tablicy organizacyjnej mogą być pamiętane na jednym bajcie, gdy założymy, że liczba rekordów nie przekroczy 223.

Wtedy przykładowo:

```
a$ (i, 3) = CHR$ (n + 32)
n= CODE (a$ (i, 3)) — 32
```

gdzie: i — nr wiersza w tablicy a$,
n — nr wiersza w tablicy r$, pod którym zapisane są informacje dotyczące nazwiska i tytułu.

Ograniczenie liczby rekordów do 223 wynika z prób przeprowadzonych na mikrokomputerze ZX Spectrum 48K.

Przykładowy schemat powiązań w omówionych strukturach danych przedstawiony został poniżej (uwaga: w celu zwiększenia przejrzystości wszystkie adresy podane są dziesiętnie, a nie znakowo):

n$ = "Naur. P.; Turski. W.M.; Strizenec. M.;"
t$ = "Zarys, metod, informatyki; Propedeutyka, informatyki; System: człowiek — komputer;"

Tablica a$

adres początku nazwiska w zmiennej n$ (2 bajty)	odsyłacz do tablicy r$ (1 bajt)
1	1
21	2
9	9

dokończenie ze str. 5

Tablica b$

adres początku tytułu w zmiennej t$ (2 bajty)	odsyłacz do tablicy r$ (1 bajt)
25 50 1	3 2 1

Tablica r$

adres początku nazwiska w n$ (2 bajty)	adres początku tytułu w t$ (2 bajty)
1 21 9	1 50 25

Wyszukiwanie informacji wg nazwiska odbywa się zgodnie ze schematem:

a) znaleźć n-nr wiersza w tablicy a$, w którym zapisany jest adres początku podanego nazwiska,

b) odczytać p — odsyłacz do tablicy organizacyjnej (odsyłacz ten zapisany jest w a$ (n, 3)),

c) odczytać z tablicy organizacyjnej adres początku tytułu — ap (adres ten zapisany jest w elementach r$ (p, 3) i r$ (p, 4)),

d) odczytać tytuł ze zmiennej t$, począwszy od elementu ap aż do pierwszego znaku średnika.

Wyszukiwanie informacji wg tytułu odbywa się podobnie, przy czym odsyłacza do tablicy organizacyjnej szukamy i odczytujemy go z tablicy b$.

Zaletą przedstawionego rozwiązania jest całkowite wykorzystanie pamięci oraz szybki dostęp do rekordów zarówno wg nazwisk, jak i tytułów (przy założeniu, że obydwie tablice, tj. a$ i b$ są posortowane). Wadą jest konieczność deklarowania dodatkowych tablic do pamiętania odsyłaczy i adresów początków rekordów oraz wydłużenie obliczeń, wynikające z kodowania i dekodowania adresów.

Rozwiązanie to jest jednak eleganckie i wykorzystuje typowe metody informatyczne (odsyłacze). Jest godne polecenia we wszystkich programach, gdzie mogą wystąpić rekordy o zmiennej długości, np. przy budowie wielojęzycznych słowników, wszelkiego rodzaju katalogów itp.

Janusz MORBITZER

PROGRAM 16

SPECTRUM

Program „Biblioteka" napisany w języku BASIC na ZX Spectrum umożliwia skomputeryzowanie domowej biblioteki. Informacje związane z jedną książką (nazwisko lub nazwiska autora oraz tytuł) tworzą rekord.

Struktura wewnętrzna programu obejmuje dwie zmienne znakowe do przechowywania nazwisk i tytułów oraz tablice odsyłaczy: pamiętane są w nich adresy początków nazwisk i początków tytułów oraz powiązania autor — tytuł. Jest to przykład wykorzystania rekordów o zmiennej długości.

Maksymalna liczba rekordów nie może przekroczyć 200 (ponieważ niektóre książki mają więcej niż jednego autora, maksymalna liczba nazwisk autorów wynosi 230). Oczywiście, istnieje możliwość zrezygnowania z tych ograniczeń. Wiąże się to z niewielkimi zmianami w programie: adresy odsyłaczy do tablicy organizacyjnej muszą być pamiętane na dwóch bajtach. Celowe byłoby wówczas zwiększenie pamięci przez usunięcie wszystkich komentarzy.

Program umożliwia założenie zbioru, dopisywanie i kasowanie rekordów, wyszukiwanie informacji oraz uzyskanie danych ogólnych o zbiorze (liczba tytułów, liczba nazwisk autorów, wielkości wolnej pamięci). Trzy pierwsze operacje to tzw. *czynne korzystanie* **ze zbioru (dokonywanie zmian w zbiorze).** *Tryb czynny* **dostępny jest tylko po podaniu hasła — użytkownik może zmienić hasło (wiersz 240), pamiętając jednocześnie o zmianie wartości zmiennej pp, w której pamiętana jest długość zadeklarowanego hasła (wiersz 230).**

Nazwiska i tytuły należy pisać zaczynając od dużej litery. Pierwszy znak musi należeć do alfabetu angielskiego, kolejne mogą być również z zakresu alfabetu polskiego (ą, ć, ę itd.).

Informacje można wyszukiwać wg nazwiska lub tytułu. Obydwie metody zapewniają jednakowo szybki dostęp. W wypadku książek posiadających kilku autorów wyszukuje się nazwiska przez podanie dowolnego z nich. Należy zwrócić uwagę, aby przy kasowaniu rekordu uniknąć niejednoznaczności, tzn. jeżeli nazwisko danego autora występuje w zbiorze kilkakrotnie, należy posłużyć się tytułem książki (o ile oczywiście jest on jednoznaczny). Przy wyszukiwaniu informacji problem niejednoznaczności został rozwiązany w ten sposób, że drukowane są wszystkie rekordy z zadanym kluczem wyszukiwania (można wyszukiwać np. wszystkie książki, których nazwiska autorów zaczynają się od zadanej litery lub grupy liter).

Program ma wbudowane mechanizmy w dużej mierze zapewniające jego niezawodność (sprawdzenie legalności operacji, poprawności niektórych wprowadzanych danych itp.). Przed uruchomieniem programu po raz pierwszy należy z klawiatury wprowadzić instrukcję LET wr=0. Instrukcja LOAD " " CODE w wierszu 5. służy do automatycznego wczytania bloku zawierającego polskie znaki diakrytyzowane (ą, ć, ę itd.). Mniej wtajemniczeni czytelnicy mogą tę instrukcję pominąć i posługiwać się wyłącznie znakami alfabetu angielskiego.

Opis ważniejszych zmiennych i tablic zawarty jest w komentarzu w wierszu 1036.

```
   1 REM "Biblioteka"
 ****************
 (c) J.MORRITZER
   5 LOAD "znaki-pol"CODE
  10 BEEP .4,10: POKE 23609,50
  15 CLS : LET i=3: PRINT AT i,3
;"Program umozliwia zalozenie"
;AT i+2,0;"katalogu ksiazek, szy
bkie wyszu-";AT i+4,0;"kiwanie i
nformacji wg nazwiska";AT i+6,
0;"autora lub tytulu ksiazki,akt
ua-";AT i+8,0;"lizacje zbioru
oraz uzyskanie";AT i+10,0;"info
rmacji ogólnych o zbiorze";AT
 i+12,0;"( ilosc rekordów, ilos
c wolnej";AT i+14,0;"pamieci it
p.)"
  20 GO SUB 150
  90 DEF FN f(n)=200*(CODE r$(n,
1)-32)+(CODE r$(n,2)-32)
  92 DEF FN g(n)=200*(CODE r$(n,
3)-32)+(CODE r$(n,4)-32)
  95 LET zm=0
 100 LET kas=0: LET dr=0: BEEP .
4,10: INK 0: CLS : PRINT AT 2,6;
 BRIGHT 1;"Wybierz tryb pracy:":
 PRINT AT 3,6;"_______________
__"

 105 LET i=6: LET j=5: PRINT AT
i,j;"1-zakladanie zbioru";AT i+2
,j;"2-dopisywanie rekordów";AT i
+4,j;"3-kasowanie rekordów";AT i
+6,j;"4-wyszukiwanie rekordów";A
T i+8,j;"5-wydruk rekordów";AT i
+10,j;"6-informacje o zbiorze";A
T i+12,j;"7-koniec"
 110 LET t$=INKEY$: IF LEN t$=0
THEN GO TO 110
 115 IF t$<"1" OR t$>"7" THEN GO
 TO 110
 120 IF t$<"4" THEN LET zm=1: GO
 SUB 200
 125 GO TO 1000*VAL t$
 150 REM podprogram
 155 PRINT #1;AT 0,4; BRIGHT 1;
FLASH 1;"Nacisnij dowolny klawis
z": PAUSE 0: RETURN
 170 REM podprogram
 175 BEEP .1,20: PRINT #1;AT 0,2
; BRIGHT 1;"informacje zapisano
do zbioru": PAUSE 50: RETURN
 180 REM podprogram
 185 PRINT #1;AT 0,0; BRIGHT 1;
FLASH 1;"m - menu,pozostale - ko
ntynuacja": PAUSE 0: IF INKEY$="
m" THEN GO TO 100
 190 RETURN

 200 REM haslo
 205 BEEP .4,5: CLS : PRINT AT 6
,4; BRIGHT 1;"Dostep czynny do z
bioru";AT 9,4;"- wymaga podania
hasla!";AT 15,10;"PODAJ HASLO:";
AT 18,11;"           "
 210 LET q$=""
 215 IF INKEY$<>"" THEN GO TO 21
5
 220 IF INKEY$="" THEN GO TO 220
 225 LET h$=INKEY$: BEEP .05,10:
 IF CODE h$<32 THEN GO TO 215
 230 LET q$=q$+h$: LET pp=LEN q$
: IF pp=11 THEN GO TO 240
 235 GO TO 215
 240 IF q$="scisletajne" THEN PR
INT AT 18,8; INVERSE 1;"haslo pr
awidlowe": BEEP .4,20: PAUSE 50:
 RETURN
 245 CLS : PRINT AT 8,2; BRIGHT
1;"Dostep do zbioru niemozliwy";
AT 13,5;"- podano bledne haslo!"
 250 FOR i=1 TO 30: BEEP .1,0: B
EEP .15,15: NEXT i
 255 GO SUB 150: GO TO 100
 260 REM podprogram
 265 LET ll=dy: IF dx>dy THEN LE
T ll=dX
 270 LET zb=65535-USR 7962: IF w
r-1=m OR 2*ll+200>zb OR al>=m+30
 THEN CLS : PRINT AT 8,4; BRIGHT
 1;"Dopisywanie niemozliwe!": P
RINT AT 12,9; BRIGHT 1;"(zbiór
pelny)": BEEP .8,0: GO SUB 150:
GO TO 100
 275 RETURN
 300 REM wczytanie rekordu
 305 BEEP .4,10: DIM p$(m+30,3)
 310 GO SUB 260: LET ddx=dX: LET
 ddy=dy: LET jj=1
 315 CLS : PRINT AT 3,1; BRIGHT
1;"Podaj liczbe autorów ksiazki:
"

 320 INPUT "              . "; LINE l
$: LET d=LEN l$: IF d=0 THEN GO
TO 320
 325 FOR i=1 TO d: IF l$(i)>="0"
 AND l$(i)<="9" THEN NEXT i: LET
 l=VAL l$: GO TO 335
 330 GO TO 320
 335 IF l<=0 OR l>10 THEN GO TO
320
 340 PRINT AT 3,30; BRIGHT 1;l
 345 FOR i=1 TO l
 350 PRINT AT 6,4; BRIGHT 1;"Pod
aj nazwisko ";i;". autora:"
 355 INPUT "   ";(i);". "; LI
NE s$: IF LEN s$=0 THEN GO TO 35
5
 360 PRINT AT 9+i,5;i;".";s$
 365 IF i=l AND l<>1 THEN LET s$
=s$+"#;": GO TO 375
 367 IF l=1 THEN LET s$=s$+";":
GO TO 375
 370 IF i>1 THEN LET s$=s$+"#,":
 GO TO 375
 372 LET s$=s$+","
 375 GO SUB 550: GO SUB 450: GO
SUB 170

 380 NEXT i
 385 LET jj=2: CLS : PRINT AT 10
,6; BRIGHT 1;"Podaj tytul ksiazk
i:"
 390 INPUT LINE s$: LET dd=LEN s
$: IF dd=0 THEN GO TO 390
 395 LET d=(32-dd)/2-1: IF d<0 T
HEN LET d=0
 400 PRINT AT 13,d;"""";s$;"""":
 LET s$=s$+";": GO SUB 550: GO S
UB 500: GO SUB 170
 405 GO SUB 420: LET ddX=ddX+dx:
 LET ddy=ddy+dy: LET wr=wr+1
 410 RETURN
 420 REM zapis do tablicy organi
zacyjnej
 425 LET i$=CHR$ (32+INT (ddX/20
0)): LET j$=CHR$ (32+ddX-200*(CO
DE i$-32)): LET r$(wr,1)=i$: LET
 r$(wr,2)=j$
 430 LET i$=CHR$ (32+INT (ddy/20
0)): LET j$=CHR$ (32+ddy-200*(CO
DE i$-32)): LET r$(wr,3)=i$: LET
 r$(wr,4)=j$
 435 RETURN
 450 REM zapis nazwiska do tabl.
odsylaczy
 455 LET n=poz: LET m$=CHR$ (32+
wr)
 460 LET dl=LEN s$: LET x$=x$+s$
 465 LET u$=CHR$ (32+INT (dx/200
)): LET v$=CHR$ (32+dx-200*(CODE
 u$-32)): LET dX=dX+dl: LET al=a
l+1
 470 IF n>=al THEN LET a$(n,1)=u
$: LET a$(n,2)=v$: LET a$(n,3)=m
$: GO TO 490
 475 FOR k=n TO al-1: LET p$(k)=
a$(k): NEXT k
 480 LET a$(n,1)=u$: LET a$(n,2)
=v$: LET a$(n,3)=m$
 485 FOR k=n+1 TO al: LET a$(k)=
p$(k-1): NEXT k
 490 RETURN
```

```
 500 REM zapis tytulu do tabl. o
dsyłaczy
 505 LET n=poz: LET m$=CHR$ (32+
wr)
 510 LET dl=LEN s$: LET y$=y$+s$
 515 LET u$=CHR$ (32+INT (dy/200
)): LET v$=CHR$ (32+dy-200*(CODE
 u$-32)): LET dy=dy+dl: LET bl=b
l+1
 520 IF n>=bl THEN LET b$(n,1)=u
$: LET b$(n,2)=v$: LET b$(n,3)=m
$: GO TO 540
 525 FOR k=n TO bl-1: LET p$(k)=
b$(k): NEXT k
 530 LET b$(n,1)=u$: LET b$(n,2)
=v$: LET b$(n,3)=m$
 535 FOR k=n+1 TO bl: LET b$(k)=
p$(k-1): NEXT k
 540 RETURN
 550 REM określenie miejsca wsta
wienia informacji
 555 LET gdzie=1: IF jj=2 THEN G
O TO 570
 560 IF al=0 THEN LET poz=1: GO
TO 580
 565 GO SUB 600: GO TO 580
 570 IF bl=0 THEN LET poz=1: GO
TO 580
 575 GO SUB 750
 580 LET gdzie=0: RETURN
 600 REM wyszukiwanie w a$
 605 LET x=1: LET y=al: LET z=LE
N s$
 610 IF (y-x)>=0 THEN GO TO 625
 615 IF gdzie=1 THEN LET poz=x:
RETURN
 620 GO SUB 720: GO TO 100

 625 LET w=1+INT ((x+y-1)/2): LE
T a=200*(CODE a$(w,1)-32)+(CODE
a$(w,2)-32)
 630 IF a+z-1>LEN x$ THEN LET t$
=x$(a TO ): GO TO 640
 635 LET t$=x$(a TO a+z-1)
 640 IF t$=s$ THEN LET poz=w: GO
TO 655
 645 IF t$<s$ THEN LET x=w+1: GO
TO 610
 650 LET y=w-1: GO TO 610
 655 IF gdzie=1 OR kas=1 THEN RE
TURN
 660 BORDER 7: PAPER 7: INK 0: C
LS : LET n=CODE a$(w,3)-32: LET
p=FN f(n): LET q=FN g(n): GO SUB
 920: BEEP .2,10: GO SUB 150
 665 LET i=poz+1: LET j=poz-1
 670 FOR k=i TO al
 675 LET a=200*(CODE a$(k,1)-32)
+(CODE a$(k,2)-32): LET az=a+z-1
: IF az>dx-1 THEN GO TO 690
 677 IF x$(a TO az)=s$ THEN LET
n=CODE a$(k,3)-32: LET p=FN f(n)
: LET q=FN g(n): CLS : GO SUB 92
0: BEEP .2,10: GO SUB 150: GO TO
 685
 680 GO TO 690
 685 NEXT k
 690 FOR k=j TO 1 STEP -1
 695 LET a=200*(CODE a$(k,1)-32)
+(CODE a$(k,2)-32): LET az=a+z-1
: IF az>dx-1 THEN RETURN
 697 IF x$(a TO az)=s$ THEN LET
n=CODE a$(k,3)-32: LET p=FN f(n)
: LET q=FN g(n): CLS : GO SUB 92
0: BEEP .2,10: GO SUB 150: GO TO
 705
 700 GO TO 710
 705 NEXT k
 710 RETURN
 720 REM podprogram
 725 BORDER 7: PAPER 7: INK 0: C
LS : PRINT AT 10,2; BRIGHT 1;"W
zbiorze brak poszukiwanego";AT 1
4,11;"rekordu !": BEEP .5,10: GO
 SUB 150: RETURN
 730 REM podprogram
 735 BORDER 0: PAPER 0: INK 7: C
LS : PRINT AT 10,9; FLASH 1;"PRO
SZE CZEKAC!": RETURN
 750 REM wyszukiwanie w b$
 755 LET x=1: LET y=bl: LET z=LE
N s$
 760 IF (y-x)>=0 THEN GO TO 775
 765 IF gdzie=1 THEN LET poz=x:
RETURN
 770 GO SUB 720: GO TO 100
 775 LET w=1+INT ((x+y-1)/2): LE
T a=200*(CODE b$(w,1)-32)+(CODE
b$(w,2)-32)
 780 IF a+z-1>LEN y$ THEN LET t$
=y$(a TO ): GO TO 790
 785 LET t$=y$(a TO a+z-1)
 790 IF t$=s$ THEN LET poz=w: GO
TO 805
 795 IF t$<s$ THEN LET x=w+1: GO
TO 760
 800 LET y=w-1: GO TO 760
 805 IF gdzie=1 OR kas=1 THEN RE
TURN
 810 BORDER 7: PAPER 7: INK 0: C
LS : LET n=CODE b$(w,3)-32: LET
p=FN f(n): LET q=FN g(n): GO SUB
 920: BEEP .2,10: GO SUB 150
 815 LET i=poz+1: LET j=poz-1
 820 FOR k=i TO bl

 825 LET a=200*(CODE b$(k,1)-32)
+(CODE b$(k,2)-32): LET az=a+z-1
: IF az>dy-1 THEN GO TO 840
 827 IF y$(a TO az)=s$ THEN LET
n=CODE b$(k,3)-32: LET p=FN f(n)
: LET q=FN g(n): CLS : GO SUB 92
0: BEEP .2,10: GO SUB 150: GO TO
 835
 830 GO TO 840
 835 NEXT k
 840 FOR k=j TO 1 STEP -1
 845 LET a=200*(CODE b$(k,1)-32)
+(CODE b$(k,2)-32): LET az=a+z-1
: IF az>dy-1 THEN RETURN
 847 IF y$(a TO az)=s$ THEN LET
n=CODE b$(k,3)-32: LET p=FN f(n)
: LET q=FN g(n): CLS : GO SUB 92
0: BEEP .2,10: GO SUB 150: GO TO
 855
 850 RETURN
 855 NEXT k
 858 RETURN
 860 REM wydruk nazwisk autorów
 865 LET dl=LEN x$
 867 IF p>=dl THEN RETURN
 870 IF x$(p)=";" THEN RETURN
 875 IF dr=1 THEN IF x$(p)="," T
HEN RETURN
 880 IF x$(p)="#" THEN LET p=p+1
: GO TO 870
 885 PRINT x$(p);: LET p=p+1: GO
TO 870
 890 REM wydruk tytułów
 895 LET dl=LEN y$
 897 IF q>=dl THEN RETURN
 900 IF y$(q)=";" THEN RETURN
 905 PRINT y$(q);: LET q=q+1: GO
TO 900
 920 REM wydruk rekordu nr n
 925 PRINT INVERSE 1;AT 4,12; IN
K 4;"AUTOR:": PRINT : PRINT : GO
 SUB 860: PRINT : PRINT : PRINT
: PRINT
 930 PRINT "           ";: PRIN
T INVERSE 1; INK 4;"TYTUL:": PRI
NT : PRINT : GO SUB 890
 935 RETURN
 940 REM podprogram
 945 FOR i=p TO dx-1
 950 IF x$(i)="#" THEN LET wy=0:
RETURN
 955 IF x$(i)="," OR x$(i)=";" T
HEN LET wy=1: RETURN
 960 NEXT i
1000 REM zakładanie zbioru
1005 IF wr<=1 THEN GO TO 1030
```

```
1010 CLS : PRINT AT 1,13; BRIGHT 1;"UWAGA:": PRINT AT 2,13;"———";AT 4,3; INVERSE 1,"Zbiór został już założony!": BEEP .6,10: PRINT AT 6,5;"Ponowne założenie zbioru";AT 8,3;"spowoduje zniszczenie jego";AT 10,3;"starej zawartości.";AT 14,13; BRIGHT 1;"Podaj": PRINT AT 15,13;"————";AT 17,5;"0-powrót do menu";AT 19,5,"1-zakładanie zbioru"
1015 LET t$=INKEY$: IF LEN t$=0 THEN GO TO 1015
1020 IF t$<>"0" AND t$<>"1" THEN GO TO 1015
1025 IF t$="0" THEN GO TO 100
1030 LET m=200: DIM a$(m+30,3): DIM b$(m,3): DIM r$(m,4)
1035 LET x$="": LET y$="": LET wr=1: LET dx=1: LET dy=1: LET al=0: LET bl=0

1036 REM x$-nazwiska autorów,y$-tytuły; a$(i,1),a$(i,2)-adres początku nazwiska w x$; a$(i,3)-odsyłacz do tablicy organizacyjnej; b$(i,1),b$(i,2)-adres pocz. tytułu w y$; b$(i,3)-odsyłacz do tablicy organizacyjnej; r$-tablica organizacyjna (przechowuje zakodowane adresy nazwisk autorów i skojarzonych z nimi tytułów); p$-tablica robocza; al-aktualna ilość wierszy w a$; bl-aktualna ilość wierszy w b$; wr-aktualna ilość wierszy w r$; dx-1-długość x$; dy-1-długość y$
1040 GO SUB 300: GO SUB 180
1045 GO TO 1040
2000 REM dopisywanie rekordów
2005 IF wr>1 THEN GO TO 2015
2010 CLS : PRINT AT 8,4; BRIGHT 1,"Dopisywanie niemożliwe!": PRINT AT 13,3; BRIGHT 1;"(zbiór nie został założony)": BEEP .8,10: GO SUB 150: GO TO 100
2015 GO SUB 300: GO SUB 180: GO TO 2015
3000 REM kasowanie
3005 CLS : LET zb=65535-USR 7962: IF 2*ll+200>zb THEN PRINT BRIGHT 1;AT 8,4;"Kasowanie niemożliwe!": PRINT AT 12,4; BRIGHT 1;"(brak miejsca w pamięci)": BEEP .8,10: GO SUB 150: GO TO 100
3010 BEEP .4,10: CLS : PRINT AT 6,5; BRIGHT 1;"Podaj sposób kasowania:": PRINT AT 7,5;"————————";AT 11,6;"1-wg nazwiska autora";AT 14,6;"2-wg tytułu"
3015 LET t$=INKEY$: IF t$<>"1" AND t$<>"2" THEN GO TO 3015
3020 LET kas=1: LET gdzie=0: IF t$="2" THEN GO TO 3031
3025 CLS : PRINT AT 11,2; BRIGHT 1;"Podaj nazwisko do skasowania:"
3027 INPUT " "; LINE s$: IF LEN s$=0 THEN GO TO 3027
3029 GO SUB 730: PRINT INK 7;AT 13,6;"(kasowanie rekordu)": PRINT AT 16,4; INVERSE 1;"Nie wolno używać BREAK !": GO SUB 600: LET n=poz: LET ww=CODE a$(n,3)-32: GO TO 3037
3031 CLS : PRINT BRIGHT 1;AT 11,3;"Podaj tytuł do skasowania:"
3033 INPUT LINE s$: IF LEN s$=0 THEN GO TO 3033
3035 GO SUB 730: PRINT INK 7;AT 13,6;"(kasowanie rekordu)": PRINT AT 16,4; INVERSE 1;"Nie wolno używać BREAK !": GO SUB 750: LET n=poz: LET ww=CODE b$(n,3)-32
3037 LET pn=FN f(ww): LET pt=FN g(ww)
3040 DIM p$(m,4): REM kasowanie odsyłaczy z r$
3045 IF ww=1 THEN GO TO 3060
3050 IF ww=wr-1 THEN LET wr=wr-1: GO TO 3075
3055 FOR k=1 TO ww-1: LET p$(k)=r$(k): NEXT k
3060 FOR k=ww+1 TO wr-1: LET p$(k-1)=r$(k): NEXT k
3065 LET wr=wr-1
3070 FOR k=1 TO wr-1: LET r$(k)=p$(k): NEXT k
3075 LET d=0: LET p=pn: LET d1=0: REM kasowanie odsyłaczy w a$
3080 FOR k=pn TO 100000

3085 LET d=d+1: LET d1=d1+1
3090 IF x$(k)=";" THEN LET s$=x$(p TO p+d-2): GO SUB 600: LET n=poz: GO SUB 3280: GO TO 3105
3095 IF x$(k)="," THEN LET s$=x$(p TO p+d-2): GO SUB 600: LET n=poz: GO SUB 3280: LET p=p+d: LET d=0
3100 NEXT k
3105 REM kasowanie nazwiska z x$
3110 IF pn>1 THEN GO TO 3120
3115 LET x$=x$(d1+1 TO ): GO TO 3130
3120 IF pn+d1=dx THEN LET x$=x$( TO pn-1): GO TO 3130
3125 LET x$=x$( TO pn-1)+x$(pn+d1 TO )
3130 GO SUB 3200: LET dx=dx-d1: LET dn=d1
3135 REM kasowanie odsyłaczy z b$
3140 LET d=0: LET p=pt
3145 FOR k=pt TO 100000
3150 LET d=d+1
3155 IF y$(k)=";" THEN LET s$=y$(p TO p+d-2): GO SUB 750: LET n=poz: GO SUB 3330: GO TO 3165
3160 NEXT k
3165 REM kasowanie tytułu z y$
3170 IF pt>1 THEN GO TO 3180
3175 LET y$=y$(d+1 TO ): GO TO 3190
3180 IF pt+d=dy THEN LET y$=y$( TO pt-1): GO TO 3190
3185 LET y$=y$( TO pt-1)+y$(pt+d TO )
3190 GO SUB 3240: LET dy=dy-d: LET dt=d: GO SUB 3380
3195 BEEP .8,10: BORDER 7: PAPER 7: INK 0: CLS : PRINT AT 11,4; BRIGHT 1;"Rekord został skasowany": GO SUB 150: GO TO 100
3200 REM zmiana odsyłaczy w a$
3205 FOR q=1 TO al
3210 LET x=200*(CODE a$(q,1)-32)+(CODE a$(q,2)-32): LET y=CODE a$(q,3)-32
3215 IF x>pn THEN LET x=x-d1: LET m$=CHR$ (32+INT (x/200)): LET n$=CHR$ (32+x-200*(CODE m$-32)): LET a$(q,1)=m$: LET a$(q,2)=n$
3220 IF y>ww THEN LET a$(q,3)=CHR$ (31+y)
3225 NEXT q
3230 RETURN
3240 REM zmiana odsyłaczy w b$
3245 FOR q=1 TO bl
3250 LET x=200*(CODE b$(q,1)-32)+(CODE b$(q,2)-32): LET y=CODE b$(q,3)-32
3255 IF x>pt THEN LET x=x-d: LET m$=CHR$ (32+INT (x/200)): LET n$=CHR$ (32+x-200*(CODE m$-32)): LET b$(q,1)=m$: LET b$(q,2)=n$
3260 IF y>ww THEN LET b$(q,3)=CHR$ (31+y)
3265 NEXT q
3270 RETURN
3280 REM kasowanie wiersza z a$
3285 DIM p$(m+30,3)
```

```
3330 REM [illegible]
3335 DIM p$(m+30,3)
3340 IF n=bl THEN LET bl=bl-1: RETURN
3345 IF n=1 THEN GO TO 3355
3350 FOR q=1 TO n-1: LET p$(q)=b$(q): NEXT q
3355 FOR q=n+1 TO bl: LET p$(q-1)=b$(q): NEXT q
3360 LET bl=bl-1
3365 FOR q=1 TO bl: LET b$(q)=p$(q): NEXT q
3370 RETURN
3380 REM [illegible]
3385 FOR q=1 TO wr-1
3390 LET x=FN f(q): LET y=FN g(q)
3395 IF x>pn THEN LET x=x-dn: LET m$=CHR$ (32+INT (x/200)): LET n$=CHR$ (32+x-200*(CODE m$-32)): LET r$(q,1)=m$: LET r$(q,2)=n$
3400 IF y>pt THEN LET y=y-dt: LET m$=CHR$ (32+INT (y/200)): LET n$=CHR$ (32+y-200*(CODE m$-32)): LET r$(q,3)=m$: LET r$(q,4)=n$
3405 NEXT q
3410 RETURN
4000 REM wyszukiwanie informacji
4005 LET gdzie=0: BEEP .4,10: CLS : PRINT AT 5,2; BRIGHT 1;"Wybierz sposób wyszukiwania:": PRINT AT 6,2;"____________________________"
4010 PRINT AT 11,6;"1-wg nazwisk autorów";AT 15,6;"2-wg tytułów"
4015 LET t$=INKEY$: IF LEN t$=0 THEN GO TO 4015
4020 IF t$<>"1" AND t$<>"2" THEN GO TO 4015
4025 IF t$="2" THEN GO TO 4045
4030 CLS : PRINT BRIGHT 1;AT 10,5;"Podaj nazwisko autora:"
4035 INPUT " : "; LINE s$: IF LEN s$=0 THEN GO TO 4035
4040 GO SUB 730: GO SUB 600: GO SUB 180: GO TO 4005
4045 CLS : PRINT AT 10,6; BRIGHT 1;"Podaj tytuł książki:"
4050 INPUT " : "; LINE s$: IF LEN s$=0 THEN GO TO 4050
4055 GO SUB 730: GO SUB 750: GO SUB 180: GO TO 4005
5000 REM wydruk rekordów
5005 BEEP .6,10: CLS : PRINT BRIGHT 1;AT 4,5;"Podaj sposób wydruku": PRINT AT 5,5;"____________________";AT 9,0;"1-nazwiska wszystkich autorów";AT 13,0;"2-wszystkie tytuły";AT 17,0;"3-wszystkie rekordy(autor-tytuł)"
5010 LET t$=INKEY$: IF LEN t$=0 THEN GO TO 5010
5015 IF t$<"1" OR t$>"3" THEN GO TO 5010
5020 CLS : IF t$>"1" THEN GO TO 5045
5022 LET dr=1: PRINT AT 0,5; BRIGHT 1;"Wykaz nazwisk autorów:": PRINT AT 1,5;"______________________"
5025 FOR n=1 TO al
5030 PRINT : LET p=200*(CODE a$(n,1)-32)+(CODE a$(n,2)-32): GO SUB 860
5035 NEXT n
5040 BEEP .4,10: GO SUB 150: GO TO 100
5045 IF t$="3" THEN GO TO 5067
5047 PRINT AT 0,9; BRIGHT 1;"Wykaz tytułów:": PRINT AT 1,9;"______________"
5050 FOR n=1 TO bl
5055 PRINT : LET q=200*(CODE b$(n,1)-32)+(CODE b$(n,2)-32): GO SUB 890
5060 NEXT n
5065 BEEP .4,10: GO SUB 150: GO TO 100
5067 LET dr=0: LET nr=1
5070 FOR n=1 TO al
5075 LET p=200*(CODE a$(n,1)-32)+(CODE a$(n,2)-32): GO SUB 940: IF wy=0 THEN GO TO 5090
5080 CLS : PRINT BRIGHT 1;AT 1,10;"Rekord nr ";nr: PRINT : PRINT : LET nr=nr+1
5085 LET t=CODE a$(n,3)-32: LET q=200*(CODE r$(t,3)-32)+(CODE r$(t,4)-32): GO SUB 920: BEEP .2,10: GO SUB 180
5090 NEXT n
5095 CLS : PRINT AT 11,4; BRIGHT 1;"Koniec wydruku rekordów": BEEP .6,10: GO SUB 150: GO TO 100
6000 REM informacje o zbiorze
6005 BEEP .5,10: CLS : PRINT AT 3,6; BRIGHT 1;"INFORMACJE O ZBIORZE": PRINT AT 4,6;"____________________"
6010 LET i=8: LET j=2: LET d=30-LEN (STR$ (wr-1)): PRINT AT i,j;"Ilość pozycji..................";AT i,d;wr-1: LET d=30-LEN (STR$ al): PRINT AT i+3,j;"Ilość autorów.................";AT i+3,d;al: LET d=30-LEN (STR$ bl): PRINT AT i+6,j;"Ilość tytułów...................";AT i+6,d;bl
6015 LET pp=65535-USR 7962: LET pd=30-LEN (STR$ pp): PRINT AT i+9,j;"Wolna pamięć (bytes)........";AT i+9,pd;pp
6020 GO SUB 150: GO TO 100
7000 REM koniec
7005 IF zm=0 THEN CLS : PRINT AT 9,8; BRIGHT 1;"Zawartość zbioru": PRINT AT 13,7; BRIGHT 1;"nie uległa zmianie": BEEP .6,10: GO SUB 150: GO TO 7030
7010 CLS : PRINT BRIGHT 1;AT 7,8;"Zawartość zbioru": PRINT AT 9,9; BRIGHT 1;"uległa zmianie": PRINT AT 14,6; BRIGHT 1;"Czy chcesz nagrać nową": PRINT AT 16,1; BRIGHT 1;"wersję na taśmę? (1-tak,0-nie)": BEEP .5,10
7015 LET t$=INKEY$: IF t$<>"0" AND t$<>"1" THEN GO TO 7015
7020 IF t$="0" THEN GO TO 7030
7025 BORDER 0: PAPER 0: INK 7: CLS : PRINT AT 11,2; FLASH 1;"Trwa zapis programu i zbioru": SAVE "Biblioteka" LINE 1: BEEP .1,10: SAVE "Znaki-pol"CODE 65368,168: BEEP .6,10: BORDER 7: PAPER 7: INK 0
7030 CLS : PRINT BRIGHT 1;AT 10,13;"KONIEC": PRINT AT 13,12; BRIGHT 1;"PROGRAMU": BEEP .6,15: PAUSE 0
7035 REM koniec programu
```

PROGRAM 17

ATARI

POLE MINOWE jest grą napisaną w języku ATARI BASIC. Zadaniem gracza jest przejście pola minowego od furtki wejściowej do wyjściowej. Przechodzący jest przedstawiony na ekranie znakiem „ ". Sterowanie jego ruchem odbywa się klawiszami: „ –, =, +, *". (Klawiatura ATARI 800 XL zawiera na tych klawiszach również oznaczenia: „↑, ↓, ←, →", które wskazują na kierunek ruchu). Na polu minowym ukrytych jest losowo 100 min, niewidocznych na ekranie. Ustawienie znaku na polu sąsiadującym z miną sygnalizowane jest dźwiękiem ostrzegawczym i migotaniem. Takie „otarcie się" o minę kosztuje przechodzącego sporo nerwów i stratę jednego punktu ze 100 otrzymanych na początku gry. Droga, którą gracz przebył bezpiecznie, zaznaczona jest kropkami. Wybór dalszej drogi zależy od gracza. Wejście na minę to koniec gry.

Po zakończeniu gry na ekranie ukazuje się uzyskana liczba punktów oraz liczba ruchów wykonanych przez gracza. Wyświetla się również położenie ukrytych min oraz przebyta droga. Można zobaczyć, jak należało iść. Ponowne rozpoczęcie gry następuje po naciśnięciu klawisza START.

Program jest prosty. Niewątpliwie zachęci on użytkowników ATARI do jego rozbudowy. A możliwości jest tutaj wiele. Od prostych usprawnień do większych przeróbek. Najprostsze i najłatwiejsze do wykonania jest wprowadzenie sterowania za pomocą joysticka. Wystarczy dodać poniższą sekwencję:

```
40
170
230 RU=STCK(0)
240 IF NR=0 AND RU<>14 THEN 230
360 RU=STICK(0)
370 IF RU=14 AND I=20 AND J=1 THEN
    700
380 IF RU=14 AND J=1  THEN 360
390 IF RU=13 AND J=21 THEN 360
400 IF RU=11 AND I=1  THEN 360
410 IF RU=7  AND I=37 THEN 360
510 IF RU=14 AND I=20 AND J=1 THEN
    700
515 IF RU<>14 THEN 525
525 IF RU<>13 THEN 535
535 IF RU<>11 THEN 545
545 IF RU<>7 THEN RETURN
765
```

Gra ma jeden poziom trudności. Można go zwiększyć, rozmieszczając na polu więcej niż 100 min (zmiana wartości w linii 110 programu). Można się wtedy przekonać, jak trudne jest pokonanie pola minowego już przy 200 minach. A gdyby tak po pozytywnym zakończeniu gry przechodzić do następnej o wyższym poziomie trudności? Ponieważ zliczana punktacja odnosi się do każdej gry osobno, można by wprowadzić punktację globalną po 2, 3 partiach. Niewątpliwą atrakcją byłby drugi gracz, oczywiście z oddzielnie prowadzoną punktacją. Również zasady punktowania można uzależnić od ogólnej liczby ruchów wykonanych przez gracza (im mniej tym lepiej).

Dla bardziej zaawansowanych w programowaniu proponuję uzyskanie lepszych efektów dźwiękowych, zwłaszcza melodii końcowej.

Wyposażenie gracza w możliwości rzucania granatów (np. pięciu w jednej partii) o dwa pola elementarne w wybranym kierunku, które unieszkodliwiałyby znajdującą się tam minę, pozwoli na obranie lepszej taktyki gry. Zniszczenie miny powinno być wtedy odpowiednio punktowane na korzyść gracza.

Inne poprawki podpowie wam wyobraźnia. Życzę dobrej zabawy.

```
  1 REM GRA "POLE MINOWE"

  3 CLR
  4 DIM A(37,21), B$(37)
  5 FOR I=1 TO 37:FOR J=1 TO 21:
    A(I,J)=0:NEXT J:NEXT I
  6 GRAPHICS 0

  8 REM PLANSZA TYTULOWA
  9 POKE 752,1:PRINT "[CLEAR]"
 10 POS.11,5:PRINT "[Q][R][E] [Q][R]
    [E] |   [Q][R][R]";
 11 POS.11,6:PRINT "| |  | |  |    |";
 12 POS.11,7:PRINT "[A][R][C] | | |
      [A][R]";
 13 POS.11,8:PRINT "|   | | |   |";
 14 POS.11,9:PRINT "|   [Z][R][C] [Z]
    [R][R] [Z][R][R]";
 15 POS.7,11:PRINT "[Q]  [E] | [Q]  [E]
     [Q][R][E] | | [Q][R][R]";
 16 POS.7,12:PRINT "|[G][F]| | |[G] | |
     | |  | |";
 17 POS.7,13:PRINT "|  | | | [G]| | | |  |
     [A][R]";
 18 POS.7,14:PRINT "|  | | | [C] | | |[F]
    [G]  |";
 19 POS.7,15:PRINT "| | | |  | [Z][R][C]
     [Z]  [C] [Z][R][R]";

 20 FOR N=1 TO 37
 25 POS.N,1:PRINT "[R]";
 30 POS.N,23:PRINT "[R];
 35 NEXT N
 40 OPEN #1,4,0,"K:"
 45 POLE=0
 50 FOR N=2 TO 22
 55 POS.0,N:PRINT "|";
 60 POS.38,N:PRINT "|";
 65 NEXT N
 70 POS.0,1:PRINT "[Q]";:POS.38,1:
    PRINT "[E]";
 75 POS.0,23:PRINT :[Z]";:POS.38,23:
    PRINT "[C]";
 80 POS.19,1:PRINT "[D] [A]";
 85 POS.19,23:PRINT "[D][T][A]";

100 REM ROZMIESZCZENIE MIN
110 FOR N=1 TO 100
120 SETCOLOR 4,2,INT(RND(0)*16)
```

```
130 I=INT(RND(0)*37)
140 IF I=0 THEN 130
150 J=INT(RND(0)*21)
152 IF J=0 THEN 150
153 IF I=20 AND J=1 THEN 130
154 IF I=20 AND J=21 THEN 130
155 A(I,J)=1
156 NEXT N
160 NR=0:PKT=0
170 DIM RUCH$(1)
180 DIM SP$(25)
190 SP$="                         "
200 FOR N=5 TO 15

210 POS.7,N:PRINT SP$;
220 NEXT N
230 POS.0,0:GET #1,B:RUCH$=CHR$(B)
240 IF NR=0 AND RUCH$<>"-" THEN 230

250 REM OBSLUGA PIERWSZEGO RUCHU
260 NR=1:I=20:J=21
270 A(20,21)=3
280 POS.20,22:PRINT "[T]";
290 POS.20,23:PRINT ".";
300 IF A(I-1,J)=1 THEN GOSUB 450
305 Y=J-1
310 IF A(I-1,J-1)=1 THEN GOSUB 450
315 X=I
320 IF A(I,J-1)=1 THEN GOSUB 450
325 X=I+1
340 IF A(I+1,J-1)=1 THEN GOSUB 450
345 Y=J
350 IF A(I+1,J)=1 THEN GOSUB 450
360 POS.0,0:GET #1,B:RUCH$=CHR$(B)
370 IF RUCH$="-" AND I=20 AND J=1
    THEN 700
380 IF RUCH$="-" AND J=1 THEN 360
390 IF RUCH$="=" AND J=21 THEN 360
400 IF RUCH$="+" AND I=1 THEN 360
410 IF RUCH$="*" AND I=37 THEN 360
420 GOSUB 500
430 GOTO 360

450 REM OSTRZEZENIE O MINIE
455 FOR N=1 TO 5
460 SOUND 0,10,0,12
465 FOR Z=1 TO 5:NEXT Z
470 POS.I,J+1:PRINT " ";
475 SOUND 0,0,0,0
480 POS.I,J+1:PRINT 6[T]";
485 NEXT N
490 RETURN
500 REM OBSLUGA RUCHU
505 IP=I:JP=J
510 IF RUCH$="-" AND J=1 AND J=20
    THEN 700
515 IF RUCH$<>"-" THEN 525
520 J=J-1:GOTO 555
525 IF RUCH$<>"=" THEN 535
530 J=J+1:GOTO 555
535 IF RUCH$<>"+" THEN 545
540 I=I-1:GOTO 555
545 IF RUCH$<>"*" THEN RETURN
550 I=I+1
555 SOUND 0,10,12,14:SOUND 0,0,0,0:
    NR=NR+1
560 Y=J+1:POS.I,Y:PRINT "[T]";
565 X=IP:Y=JP+1:POS.X,Y:PRINT ".";
570 IF A(I,J)=1 OR A(I,J)=2 THEN 900
575 A(I,J)=3
```

```
580 REM SPRAWDZENIE OKOLICY
585 SYGN=0
590 FOR I1=-1 TO 1 STEP 2
595 IF I=1 AND I1=-1 THEN 625
600 IF I=37 AND I1=1 THEN 625
605 IF A(I+I1,J)<>2 AND A(I+I1,J)<>1
    THEN 625
610 A(I+I1,J)=2:PKT=PKT+1
615 IF SYGN=0 THEN GOSUB 450
620 SYGN=1
625 IF J=1 AND I1=-1 THEN 655

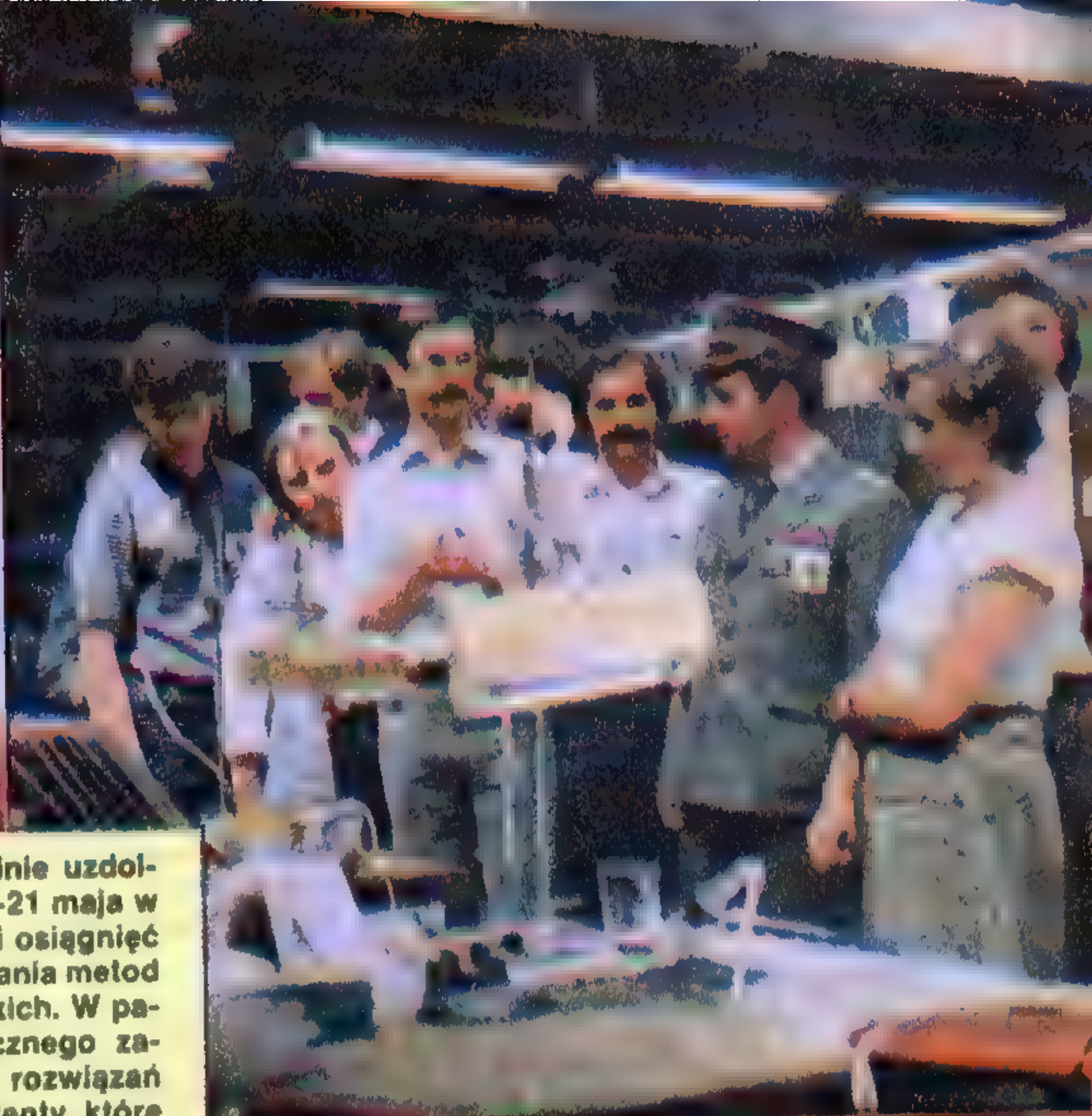
630 IF J=21 AND I1=1 THEN 655
635 IF A(I,J+I1)<>2 AND A(I,J+I1)<>1
    THEN 655
640 A(I,J+I1)=2:PKT=PKT+1
645 IF SYGN=0 THEN GOSUB 450
650 SYGN=1
655 NEXT I1
660 RETURN

700 REM OBSLUGA RUCHU KONCZACEGO
710 POS.20,2:PRINT " ";
720 POS.20,1:PRINT "[T]";
730 POS.0,0:PRINT 100-PKT;" ";"W ";NR;
    " RUCHACH";
735 GOSUB 800
740 SOUND 0,RND(0)*255,14,10
750 IF PEEK(53279)<>6 THEN 740
760 SOUND 0,0,0,0
765 CLOSE #1
770 GOTO 3

800 REM WYSWIETLENIE CALEGO POLA
805 IF POLE=1 THEN 895
806 FOR J=1 TO 21
807 POS.1,J+1
810 FOR I=1 TO 37
815 B$(I)=" "
830 IF A(I,J)<>1 THEN 850
840 B$(I)="*":GOTO 890
850 IF A(I,J)<>2 THEN 860
855 B$(I)="*":GOTO 890
860 IF A(I,J)<>3 THEN 870
865 B$(I)="[T]":GOTO 890
870 IF A(I,J)<>4 THEN 890
880 B$(I)="[S]"
890 NEXT I
891 PRINT B$;
892 NEXT J
894 POLE=1
895 RETURN

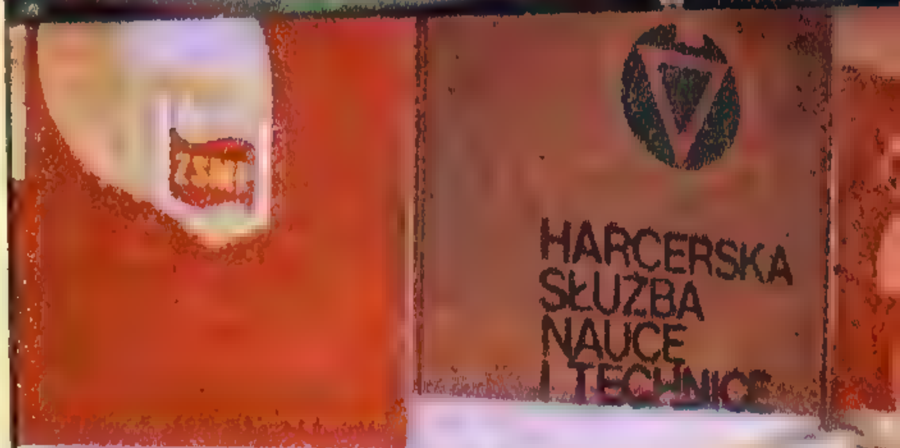
900 REM WYBUCH
905 FOR N1=1 TO 255 STEP 14
910 FOR N=1 TO 4
915 POS.I-1,J:PRINT "[G]|[F]";
920 POS.I-1,J+1:PRINT "[R]+[R]";
925 POS.I-1,J+2:PRINT "[F]|[G]";
930 SOUND 0,N1,8,10
935 POS.I-1,J:PRINT "   ";
940 POS.I-1,J+1:PRINT " [T] ";
945 POS.I-1,J+2:PRINT "   ";
950 NEXT N
955 NEXT N1
960 SOUND 0,0,0,0
965 A(I,J)=4
970 GOSUB 800
975 PKT=100
980 GOTO 700
```





FORUM MŁODYCH TWÓRCÓW TECHNIKI

Ideą spotkania młodzieży szczególnie uzdolnionej jakie odbyło się w dniach 20—21 maja w Katowicach, było ukazanie dorobku i osiągnięć młodych wynalazców oraz wypracowania metod wdrażania pomysłów racjonalizatorskich. W pawilonach Ośrodka Postępu Technicznego zaprezentowano ponad 250 unikalnych rozwiązań technicznych, w tym urządzenia i patenty, które zdobyły 10 złotych medali na Światowej Wystawie Młodzieży w Płowdiw (Bułgaria). Prawie wszystkie prezentowane wynalazki i pomysły, zawierały elementy ze świata elektroniki. Laureaci Turnieju Młodych Mistrzów Techniki, członkowie klubów mikrokomputerowych ZHP i ZSMP, zdobywcy Nagród im. M. Kopernika, odwiedzili również kilka wybranych zakładów pracy, zapoznając się z nowoczesnymi rozwiązaniami technologicznymi w przemyśle. Życzymy wszystkim twórcom i wynalazcom nowych odkryć, wielu pomysłów i cierpliwości w oczekiwaniu na ich... wdrożenie.









TARGOWA OFERTA

O **Juniorze** „IKS" rozmawia z zastępcą dyrektora ds. handlowych wrocławskiej ELWRO, **Jerzym Chełchowskim**

X: — Czy Junior podbije rynek?

J. Ch.: — Polski z pewnością

X: — A inne?

— Nie ma o tym mowy, z kilku powodów, a przede wszystkim dlatego, że nie mamy takiego zamiaru

X: — To tylko skromność?

— Nie, nasza produkcja w całości skierowana będzie na polski rynek, a w większości adresowana do szkół — jest to przecież komputer edukacyjny.

X: — Czyli jaki?

— Przede wszystkim prosty w obsłudze i nowoczesny w swej konstrukcji. Musi spełniać też kilka innych warunków: powinien mieć otwartą strukturę, pozwalającą na sprzętową rozbudowę o dodatkowe bloki funkcjonalne (dysk twardy, grafika wysokiej rozdzielczości itp.), a poza tym przyjęliśmy zasadę, że tego typu komputer powinien być elementem lokalnej sieci komputerowej. Taki jest Junior.

X: — A konkretnie?

— Podstawowe parametry, to mikroprocesor Z-80A, stronicowanie pamięci operacyjnej (64 KB RAM, 24 KB EPROM), układy grafiki kolorowej i generatora dźwięku, jednostka sterująca pamięcią na dyskach elastycznych i pamięcią kasetową, układy pozwalające na bezpośrednie dołączenie drukarki znakowej i graficznej, joysticka, pióra świetlnego, układu „myszki".

X: — Dodajmy, że oprogramowanie rezydujące swiera edytor, interpreter Basica, oraz programy sterujące pamięcią dyskową, magnetofonem kasetowym, lokalną siecią komputerową i drukarką znakową, a system operacyjny, w który wyposażony jest JUNIOR (CPOS) jest kompatybilny z systemem CP/M V2.2.

— To standard światowy. Dzięki temu dla Juniora dostępny będzie ogromny zbiór oprogramowania, będzie można używać także innych języków programowania: LOGO, Pascal, Fortran, C czy Makroasembler.

X: — I jeszcze cena.

— Zależy oczywiście od konfiguracji. Jednostka centralna nie powinna być droższa niż 120—130 tysięcy Wielkość produkcji ustali ją ostatecznie

Andrzej BIER — st. konstruktor — Centrum Naukowo-Produkcyjne Systemów Sterowania „MERASTER":

— Firmowy „przebój" targowy to plotter Mera 621. Zbudowany został na częściach krajowych, a reprezentuje... przyzwoity światowy poziom. Może współpracować z każdym profesjonalnym komputerem. Charakteryzuje się dużą szybkością kreślenia, łatwością obsługi i jest przystosowany do standardowego języka graficznego. „Urządzenie" powinno zdobyć uznanie fachowców i... księgowych (koszty produkcji skalkulowano w granicach 50 proc. ceny, sprowadzanych dotychczas tego typu plotterów). A wszystko to będzie można kupić już... w 1987 r. Dzisiaj proponujemy profesjonalną sieć MERANET 660 w oparciu o komputer MERA 660 lub terminal MERA 7953 N

Włodzimierz TRUSZ — przedsiębiorstwo polonijno-zagraniczne „KEBA":

— Potencjalnym klientem dla nas jest przemysł. Oferujemy sprzęt profesjonalny. Chcemy również uczestniczyć w wyposażaniu szkół w komputery. Współpracujemy z amerykańską firmą **ACO** Polecamy wszystkie możliwości grafiki komputerowej. Największe zainteresowanie naszym sprzętem (głównie firmy IMB) wykazuje Fabryka Samochodów Osobowych w Warszawie, która zamierza skomputeryzować procesy projektowania.

Mariusz JAWORSKI — Computer system desing — consultant:

— Nasz rynek, to praktycznie „pustynia" informatyczno-komputerowa Mam na myśli przede wszystkim stan świadomości w tej dziedzinie naszego społeczeństwa. Dlatego też, nastawiliśmy się na udzielanie w szerokim zakresie porad. Nasze usługi obejmują

— doradztwo w sprawie zakupu sprzętu i oprzyrządowania,

— organizację programów w zależności od profilu przedsiębiorstwa i charakteru komputeryzowanego działu,

— zapewnienie kompatybilności zakupionym urządzeniom,

— wprowadzenie sprawdzonych sieci komputerowych

Sieci w zakładach pracy mogą powstać w przypadku istnienia przynajmniej 3 komputerów. Najczęściej komputeryzuje się:

— księgowość,

— gospodarkę materiałową,

— finanse,

— stanowiska zarządzania, centra obliczeniowe, itp

Polecamy sprzęt za dewizy i za złotówki. Doradzamy urządzenia sprawdzone z gwarantowanym serwisem Prowadzimy również kursy i przeszkolenia dla ludzi w zakresie obsługi zamawianego sprzętu. Gwarantujemy pomoc w uruchomieniu komputerowych systemów. Koszty udzielania porad i wprowadzania określonych systemów do zakładów pracy, wraz z przeszkoleniem kilku osób i początkowym treningiem, zamykają się w sumie około 1 mln zł. Polecamy sprzęt w cenach od 5 do 30 mln złotych. I tutaj uwaga — zawsze taniej niż w firmach polonijnych.

Najbliższe i najważniejsze zadanie, jakie nas obecnie czeka, to komputeryzacja Wojewódzkiego Związku Spółdzielni Rolniczej w Lesznie wraz z wprowadzeniem sprawdzonej sieci profesjonalnej typu CORVUS



ELWRO 800 Junior



Nawiązaliśmy ścisłą współpracę z „Bomisem", w zakresie świadczeń gwarantujących wysoką jakość wykonywanych przeglądów

Peter KASYK — amerykańska firma „WANG":

— W Polsce firma działa już 10 lat. Oferuje tylko komputerowe systemy profesjonalne dla dużych zakładów pracy, działające w określonych sieciach. Pierwszym klientem „WANG-a" był w latach 70-tych „Budopol". Obecnie komputery firmy sterują procesami produkcyjnymi w „Polkolorze", który to zakład zamierza rozszerzyć swoją bazę komputerową o dalszy sprzęt „WANG-a". Polskie Linie Oceaniczne podpisały z firmą kontrakt wartości 1 mln dol obejmujący dostawę urządzeń komputerowych i stworzenie systemu zarządzania przedsiębiorstwem. Przeszkodą w handlu z Polską są zachodnie bariery licencyjne. Obecnie sytuacja w tej dziedzinie znacznie się poprawiła. Firma nie obawia się konkurencji ze strony przedsiębiorstw europejskich. Urządzenia, które oferujemy są tak wysokiej klasy, że nie mają konkurentów

MAZOVIA

Andrzej STOBIŃSKI — Zakład Mechaniki Precyzyjnej „UNITRA-MAGMOR":

— Atrakcyjna oferta firmy to przede wszystkim terminal ekranowy AN-2000 Składa się on z dwu podstawowych części: monitora Neptun-M 157 oraz klawiatury z mikrokomputerem zbudowanym w oparciu o mikroprocesor INTEL 8080 posiadającym 2 kilobajty pamięci RAM (pamięć obrazu) oraz 2 kilobajty pamięci EPROM (w niej zapisany jest program, który zarządza pracą tego urządzenia) Terminal przeznaczony jest do wprowadzania danych z klawiatury, przesyłania ich do komputera oraz wyświetlania na monitorze informacji odbieranych z systemu komputerowego. Pojemność ekranu wynosi 2000 znaków rozmieszczonych w 25 wierszach po 80 znaków w wierszu. Jest możliwe elastyczne redagowanie informacji na ekranie. Dostępny jest alfabet polski i angielski, duże i małe litery, oraz znaki semigraficzne. Niewątpliwą dodatkową zaletą tego urządzenia jest możliwość automatycznego testowania poprawności poszczególnych elementów układów elektronicznych. Zabezpiecza to poprawność działania a w przypadku awarii, łatwą lokalizację błędu. Cały komplet kosztuje 1,4 mln zł. Wyprodukowano już 250 sztuk tych urządzeń jako serię informacyjną. Na targach jest prezentowany po raz pierwszy

Ryszard CHWALKO — Spółdzielnia Rzemieślnicza „MOTGOS":

— Poprzez naszą działalność chcemy chociaż częściowo uporządkować „dziki" rynek komputerowy, jaki u nas zapanował. Nastawiamy się raczej na klienta indywidualnego, ewentualnie na małe zakłady rzemieślnicze. Oferujemy:

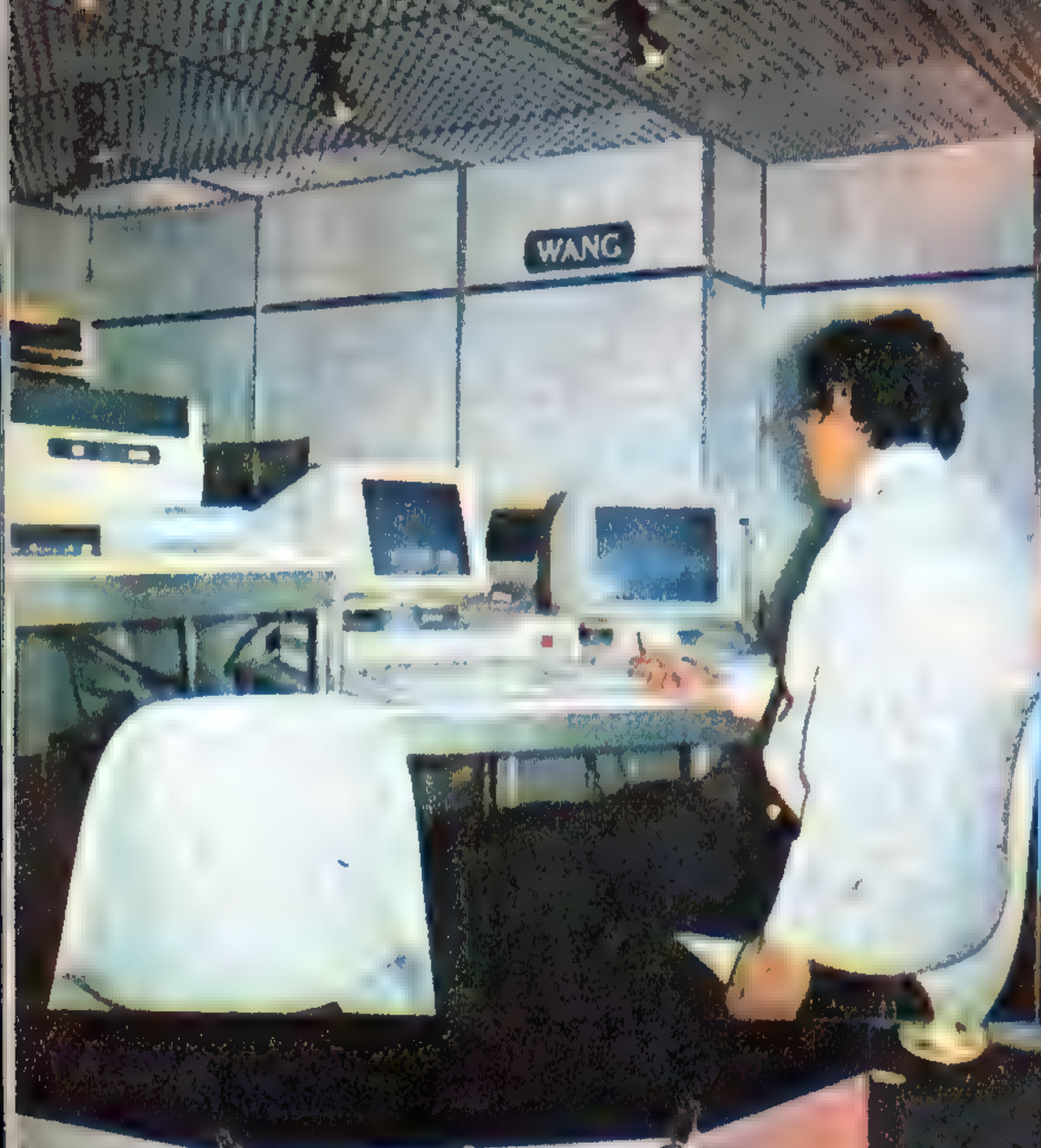
— usługi serwisowe (za złotówki),
— informację techniczną,
— przeglądy przedsprzedażne,
— pokaz sprzętu i jego zastosowanie,
— zapewniamy import części zamiennych,
— naprawy gwarancyjne i pogwarancyjne

Zapewniamy serwis dla komputerów. Commodore C-64, C-128, Schneider CPC 464 CPC 6128 oraz dla wielu urządzeń peryferyjnych: drukarki, stacje dysków, pamięci kasetowe, monitory, itp.

Opracowali:
Wiesław Cetera
Ryszard Rogoń



Zdjęcia: R. Rogoń





58 MIĘDZYNARODOWE

(Elektronika, ko

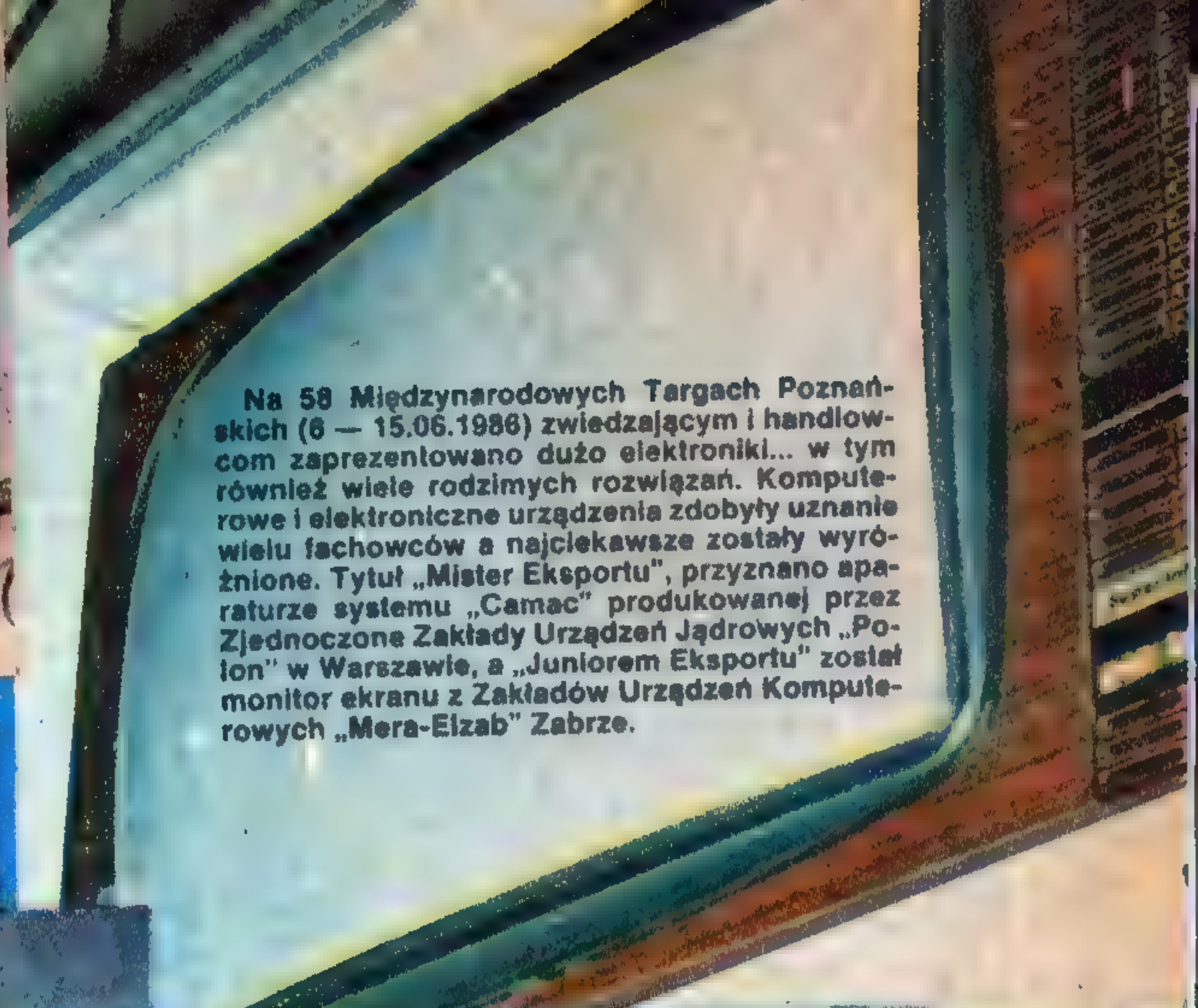








TARGI POZNANSKIE
mputery, firmy)



Na 58 Międzynarodowych Targach Poznańskich (6 — 15.06.1986) zwiedzającym i handlowcom zaprezentowano dużo elektroniki... w tym również wiele rodzimych rozwiązań. Komputerowe i elektroniczne urządzenia zdobyły uznanie wielu fachowców a najciekawsze zostały wyróżnione. Tytuł „Mister Eksportu", przyznano aparaturze systemu „Camac" produkowanej przez Zjednoczone Zakłady Urządzeń Jądrowych „Polon" w Warszawie, a „Juniorem Eksportu" został monitor ekranu z Zakładów Urządzeń Komputerowych „Mera-Elzab" Zabrze.







2

LOGO

W poprzednim odcinku celowo pominąłem kilka ogólnych uwag odnośnie do programowania w LOGO. Pierwsza dotyczy wywołania LOGO, czyli uruchomienia translatora języka. Od tej pory pisząc LOGO, będę miał na myśli polskie LOGO, a jest ono dostępne w postaci interpretera (program PTI LOGO) na mikrokomputerze ZX Spectrum. Jak zatem rozpocząć konwersację z LOGO? Na pewno prawie wszyscy użytkownicy Spectrum uśmiechną się wyrozumiale i odpowiedzą: trzeba je wgrać z kasety. Oczywiście tak jest, ale dla porządku oraz dla tych, którzy może nie wiedzą, przypomnę te podstawowe rzeczy. Mikrokomputer ZX Spectrum w wersji 48 K lub Plus ma ok. 48 000 bajtów (nie licząc pamięci typu ROM), czyli komórek pamięci. Tyle też wymaga interpreter LOGO. Jasno więc z tego wynika, że Spectrum 16 K (jest ich zresztą coraz mniej) nie nadaje się do korzystania z LOGO. Mając już komputer włączamy go do sieci i widzimy napis „© 1982 Sinclair Research Ltd". W tym momencie możemy podawać rozkazy lub układać programy w języku Basic, który jest automatycznie dostępny. Firma Sinclair wyposażyła swój komputer w interpreter (rodzaj translatora) tego języka, który jest na stałe w pamięci — jest zapisany w ROM-ie. Siłą rzeczy, aby uruchomić LOGO, musimy napisać rozkaz w Basic'u „LOAD "" ". Mając kasetę z programem PTI LOGO, wgrywamy go, podobnie jak to czynimy z każdym innym programem czy grą. Program PTI LOGO (nazwa błyskająca) wgrywa się około 4 minut. Po tym czasie na ekranie powinien pojawić się napis „Polskie LOGO wer. 9 cPTI/1986". Po tym też można poznać LOGO, o którym piszemy, gdyż krąży wiele wersji polskiego LOGO, będących atrapami, różniących się od siebie, wprowadzających tym samym zamieszanie. Wersje te są złe i nie warto zwracać na nie uwagi.

Kiedy widzimy już napis obwieszczający, że LOGO jest wgrane, widzimy również znak „?" i migający kursor. Taka sytuacja na ekranie informuje nas, że LOGO gotowe jest przyjmować komendy. Wykonanie komendy następuje po wciśnięciu klawisza ENTER. Pisząc zatem jakiś ciąg znaków postępujemy podobnie jak przy pisaniu listu; sami wiemy, co napisaliśmy i aby ktoś przeczytał, trzeba list wysłać — wcisnąć ENTER. Poprzednio bawiąc się żółwiem — zauważyliśmy, że wprowadzane przez nas komendy i w ogóle wszystkie teksty widoczne były tylko w dwóch dolnych liniach ekranu, LOGO bowiem ma dwa tryby: graficzny i tekstowy. Wykonanie jakiejkolwiek komendy graficznej (związanej z żółwiem) powoduje przejście do trybu graficznego — tylko dwie linie przeznaczone są na tekst. W trybie tekstowym cały ekran dostępny jest na napisy, a przechodzi się do tego trybu, pisząc TEKSTY lub w skrócie TS (w angielskim LOGO TEXTSCREEN TS). Spróbujmy napisać „PŻ", a następnie „XXX". Po wprowadzeniu komendy „pokaż żółwia", ukazuje się on na środku ekranu. Następnie po wprowadzeniu XXX w dolnych liniach wyświetla się: „Nie wiem jak zrobić XXX". Tylko w tych dwóch liniach widzimy naszą konwersację. Napiszmy teraz „TS", a następnie „XXX". Żółw znika, znak zapytania pokazuje się u góry ekranu, a komunikat o błędzie w następnej górnej linii. Teraz jesteśmy w trybie tekstowym. Zostawmy na chwilę nasze rysunki i żółwia. Napiszmy:

PISZ 24

Po wciśnięciu ENTER, na ekranie pojawi się 24. Napiszmy teraz:

PISZ "LOGO

Jeden cudzysłów przed słowem oznacza, że słowo ma być wzięte dosłownie, tzn. traktowane jako ciąg znaków — napis. Teraz na ekranie po wciśnięciu ENTER pojawi się LOGO.

Poznaliśmy komendę PISZ, która ma też skrót PP (angielskie LOGO — PRINT PR). Powoduje ona wyświetlenie wartości, jaka jest podana. Wartość ta może być liczbą, słowem, ale może być też ciągiem słów — co w LOGO nazywa się listą. Napiszmy:

PP [Dzisiaj jest ładna pogoda]

W nawiasach kwadratowych jest lista wyrazów, które wyświetlone zostaną po naciśnięciu ENTER. Z poprzedniego odcinka pamiętamy, że można powtarzać jakieś czynności dowolną ilość razy, pisząc komendę POWTÓRZ. Napiszmy:

POWTÓRZ 15 [pisz [trala lala la]]

Komenda znaczy: powtórz 15 razy instrukcje podane w nawiasie kwadratowym. W nawiasie kwadratowym jest tylko jedna instrukcja — PISZ, która ma wpisać ciąg wyrazów ujęty w nawiasy kwadratowe. Wynik jest zatem zgodny z tym, co widzimy na ekranie — 15 napisów „trala lala la" jeden pod drugim. LOGO umie oczywiście liczyć. Spróbujmy napisać:

PISZ 12+13

otrzymamy wynik 25. Podobnie możemy napisać PISZ 34/2 lub PP 1.3*3. Kropka dziesiętna umożliwia nam wprowadzanie dowolnej liczby ułamkowej. Weźmy tak:

PISZ 2.23+3.33—5—7*8/100 powinien wyświetlić się wynik 0.

Wyrażenie może być dowolnie długie, może zawierać nawiasy np:

PP (3+2)*5 daje oczywiście 25.

Tyle na razie o arytmetyce w LOGO. W następnych odcinkach dowiemy się o innych matematycznych możliwościach, o różnych funkcjach, które LOGO zna i o sposobie ich wykorzystania. Teraz jeszcze przyda nam się znajomość jednej funkcji pomocnej przy rysowaniu koła.

Wprowadźmy:

PISZ PI↑

Wyświetli się 3.1415927. Funkcja pi↑oznacza stałą pi i zamiast pisać 3.141 możemy jej używać.

Wróćmy teraz do rysunków. Umiemy już poruszać żółwia, obracać, podnosić pióro, by nie ciągnął kreski i właściwie możemy wszystko narysować. Jednak narysowanie choćby kwadratu wymaga napisania jednej dość długiej linii. Jeśli chcemy narysować kilka kwadratów w różnych miejscach, musimy za każdym razem powtarzać ten sam ciąg dobrze znanych komend. Na-

uczmy zatem LOGO rysować kwadrat tak, żeby nie tracić czasu na powtarzanie w kółko tego samego. Napiszmy:
? OTO kwadrat
> POWTÓRZ 4 [naprzód 40 prawo 90]
> JUŻ
KWADRAT zdefiniowane
Po napisaniu pierwszej linii i jej wprowadzeniu pojawia się znaczek ">" zamiast znaku zapytania. Oznacza to, że LOGO zapamiętuje wprowadzaną linię, lecz jej nie wykonuje — jesteśmy w trakcie definicji procedury. Po napisaniu słowa „już", które oznacza, że skończyliśmy opisywać treść procedury — LOGO pisze np. KWADRAT zdefiniowane — od tej pory zna nowe słowo. Napiszmy zatem

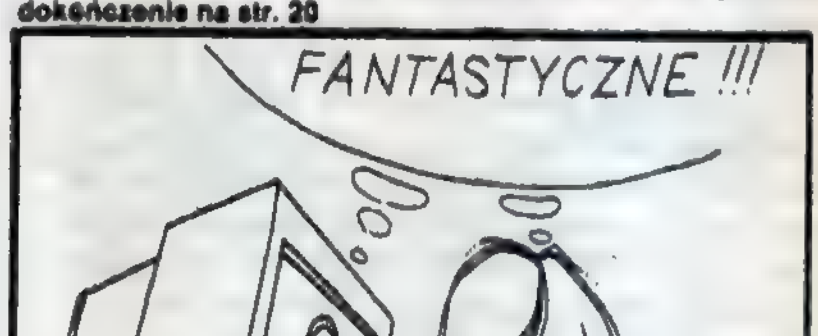
? kwadrat

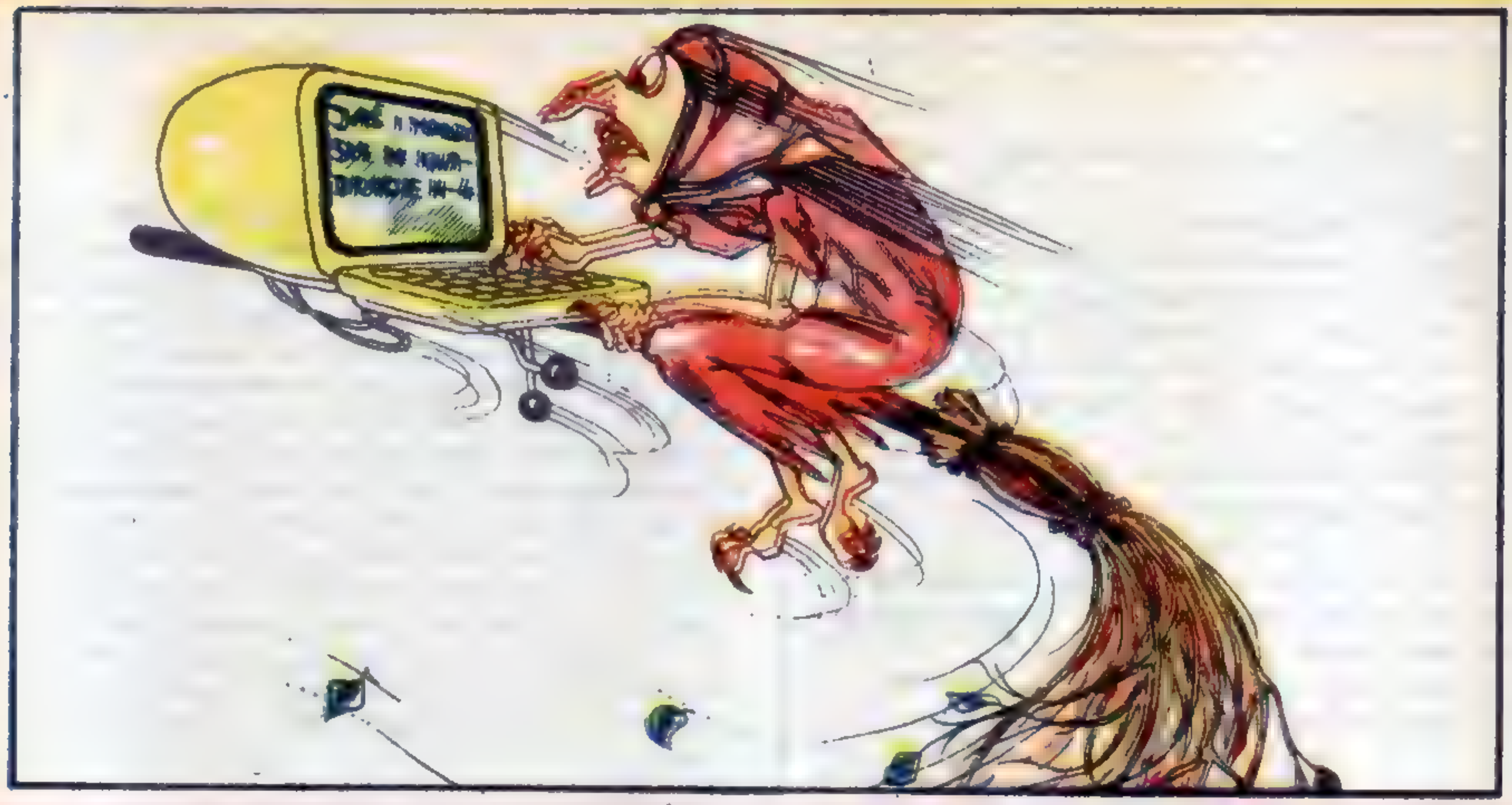
i faktycznie rysuje się od razu cały kwadrat. Słowa kwadrat możemy teraz używać i w nowych procedurach, na przykład:
OTO ROZETA
POWTÓRZ 8 [kwadrat pw 45]
JUŻ
Wywołajmy
? rozeta
i rysuje się kwadrat za kwadratem, obracany o 45 stopni. Zdefiniujmy teraz taką procedurę:
OTO SERWETKA
POWTÓRZ 9 [rozeta prawo 10]
JUŻ
Po wywołaniu „serwetka", na ekranie powstaje dość efektowny rysunek, przypominający ręcznie robioną serwetkę. Spróbujmy jeszcze:
OTO REKLAMA
SERWETKA
PISZ [ALE ŁADNA SERWETKA!]
JUŻ
Teraz mamy serwetkę z podpisem. W tym momencie radzę poeksperymentować sobie i korzystając z możliwości definicji procedur, pohasać żółwiem po ekranie. Dobrze jest definiować procedury po uprzednim przejściu do trybu tekstowego (pisząc TEKSTY lub TS), bo wszystkie teksty widzimy. Odpowiednikiem „OTO" — początku definicji procedury — jest w angielskim LOGO „TO", a odpowiednikiem „JUŻ" — „END".

Narysujmy teraz kwadrat i „wjedźmy" żółwiem do jego środka. Spróbujmy na przykład:
? kwadrat pod prawo 45 naprzód 20
Podane w jednej linii komendy spowodują narysowanie kwadratu, podniesienie rysującego piórka, obrócenie żółwia o 45 stopni w prawo i przeniesienie go wewnątrz kwadratu. Podnieśliśmy piórko, by żółw nie mazał „wchodząc" do wnętrza. Napiszmy teraz:
ZAMALUJ
i cały kwadrat zostanie wypełniony kolorem kreski. Komenda ZAMALUJ lub w skrócie ZAM powoduje wypełnienie kolorem kreski figury, wewnątrz której znajduje się żółw. Pozwala nam ona dowolnie kolorować rysunki na ekranie, ale trzeba o niej wiedzieć jeszcze coś więcej. Zanim opowiemy sobie o zmienianiu kolorów piórka-pisaka, tła, ramki i o sposobie, w jaki komenda ZAMALUJ działa, spróbujmy narysować na ekranie koło. Ponieważ ekran składa się z punktów, więc na pewno idealnego koła nie uda nam się narysować. Wystarczy więc przyjąć, że koło to wielokąt foremny o dość dużej liczbie boków, np. 36. Zdefiniujmy procedurę:
OTO KOŁO
POWTÓRZ 36 [NP 10 PW 10]
JUŻ
Zdefiniowaliśmy 36-kąt. Żółw „idzie" 10 kroków do przodu i skręca o 10 stopni w prawo. Jeśli 36 razy to powtórzyć, narysuje figurę, która może być uważana za koło. Ile kroków wynosi promień tego koła? Wykorzystując znajomość elementarnego wzoru na obwód koła, mamy: obwód = 2*pi*r, zatem r = obwód/(2*pi). No, ale obwód naszego koła, to tyle, ile „przeszliśmy" żółwiem, czyli 36*10. Stąd promień: r = 36*10/(2*3.14). Zauważmy, że wzór ten pozwala rysować koła o dowolnych promieniach, bo liczba kroków w jednym ruchu wynosi podobnie: liczba kroków = 2*pi*r/36. W jaki sposób narysować koło o dowolnym promieniu, korzystając tylko z jednej uniwersalnej procedury, dowiemy się w następnym odcinku. Na razie zdefiniujmy procedurę rysującą koło o promieniu np. 40. Środek tego koła ma znajdować się w punkcie, gdzie znajduje się żółw. A zatem:

OTO KOŁO	
POD NP 40 PW 90 OPU	OPU — opuść
POWTÓRZ 36	POD — podnieś
NP 2*PI*40/36 PW 10	LW — lewo
POD LW 90 WS 40 OPU	WS — wstecz
JUŻ	

Żółw był w środku, ale trochę z boku. I dlatego pewne miejsca nie zostały pokolorowane. Sposób, w jaki rysunek zostaje zamalowany, jest następujący: 1. Z miejsca, gdzie stoi żółw, prowadzona jest pionowa kreska (w górę i w dół) do brzegów figury.
2. Z każdego punktu tej kreski prowadzone są kreski poziome do brzegów figury.
Zauważmy, że jeśli wewnątrz są jakieś niepotrzebne kreski (jak w ostatnim przykładzie z kołem) mają one najczęściej wpływ na kolorowanie. Dlatego też, aby pokolorować „wklęsłe" figury, należy żółwiem kilkakrotnie „najeżdżać" na kawałki niezamalowane. Sposób ten nie jest zbyt wygodny, ale w angielskim LOGO w ogóle nie można figur zamalowywać (nie ma odpowiednika ZAMALUJ). Algorytmy, skutecznie wypełniające kolorem figurę zamkniętą, wymagają dość dużo pamięci
dokończenie na str. 20





czy to na swą treść (dość skomplikowana strategia), czy na pamiętanie rysunku (algorytm „pożaru prerii"). Dla potrzeb LOGO na Spectrum zaadaptowany został ten prosty, ale i bardzo mało kosztowny sposób kolorowania. Siłą rzeczy nie jest najlepszy.

Jeśli dysponujemy telewizorem lub monitorem kolorowym, możemy pobawić się w robienie różnokolorowych rysunków (w telewizorach czarno-białych będą to odcienie). Mamy do dyspozycji 8 kolorów. Komenda ustawiająca kolor pisaka to po prostu PISAK n, gdzie n to numer koloru widoczny na klawiaturze. Tak więc PISAK 2 spowoduje, że żółw i jego ewentualna kreska będą od tej chwili czerwone. Podobnie jest z ustawieniem koloru tła — TŁO 6 — spowoduje, że tło rysunku stanie się żółte. I jeszcze tzw. BORDER, czyli margines-ramka. Piszemy po prostu RAMKA 1 i „border" stanie się niebieski. Wszystkie te komendy odnoszą się do grafiki. W trybie tekstowym również możemy ingerować w kolory wyświetlanych tekstów. Czynimy to komendą:
KOLORYT [n m], gdzie n to numer koloru tła tekstów, a m to numer koloru atramentu. Osobiście radzę przy dłuższej pracy, zwłaszcza z telewizorem używać do tła i ramki koloru czarnego, a do atramentu żółtego, — mniej męczą się oczy. Należy więc wykonać: PISAK 6 TŁO 0 KOLORYT [0 6]. Warto przy okazji wiedzieć, że informacje o niektórych kolorach są dostępne dzięki funkcjom: TŁO↑, PISAK↑, KOLORYT↑.
Spróbujmy napisać:
(PISZ "TŁO TŁO↑ "PISAK PISAK↑ "KOLORYT KOLORYT↑)
a potem zróbmy np. PISAK 4 i ponownie powyższą linię. Kolor pisaka zostanie już zaktualizowany. Cała linia, a raczej komenda PISZ została wzięta w nawiasy okrągłe. Normalnie komenda PISZ ma jeden parametr, którego wartość wyświetla. Postawienie nawiasu przed komendą powoduje, że wszystko, aż do nawiasu zamykającego, zostanie potraktowane jako parametr PISZ. Tak więc tutaj zostało wypisane po kolei: słowo TŁO (bo było "TŁO), wartość liczbowa dla koloru tła (wartość funkcji TŁO↑, której działanie jest identyczne jak funkcji PI↑, czyli zwrócenie wartości), słowo PISAK ("PISAK), numer pisaka (PISAK↑), słowo KOLORYT ("KOLORYT) i dwie liczby — kolor tła i atramentu dla napisów (KOLORYT↑).
Zdefiniujmy teraz następującą procedurę:

```
OTO KOLORY
POWTÓRZ 8 [PISAK PISAK↑+1 NP 40 PW 45]
JUŻ
```

Zobaczymy ośmiokąt z każdym bokiem innego koloru. Zauważmy, że za każdym razem ustawiamy kolor pisaka na bieżący (PISAK↑) plus jeden, tzn. zmieniamy kolor. Gdy chcemy go zmienić z białego na czarny, dodajemy do siódemki jeden, ale nie zrobi się osiem, lecz zero — kolory działają w kółko. Zróbmy teraz inne doświadczenie. Wywołajmy zdefiniowaną wcześniej procedurę KOŁO — rysującą koło o promieniu 40 kroków, a następującej procedurze:

```
OTO ZGRZYT
CS PISAK 6 TŁO 7 KOŁO 2 ZAMALUJ
POD NP 40 PISAK 0 OPU KOŁO 2 ZAMALUJ
JUŻ
```

Cóż się dzieje? Kolory nakładając się, zmieniają całe kwadraciki. Dzieje się tak dlatego, że w obrębie takiego kwadracika (8 punktów na 8) są dostępne tylko dwa kolory — tła i kreski — co za tym idzie, któryś z kolorów kresek wybiera się ostatni. Napiszmy teraz
NEG
i zobaczymy, jak przedziwny jest negatyw tego rysunku. NEG to komenda powodująca zrobienie negatywu ekranu. Powrót do „pozytywu", to znów NEG. Spróbujmy wprowadzić taką procedurę:

```
OTO MIGMIG
POWTÓRZ 100 [NEG]
JUŻ
```

Przy wywołaniu widać jak ekran jest podzielony na trzy pasy — co zresztą widać najlepiej przy wgrywaniu jakiejkolwiek gry (winiety-obrazki reklamowe wgrywają się partiami od góry).

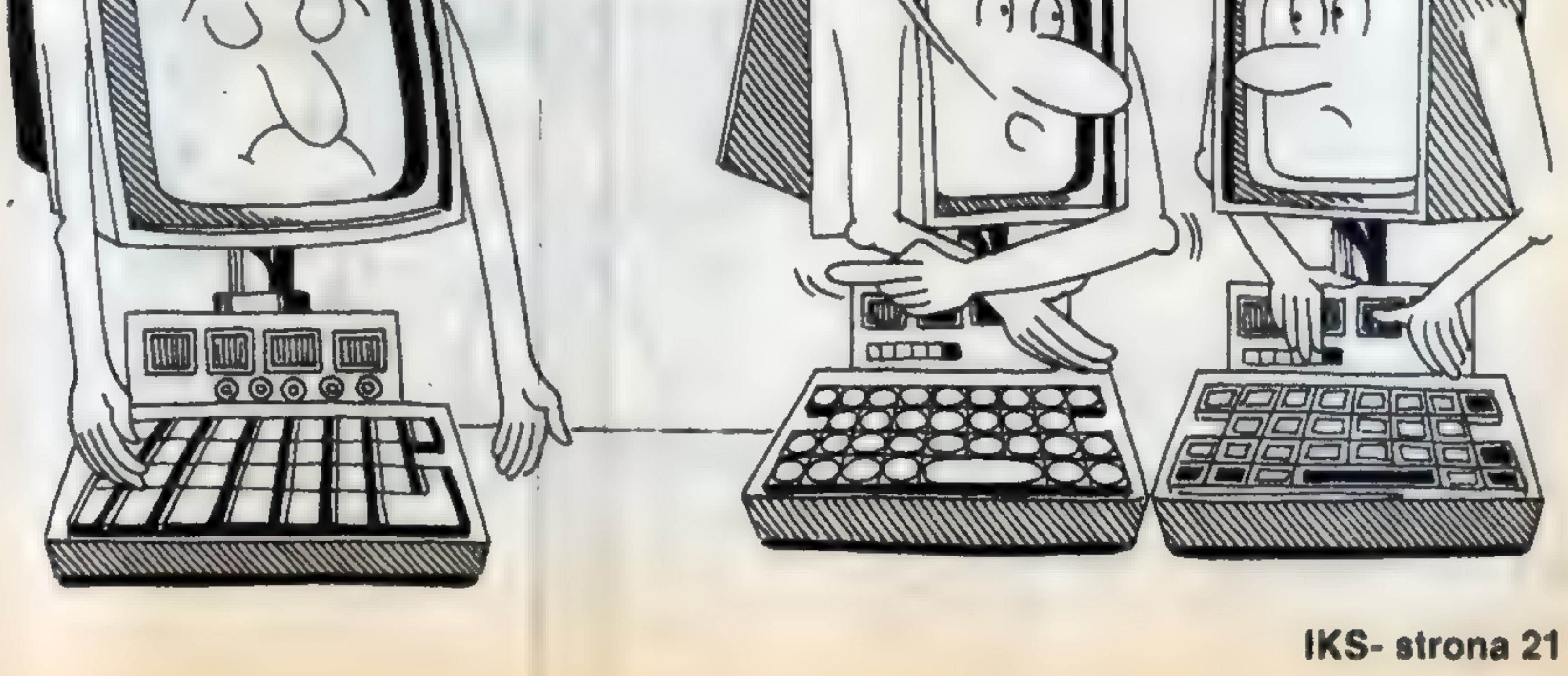
Przypomnijmy znane nam już słowa języka LOGO:

PTI LOGO	angielskie LOGO	znaczenie
PŻ	SHOWTURTLE ST	pokaż żółwia
SŻ	HIDETURTLE HT	schowaj żółwia
NAPRZÓD NP	FORWARD FD	naprzód o n kroków
WSTECZ WS	BACK BK	wstecz o n kroków
PRAWO PW	RIGHT RT	w prawo o n stopni
LEWO LW	LEFT LT	w lewo o n stopni
CZYŚĆ CS	CLEARSCREEN CS	wyczyść ekran
ZMAŻ	CLEAN	wyczyść ekran, ale żółw pozostaje gdzie był
WRÓĆ	HOME	żółw na środek
POD	PENUP PU	podnieś pisak
OPU	PENDOWN PD	opuść pisak
POWTÓRZ	REPEAT	powtórz n razy coś
ŚCIERANIE	PENERASE PE	żółw ściera — gumka
TEKSTY TS	TEXTSCREEN TS	tryb tekstowy
PISZ PP	PRINT PR	pisz — wyświetl coś
PI'	————	stała pi
OTO	TO	początek definicji — nagłówek
JUŻ	END	koniec definicji
ZAMALUJ ZAM	————	zamalowanie figury
PISAK	SETPC	ustala kolor pisaka
PISAK'	PENCOLOUR PC	zwraca numer koloru pisaka
TŁO	SETBG	ustala kolor tła
TŁO'	BACKGROUND BG	zwraca numer koloru tła
RAMKA	SETBORDER SETBG	ustala kolor ramki
KOLORYT	SETTC	ustala kolory tła i atramentu dla napisów
KOLORYT'	TEXTCOLOUR TC	zwraca numery kolorów
NEG	————	negatyw ekranu

Pracując w LOGO możemy całkowicie zapomnieć, że na klawiszach ZX Spectrum są jakieś słowa np. INPUT czy RETURN. Dla nas ważne są tylko litery (duże po wciśnięciu CAPS SHIFT) i niektóre inne znaki np. + / itp. Wszystko w LOGO wprowadza się znaczek po znaczku, co wcale nie jest tak niewygodne, jak się niektórym może zdawać. Wszystkie profesjonalne komputery czy mikrokomputery mają klawiaturę podobną do maszyny do pisania, a nie do słownika. W Spectrum zastosowano taką klawiaturę (a raczej taki sposób pracy z Basic'em) ze względu na małą pamięć (każde słowo BASIC'a kodowane jest jako jeden znak) oraz ze względu na analfabetów, którzy napisaliby np. SAFE zamiast SAVE.

W przyszłym miesiącu poznamy procedury z parametrami, nauczymy się też posługiwać edytorem — by móc poprawiać treści procedur oraz wyjaśnimy sobie to co na pewno każdy zauważył. Po kilku chwilach zabawy żółwiem „wyjeżdża" on nam poza ekran (celowo lub nie) i wtedy pojawia się po przeciwnej stronie. Dlaczego tak się zachowuje i czy może inaczej — to właśnie wyjaśnimy sobie w przyszłym odcinku.

„SIS"



3

BASIC

(Podstawy programowania)

Kontynuujemy wykład z programowania w języku BASIC na mikrokomputer ZX Spectrum. Zaczniemy od kilku uwag na temat algorytmów i schematów blokowych, a potem dalsze instrukcje BASICa.

W pierwszym numerze „IKS-a" przeczytaliśmy artykuł pt. „Co trzeba wiedzieć?". Zawiera on podstawowe zasady programowania dla początkujących programistów. Przypomnijmy początek: „Przede wszystkim trzeba zrozumieć problem. Sformułowanie zadania nie może bowiem pozostawiać wątpliwości co do jego sensowności, poprawności formalnej i merytorycznej oraz kompletności sformułowań".

„Drugim, niezmiernie ważnym etapem jest analiza problemu i dobór metody rozwiązania"... I wiele innych ważnych sformułowań.

Wykorzystanie komputera do rozwiązywania różnorodnych, często bardzo skomplikowanych zadań jest możliwe po uprzednim przedstawieniu każdego z tych zadań w postaci algorytmu.

Mianem algorytmu określamy zwykle przepis przekształcenia pewnych danych w celu uzyskania pewnego wyniku.

Dokładnie mówiąc, nazywamy tak przepis sprowadzający się do tego, by dane uznane za dopuszczalne przekształcać krok po kroku według wskazanej reguły. Dane, które mamy przekształcać, nazywamy początkowymi (wejściowymi), dane, które uzyskujemy w kolejnych krokach — wynikami pośrednimi, a poszukiwany wynik — wynikiem końcowym (danymi wyjściowymi).

Jako najprostsze przykłady możemy podać znane każdemu algorytmy arytmetyczne — dodawania, odejmowania, mnożenia i dzielenia dla liczb zapisanych w systemie np. dziesiętnym, czy też algorytm rozwiązania równań z jedną niewiadomą

podamy teraz przykłady algorytmów.

Przykład 3.1

Dane jest równanie

$$ax + b = 0$$

dla danych a i b znaleźć wartość x

Opiszemy ten algorytm w punktach

1. Sprawdzamy czy $a \neq 0$,
   jeżeli tak, to przejdź do punktu 2,
   jeżeli nie, to przejdź do punktu 4.
2. Oblicz $X = -\frac{b}{a}$
3. Zakończ obliczenia
4. Gdy $a = 0$ i $b \neq 0$ równanie sprzeczne, brak rozwiązań, gdy $a = 0$ i $b = 0$ każda wartość x spełnia dane równania.
5. Zakończ obliczenia

Przykład 3.2

Obliczyć pole trójkąta (F), promień okręgu opisanego na trójkącie (R), promień okręgu wpisanego w trójkąt (r), gdy dane są długości jego boków a, b, c.

Problem sformułowany jest jednoznacznie

a) pole trójkąta obliczamy według wzoru Herona:

$$F = \sqrt{p(p-a)(p-b)(p-c)}$$

$$\text{gdzie: } p = \frac{a+b+c}{2}$$

b) promień okręgu opisanego na trójkącie według wzoru:

$$R = \frac{abc}{4F}$$

c) promień okręgu wpisanego w trójkąt według wzoru:

$$r = \frac{F}{p}$$

Danymi są tu wielkości a, b, c, a wynikiem wartości F, R, r

Czynności prowadzące do uzyskania wyniku są następujące

1. Pobranie wartości a, b, c.
2. Obliczenie p:

$$p = \frac{a+b+c}{2}$$

3. Obliczenie wartości występującej pod znakiem pierwiastka:

$$p = (p-a)(p-b)(p-c)$$

4. Pierwiastek kwadratowy obliczany z liczb dodatnich, wobec tego sprawdzamy:

czy $z \leq 0$

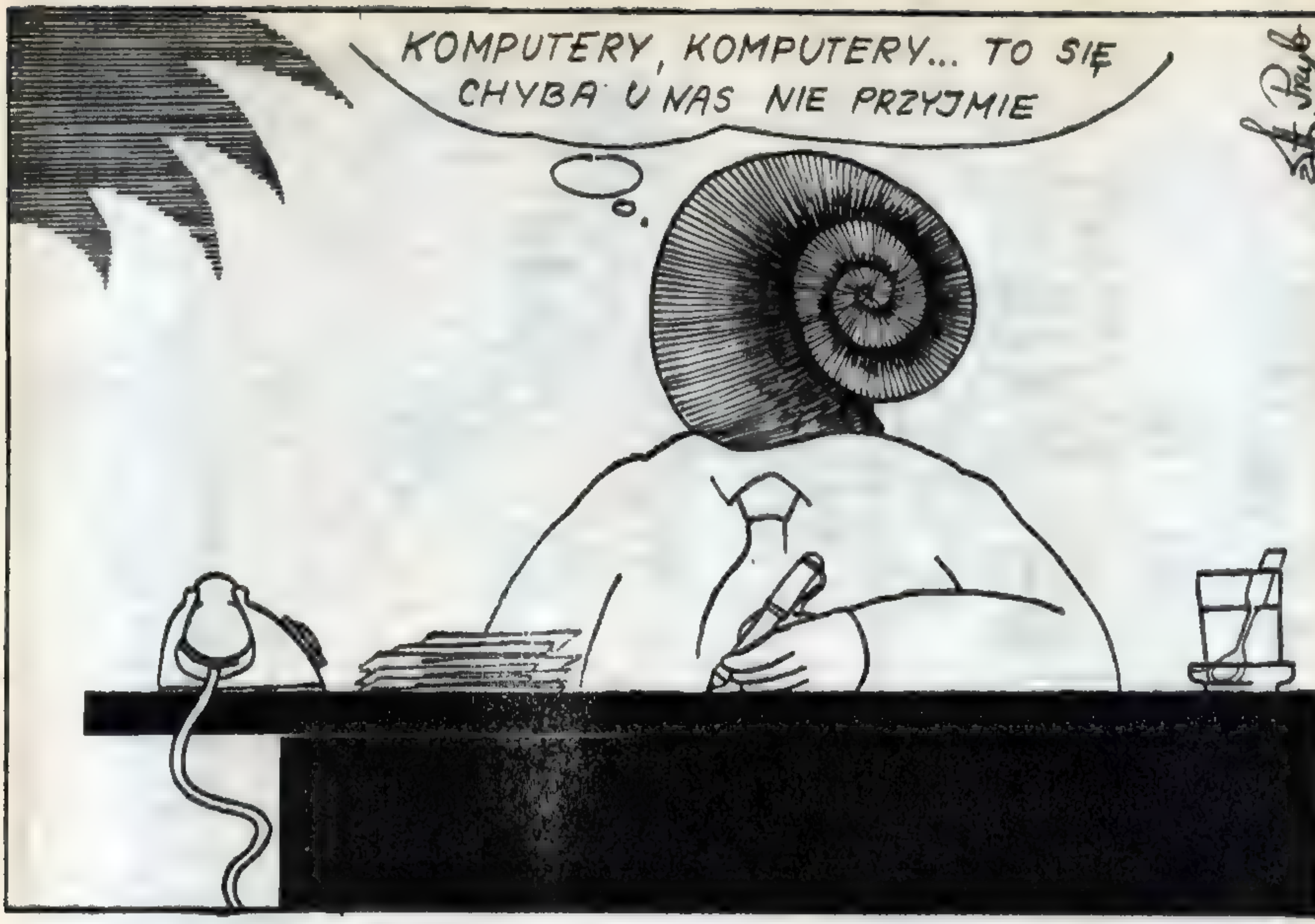
jeżeli tak, to przechodzimy do punktu 9

jeżeli nie, to przechodzimy do punktu 5

5. Obliczenie wartości F:

$$F = \sqrt{z}$$





6. Obliczenie wartości R:

$$R = \frac{a \cdot b \cdot c}{4 \cdot F}$$

7. Obliczenie wartości r

$$r = \frac{F}{p}$$

8. Zakończenie obliczeń.
9. Spełnienie nierówności $z \leq 0$ oznacza, że z podanych wartości boków nie można zbudować trójkąta (trójkąt można zbudować z podanych boków a, b, c, jeżeli: $a + b > c$ i $a + c > b$ i $c + b > a$), należy więc zakończyć obliczenia.

Wprowadzono tu wielkości pomocnicze p i z. Są to wyniki pośrednie otrzymane w czasie liczenia.

Ten sam proces liczenia można byłoby rozdrobnić przez podzielenie go na jeszcze bardziej szczegółowe czynności, np. liczenie poszczególnych różnic osobno itp.

W przykładach 3.1 i 3.2 opisaliśmy algorytmy w języku polskim w postaci słownej. Jest to najprostsza metoda opisu algorytmu.

Zauważmy, że z chwilą podania algorytmu proces liczenia staje się czynnością zupełnie automatyczną. Aby przeprowadzić obliczenia, wystarczy zrealizować ten algorytm krok po kroku dla posiadanych danych i nie ma potrzeby zagłębiania się w istotę rozwiązywanego problemu.

Formułując algorytmy należy pamiętać o tym, że algorytm musi być jednoznaczny i kompletny. Jednoznaczność polega na tym, że wielokrotne zastosowanie algorytmu dla tych samych danych doprowadza do identycznych wyników. Kompletność algorytmu wymaga uwzględnienia w nim wszystkich przypadków, jakie mogą zaistnieć w trakcie jego wykonania.

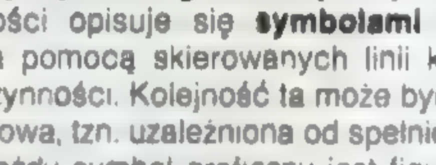
Przygotowany algorytm zapisujemy w postaci „zrozumiałej" dla maszyny, tzn. w danym języku programowania. Tak zapisany algorytm nazywamy **programem.** Z kolei program może być wprowadzony do komputera, a następnie wielokrotnie w nim wykonywany.

Przyjrzyjmy się jeszcze raz algorytmom przedstawionym w postaci słownej. Zauważamy, że taki algorytm jest dość niewygodny, wymaga użycia dużej liczby symboli i słów, może być nieprzejrzysty.

Dla uzyskania przejrzystości **algorytmy przedstawia się zazwyczaj w postaci graficznej**. W tym celu elementarne czynności opisuje się **symbolami graficznymi**, wskazując za pomocą skierowanych linii kolejność wykonywania czynności. Kolejność ta może być w szczególności warunkowa, tzn. uzależniona od spełnienia pewnych warunków. Każdy symbol graficzny jest figurą geometryczną, wewnątrz której znajduje się opis czynności; dla przejrzystości można go numerować bądź oznaczać symbolem. Taki **graficzny schemat algorytmu nazywa się schematem blokowym (lub siecią działań)**

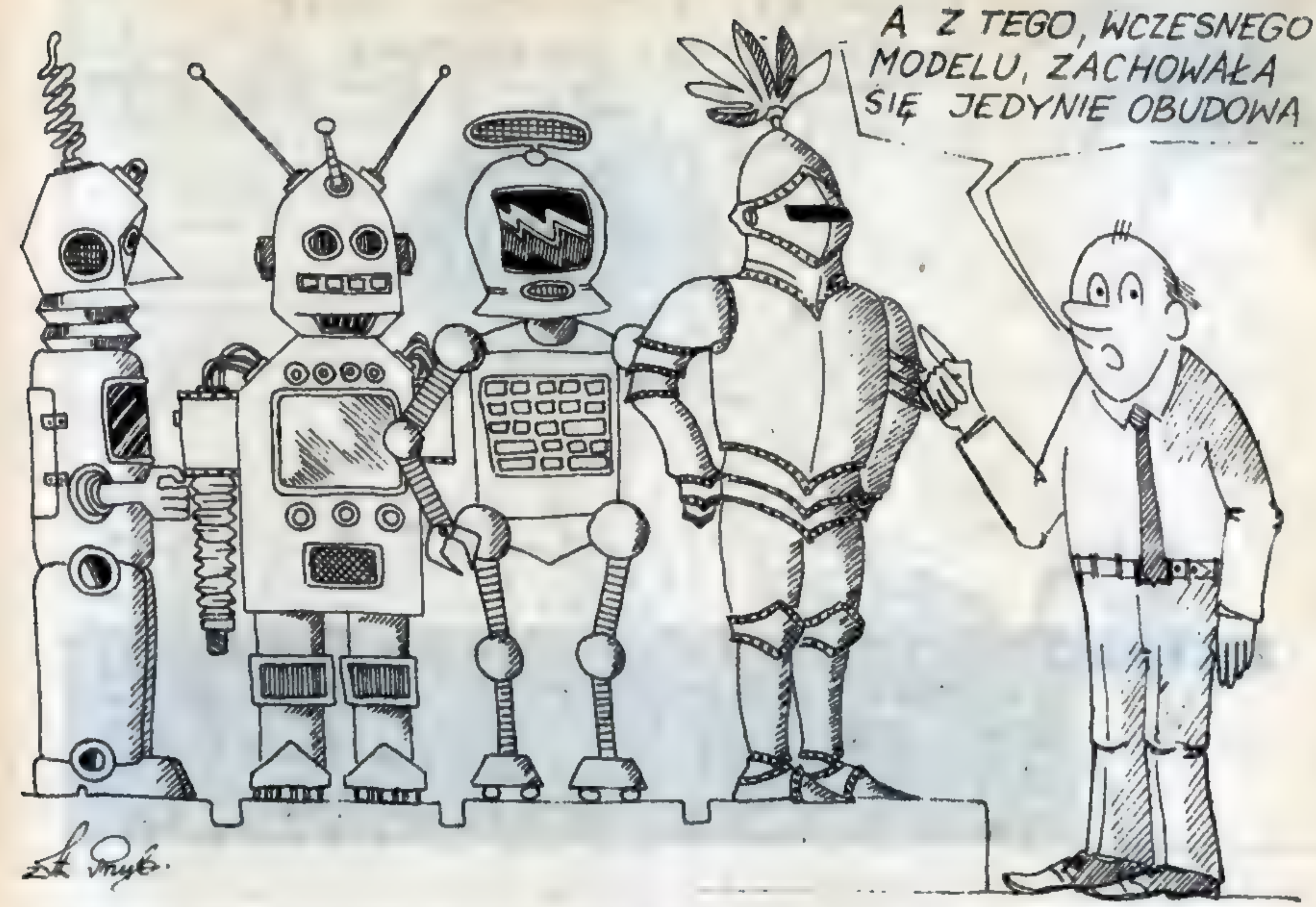
Do przedstawienia operacji używa się następujących symboli graficznych:

1. Operacja lub grupa operacji, w wyniku których zmienia się wartość, postać lub miejsce zapisu danych.



Operacje wpisane do jednego prostokąta są wykonywane wg kolejności zapisu.

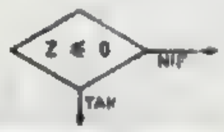
2. Wprowadzanie lub wyprowadzanie danych:



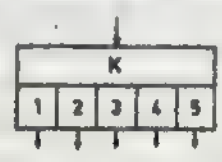


3 Decyzja — operacja określająca wybór jednej z alternatywnych dróg działania

a)

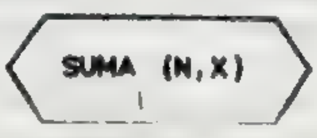


b)



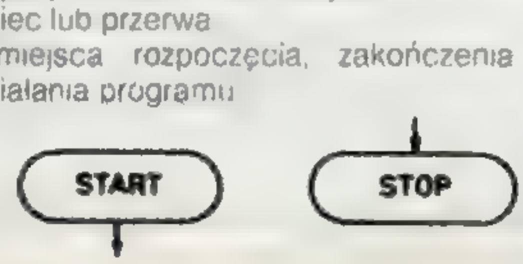
Te symbole noszą czasami nazwę logicznych lub warunkowych. Z takiego symbolu mogą wychodzić dwie lub więcej linii, każda dla jednej z możliwych dróg wyjścia

4 Wywołanie podprogramu

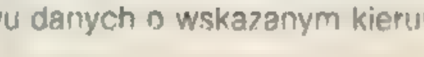


Wewnątrz symbolu graficznego podaje się nazwę podprogramu oraz spis parametrów aktualnych

5. Początek, koniec lub przerwa
oznaczenie miejsca rozpoczęcia, zakończenia lub przerwania działania programu



6. Droga przepływu danych o wskazanym kierunku przepływu



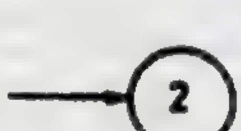
7 Skrzyżowanie dróg przepływu danych bez powiązania logicznego między nimi



8 Łączenie dróg przepływu



9 Łącznik wewnątrzstronicowy
wejście (do) lub wyjście z wyodrębnionych fragmentów schematu znajdujących się na jednej stronie



lub



10 Łącznik międzystronicowy
wejście lub wyjście z wyodrębnionych fragmentów schematu znajdujących się na różnych stronach



11. Komentarz

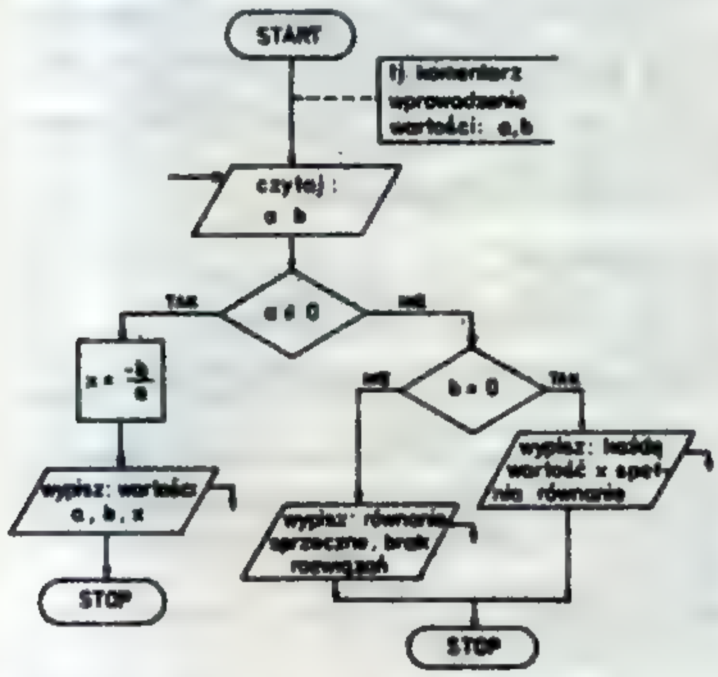


Komentarzy używa się w celu opisania pewnych części schematu. Ułatwia to zrozumienie schematu blokowego.

Powyżej podaliśmy symbole graficzne, najczęściej używane w schematach blokowych. Symbole graficzne 1, 2, 3, 4, 5, 11 nazywa się blokami. Kształt bloku określa rodzaj czynności, napis wewnątrz bloku precyzuje tę czynność oraz jej parametry. Do każdego bloku może dochodzić dowolna liczba strzałek, z każdego bloku z wyjątkiem decyzyjnego (3) może wychodzić jedna strzałka (jedno wyjście). Zaleca się uzupełnienie schematów blokowych listą nazw i oznaczeń użytych w schemacie.

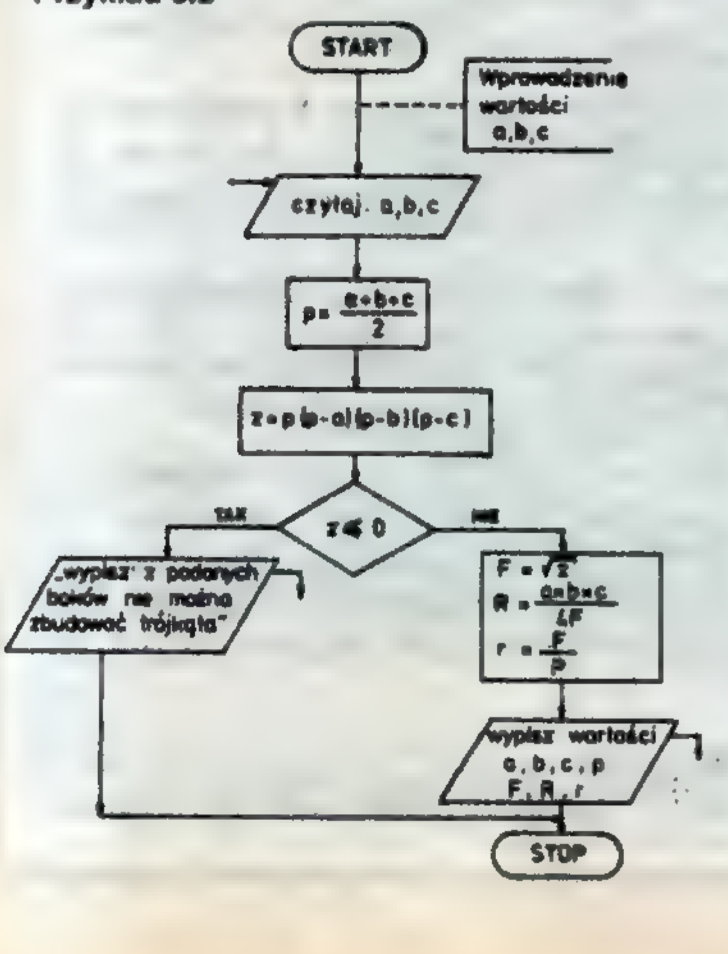
Zapiszemy teraz w postaci schematów blokowych algorytmy podane w przykładach 3.1 i 3.2.

Przykład 3.1



Oznaczenia
a, b — współczynniki równania
X — pierwiastek równania

Przykład 3.2



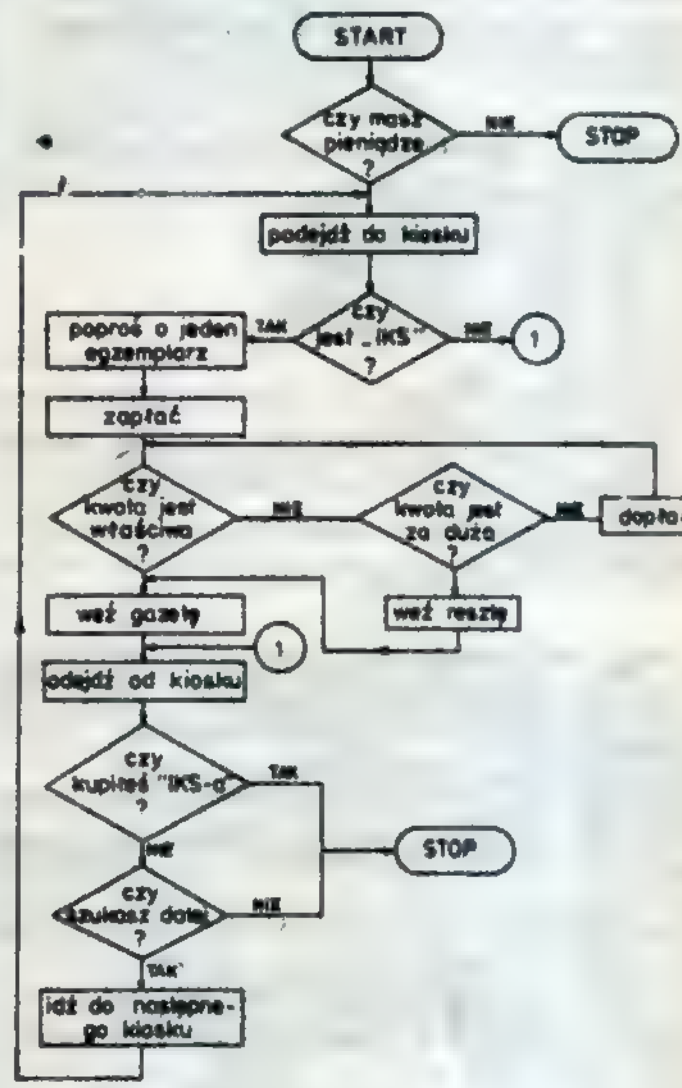
Oznaczenia
a, b, c — boki trójkąta
p z — zmienne pomocnicze
F — pole trójkąta
R — promień okręgu opisanego na trójkącie
r — promień okręgu wpisanego w trójkąt

Schematy blokowe mogą być używane do zapisu algorytmów nienumerowanych. Najlepszym przykładem zbioru algorytmów jest książka kucharska, algorytmów gotowania i przyrządzania potraw, tzn. przepisów na wykonanie pewnych czynności w dokładnie określonej kolejności, by za każdym razem, gdy je wykonamy, osiągnąć pożądany rezultat. Także wykonując codzienne czynności stosujemy zwykle świadomie lub podświadomie pewną metodę postępowania, która ma nas doprowadzić do określonego celu.

I tak np. według określonego schematu (algorytmu) uruchamiamy i prowadzimy samochód, ubieramy się, robimy zakupy itp.

Przykład 3.3.

Napisać algorytm w postaci sieci działań czynności związanych z zakupem „IKS-a" w kiosku RUCHU.



Podany wyżej algorytm może być rozszerzony np. o wyjście z domu czy o brak reszty u kioskarza, ale kolejności czynności wyszczególnionych w poszczególnych punktach nie da się zmienić.

Opierając się na powyższych przykładach, powiemy, że algorytm (w najprostszym ujęciu) jest to metoda prowadząca do rozwiązania określonego zagadnienia lub do osiągnięcia zamierzonego celu.

dokończenie w piątym numerze IKS-a

PROGRAM 18

COMMODORE C-64

Układ równań

Rozwiązanie układu równań liniowych nastręcza wiele trudności. Program ten zrealizowany na C-64 może to zadanie w wielu wypadkach ułatwić. Sposób korzystania z programu i jego szczegółowy opis uzyskuje się po uruchomieniu programu.

```
100 POKE53280,7:POKE53281,0:PRINT""
105 GOSUB2000
110 PRINT"ROZWIAZYWANIE"SPC(67)"UKLADU ROWNAN"
115 FORI=55521TO56295STEP40:FORJ=0TO14:POKEI+J,1:NEXT:NEXT
120 FORI=1249TO2023STEP40:FORJ=0TO14:POKEI+J,160:NEXT:NEXT
130 PRINT""TAB(28)"FUNKCJE:"
140 PRINT""TAB(26)"F1"SPC(1)"ZAPIS"
150 PRINT""TAB(26)"F2"SPC(1)"INSTRUKCJA"
160 PRINT""TAB(26)"F3"SPC(1)"ZMIANY"
170 PRINT""TAB(26)"F4"SPC(1)"KONIEC"
180 PRINT""TAB(26)"F5"SPC(1)"OBLICZENIE"
200 PRINT""TAB(26)"F7"SPC(1)"LIST"
260 FORI=1TO1000:NEXT
265 GOSUB2000

270 PRINT"   WYBIERZ":PRINTTAB(28)"FUNKCJE"
280 GETZ$:IFZ$=""THEN280
290 IFASC(Z$)=133THEN390
300 IFASC(Z$)=137THEN1220
310 IFASC(Z$)=134THEN560
320 IFASC(Z$)=138THEN380
330 IFASC(Z$)=135THEN1050
350 IFASC(Z$)=136THEN780
370 GOTO280
380 PRINT"":END
390 GOSUB2000:PRINT"F1  ZAPIS"SPC(35)"UKLADU"SPC(34)"ROWNAN"
400 POKE198,0:PRINT"PODAJ LICZBE ROWNAN:":PRINT:WER=7:KOL=0:GOSUB2100:N=L
410 DIMA(N-1,N-1),B(N-1),A1(N-1,N-1),B1(N-1),X(N-1)
420 PRINT"PODAJ WARTOSCI WSPOLCZYN-":PRINT"NIKOW:"
430 I=0:J=0:WER=6:KOL=11

440 POKE198,0:PRINT"A"I+1"-"J+1"="":GOSUB2100:A(I,J)=L:A1(I,J)=L
450 IFJ<N-1THENJ=J+1:GOTO470
460 J=0:IFI<N-1THENI=I+1:GOTO470
465 GOTO490
470 IFWER<23THENWER=WER+1:GOTO440
480 P=1264:K=2008:GOSUB2300:WER=6:PRINT"":GOTO440
490 P=1264:K=1024+40*WER:GOSUB2300:PRINT"PODAJ WARTOSCI WYRAZOW  ":PRINT"WOLNYCH"
500 I=0:WER=6
510 POKE198,0:PRINT"B"I+1"="":GOSUB2100:B(I)=L:B1(I)=L
520 IFI<N-1THENI=I+1:GOTO540
530 P=1024:K=2008:GOSUB2300:GOTO265
540 IFWER<23THENWER=WER+1:GOTO510
550 WER=6:P=1264:K=2008:GOSUB2300:PRINT"":GOTO510

560 GOSUB2000:PRINT"F3  ZMIANA"SPC(33)"WARTOSCI"SPC(33)"DANYCH"
570 POKE198,0:PRINT"ZMIANA WARTOSCI WSPOLCZYN-"SPC(15)"NIKOW (T/N)"
580 GETZ$:IFZ$=""THEN580
590 IFZ$="T"THEN710
600 IFZ$<>"N"THEN580
605 P=1024:K=1368:GOSUB2300
610 POKE198,0:PRINT"ZMIANA WARTOSCI WYRAZOW"SPC(15)"WOLNYCH (T/N)"
620 GETZ$:IFZ$=""THEN620
630 IFZ$="N"THENP=1024:K=2008:GOSUB2300:GOTO265
640 IFZ$<>"T"THEN620
650 KOL=13
660 WER=3:POKE198,0:PRINT"NR WIERSZA=":GOSUB2100:I=L
670 IFI<1ORI>NTHENPRINT"";:FORI=1TO25:PRINT" ";:NEXT:GOTO660

680 PRINT"JEST:"B1(I-1):PRINT"MA BYC:":WER=8:KOL=9:GOSUB2100:B1(I-1)=L:GOTO605
710 KOL=13
720 WER=4:POKE198,0:PRINT"NR WIERSZA=":GOSUB2100:I=L
730 IFI<1ORI>NTHENPRINT"";:FORI=1TO25:PRINT" ";:NEXT:GOTO720
740 WER=5:POKE198,0:PRINT"NR KOLUMNY=":GOSUB2100:J=L
750 IFJ<1ORJ>NTHENPRINT"";:FORJ=1TO25:PRINT" ";:NEXT:GOTO740
760 PRINT"JEST:"A1(I-1,J-1):PRINT"MA BYC:":WER=8:KOL=9:GOSUB2100:A1(I-1,J-1)=L
770 P=1024:K=1368:GOSUB2300:GOTO570
780 GOSUB2000:PRINT"F7  LIST"SPC(32)"WPROWADZONYCH"SPC(31)"DANYCH"
790 WER=1:I=0:J=0
800 PRINT"WSPOLCZYNNIKI:"

805 PRINT""
810 PRINT"A"I+1"-"J+1"="A1(I,J)
820 IFJ<N-1THENJ=J+1:GOTO850
830 IFI<N-1THENJ=0:I=I+1:GOTO850
840 GOTO890
850 IFWER<15THENWER=WER+1:GOTO810
860 PRINT"PRZYCISNIJ JAKIS KLAWISZ":WER=1
870 GETZ$:IFZ$=""THEN870
880 P=1224:K=2008:GOSUB2300:PRINT"";:GOTO805
890 PRINT"PRZYCISNIJ JAKIS KLAWISZ"
900 GETZ$:IFZ$=""THEN900
910 P=1024:K=2008:GOSUB2300
920 WER=1:I=0
930 PRINT"WYRAZY WOLNE:"
935 PRINT""
940 PRINT"B"I+1"="B1(I)
950 IFI=N-1THEN1000

960 I=I+1:IFWER<15THENWER=WER+1:GOTO940
970 PRINT"PRZYCISNIJ JAKIS KLAWISZ":WER=1
980 GETZ$:IFZ$=""THEN980
990 P=1224:K=2008:GOSUB2300:PRINT"";:GOTO935
1000 PRINT"POWTORZYC (T/N)"
1010 GETZ$:IFZ$<>"T"ANDZ$<>"N"THEN1010
1020 P=1024:K=2008:GOSUB2300
1030 IFZ$="T"THEN780
1040 GOTO265
1050 GOSUB2000:PRINT"F5 OBLICZENIE"SPC(31)"UKLADU"SPC(34)"ROWNAN"
1060 FORI=0TON-1:B(I)=B1(I):FORL=0TON-1:A(I,L)=A1(I,L):NEXT:NEXT
1070 FORE=0TON-2:FORI=E+1TON-1:D=-(A(I,E)/A(E,E)):B(I)=D*B(E)+B(I):FORL=E+1TON-1

1080 A(I,L)=D*A(E,L)+A(I,L):NEXT:NEXT:NEXT
1090 FORI=0TON-1:X(I)=0:NEXT
1100 FORI=N-1TO0STEP-1:FORL=ITON-1:X(I)=X(I)+X(L)*A(I,L):NEXT
1110 X(I)=(B(I)-X(I))/A(I,I):NEXT
1120 PRINT"WYNIKI:":I=0
1130 WER=1:PRINT""
1140 PRINT"X"I+1"="X(I):IFI=N-1THEN1180
1145 I=I+1:IFWER<15THEN1140
1150 PRINT"PRZYCISNIJ JAKIS KLAWISZ"
1160 GETZ$:IFZ$=""THEN1160
1170 P=1224:K=2008:GOSUB2300:GOTO1130
1180 PRINT"POWTORZYC (T/N)"
1190 GETZ$:IFZ$="T"THENP=1224:K=2008:GOSUB2300:GOTO1120
1200 IFZ$="N"THENP=1024:K=2008:GOSUB2300:GOTO265

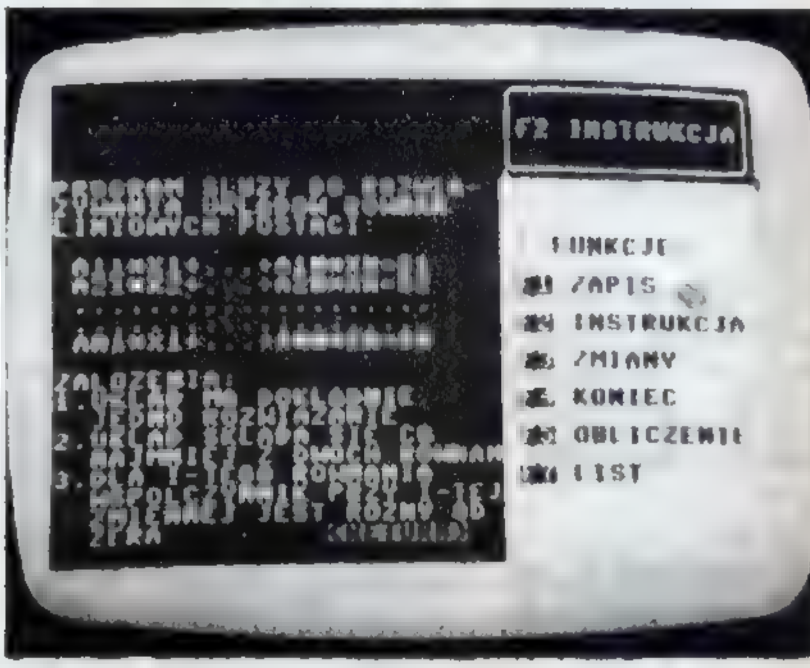
1210 GOTO1190
1220 GOSUB2000:PRINT"F2 INSTRUKCJA"
1230 PRINT"PROGRAM SLUZY DO ROZWIA-"
```

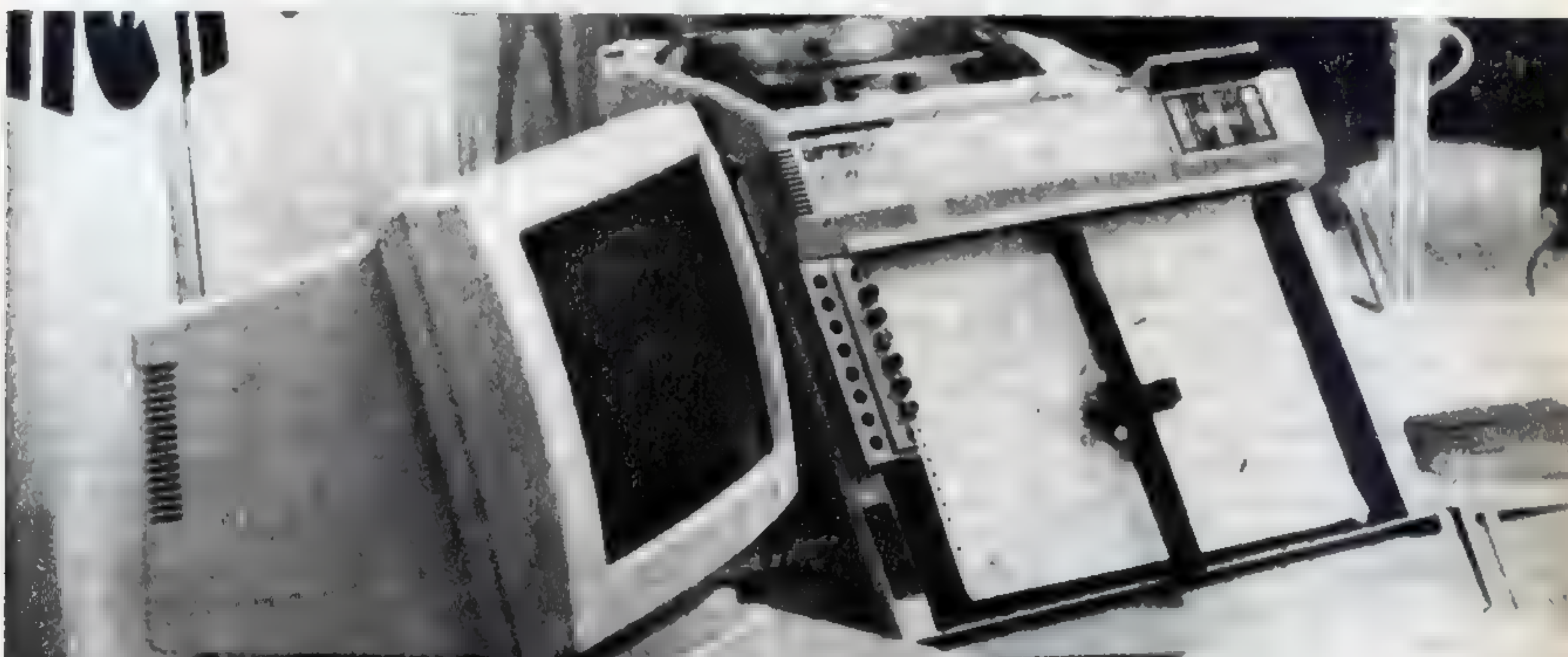
```
1240 PRINT"ZYWANIA UKLADOW ROWNAN"
1250 PRINT"LINIOWYCH POSTACI: "
1260 PRINT"[] A11*X1+...+A1N*XN=B1"
1270 PRINT" A21*X2+...+A2N*XN=B2"
1280 PRINT" ........................"
1290 PRINT" ........................"
1300 PRINT" AN1*X1+...+ANN*XN=BN"
1310 PRINT"[]ZALOZENIA: "
1320 PRINT"1.UKLAD MA DOKLADNIE"
1330 PRINT"  JEDNO ROZWIAZANIE"
1340 PRINT"2.UKLAD SKLADA SIE CO"
1350 PRINT"  NAJMNIEJ Z DWOCH ROWNAN"
1360 PRINT"3.DLA I-TEGO ROWNANIA"
1370 PRINT"  WSPOLCZYNNIK PRZY I-TEJ"
1380 PRINT"  ZMIENNEJ JEST ROZNY OD"

1390 PRINT"  ZERA          []<RETURN>[]"
1400 GETZ$:IFZ$<>CHR$(13)THEN1400
1410 P=1224:K=2008:GOSUB2300
1420 PRINT"[][][][][][]UWAGI DLA UZYTKOWNIKA:"

1430 PRINT"[]PROGRAM POZWALA WYKONAC"
1440 PRINT"OBLICZENIA DLA KILKU PR-"
1450 PRINT"DOBNYCH UKLADOW ROWNAN"
1460 PRINT"BEZ KONIECZNOSCI PONOW-"
1470 PRINT"NEGO WPROWADZANIA DANYCH."
1480 PRINT"WYKORZYSTAJ W TYM CELU"
1490 PRINT"FUNKCJE 'F3 ZMIANY' "
1500 PRINT"JAKO PIERWSZA WYKONAJ"
1510 PRINT"FUNKCJE 'F1 ZAPIS' "
1520 PRINT"I WPROWADZ DANE. FUNKCJI"
1530 PRINT"TEJ NIE NALEZY UZYWAC"
1540 PRINT"POWTORNIE. "
1550 PRINT"FUNKCJA 'F7 LIST' POZWALA"
1560 PRINT"PRZEGLADNAC AKTUALNA POS-"

1570 PRINT"TAC UKLADU ROWNAN."
1580 PRINT"                []<RETURN>[]"
1590 GETZ$:IFZ$<>CHR$(13)THEN1590
1600 P=1224:K=2008:GOSUB2300:GOTO265
1999 GOTO1999
2000 PRINT"[]"TAB(25)"┌──────────┐";
2010 PRINTSPC(25)"│          │";
2020 PRINTSPC(25)"│          │";
2030 PRINTSPC(25)"│          │";
2040 PRINTSPC(25)"└──────────┘[][][][][][]
[][][][][][][][][][][]";
2050 FORI=55321TO55495STEP40:FORJ=0TO14:
POKEI+J,1:NEXT:NEXT:RETURN
2100 C=1024+KOL+40*WER:M=1048+40*WER:MD=
C:L$=LEFT$(L$,0)
2110 GETZ$
2120 IF(Z$>="0"ANDZ$<="9")ORZ$="."ORZ$="
-"ORZ$="+"ORZ$="E"THEN2150
2130 IFZ$=CHR$(13)THEN2200
```





Sposób na projektowanie... komputer + plotter.

Z historii mikrokomputerów

(Wielka Brytania)

Dostarczane zza Oceanu do Anglii komputery, takie jak PET, APPLE II czy TRS-80 model 1, kosztowały aż 500 funtów. Z pewnością fakt ten znacznie przyspieszył w Wielkiej Brytanii prace nad własnymi mikrokomputerami, a konstrukcje brytyjskiej produkcji na przykład: NASCOM 1, Mk 14, UK 101 były niemal trzykrotnie tańsze od amerykańskich. Budowano je na jednym pakiecie, który poza procesorem mieścił 2 lub 4K RAM i ROM. Komputery te akceptowały kod ASC II.

Zapamiętajcie!

Bit — najmniejsza jednostka informacji, przyjmuje wartość 1 lub 0 (tak lub nie)

Bajt to 8 Bitów

KBajt: 2^{10} = 1024 Bajty = 8 · 1024 = = 8192 Bity

MBajt: 1024 KBajty = 1024 · 1024 = = 1 048 576 Bajtów = 8 · 1 048 576 = = 8 388 608 Bitów

GBajt: 1024 MBajty = 1024 · 1024 = = 1 048 576 KBajtów = = 1024 · 1024 · 1024 = = 1 073 741 824 Bajty = = 8 · 1 073 741 824 = = 8 589 934 592 Bity

Znaczącą datą w historii rozwoju mikrokomputerów jest rok powstania pierwszego pisma im poświęconego. W Wielkiej Brytanii stało się to w 1978 roku, kiedy ukazało się pismo „PERSONAL COMPUTER WORLD".

W niecałe dwa lata potem pojawiają się pierwsze przenośne mikrokomputery ACORN ATOM i SINCLAIR Z80. Już wówczas miały one „wbudowany" BASIC w pamięci ROM. ACORN ATOM bazował na procesorze 6502; miał 2K RAM (z możliwością rozbudowy do 12K), monochromatyczny telewizor.

Sir Clive Sinclair (jeszcze wówczas po prostu „Pan") wymarzył sobie mikrokomputer, który kosztowałby mniej niż magiczna cena 100 funtów. Po wielu miesiącach ciężkiej pracy jego firma „Sinclair Research" wyprodukowała mikrokomputery ZX80 o pamięci 1K RAM z BASIC-em i interfejsem do typowych, domowych telewizorów i magnetofonów. Powodzenie tego sprzętu przyczyniło się do uruchomienia produkcji ZX81, który był już o 50% tańszy od swego poprzednika.

W 1982 roku na rynku niepodzielnie panował mikrokomputer ZX Spectrum z pamięcią 48K RAM + kolory za 125 funtów. W tym samym czasie komputer ten jako TIMEX SINCLAIR 2000 zawędrował do USA.

Jak dotychczas do Spectrum napisano największą liczbę komputerowych gier, dla dzieci i dorosłych — jedynie Commodore może być dla niego konkurencją.

Tymczasem ACORN wyprodukował na zlecenie rozgłośni BBC mikrokomputer o tej samej nazwie — BBC MICRO. Komputer ten i jego oprogramowanie konstruowano z myślą o sprzężeniu nowoczesnej technologii z telewizją dydaktyczną. BBC zbudowany jest na procesorze 6502, a jego zalety podkreśla doskonała kolorowa grafika.

Jednak wadą nie do pokonania dla BBC okazała się zbyt wysoka cena, a ograniczoność zastosowania doprowadziła producenta niemal do bankructwa. Ostatecznie BBC został wykupiony przez firmę OLIVETTI i ślad po nim zaginął.

W historii brytyjskich mikrokomputerów znaczące były także inne konstrukcje: NEW BRAIN i Jupiter ACE, który zbudowano na podstawie procesora ZX80 (zamiast BASICA w ROM-ie mieścił się FORTH). Konieczność zdobycia przez każdego użytkownika mikrokomputera ACE odpowiednich kwalifikacji okazała się zgubna dla producenta — Jupiter zbankrutował.

Potem pojawiły się mikrokomputery DRAGON 32 na procesorze 6809. ORIC i pierwszy, składany, do dziś produkowany, przenośny (lap — held) mikrokomputer HUSKY. Jego produkcję rozpoczęto jeszcze przed rokiem 1981, a brak wcześniejszych informacji o nim tłumaczy podstawowe przeznaczenie komputera HUSKY — wojsko. Przystosowany jest do pracy na wolnym powietrzu, można nim rzucać o podłogę, jest po prostu niezawodny. Elektronika tego urządzenia oparta jest na układach CMOS i układach kompatybilnych z Z80 (procesor C800 firmy National Semiconductor, zegar 4MHz, system CP/M).

Obudowa komputera HUSKY jest prosta, bez mechanicznych pamięci masowych. Ta pozorna wada czyni sprzęt mobilnym, a niezbędne dane stałe i programy są zarejestrowane w trwałej pamięci CMOSRAM.

Już w 1981 roku prezentowany mikrokomputer miał wbudowaną pamięć RAM o wielkości 144K, a ostatni model HUSKY HUNTER jest wielkości połowy arkusza A4, waży 1200 g, pracuje pod wodą, a jego baterie zapewniają pracę przez 50 godzin. Ekran typu LCD wyświetla informacje w 8 wierszach po 40 znaków lub 240×64 piksele. Pamięć ROM ma 48K i mieści się w niej edytor tekstów, program komunikacji, interpreter BASICA, file manager DEMOS kompatybilny z CP/M 2.2 Pamięć RAM ma wielkość 256K w tym 48K z przeznaczeniem na symulację dysku.

Kolejne modele mikrokomputera HUSKY to:

— HUSKY SP (special performance), 352K RAM, baterie na 100 godzin pracy,
— HUSKY IS (Intrinsically Safe) przeznaczony do zabezpieczenia elektrowni, petrochemii, ko-

pam itp wyposażony w RS — 422,

- HUSKY AD (analog/digital) ma osiem kanałów zamiany sygnałów analogowych na cyfrowe,
- HUSKY M208, pierwszy komputer odporny na krańcowo złe warunki zewnętrzne; drgania, uderzenia, temperaturę. Producent twierdzi, że sprzęt ten wytrzyma warunki „3 wojny".

Brytyjskie czekanie na 16-bitowy komputer zakończyło się 4 lata temu wyprodukowaniem SIRIUSA. Natychmiast firma ACT odpowiedziała mikrokomputerem APRICOT, którego w 1983 roku sprzedano 35 000 szt. Komputer ten jest ciągle sprzedawany w swej oryginalnej obudowie z bliźniakami jednostronnych dysków 256 KB plus 9-calowy monitor z zielonym ekranem.

W marcu 1984 roku firma ACT rozpoczęła sprzedaż modeli Xi20s o pamięci 1 MB RAM, twardy dysk 20MB i 720 KB floppy dysk. Brak kompatybilności z IBM producent zastąpił dwoma tysiącami własnych programów. Było to kolejnym objawem procesu wypierania z rynku konkurentów ACT, w tym IBM. Duży wpływ na te zmiany miała także produkcja lokalnych sieci POINT 32, które łączą do 32 komputerów APRICOT.

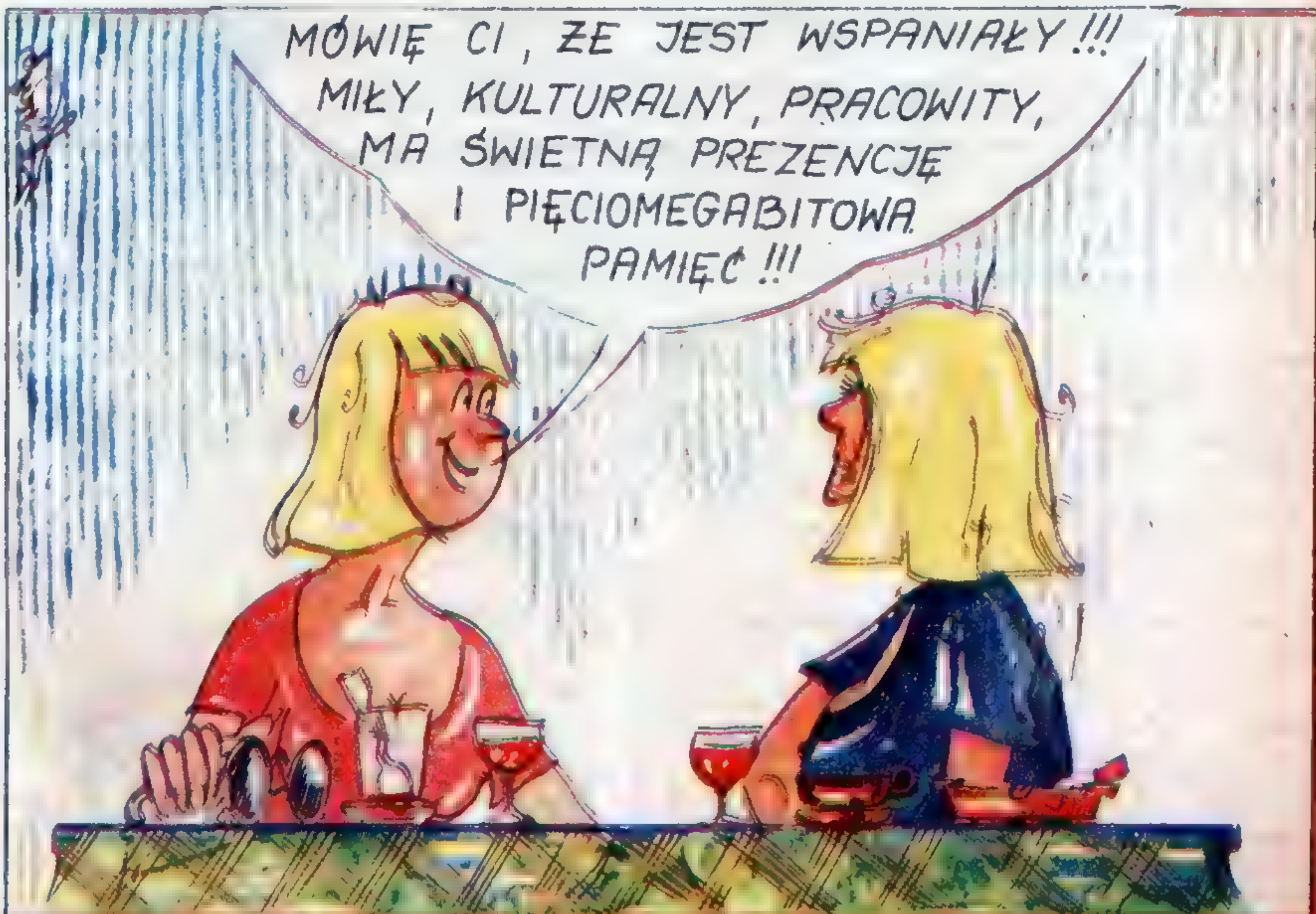
Użytkownicy mikrokomputera APRICOT chwalą sobie jego małe rozmiary i łatwość transportu. Ten typ komputera między innymi upodobali sobie aptekarze. Pomaga im w tym skrupulatnie dostarczane fachowe oprogramowanie.

Warto podkreślić, że procesor w APRICOT 8086 jest szybszy niż 8088 w IBM PC. Ponadto prezentowany mikrokomputer został ostatnio uzupełniony o bardzo użyteczne oprogramowanie, COMMUNIQUE, pozwalające korzystać z elektronicznej poczty — Telcom Gold, a także z terminali VT100, rozpowszechniających oprogramowanie i innych źródeł cennych informacji (MICROMAIL, MULTISTREAM).

Na zakończenie warto zwrócić uwagę na dziwną historię jednej z najbardziej popularnych firm mikrokomputerowych w Polsce — Sinclair Research. Po olbrzymich sukcesach rynkowych, których uwieńczeniem był ZX Spectrum 48 KB RAM, firma zainwestowała między innymi w wersję QL. Zdarzenie to odegrało niebagatelną rolę w kryzysie finansowym Sinclair'a. Firmie groziło bankructwo. Uzyskany w ubiegłym roku kredyt około 15 milionów funtów pozwolił na pokrycie długów i radykalną obniżkę ceny QL z 399 na 199 funtów! Wkrótce firma otrzymała zamówienie na sprzęt opiewające na 10 milionów funtów. Ku wielkiemu zaskoczeniu okazało się jednak, że nie QL jest przedmiotem zamówienia, stanowi bowiem tylko 15% całości — reszta to znane u nas Spectrum i Spectrum plus. Tymczasem ZX80 jest nadal najlepiej sprzedawanym komputerem domowym.

Niepowodzenia mające swe źródło w QL odzwierciedla między innymi opinia, że nowa wersja AMSTRADA PCW 8256, mimo ceny 399 funtów, nie będzie miała konkurencji w QL — który jest tylko zabawką, ma kolorowy monitor, jego przeznaczenie to gry, a nie zastosowanie profesjonalne.

W. GOGOŁEK



Odpowiedzi na zadania z drugiego numeru „IKS-a"

Zadanie 1

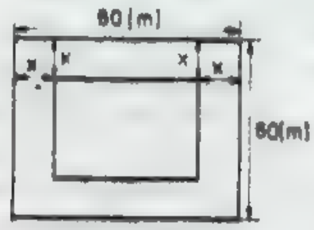
Oznaczamy szerokość drogi przez x. Z warunków zadania wynika, że.

$$(80 - 2x)(60 - 2x) = \frac{80 \cdot 60}{2}$$

stąd otrzymujemy równanie kwadratowe

$$x^2 - 70x + 600 = 0$$

Jego rozwiązaniem są dwie wartości: $x_1 = 10$ i $x_2 = 60$. Uwzględniając treści zadania odrzucamy wartość $x_2 = 60$, czyli szerokość drogi wynosi 10 m.



Zadanie 2

Załóżmy, że pierwszy przedmiot kupiono za x zł, wtedy drugi przedmiot kupiono za 225 — x zł. Ze sprzedaży pierwszego przedmiotu otrzymano 25 proc. dochodu, tzn., że został on sprzedany za 1,25x zł. Ze sprzedaży drugiego przedmiotu otrzymano 50 proc. dochodu, czyli sprzedano go za 1,5 (225 — x) zł.

W zadaniu podano, że całkowity dochód (w stosunku do ceny zakupu wynoszącej 225 zł) wynosi 40 proc.
Zatem do kasy antykwariatu wpłynęło za oba przedmioty:

1,40 • 225 = 315 (zł), stąd:

$$1\tfrac{1}{4}x + 1\tfrac{1}{2}(225 - x) = 315$$

Zatem pierwszy przedmiot kupiono za 90 zł a drugi za 135 zł.

Zadanie 3

Po 2 godzinach napełniła się $\frac{1}{6}$ zbiornika (woda wlewała się przez pierwszy kran). Oba krany w ciągu godziny napełniały $\frac{1}{12} + \frac{1}{8} = \frac{5}{24}$ zbiornika stąd $\frac{5}{6}$ zbiornika napełniły oba krany w ciągu 4 godzin, bo

$$\frac{5}{6} = \frac{5}{24}x; \text{ a zatem } x = 4$$

zatem zbiornik zostanie napełniony całkowicie po 6 godzinach.

Zadanie 4

W zadaniu tym należało podać algorytm, czyli dokładny przepis wykonania w określonym porządku skończonej liczby operacji, pozwalających na rozwiązanie równania kwadratowego $ax^2 + bx + c = 0$. Poniżej przedstawiony został algorytm w postaci schematu blokowego. Algorytm ten można też było przedstawić w innej postaci np. jako opis słowny kolejnych czynności w określonym porządku.

W następnym numerze dalsze odpowiedzi

Liga Myślących

Zadanie nr 1

Poprzedniego dnia zostały wyregulowane dwa zegary ścienny i budzik. Zegar ścienny spóźnia się o 2 minuty na godzinę, budzik zaś spieszy się o 1 minutę na godzinę. Następnego dnia rano oba zegary stanęły. Zegar ścienny wskazywał godzinę siódmą rano, budzik zaś ósmą.

W jakim czasie zostały wyregulowane i nakręcone dnia poprzedniego oba zegary?

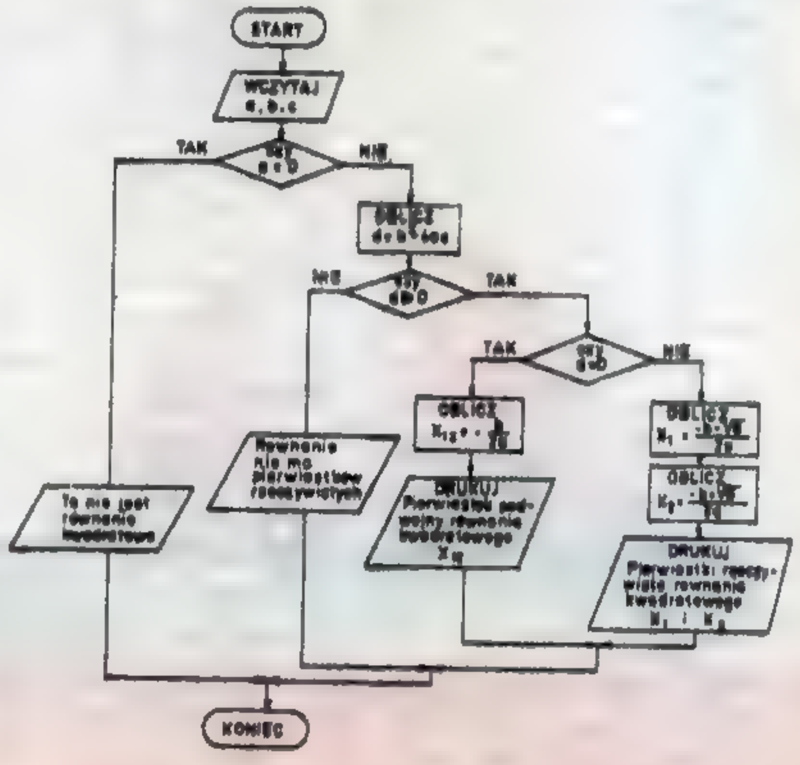
Zadanie nr 2

Jakie jest prawdopodobieństwo losowego wyciągnięcia spośród 10 cyfr od 0 do 9 trzech takich, aby można było z nich ułożyć liczbę podzielną przez 9?

Zadanie nr 3

W turnieju szachowym bierze udział n zawodników, przy czym n — 2 zawodników rozegrało między sobą partie każdy z każdym, zaś z dwóch pozostałych, jeden rozegrał 10 partii a drugi jedną. Ilu było zawodników i czy wspomniani dwaj gracze grali ze sobą, jeżeli wiadomo, że rozegrano w sumie 55 partii?

Rozwiązania zadań prosimy przesyłać do redakcji do końca września br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody - niespodzianki



Pocztowa giełda

Rozwiązanie krzyżówki nr 2. Hasło brzmi „**BAW SIĘ z NAMI**". Bony pieniężne 1000 zł wylosowali: **Julita Dryggas** Gdańsk, **Karol Piekarz** Rybnik, **Miron Rogacki** Konin, **Witold Bardzki** Łódź, **Paweł Szpakowski** Warszawa

Nagrody książkowe otrzymują **Krzysztof Awodziewicz** Szczecin, **Adam Szałamacha** Chorzów-Batory, **Janusz Kita** Stalowa Wola, **Leonard Radliński** Wrocław, **Wacław Uchlik** Rumia, **Tomasz Bubiak** Jaworzno, **Piotr Burek** Kamienna Góra, **Mariusz Ladziński** Rzeszów, **Robert Soliński** Grajewo, **Orłowska Mirosława** Kłodzko, **Anna Cholewińska** Lublin

---

Niebywałym powodzeniem cieszy się „**Liga Myślących**". Zadania z pierwszego numeru „IKS-a" rozwiązało prawidłowo ponad 400 osób. Z drugiego numeru, mimo że przysłano więcej rozwiązań, tylko kilkanaście uzyskało maksymalną liczbę punktów. Na czołówkę i „liderów" przyjdzie jeszcze trochę poczekać

Przepraszamy za mylną „stopkę" w „Lidze Myślących" w nr. 3 „IKS-a"

Warszawskie (czerwcowe) notowania „komputerowe":
ZX Spectrum Plus — 110 tys.
ZX Spectrum 48 + magnetofon — 105 tys.
ZX Spectrum 128 K — 195 tys.

Atari 800 XL + magnetofon — 115 tys.
Atari 130 XE — 135 tys.

Commodore C-64 + magnetofon + 2 joysticki — 290 tys.
Commodore 16 + magnetofon — 60 tys.

Meritum 1 — 155 tys.

Prosimy o nadsyłanie notowań „komputerowych" z innych miast (z adresem giełdy).

Ogłoszenia

Sprzedam programy, literaturę do: Commodore C-64, C-16, C-116, Plus 4, Atari 800 XL. **Marek Górny skr. p. 1170 40-001 Katowice 1.**

Kupię grę elektroniczną. **Andrzej Dopke, ul. Rejtana 7/17 84-200 Wejherowo, woj. Gdańsk.**

Kupię oprogramowanie i moduł ROM-u (TV Extended Basic) na komputer T 199/4A **Piotr Uchyta, ul. Kościuszki 3 44-240 Żory.**

Wymienię programy na Commodore C-64. **Michał Łaszewicz, ul Szpitalna 35a/54 15-295 Białystok tel. 215-58.**

Sprzedam Commodore C-116 + magnetofon + książkę Basic i kasety. **Robert Zdziński Centr. B 1/100 31-926 Kraków.**

Lista „przebojów"

W naszych notowaniach systematycznie zachodzą zmiany. Na czoło wysunął się program z trzeciego numeru „**Łamigłówka**". Czytelnicy lubią programy „myślące" — taki wniosek nasuwa się z otrzymanych listów. Chwalicie „**Odwracankę**", która z pierwszego miejsca spadła na drugie. — oba programy przeznaczone są na komputer „**Atari**", może nasi czytelnicy pokuszą się o adaptowanie ich na inne typy mikrokomputerów — wytrwałym życzymy powodzenia. Na kolejnych miejscach na naszej liście, z niewielką różnicą otrzymanych głosów znajduje się program dla statystyków „**Współczynniki korelacji**" i dwa programy na Commodore C-64 „**Projektant sprites**" i „**Projektant — znaki**".

Uwaga! Już od kilku tygodni komputerowe programy prezentujemy w sobotnio-niedzielnym wydaniu „**Żołnierza Wolności**" — **Horyzont**. Na łamy zapraszamy naszych czytelników. Wasze najciekawsze propozycje opublikujemy. Liczy się każdy dobry pomysł — zapraszamy do współpracy.

W naszym komputerlandzie

W naszym komputerlandzie sensacja! Najpoważniejsze zespoły robotów ustaliły, że pieniądze spadają z niebios. W związku z tym w bardzo prosty sposób, nie wymagający żadnego wysiłku poza szukaniem frajerobów, można zdobyć fortunę.

Wystarczy tylko ułożyć listę 12 robotów, najlepiej z kręgów własnej robociej rodziny, po czym sprzedać listę frajerobom, którzy dopasują podobne sobie typy i generacje robotów i mają obowiązek wysłać gotówkę do Pierwszego (w liście).

Poczta jest zadowolona bo kasuje opłaty. Dziełoroby zwane listonoszami są zadowolone, bo dostają od Pierwszego dolę.

Pierwszy jest najbardziej zadowolony, bo frajeroby przysyłają mu szmal, który upycha w podwójne dno obudowy monitora i pęka ze śmiechu na widok reszty, która biega z opracowanymi przez siebie fałszywymi listami żądając opłat.

Dla zapewnienia reklamy w przedsięwzięciu Pierwszy rozsiewa pogłoski o wielomilionowych fortunach, jakie roboty porobiły w naszym komputerlandzie i interes kwitnie.

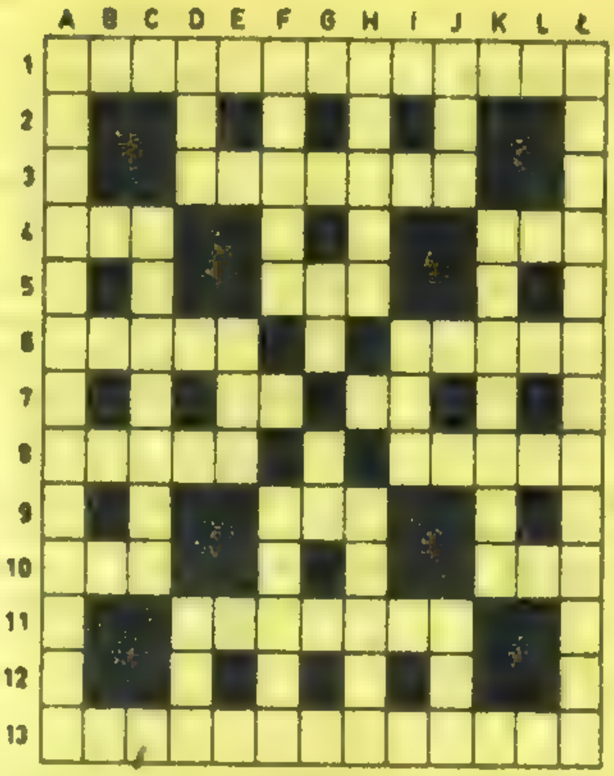
Pomysł był do tego stopnia dobry, że nawet pracowity Spektruś postanowił taką skróconą listę uruchomić i umieścił na niej: redaktorów „IKS-a" a następnie siebie.

Nie jest jednak wydrwigroszem i ową listę oferował za darmo uczciwie mówiąc, że pragnie dorobić się kosztem czytelników, aby kupić wreszcie redakcji „IKS-a" niezbędny sprzęt.

Pełen nadziei Spektruś czeka na miliony...

Podglądał: **Eugeniusz MLĘCZAK**

Krzyżówka nr 4



POZIOMO: A-1 Mikrokarta przejrzysta, D-3 Podstawowa komórka społeczna, A-4 Patetyczny utwór poetycki, K-4 Autor trylogii „Czas nieutracony", F-5 Znany polski przedwojenny bokser, A-6 Organ dowodzenia sił zbrojnych, I-6 Drapieżna ryba morska z rzędu chrzęstnoszkieletowych, E-7 Miara powierzchni gruntu, H-7 Matka bogów, A-8 Miasto w Turcji. Dawna Smyrna, I-8 To co jest unoszone przez wiatr lub wodę, F-9 Rodzaj sieci na ryby, A-10 Rozebrana lala na fotce, K-10 4840 jardów kwadratowych, D-11 Przenośnie grubas kojarzy się z masłem, A-13 Metodyka dokonywania pomiarów na człowieku.

PIONOWO: A-1 Wyjawienie czegoś ukrywanego, C-4 Pracuje bez udziału człowieka, D-1 Jednostka podstawowa makrocząsteczki otrzymanej przez polimeryzację, D-11 Duży gęsty stary las, F-1 Służy do zmiany alfabetu, F-9 Np. Gujazyl, G-5 Pies z elementarza, H-1 Myśliwy beocki, H-9 U ludów tureckich okup płacony dawniej rodzicom narzeczonej, I-6 Pierwiastek, J-1 Papuga, J-11 Piłka poza linią boiska, G—8 Szósty, stopień gamy C-dur, K-4 Słownictwo, Ł-1 Techn. otrzymywanie metali z tlenków przy użyciu metalicznego glinu.

HASŁO: 3—G, 1—I, 1—H, 4—Ł, 13—F, 5—C, 1—L, 4—F, 2—H, 2—A, 11—Ł, 3—F, 13—G, 7—K, 6—B, 10—L, 4—H, 11—H, 10—F.

[illegible]



TANDY 1000, wyprodukowany przez firmę Tandy (Radio Shack, jest jednym z tańszych komputerów tej klasy, jego cena, z pojedyńczą, 128 kilobajtową stacją dysków, wynosi 999 dolarów lub 1746 z podwójną — 256 kilobajtową. Jest w pełni kompatybilny z innymi komputerami osobistymi tej klasy. Działa w oparciu o 16/8 bitowy procesor 8088. Ma możliwości rozszerzenia pamięci do 640 KB. Zastosowany ekran jest monochromatyczny i zawiera 80 znaków w 25 wierszach. Możliwe jest zastosowanie monitora kolorowego.

(c.)



„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera, Ryszard Rogoń. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 7821. Nr ind. 382809 P-73

IKS
INFORMATYKA KOMPUTERY SYSTEMY

CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 5/1986/ ISSN0860—2794



Rys. Michał Przybyłowski

Przyjemności lata mamy już za sobą. Co to były za wakacje! Po raz pierwszy tak wielu spędziło je przed komputerem. Na licznych obozach te cacka naszych czasów były ważniejsze od kucharza i pogody razem. A że ta ostatnia nie zawsze dopisywała, tym intensywniej stukano w komputerowe klawisze.

Kończymy pierwszą piątkę numerów „IKS-a". Każdy numer był inny, i mamy nadzieję, że lepszy. Jeśli tak jest, to bez wątpienia stało się to także przy pomocy naszych Czytelników, którzy nie skąpili nam rad i nie ukrywamy, że z nich korzystamy. Na barki skromnej redakcji spadł dosłownie grad korespondencji. I choć przeczytaliśmy każdy list (do dziś czujemy w dłoniach trud otwierania niezwykle starannie zaklejanych kopert), nie na każdy zdołaliśmy odpowiedzieć — prosimy o cierpliwość — już wkrótce nadrobimy zaległości. Ale już teraz dziękujemy za rady, życzenia i wszystkie uwagi.

Mamy jednak prośbę, rozwiązania krzyżówek, przysyłajcie nam na kartkach pocztowych, a rozwiązania „Ligi Myślących" opatrujcie tym właśnie dopiskiem.

Już wkrótce na łamach „IKS-a" znajdziecie najciekawsze propozycje naszych czytelników. Zapraszamy do dzielenia się swoimi doświadczeniami w programowaniu: Jeśli coś można zrobić lepiej, to zróbmy — rusz głową.

W naszym piśmie więcej miejsca poświęcać będziemy programowaniu i zgodnie z życzeniem Czytelników rozszerzamy nasz „komputerowy asortyment". Prosicie nas często o gry, dobre, graficzne. No cóż dobra atrakcyjna gra, z dobrą grafiką, to przede wszystkim bardzo dobry program i to napisany w języku wewnętrznym — jego objętość byłaby tak duża, że „pożarłby" większość naszych kolumn. Zresztą o kasety z atrakcyjnymi programami coraz łatwiej, a ich sprzedażą zajmują się również Kluby Międzynarodowej Prasy i Książki (o czym piszemy obok). A jeśli już gry, to proste, logiczne. Tym lepsze im błyskotliwszy projektant i zdolniejszy programista. Może wśród Waszych propozycji znajdzie się pomysł na miarę Rubika. Czekamy na Wasze pomysły i otwieramy dla nich naszą „Listę przebojów". Nadal prezentować będziemy, przykłady praktycznego zastosowania komputerów. Zaczynamy od problemów nam najbliższych, bo komputer w domu nie może być jedynie kosztowną zabawką, ale w miarę możliwości powinien nas wspierać : Biblioteka, obliczenia, programy edukacyjne, to tylko niektóre przykłady jego zastosowania.

Jeśli ułatwimy wam poznanie komputera uznamy to za swój sukces, bo dialog z tym urządzeniem to realna rozmowa o przyszłości.

REDAKCJA

„IKS" w ciągłej sprzedaży

Już wkrótce niektóre Kluby Międzynarodowej Prasy i Książki będą wyspecjalizowanymi punktami sprzedaży czasopism i publikacji informatycznych, oraz kaset z programami komputerowymi. W wytypowanych przez Zarząd Główny RSW klubach zawsze będzie można kupić **„IKS-a"**, **„Bajtka"** i **„Komputer"**. Oto adresy placówek, które w pierwszej kolejności zajmą się sprzedażą kaset i czasopism

1. **Bydgoszcz — Al. 1 Maja 10**
2. **Gdańsk — ul. Długi Targ 25/27**
3. **Katowice — ul. Wawelska 2**
4. **Gliwice — Rynek Główny**
5. **Kielce — ul. Rewolucji Październikowej 3/5**
6. **Koszalin — ul. Zwycięstwa 106/108**
7. **Kraków — Mały Rynek 4**
8. **Lublin — Krakowskie Przedmieście 20**
9. **Łódź — ul. Narutowicza 8/10**
10. **Olsztyn — ul. 1 Maja 18/19**
11. **Poznań — ul. Ratajczaka 39**
12. **Rzeszów — ul. Słowackiego 11**
13. **Szczecin — al. Wojska Polskiego 2**
14. **Warszawa — ul. Bagatela 14**
    ul. Marszałkowska 116/122
15. **Wrocław — plac Kościuszki 21/23**

Dodajmy, że ciągłą sprzedaż naszego miesięcznika prowadzi księgarnia Wydawnictwa MON — „Bellona" — ul. Grzybowska 77 Warszawa.

Już wkrótce kolportażem wydawnictw informatycznych zajmą się również KMPiK w Opolu, Zielonej Górze, Białymstoku, Toruniu, Tarnowie, Kaliszu, Pile, Świnoujściu. Mamy nadzieję, że każdy miłośnik informatyki znajdzie w nich coś dla siebie.

UWAGA! „IKS" w prenumeracie! Prenumeratę przyjmują oddziały RSW „Prasa-Książka-Ruch", a w miejscowościach, gdzie ich nie ma urzędy pocztowe i doręczyciele.



Klucz do transmisji

W komputerowym świecie liczą się głównie fakty. Informatyczne, matematyczne, elektroniczne... Od czasu do czasu wkradają się doń jednak także plotki i pogłoski. Jedna z takich wieści dotyczy resortu łączności. Niektórzy posiadacze mikrokomputerów twierdzą z nieukrywanym oburzeniem, iż łącznościowcy nie zezwalają na wykorzystywanie łączy telefonicznych do międzykomputerowej transmisji danych. By nie utrzymywać dłużej stanu niepewności, o wyjaśnienia poprosiliśmy inż. Grzegorza Wiśniewskiego — dyrektora Departamentu Służb Telekomunikacyjnych Ministerstwa Łączności.

● Panie Dyrektorze, czy to prawda, że telefony mają z komputerami „na pieńku"?

— Nic podobnego. Byłaby to przecież postawa przeciwna postępowi. Natomiast wykorzystywanie linii telefonicznej do pracy komputera rzeczywiście powoduje określone sankcje — gdy abonent nie zgłosił wcześniej takiego zamiaru. Z chwilą wykrycia takiego nadużycia sankcją natychmiastową jest odłączenie linii telefonicznej od sieci. Dalszą konsekwencją może być trwałe wymówienie usług takiemu abonentowi.

● Dlaczego tak się dzieje?

— Po pierwsze dlatego, że takie są przepisy międzynarodowe. Po drugie — przedsiębiorstwo Polska Poczta Telegraf i Telefon nie ma charakteru dobroczynnego. Musi na siebie zarabiać, a w związku z tym nie świadczy usług darmo.

● Co zatem należy uczynić, by przyłączyć mikrokomputer do linii telefonicznej zgodnie z przepisami?

— Częścią łączącą mikrokomputer z siecią jest modem. Można dołączyć go do sieci telefonicznej galwanicznie, albo poprzez tzw. sprzęgacz akustyczny. Należy zatem zaopatrzyć się w któryś z tych modemów. Sprzęgacz akustyczny ma tę istotną z uwagi na temat naszej rozmowy zaletę, że nie wymaga homologacji. Natomiast modem połączony galwanicznie może być eksploatowany jedynie po homologacji dokonanej przez Instytut Łączności.

● Gdy wymogi techniczne zostaną spełnione...

— Nasz klient musi zdeponować w resorcie dyskietkę lub taśmę zawierającą procedurę wymiany informacji. Po spełnieniu tego obowiązku abonenci uiszczać będą niewielką opłatę miesięczną. Natomiast opłaty za poszczególne połączenia dla indywidualnych posiadaczy mikrokomputerów obliczane są dokładnie tak samo jak za rozmowy telefoniczne. Jedynie wobec instytucji stawiane są w tej kwestii ostrzejsze wymagania.

● Ilu jest obecnie w kraju abonentów tego rodzaju?

— Dopiero kilkudziesięciu, głównie z dużych ośrodków miejskich. Niektórzy z nich realizowali już bardzo dalekie połączenia, nawet ze Stanami Zjednoczonymi.

● Czy wszystkie centrale telefoniczne naszego kraju są zdolne do realizacji połączeń komputerowych?

— Niestety nie. Mamy przecież jeszcze centrale ręczne i gdyby telefonistka włączyła się w transmisję danych z pytaniem „mówi się?" — wszystko trzeba by było zaczynać od nowa.

● Jakie są zatem tendencje rozwojowe?

— Dotyczą one stworzenia wydzielonej sieci transmisji danych, początkowo tylko dla instytucji, dostosowanej do współpracy i z siecią telefoniczną i z siecią teleksową. Przy okazji chciałbym powiedzieć o innej koncepcji — interesującej na pewno wszystkich telewidzów. Jej istotą jest sieć uzupełniająca dzisiejszy system łączy telefonicznych na takiej zasadzie, na jakiej w wielu już państwach za pomocą „czarnej skrzynki" sprzedawanej przez administrację łączności do linii telekomunikacyjnych przyłączać można odbiorniki telewizyjne.

● Kiedy resort przystąpi do realizacji tych, nowych zadań?

— Początek w przyszłym roku.

● Dziękuję bardzo za informacje.

Michał PRZYBYŁOWSKI



Rys. Michał Przybyłowski

Mikro w Japonii

Światową historię mikrokomputerów rozpoczął w 1971 roku mikroprocesor 4004. Rok potem INTEL produkuje 8008, by po kilkunastu miesiącach dostarczyć na rynek jeden z najpopularniejszych mikroprocesorów 8080. Znaczącym uzupełnieniem rozwoju mikro było pojawienie się w 1973 roku procesora Ziloga.

W tym samym czasie japońscy konstruktorzy pracowali nad własnymi technologiami i już wiosną 1972 roku firma **NEC** rozpoczyna sprzedaż mikroprocesorów NEC 700. Składały się one z dwóch układów i były ekwiwalentem procesora 4004.

Maj 1973 roku to data narodzin w firmie **TOSHIBA** 12-bitowego mikroprocesora TLCS-12. Miało to zapewne istotne znaczenie w przyspieszeniu prac NECa, skoro w październiku tego samego roku rozpoczęto produkcję serii μCOM: μCOM-4, μCOM-8, μCOM-16. Są to kolejno procesory 4-, 8- i 16-bitowe. W rezultacie okazało się, że mikrokomputery NECa i Fujitsu są dwa razy szybsze niż IBM PC.

Japońskie komputery najczęściej bazują na mikroprocesorach 8086-2 lub 80186, są wyposażone w pamięć RAM 0.5 MB, floppy dyski 1 MB każdy i pracują pod systemami MS DOS lub CP/M-86 plus język japoński, którego opracowanie nie było łatwe. A poza tym angielska terminologia informatyczna jest istotnym problemem. Niestety u nas nawet nie budzi zainteresowania, a czasami jedynie uśmiech politowania „fachowców" wobec prób wprowadzania rodzimej terminologii związanej z informatyką.

Stan rozwoju komputerów w Kraju Kwitnącej Wiśni odzwierciedla bardzo interesujący system SV — 2000 (Software Vending System). Jest to sieć z terminalami w postaci budek telefonicznych. Zamiast aparatu telefonicznego umieszczony tam terminal ma wbudowany ekran, klawisz start, dwie gałki kursora, klawisze akceptacji i kasowania oraz otwory na kasety, floppy dysk. Oczywiście nie zabrakło wlotu na monety, a także kartę kredytową. Całość połączona jest z głównym komputerem kablem telefonicznym.



Za pośrednictwem tego ogólnodostępnego terminalu można dokonywać transakcji finansowych, rejestrować i odczytywać dane, programy itp.

Po naciśnięciu klawisza start, na ekranie ukazuje się lista możliwych „usług" terminalu. Można na przykład po podaniu typu posiadanego w domu komputera (koniecznie firmy NEC), wybrać nową grę, zostanie ona oczywiście zaprezentowana, ale dopiero uiszczenie odpowiedniej opłaty jest warunkiem by została zapisana na dysku lub kasecie użytkownika. Czas wykonania wszystkich tych czynności (w budce — terminalu) nie przekracza 200 sekund. Koszt tego typu terminalu wynosi 8 — 12 tys. dolarów.

Przykładami obecnie produkowanych w Japonii mikrokomputerów są: IS-11 C, T 1100, SPIRIT PC.

IS-11 C z ekranem LCD o rozdzielczości 640 × 200, ma pamięci ROM 72 KB i RAM 80 KB, 70 klawiszy, wbudowany system mikrokaset 128 KB każda. IS-11 C ma również możliwość dołączenia termicznej drukarki, innej klawiatury, interfejsu do mikrofloppów, a także lokalnego komputera za pośrednictwem łączności telefonicznej.

T 1100 również jest wyposażony w ekran LCD, mikrofloppy, dysk i jest kompatybilny z IBM PC, akceptuje MS DOS/PC DOS.

SPIRIT PC znany w Wielkiej Brytanii jako OSBORNE jest w pełni kompatybilny z IBM PC. Posiada oprogramowanie INTEGRATED 7 o siedmiu integralnych aplikacjach, między innymi baza danych, grafika zawodowa, a ponadto słownik z 35 000 słów (zamiast słownika ortograficznego — oczywiście japońskiego). Innymi słowy komputer poza realizacją typowych zadań kontroluje poprawność pisanych słów języka potocznego, na przykład podczas aktualizacji zbiorów, których elementami są słowa.

Na zakończenie dodajmy, iż ostatnio Japonię zdominował system operacyjny UNIX, który podobnie jak CAD/CAM stał się standardem.

Poza niekwestionowanymi osiągnięciami w zakresie technologii, na szczególne podkreślenie zasługuje szacunek Japończyków do własnego języka. Szkoda, że tego nie można powiedzieć o wielu producentach sprzętu elektronicznego (w tym komputerów) w Polsce. Nawet większość podstawowych opisów sprzętu domowego wykonana jest w języku angielskim. Na przykład napis „kolor" na telewizorze Neptun — „COLOR"! Dlaczego?

Włodzimierz GOGOŁEK

WYLICZENIA WIELOPOZYCYJNE

Dziś prezentujemy program szczególny. Taki, który zwiększa możliwości Twojego komputera. Obecnie będziesz mógł wyliczyć wiele, bardzo wiele pozycji „po przecinku". Ile? 100, 500, a może więcej. Nasz program ma jednak ograniczone zastosowanie — wylicza on stałą „π", stałą „e", eksponentę oraz ułamek.

Zacznijmy od końca. Ile pozycji ma ułamek 1/7? Nieskończenie wiele. Tak, ale w jego „ogonie" jest prawidłowość: stale się powtarza człon „142857" i znów „142857" itd. Ciąg ten ma **SZEŚĆ** pozycji i nie może mieć więcej, gdyż przy podziale przez 7 „reszta" wynosi 1, lub 2, 3, 4, 5 lub 6. Większej reszty nie ma.

I oto mamy regułę: okres odwrotności liczby „M" może mieć NAJWYŻEJ „M-1" pozycji.

Oczywiście może ich być mniej. (Odwrotność M=10 wynosi 0,1, okres jest ZEROWY, a więc mniejszy od 10-1).

Bardzo „złośliwa" jest odwrotność M=21. Okres ma aż 20 pozycji, Dla M=49 pozycji jest 42.

Proponujemy zacząć od wyliczeń ułamka (etykieta G). Wprowadźmy... 100 pozycji dla licznika 12345 oraz mianownika 99999. Rezultat powinien być monotonny: stale „12345.12345". Jeśli jest inny — źle wprowadziłeś program.

Gdy usunąłeś błąd, zbadaj możliwości Twojego komputera. Wylicz 500, 1000, a może więcej pozycji. Wynik ma być zawsze ten sam: „12345.12345". Jeśli na końcowych pozycjach masz inny wynik, oznacza to, że osiągnąłeś GRANICE MOŻLIWOŚCI Twojego komputera. Warto je poznać.

Dalsze programy służą do wyliczeń wielopozycyjnych stałej „π", stałej „e" oraz eksponenty.

Wyliczanie stałej „π" jest długie: mój **SHARP PC-150** wylicza 100 pozycji w ciągu 26 minut i 18 sekund. Francuski programista Renaud de LA TAILLE (od którego pożyczyłem koncepcję tego programu) podaje, iż wyliczenie 1000 pozycji trwa 14 dni, zaś 1650 pozycji jego **SHARP PC 1211** liczy 38 dni!

Niewątpliwie istotne jest, czy te wyliczenia są **POPRAWNE?** Ja postąpiłem następująco: wyszperałem w tablicach matematycznych stałe "π" oraz „e" z dokładnością do 31 pozycji. Porównałem je z moimi wyliczeniami: **IDEALNA ZGODNOŚĆ!**

Jednak poprawność 31 pozycji wcale nie oznacza, iż reszta 100-pozycyjnego „ogona" jest dobra. I oto mam problem: czy „ogon" jest dobry?

Sam nie potrafię odpowiedzieć. Spróbujmy wspólnie: napisz do „IKS-a". Podaj wyniki wyliczeń i wyszperane dane.

Wracając do programu: zwróć uwagę na trzy podprogramy (etykiety A, B, C). Są one stosowane we wszystkich naszych programach. Podprogram „A" dotyczy wielopozycyjnego mnożenia, „B" — dzielenia, zaś „C" — wydania wyników. Teoria mówi, iż za ich pomocą (A,B,C) można wielopozycyjnie wyliczać funkcje trygonometryczne (a zapewne i inne). To adresuję do tych, którzy lubią wyższą matematykę (problem „szeregi potęgowe"). No i na koniec: PO CO TAKA DOKŁADNOŚĆ? Odpowiadam: w „codziennym" życiu jest zbyteczna. Ale już dziś kartografowie uważają, iż dziesięciopozycyjne wyliczanie jest zbyt mało dokładne („przydałby się dłuższy arytmometr"). Co zatem mają powiedzieć kosmonauci?!

Czy możesz sobie wyobrazić, jak precyzyjnie był wyliczony tor WEGI, która po latach wędrówki kosmicznej **TRAFIŁA** w warkocz komety Halley'a?!

STARY NAUCZYCIEL

```
50:"A":FOR K=WTO
   0STEP -1
60:G=INT X(K):D=X
   (K)-G
70:B=D*L+P/Q:C=(G
   *L+INT B)/Q
80:P=INT C:X(K)=(
   C-P)*Q+(B-INT
   B)
90:NEXT K:RETURN
100:"B":FOR K=0TO
    W
110:G=INT X(K):D=X
    (K)-G
120:A=G+P*Q:B=INT
    (A/M)
130:E=A-B*M:F=(D+E
    )*Q
140:C=INT (F/M):P=
    F-C*M
150:X(K)=B+(C/Q)
160:NEXT K:RETURN
170:"C":TEXT :
    USING :FOR K=0
    TO W
180:LPRINT K;"*";
    USING "######.
    #####";X(K)
190:USING :NEXT K
195:T1=T1-INT (T1/
    100)*100:T2=T2
    -INT (T2/100)*
    100
200:LPRINT "OD";T1
    ;" DO";T2:END
210:"D":CLEAR :REM
     "Stala π"
220:INPUT "π Pozyc
    ji ";Z
225:T1=TIME
230:N=INT (Z*(LN 1
    0)/(LN 2)):W=
    INT (Z/10)-1
240:IF W<1LET W=1
250:DIM X(W):Q=1E5
    :USING :WAIT 0
260:L=N:PRINT "PAR
    AMETR";N:GOSUB
    "A"
270:M=2*N+1:GOSUB
    "B"
280:X(0)=X(0)+2E4:
    P=0
290:N=N-1:IF N>0
    GOTO 260
295:T2=TIME
300:TEXT :LPRINT "
    Stala π";Z;" p
    ozycji"
310:GOTO "C"
320:"F":CLEAR :REM
     "Stala e"
330:INPUT "Stala e
     Pozycji ";Z
335:T1=TIME
```

dokonczenie na str. 6

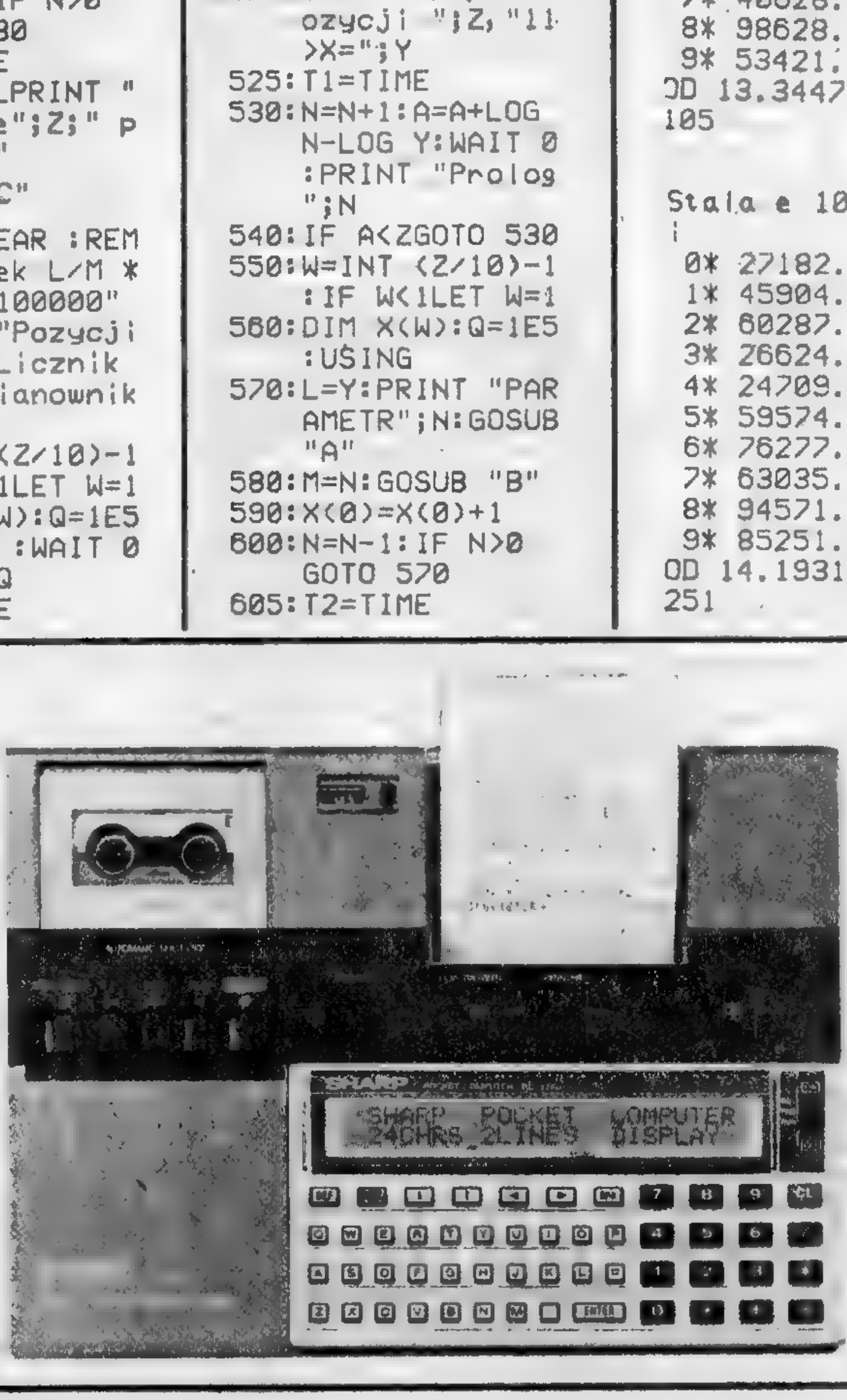
dokończenie ze str. 5

```
340:N=N+1:A=A+LOG
    N:WAIT 0:PRINT
    "Prolog";N
350:IF A<ZGOTO 340
360:W=INT (Z/10)-1
    :IF W<1LET W=1
370:DIM X(W):Q=1E5
    :USING
380:M=N:PRINT "PAR
    AMETR";N:GOSUB
    "B"
390:X(0)=X(0)+1E4:
    P=0
400:N=N-1:IF N>0
    GOTO 380
405:T2=TIME
410:TEXT :LPRINT "
    Stala e";Z;" p
    ozycji"
420:GOTO "C"
430:"G":CLEAR :REM
     "Ulamek L/M *
    0<L<M<100000"
440:INPUT "Pozycji
    ";Z,"Licznik
    ";L,"Mianownik
    ";M
450:W=INT (Z/10)-1
    :IF W<1LET W=1
460:DIM X(W):Q=1E5
    :USING :WAIT 0
    :X(0)=Q
465:T1=TIME
470:PRINT "Licznik
    ";L:GOSUB "A"
480:PRINT "Mianown
    ik";M:GOSUB "B
    "
485:T2=TIME
490:TEXT :LPRINT "
    Ulamek";L;" /"
    ;M;" Poz.";Z
500:GOTO "C"

510:"H":CLEAR :REM
     "Exp X *0<X<1
    1*"
520:INPUT "Exp X P
    ozycji ";Z,"11
    >X=";Y
525:T1=TIME
530:N=N+1:A=A+LOG
    N-LOG Y:WAIT 0
    :PRINT "Prolog
    ";N
540:IF A<ZGOTO 530
550:W=INT (Z/10)-1
    :IF W<1LET W=1
560:DIM X(W):Q=1E5
    :USING
570:L=Y:PRINT "PAR
    AMETR";N:GOSUB
    "A"
580:M=N:GOSUB "B"
590:X(0)=X(0)+1
600:N=N-1:IF N>0
    GOTO 570
605:T2=TIME
610:TEXT :LPRINT "
    Exp";Y;" Poz."
    ;Z
620:GOTO "C"
```

```
Stala π 100 pozycj
i
 0* 31415.92653
 1* 58979.32384
 2* 62643.38327
 3* 95028.84197
 4* 16939.93751
 5*~*5820.97494
 6* 45923.07816
 7* 40628.62089
 8* 98628.03482
 9* 53421.17067
OD 13.3447 DO 14.0
105

Stala e 100 pozycj
i
 0* 27182.81( '8
 1* 45904.52353
 2* 60287.47135
 3* 26624.97757
 4* 24709.36999
 5* 59574.96696
 6* 76277.24076
 7* 63035.35475
 8* 94571.38217
 9* 85251.66427
OD 14.1931-DO 14.2
251
```



PROGRAM 20

MERITUM

```
1 ' RUCH JEDNOSTAJNIE PRZYSPIESZONY
4 DIM V(10):DIM S(10)
5 CLS
6 FOR X=1 TO 7:PRINT:NEXT X
10 PRINT CHR$(23);"RUCH JEDNOSTAJNIE PRZYSPIESZONY"
11 PRINT@846,"NACISNIJ SPACJE"
13 IF INKEY$=" " THEN GOTO 19
14 GOTO 13
19 CLS
20 INPUT"PODAJ PRZYSPIESZENIE 'A'";A
25 CLS
26 PRINT TAB(12)"PRZYSPIESZENIE 'A' WYNOSI ";A
28 PRINT" T="," V="," S="," DS="
29 PRINT
30 FOR T=1 TO 10
40 V(T)=A*T:S(T)=(A*T*T)/2:DS=S(T)-(A*(T-1)*(T-1))/2
50 PRINT T,V(T),S(T),DS
60 NEXT T
65 PRINT@902,"W CELU KONTYNUOWANIA PROGRAMU NACISKAJ SPACJE "
70 IF INKEY$=" " GOTO 100
75 GOTO 70
100 CLS:PRINT@0,"V":GOSUB 300
110 FOR T=1 TO 10
111 ON ERROR GOTO 400
115 SET(T*9,44-V(T)/10)
120 NEXT T
130 IF INKEY$=" " THEN GOTO 200
140 GOTO 130
200 CLS:PRINT@0,"S":GOSUB 300
210 FOR T=1 TO 10
211 ON ERROR GOTO 510
215 SET(T*9,44-S(T)/10)
220 NEXT T
230 IF INKEY$=" " THEN RUN
240 GOTO 230
300 FOR X=0 TO 41
310 SET(X,44):SET(0,44-X)
320 NEXT X
330 FOR X=42 TO 100
340 SET(X,44)
350 NEXT X
355 PRINT@945,"T"
360 RETURN
400 IF INKEY$=" " THEN RESUME 200
410 GOTO 400
510 IF INKEY$=" " THEN RESUME 530
520 GOTO 510
530 RUN
```

Głos mają Czytelnicy:

MERITUM POTRAFI RÓWNIEŻ RYSOWAĆ

Do naszych szkół dociera coraz więcej różnych mikrokomputerów. By dobrze wykorzystać ich wszechstronne możliwości, potrzebne są nauczycielom programy dydaktyczne, opracowane szczegółowo. Wśród zainteresowanych mikrokomputerami panuje mylna opinia, że nasze rodzime MERITUM produkowane przez MERA-ELZAB nie ma możliwości graficznych.

Chciałbym zgodnie z przysłowiem „cudze chwalicie, swego nie znacie" wykazać, że na MERITUM można na lekcjach demonstrować różne interesujące rzeczy, natrafiając rzecz jasna w trakcie programowania na pewne trudności (np. nie można rysować małych okręgów, potrzebnych w schematach elektrycznych do przedstawienia przyrządów pomiarowych).

Podstawowym znakiem graficznym w MERITUM, niestety, nie jest punkt, lecz mały prostokąt. Dlatego różna ilość komórek semigraficznych w pionie i poziomie jest powodem, że chcąc na przykład narysować okrąg należy rzędne i odcięte pomnożyć przez różne współczynniki wynikające z organizacji ekranu. Można to opanować metodą wielu prób. Na zamieszczonych moich programach można prześledzić, jak ten problem rozwiązałem.

Jan GOLLA

PROGRAM 21

ATARI

Jeszcze do niedawna uzyskanie wykresu funkcji y = f (x), na podstawie jej zapisu analitycznego, było czynnością bardzo żmudną. Wymagało stablicowania funkcji, to znaczy obliczenia wartości y dla wartości x, wybieranych zwykle ze stałym krokiem, w zadanym przedziale, a następnie cierpliwego przenoszenia wyznaczonych punktów na wykres (najczęściej na papierze milimetrowym). Tablicowanie funkcji możliwe jest za pomocą każdego komputera, lecz uzyskanie jej wykresu wymaga graficznego urządzenia wyjściowego. Do niedawna takie urządzenia dostępne były tylko w nielicznych ośrodkach obliczeniowych. Dopiero pojawienie się mikrokomputerów, wykorzystujących monitor lub zwykły odbiornik telewizyjny jako urządzenie wyjściowe, umożliwiło uzyskanie bezpośrednio wykresu funkcji. Powstało wiele odpowiednich programów różniących się typem używanego komputera oraz sposobem dialogu z użytkownikiem. Przykładem może być program dla ZX Spectrum zamieszczony w 1 numerze „IKS-a".

Program tworzący wykres funkcji musi uwzględniać specyfikę komputera, w tym przypadku ATARI 800 XL. Jego wersja języka BASIC posiada instrukcję VAL o znacznie mniejszych możliwościach niż Spectrum. W rezultacie kreślona funkcja nie może być podana jako ciąg znaków. Można ją podać w postaci linii programu (przed uruchomieniem lub po zatrzymaniu), ale w takim przypadku niemożliwe jest zrobienie opisu wykresu. W książce „ATARI BASIC" wydanej przez klub użytkowników ATARI z Krakowa zamieszczony jest przykład programu dającego na ekranie wykresy funkcji. Wymaga on jednak skomplikowanej obsługi magnetofonu.

W przedstawionym programie przyjęto inne rozwiązanie, wykorzystujące właściwości ATARI oraz jego edytora ekranowego. Komputer ten umożliwia użycie ekranu telewizora jako urządzenia zewnętrznego działającego w trybie zapisu i odczytu. Linia programu, zawierająca funkcję do wykreślenia zapisywana jest na ekranie. Program zatrzymuje się i użytkownik ma możliwość zmiany funkcji, za pomocą edytora, w taki sam sposób, jak dowolnego wiersza programu. Po podaniu instrukcji CONT realizacja programu będzie kontynuowana. Zmieniona funkcja wprowadzana jest z ekranu jako ciąg znaków. Dzięki temu możliwe jest wykonanie opisu wykresu, co również wymaga dodatkowych zabiegów.

Komputery ZX Spectrum, Amstrad-Schneider CPC 464 lub 6128 pozwalają na wykonanie opisu rysunku lub osi układu współrzędnych w sposób bardzo prosty, dzięki możliwości jednoczesnej pracy w trybie graficznym i tekstowym. Komputery ATARI 600/800 XL mają odrębne tryby graficzne i tekstowe. Nie można, bez dodatkowych operacji, uzyskać na przykład opisu osi układu. W tym celu opracowano specjalny podprogram (linie 2000—2090), który umożliwia zapis dowolnego znaku, lub ciągu znaków, w wybranym miejscu ekranu. Pobiera on kolejne bajty, tworzące obraz znaku, z obszaru pamięci ROM, i przesyła je do odpowiednich komórek pamięci ekranu. Można go wykorzystać również w innych programach. **Należy tylko pamiętać, żeby w zmiennej PX podać numer kolumny, w której ma być umieszczony pierwszy znak ciągu (0—39), w zmiennej PY — pionowe położenie tekstu (0—183), a w zmiennej VS (która musi być wcześniej zadeklarowana) — ciąg znaków, który ma być wydrukowany na ekranie. Trzeba również pamiętać, aby nie używać w innym celu zmiennych: PX, PY, VS, DL, D1, D2 i D3.**

Pozostałe podprogramy mają znaczenie kosmetyczne: określenie dziedziny funkcji (linie 300—325), wyznaczenie przedziału zmienności funkcji (linie 330—370), określenie położenia osi układu współrzędnych i współczynnika skali (linie 500—550). Krok obliczeń uzależnia się od maksymalnej ilości punktów graficznych (pixeli) przeznaczonych na wykres oraz granic zmienności (linie 510—520). Po wykonaniu niezbędnych obliczeń samo kreślenie funkcji (linie 900—950) jest prostą operacją, jeśli tylko pamięta się, że punkt o współrzędnych (0,0) leży w lewym, górnym rogu ekranu. Jeżeli wykres jest zbyt krótko na ekranie, to należy poprawić linię 100, wstawiając większą liczbę w miejsce 5000.

L. PIELA

WYKRES FUNKCJI

```
10 DIM O$(1),FUN$(100),V$(40)
20 FUN$="WYKRES FUNKCJI: "
22 ? "}"
25 GOSUB 1400:REM STRONA TYTULOWA
30 GOSUB 1000 REM NOWA FUNKCJA
35 GOSUB 1900 REM WSTAWIENIE FUNKCJI
40 GOSUB 300:REM OKRESLENIE ZAKRESOW
50 GOSUB 500:REM WYZNACZENIE WSP. SKALI
55 GOSUB 1500:REM WYDRUK TEKSTU
60 GOSUB 600 REM KRESLENIE OSI
70 GOSUB 700 REM OPIS OSI X
80 GOSUB 800 REM OPIS OSI Y
90 GOSUB 900 REM KRESLENIE FUNKCJI
100 FOR I=1 TO 5000 NEXT I
110 GRAPHICS 0 GOTO 30
300 REM OKRESLENIE ZAKRESOW XZ I YZ
310 ? "}" POSITION 0,2
315 ? "PODAJ DZIEDZINE FUNKCJI."
320 POSITION 0,4 ? "Xmin=", INPUT XP
325 POSITION 24,4 ? "Xmax=", INPUT XK
330 POSITION 1,7 ? "CZY ZNASZ PRZEDZIAL ZMIENNOSCI Y? (T/N)"
335 INPUT O$ IF O$(1,1)="T" THEN GOTO 360
340 IF O$(1,1)="N" THEN GOTO 380
350 GOTO 335
360 POSITION 0,10:? "Ymin=", INPUT YD
365 POSITION 24,10:? "Ymax=", INPUT YG
370 RETURN
380 POSITION 0,11:? "PROSZE POCZEKAC CHWILE."
385 POSITION 6,14:? "MUSZE WYZNACZYC Ymin I Ymax."
390 GOSUB 400
392 POSITION 0,16:? "Ymin=";YD:POSITION 0,18 ? "Ymax=";YG
394 FOR I=1 TO 2000:NEXT I
395 RETURN
400 REM WYZNACZENIE YD I YG
410 DX=(XK-XP)/240
420 NM=1E+90:NM=-NM
430 FOR X=XP TO XK STEP DX
440 GOSUB 5000
450 IF Y>NM THEN NM=Y
460 IF Y<NN THEN NN=Y
470 NEXT X
480 YD=NN YG=NM
490 RETURN
500 REM WYZNACZENIE WSP. SKALI
510 XZ=XK-XP:YZ=YG-YD
520 DX=XZ/239 DY=YZ/159
530 CY=INT(ABS(YD*(YD<0))/DY):CY=160-C
540 CX=INT(ABS(XP*(XP<0))/DX)+40
550 RETURN
600 REM KRESLENIE OSI
605 REM GRAPHICS 24 COL.1
610 PLOT 40,CY DRAWTO 280,CY
620 PLOT CX,0 DRAWTO CX,160
625 COLOR 0 PLOT 0,0 COLOR 1
630 YSK=CY+1
640 XSK=INT(CX/8)-4
650 RETURN
700 REM OPIS OSI X
710 FOR N=0 TO 5
720 SX=(INT(10*(XP+N*XZ/5)+0.5))/10
730 PX=6*N+4 PY=YSK+1:V$=STR$(SX):GOSUB 2000
740 PLOT (6*N+5)*8,CY+2:DRAWTO (6*N+5)*8,CY
750 NEXT N
760 RETURN
800 REM OPIS OSI Y
810 FOR N=0 TO 5
820 SY=(INT(10*(YG-N*YZ/5)+0.5))/10
830 PX=XSK PY=N*32+1:V$=STR$(SY):GOSUB 2000
840 PLOT CX+1,N*32
850 NEXT N
860 RETURN
900 REM KRESLENIE FUNKCJI
905 SETCOLOR 2,1,1 SETCOLOR 4,1,1:SETCOLOR 1,12,6
910 FOR N=0 TO 239
920 X=XP+N*DX GOSUB 5000
930 Y=(Y-YD)/DY:PLOT N+40,160-Y
940 NEXT N
950 RETURN
1400 REM STRONA TYTULOWA
1410 GRAPHICS 2+32
1420 SETCOLOR 0,2,4 SETCOLOR 1,0,12:SETCOLOR 2,4,2:SETCOLOR 3,2,14:SETCOLOR 4,11
,0
1430 POSITION 3,3
1440 ? #6,"WYKRES FUNKCJI"
1450 POSITION 6,6
1460 ? #6,"y=f(x)"
1465 ? " ? "START      POCZATEK"
1470 IF PEEK(53279)=6 THEN RETURN
1480 GOTO 1470
1500 REM WARUNKI
1510 GRAPHICS 24 COLOR 1
1520 PX=1 PY=175 V$=FUN$:GOSUB 2000
1550 RETURN
1000 REM ZMIANA FUNKCJI
1005 CLOSE #1:OPEN #1,4 0,"S:"
1010 POSITION 2,0 LIST 5000
1020 POSITION 1,6
1030 ? "WSTAW NOWA FUNKCJE PRZY POMOCY EDYTORA."
1050 POSITION 0,0 ? "NACISNIJ     RETURN"
1060 POSITION 0,11 ? "WPISZ CONT I RETURN"
1070 STOP
1080 RETURN
1900 REM WCZYTYWANIE FUNKCJI JAKO CIAGU
1910 POSITION 7,1
1920 FOR I=1 TO 30 GET #1,Z
1930 FUN$(I+16,I+16)=CHR$(Z):NEXT I
1940 RETURN
2000 REM PISANIE ZNAKOW
2005 REM GRAPHICS 56 COLOR 1
2010 DL=PEEK(560)+PEEK(561)*256
2020 D1=PEEK(DL+4)+PEEK(DL+5)*256
2030 FOR U=1 TO LEN(V$)
2040 D2=57344+((ASC(V$(U,U))-32)*8)
2050 D3=D1+PY*40+PX+U-1
2060 FOR Z=0 TO 7
2070 POKE D3+Z*40,PEEK(D2+Z)
2080 NEXT Z NEXT U
2090 RETURN
5000 Y=SIN(X)+COS(X)
5010 RETURN
```



„Dialog" z komputerami.

PROGRAM 22

Program OKNO tworzy na ekranie pseudo-okienka, a mówiąc ściślej wypełnia zadane obszary proponowanymi kolorami. Napisany jest w języku wewnętrznym Z80 pod owsemblerem GENS1M21. Istotną sprawą w tym dość prostym programie jest sposób przekazywania parametrów. Zamiast zwykle używanej metody wykorzystującej komendę POKE przed uruchomieniem kodu zastosowano tu funkcję użytkownika o 5 parametrach. Kolejno: l oznacza numer linii lewego górnego pola okna, k — numer kolumny, s — szerokość okna, h — wysokość okna, zaś a — atrybut, który wypełni okno wg wzoru:
a = ink + 8 * paper + 64 * bright + 128 * flash
Parametry te wyszukuje sobie procedura, sprawdza ich poprawność i wypełnia okno kolorem. Są one umieszczone wewnątrz instrukcji IF... THEN..., która, jak widać, jest tylko atrapą. Wywoływanie kodu maszynowego w ten sposób wydaje się być najbardziej korzystne. Użycie np. RANDOMIZE powoduje narzucenie stałego ciągu liczb pseudolosowych przy korzystaniu z funkcji RND.

Program OKNO zbudowany jest następująco

linia 40 — rejestr IX wskazuje pierwszy parametr (l) w obliczanej funkcji w ( ) poprzez odczyt zmiennej systemowej DEFADD.
linie 50...60 — rejestr HL odkładany jest na stos z zawartością adresu procedury obsługi błędu przy błędnym parametrze;
linie 70...250 — sprawdzanie zasadności parametrów: eliminacja liczb ujemnych, ułamkowych i zer;
linia 260 — rejestr IX ponownie wskazuje parametr l;
linie 270...290 — eliminacja linii powyżej numeru 23;
linia 300 — rejestr HL podaje adres ekranu o numerze linii l;
linie 310...330 — eliminacja kolumny powyżej numeru 31;
linie 360...420 — przejście na adresy ekranu atrybutów;
linie 430...440 — rejestr C zawiera parametr a, zaś rejestr B wysokość okna;
linie 450...610 — wypełnienie okna atrybutem;
linie 500...520 — sprawdzenie czy okno nie wykracza poza ekran z prawej strony;
linie 580...600 — sprawdzenie czy okno nie wykra-

```
10              ORG   40000
20  DEFADD      EQU   23563
30  ERROR       EQU   10379
40              LD    IX,(DEFADD)
50              LD    HL,ERROR
60              PUSH  HL
70              LD    B,5
80  PARAM       LD    A,(IX+2)
90              AND   A
100             RET   NZ
110             LD    A,(IX+3)
120             AND   A
130             RET   NZ
140             LD    A,(IX+4)
150             AND   A
160             RET   Z
170             LD    A,(IX+5)
180             AND   A
190             RET   NZ
200             LD    A,(IX+6)
210             AND   A
220             RET   NZ
230             LD    DE,8
240             ADD   IX,DE
250             DJNZ  PARAM
260             LD    IX,(DEFADD)
270             LD    A,(IX+4)
280             CP    24
290             RET   NC
300             CALL  3742
310             LD    A,(IX+12)
320             CP    32
330             RET   NC
340             ADD   A,L
350             LD    L,A
360             LD    A,H
370             RRCA
380             RRCA
390             RRCA
400             AND   3
410             OR    #58
420             LD    H,A
430             LD    C,(IX+36)
440             LD    B,(IX+28)
450 WYS         PUSH  BC
460             PUSH  HL
470             LD    B,(IX+20)
480 SZER        LD    (HL),C
490             INC   HL
500             LD    A,L
510             AND   31
520             JR    Z,TAK
530             DJNZ  SZER
540 TAK         POP   HL
550             POP   BC
560             LD    DE,32
570             ADD   HL,DE
580             LD    A,91
590             CP    H
600             JR    Z,KONIEC
610             DJNZ  WYS
620 KONIEC      INC   SP
630             INC   SP
640             RET
```

Przykład wykorzystania programu OKNO

```
10 DEF FN w(l,k,s,h,a)= USR 6e4
20 IF FN w(5,1,30,10,39) THEN
```

PROGRAM 23

COMMODORE

Program DISASSEMBLER napisany na mikrokomputer C-64 należy do programów tłumaczących instrukcje zapisane w kodzie wewnętrznym mikrokomputera (zawarte w pamięci) na instrukcje języka symbolicznego.

DISASSEMBLER wyprowadza na monitor następujące informacje:

- adres (nr bajtu pamięci) tłumaczonej instrukcji;
- zawartość kolejnych bajtów pamięci reprezentujących tę instrukcję;
- zapis instrukcji w języku symbolicznym zgodnie z listą rozkazów i trybami adresacji mikroprocesora 6502 (6510).

Wszystkie informacje są wprowadzane i wyprowadzane w systemie dziesiętnym. Obszar pamięci przeznaczony do tłumaczenia określa się podając adres startowy i adres końcowy.

```
50 DIMSMB$(150),KOD(150,1):BAJT(2)
60 FORI=0TO150:READSMB$(I):READL$:GOSUB1
300:KOD(I,0)=L:READKOD(I,1):NEXT
70 INPUT"  ADRES STARTOWY";ADS
72 IF ADS>65535THENEND
75 INPUT"  ADRES KONCOWY";MET
77 WER=1:PRINT"  ":PRINT" ADRES   DZIESIE
TNIE      KOD MASZYNOWY"
78 PRINT"________________________________"
80 RZ=PEEK(ADS):I=0
100 IFRZ=KOD(I,0)THEN140
110 IFI<>150THENI=I+1:GOTO100
120 PRINTADS;TAB(8)RZ;"  **BLEDNY KOD RO
ZKAZU**"
130 GOTO70
140 ON KOD(I,1) GOSUB 500,510,520,530,54
0,550,560,570,580,590,600,610,620
150 ADS=ADS+1+B:WER=WER+1
160 IFADS>METTHEN70
170 IFWER<20THEN80
175 PRINT"________________________________"
180 PRINT" JESLI DALEJ PRZYCISNIJ JAKIS
KLAWISZ":Z$=""
190 GETZ$:IFZ$=""THEN190
200 GOTO77
500 B=1:GOSUB1400:PRINT" "OP:PRINT" "TAB
(29)"#":RETURN
510 B=1:GOSUB1400:PRINTOP:RETURN
520 B=1:GOSUB1400:PRINTOP",X":RETURN
530 B=1:GOSUB1400:PRINTOP",Y":RETURN
540 B=2:GOSUB1400:PRINTOP:RETURN
550 B=2:GOSUB1400:PRINTOP",X":RETURN
560 B=2:GOSUB1400:PRINTOP",Y":RETURN
570 B=1:GOSUB1400:PRINT" "OP",X)":PRINT
" "TAB(29)"(":RETURN
580 B=1:GOSUB1400:PRINT" "OP"),Y":PRINT
" "TAB(29)"(":RETURN
590 B=0:GOSUB1400:PRINT" A":RETURN
600 B=1:GOSUB1400:PRINTOP:RETURN
610 B=0:GOSUB1400:PRINT:RETURN
620 B=2:GOSUB1400:PRINT" "OP")":PRINT" "
TAB(29)"(":RETURN
1000 DATAADC,69,1,ADC,65,2,ADC,75,3,ADC,
6D,5,ADC,7D,6,ADC,79,7,ADC,61,8
1010 DATAADC,71,9,AND,29,1,AND,25,2,AND,
35,3,AND,2D,5,AND,3D,6,AND,39,7
1020 DATAAND,21,8,AND,31,9,ASL,0A,10,ASL
,06,2,ASL,16,3,ASL,0E,5,ASL,1E,6
1025 DATABCC,90,11,BCS,B0,11,BEQ,F0,11,B
IT,24,2,BIT,2C,5,BMI,30,11,BNE,D0,11
1030 DATABPL,10,11,BRK,00,12,BVC,50,11,B
VS,70,11,CLC,18,12,CLD,D8,12,CLI,58,12
1040 DATACLV,B8,12,CMP,C9,1,CMP,C5,2,CMP
,D5,3,CMP,CD,5,CMP,DD,6,CMP,D9,7
1050 DATACMP,C1,8,CMP,D1,9,CPX,E0,1,CPX,
E4,2,CPX,EC,5,CPY,C0,1,CPY,C4,2
1060 DATACPY,CC,5,DEC,C6,2,DEC,D6,3,DEC,
CE,5,DEC,DE,6,DEX,CA,12,DEY,88,12
1070 DATAEOR,49,1,EOR,45,2,EOR,55,3,EOR,
4D,5,EOR,5D,6,EOR,59,7,EOR,41,8
1080 DATAEOR,51,9,INC,E6,2,INC,F6,3,INC,
EE,5,INC,FE,6,INX,E8,12,INY,C8,12
1090 DATAJMP,4C,5,JMP,6C,13,JSR,20,5,LDA
,A9,1,LDA,A5,2,LDA,B5,3,LDA,AD,5
1100 DATALDA,BD,6,LDA,B9,7,LDA,A1,8,LDA,
B1,9,LDX,A2,1,LDX,A6,2,LDX,B6,4
1110 DATALDX,AE,5,LDX,BE,7,LDY,A0,1,LDY,
A4,2,LDY,B4,3,LDY,AC,5,LDY,BC,6
1120 DATALSR,4A,10,LSR,46,2,LSR,56,3,LSR
,4E,5,LSR,5E,6,NOP,EA,12,ORA,09,1
1130 DATAORA,05,2,ORA,15,3,ORA,0D,5,ORA,
1D,6,ORA,19,7,ORA,01,8,ORA,11,9
1140 DATAPHA,48,1,PHP,08,12,PLA,68,12,PL
P,28,12,ROL,2A,10,ROL,26,2
1150 DATAROL,36,3,ROL,2E,5,ROL,3E,6,ROR,
6A,10,ROR,66,2,ROR,76,3,ROR,6E,5
1160 DATAROR,7E,6,RTI,40,12,RTS,60,12,SB
C,E9,1,SBC,E5,2,SBC,F5,3,SBC,ED,5
1170 DATASBC,FD,6,SBC,F9,7,SBC,E1,8,SBC,
F1,9,SEC,38,12,SED,F8,12,SEI,78,12
1180 DATASTA,85,2,STA,95,3,STA,8D,5,STA,
9D,6,STA,99,7,STA,81,8,STA,91,9
1190 DATASTX,86,2,STX,96,4,STX,8E,5,STY,
84,2,STY,94,3,STY,8C,5,TAX,AA,12
1200 DATATAY,A8,12,TSX,BA,12,TXA,8A,12,T
XS,9A,12,TYA,98,12
1300 C$=LEFT$(L$,1):GOSUB1320:L=C*16
1310 C$=RIGHT$(L$,1):GOSUB1320:L=L+C:RET
URN
1320 IFC$="A"THENC=10:RETURN
1320 IFC$="A"THENC=10:RETURN
1330 IFC$="B"THENC=11:RETURN
1340 IFC$="C"THENC=12:RETURN
1350 IFC$="D"THENC=13:RETURN
1360 IFC$="E"THENC=14:RETURN
1370 IFC$="F"THENC=15:RETURN
1380 C=VAL(C$):RETURN
1400 PRINTADSTAB(8)RZ:OP=0:Q=1:KL=12
1410 IFQ>BTHEN1430
1415 OP=OP+PEEK(ADS+Q)*(256↑(Q-1))
1420 PRINT" "TAB(KL)PEEK(ADS+Q):Q=Q+1:KL
=16:GOTO1410
1430 PRINT" "TAB(25)SMB$(I);:RETURN
```

```
ADRES   DZIESIETNIE        KOD MASZYNOWY
________________________________________
65017   133 182            STA 182
65019   134 187            STX 187
65021   132 186            STY 186
65023   96                 RTS
65024   133 184            STA 184
65026   134 186            STX 186
65028   132 185            STY 185
65030   96                 RTS
65031   165 186            LDA 186
65033   201 2              CMP #2
65035   208 13             BNE 13
65037   173 151 2          LDA 663
65040   72  169            PHA #169
65042   0                  BRK
65043   141 151 2          STA 663
65046   104                PLA
65047   96                 RTS
65048   133 157            STA 157
65050   165 144            LDA 144
________________________________________
JESLI DALEJ PRZYCISNIJ JAKIS KLAWISZ
```

PROGRAM 24

SPECTRUM

Program REGRESJA LINIOWA przeprowadza statystyczną analizę regresji i korelacji.

Mamy zadane dwa zbiory danych x_i i y_i ($i = 1, 2 \ldots n$). Z wykresu tych danych wnioskujemy, że istnieje między nimi korelacja liniowa. Wobec tego należy obliczyć współczynniki a i b najlepiej dobranej prostej $y = ax + b$ w sensie minimum odchyleń punktów od tej prostej. Wyznaczanie prostej uzupełniamy badaniem jej istotności.

Dla danych wartości x_i i wartości y_i ($i = 1, 2 \ldots n$), gdzie n — liczba pomiarów obliczane są:

1. **Oceny parametrów A i B liniowej funkcji regresji,**
2. **Błędy standardowe oceny parametrów liniowej funkcji regresji (S(a), S(b)),**
3. **Wariancje i odchylenia standardowe składnika resztowego (S(z), S(z)),**
4. **Współczynnik korelacji liniowej (R(x,y)),**
5. **Wartości średnie: X i Y,**
6. **Wartości teoretyczne y_i ($i = 1, 2 \ldots n$), (z zaokrągleniem do 5 miejsc po przecinku).**

W programie zastosowano instrukcję PAUSE 1000, która zatrzymuje działanie programu umożliwiając tym samym spisanie z ekranu pierwszej części obliczeń.

```
   5 REM regresja liniowa
   8 CLS
  10 PRINT "podaj liczbe pomiaro
w"
  20 PRINT "liczba pomiarow N=";
  30 INPUT N
  40 PRINT N
  50 DIM X(N)
  55 DIM Y(N)
  60 PRINT "wprowadz X  i   Y"
  70 FOR I=1 TO N
  80 PRINT I;
  90 INPUT X(I),Y(I)
 100 PRINT TAB 4;X(I),Y(I)
 110 NEXT I
 120 PRINT AT 20,0;"Dane poprawn
e?T/N"
 130 INPUT Z$
 140 IF Z$="T" THEN GO TO 260
 150 IF Z$<>"N" THEN GO TO 120
 160 PRINT AT 20,0;"podaj nr X i
Y      "
 170 INPUT I
 180 PRINT AT 21,0;"podaj X i Y"
 190 INPUT X(I)
 200 INPUT Y(I)
 205 PRINT TAB 4;X(I),Y(I)
 210 GO TO 120
 260 LET S1=0
 270 LET S2=0
 280 LET S3=0
 290 LET S4=0
 295 LET S5=0
 300 FOR I=1 TO N
 310 LET S1=S1+X(I)
 320 LET S2=S2+Y(I)
 330 LET S3=S3+X(I)*X(I)
 340 LET S4=S4+X(I)*Y(I)
 345 LET S5=S5+Y(I)*Y(I)
 350 NEXT I
 360 LET R=N*S3-S1*S1
 370 LET A=(N*S4-S1*S2)/R
 380 LET B=(S2-A*S1)/N
 390 LET SZ=0
 400 FOR I=1 TO N
 410 LET ROB=Y(I)-A*X(I)-B
 420 LET SZ=SZ+ROB*ROB
 430 NEXT I
 440 LET SZ=SZ/(N-2)
 450 LET SA=SQR (SZ*N/R)
 460 LET SB=SQR (SZ*S3/R)
 470 LET RXY=(S4-S1*S2/N)/SQR ((
S3-S1*S1/N)*(S5-S2*S2/N))
 475 PRINT
 480 PRINT "ROWNANIE REGRESJI Y=
A*X+B"
 490 PRINT '"OCENY PARAMETROW :"
 500 PRINT "A=";A'"B=";B
 510 PRINT
 520 PRINT "BLEDY STANDARDOWE OC
ENY PARAM."
 530 PRINT "S(A)=";SA'"S(B)=";SB
 540 PRINT '"WARIANCJA I ODCHYLE
NIE STAN-    DARDOWE SKLADNIKA R
ESZTOWEGO:"
 550 PRINT "S↑2(Z)=";SZ'"S(Z)   =
";SQR (SZ)
 560 PRINT '"WSPOLCZYNNIK KORELA
CJI LINIOWEJ:"
 570 PRINT "R(X,Y)=";RXY
 580 PRINT '"WARTOSCI SREDNIE:"
 590 PRINT "XSR=";S1/N'"YSR=";S2
/N
 595 PAUSE 1000
 600 PRINT
 605 LET K=100000
 610 PRINT "    X         Y(EMP)
  Y(TEORT)"
 620 FOR I=1 TO N
 630 LET RB=A*X(I)+B
 640 LET RB=K*RB+0.5
 650 LET RB=INT (RB)/K
 660 PRINT X(I);TAB 8;Y(I);TAB 2
0;RB
 670 NEXT I
 675 PRINT
 680 PRINT AT 21,0;"liczymy jesz
cze ?T/N"
 700 INPUT Z$
 710 IF Z$="T" THEN GO TO 8
 720 IF Z$<>"N" THEN GO TO 700
 730 PRINT "KONIEC"
 740 STOP
```

KOMPUTER W DYDAKTYCE

Początkowo koszt produkcji pojedynczego mikroprocesora przekraczał sumę stu dolarów. Obecnie kształtuje się w granicach jednego. To tylko jedna, chociaż nie najważniejsza przyczyna, z powodu której mikrokomputer stał się tani i zarazem ogólnodostępny.

Personal Computer

GWAŁTOWNY rozwój techniki, zastosowanie nowych materiałów i technologii, spowodowały uzyskanie niewyobrażalnych wprost możliwości w miniaturyzacji elektronicznego sprzętu. Na Zachodzie przedsiębiorstwa, które jeszcze do niedawna inwestowały grube miliardy w komputery-szafy, sprzedają je obecnie za symboliczne centy. W wyspecjalizowanych firmach, za pomocą młotka i sprawnych rąk robotników, odzyskuje się z nich tylko metale szlachetne (głównie złoto), a reszta wędruje na złom. Szczytowe do niedawna osiągnięcie myśli technicznej w elektronice, traktuje się obecnie jako surowiec, z którego można uzy[illegible] gramów poszukiwanych metali.

W gospodarce i przemyśle, [illegible] nictwie i biurze, w wojsku i wszędzie tam gdzie jest potrzebny, kró[illegible] mikroprocesor. Niewielki, eleg[illegible] zapakowany, wygodny w użyciu, łatwy w obsłudze i coraz częściej wchodzący w skład osobistego wyposażenia człowie[illegible] lat będzie rzeczą tak n[illegible] zegarek. A może na[illegible] jeszcze [illegible]dziej potrzebną. Będzie doradcą, ekspertem, [illegible] narzędziem o wyjąt[illegible].

Mikrokomputer [illegible] to [illegible]isty, pomocn[illegible] mózg [illegible] [illegible]mu cz[illegible] [illegible] [illegible] nie [illegible].

W dydaktyce mikrokomputer wkrada się w łaski nauczycieli, uczniów. Nie wszyscy jeszcze go akceptują, ale wszystkich bardzo ciekawi.

Jakie są możliwości zastosowania elektroniki w szkole? Czy efekty kształcenia wzrosną przy zastosowaniu nowych środków? Komputery będą pomagać czy tylko staną się elementem dekoracyjnym w klasie?

Zaakceptowanie szklanego ekranu z przyciskami, w szkolnej rzeczywistości oznacza sięgnięcie po nowe wartości w nauczaniu i uczeniu się. Jednak efekty kształcenia w dalszym ciągu zależeć będą przede wszystkim od ucznia i nauczyciela. Możliwości współpracy człowieka z bezduszną „maszyną" jest wiele.

Z[illegible] [illegible]prowad[illegible] [illegible] jest [illegible] [illegible]ć b[illegible]. Maszyna powinna „zmusić" użytkownika do odpowiadania na pytania. Od trafności odpowiedzi uzależniona jest dalsza realizacja programu. Uczeń może sam wielokrotnie powtarzać określone lekcje lub też po bezbłędnym opa[illegible] [illegible]riału [illegible] zadan[illegible] [illegible] [illegible] [illegible] przez ucznia odpo[illegible] może ocen[illegible] sto[illegible] opanowania przez niego wiedzy [illegible]odą [illegible] jest [illegible] krót[illegible] [illegible] [illegible]kterze [illegible]cyjnym lub pochlebne słowa [illegible] dźwięki. Taka forma uznania pow[illegible] działać mobilizująco ale zdarza [illegible] odwrotnie. Słuchanie hymnu pochwalnego na cześć swoich zdolności może doprowadzić do niezbyt szybkiej reakcji na kolejne pytania. Jednak po opanowaniu materiału, uczeń już nie daje się zaskoczyć, nawet przy znacznym skróceniu czasu na odpowiedź.

Coraz częściej mikrokomputery wykorzystuje się do naśladowania ludzkiej mowy. Mogą wyręczać nauczyciela, nie mogą jednak całkowicie go zastąpić. Amerykańska firma IBM przeprowadziła eksperyment, w którym uczyła czytać małe dzieci, za pomocą mikrokomputera z układem syntezy mowy. W ciągu 3 miesięcy „pociechy" poddane tej próbie, nauczyły się samodzielnie czytać.

Nowe możliwości w nauczaniu stwarzają najnowsze komputery, które rozpoznają i „rozumieją" ludzką mowę. Należą do nich komputery piątej generacji. Największe osiągnięcia w tej dziedzinie odnotowano w Japonii i USA. Zastosowanie tego typu urządzeń w szkolnictwie doprowadzi prawdopodobnie do rewolucji w dotychczasowych systemach dydaktycznych.

NA zmysły i wyobraźnię ucznia najbardziej oddziałują barwne i ciekawe obrazy, ukazujące się w różnych kombinacjach i w nieregularnym czasie. Odpowiednio przemyślane i opracowane, nie tylko uatrakcyjniają proces kształcenia ale zmuszają do aktywności i intensywnego myślenia. Dlatego też, opracowując program edukacyjny, należy pamiętać o tym, aby zachować odpowiednie proporcje formy i treści. Nieprzestrzeganie tych reguł może poważnie ograniczyć efekty.

„Elektroniczne liczydło" stało się nie tylko modne ale i konieczne w zakładzie pracy, w biurze czy nawet w szkole. Obecnie za pomocą komputerów kształci się nie tylko informatyków czy matematyków. Swobodnie posługują się [illegible]mi lekarze i inżynierowie pra[illegible] [illegible] róż[illegible]ch branżach przemy[illegible]. Obl[illegible]enia i analizy, które [illegible] najwięcej czasu, dziś sprowadzają się do wykorzystania określonego programu. Wyniki obliczane są z szybkością kilkunastu milionów operacji na sekundę.

Przetwarzan[illegible] nie dotyczy tylko da[illegible]nych matematycznych ale również programów edu[illegible]cyjnych [illegible]e z [illegible] [illegible]em języków ci[illegible]cych [illegible] [illegible] komputera można [illegible] wprowadzić [illegible]ny zbiór znaków. Po zastosowaniu [illegible] programu [illegible] [illegible] lub otrzymać odpowiednie wydruki.

WŚRÓD programów edukacyjnych kilka zasługuje na szczególną uwagę. Najbardziej popularne są „lekcje komputerowe". Na ekranie wyświetlane są pewne partie materiału, a następnie poprzez różnego rodzaju testy sprawdza się stopień jego opanowania.

dokończenie na str. 14

dokończenie ze str. 13

Ucznia kontroluje komputer. Kreda i tablica zostaje zastąpiona częściowo przez monitor i mikroprocesor W ten sposób nowoczesne środki dydaktyczne są „przemycane" do starych metod nauczania. Z badań jednak wynika, że taka forma kształcenia nie jest zbyt efektywna

Większe walory dydaktyczne posiadają programy zawierające ćwiczenia i treningi oraz gry uczące. W nauczaniu języków obcych wykorzystuje się je do ćwiczeń gramatycznych oraz sprawdzania opanowania słownictwa.

Na bajecznie kolorowym ekranie monitora ukazuje się obcojęzyczne zdanie Wszystkie wyrazy napisane zostały z małej litery Należy znaleźć te, które pisze się z dużej. Skaczącego na ekranie „krasnala" można skierować nad dowolny wyraz. Jeżeli uczeń zdecyduje się na konkretną literę, akcentuje to „ludzikiem" i powiadamia mikrokomputer. Albo słyszy triumfalny hymn świadczący o trafności odpowiedzi lub tylko ostrzegawcze i złowróżbne, na cały ekran rozlewające się — nie. Do trzech razy sztuka — ta zasada obowiązuje również w tym systemie. Po bezskutecznych próbach odgadnięcia „cierpliwy" komputer drukuje prawidłowo napisane zdanie, podkreślając wyrazy, które pisze się z dużej litery Dodatkowo „przemądrzała maszyna" rejestruje słabszą formę psychointelektualną uczącego się i dla odprężenia proponuje prostą zabawę. Na ekranie „wyrasta" zielone drzewko z jabłkami. Jednocześnie ukazuje się litera lub cyfra, którą należy odnaleźć na klawiaturze komputera. Jeżeli uczący się, w odpowiednim i stale skracającym się czasie, potrafi naciskać zadawane znaki, spadające z drzewka jabłka gromadzone są w odpowiednim pojemniku. Pomyłki lub zbyt długi czas szukania, znaczone są rozbitymi owocami. „Mecz" kończy wyświetlenie wyniku, na podstawie którego komputer decyduje czy dalej już można się uczyć czy wybrać inną zabawę edukacyjną

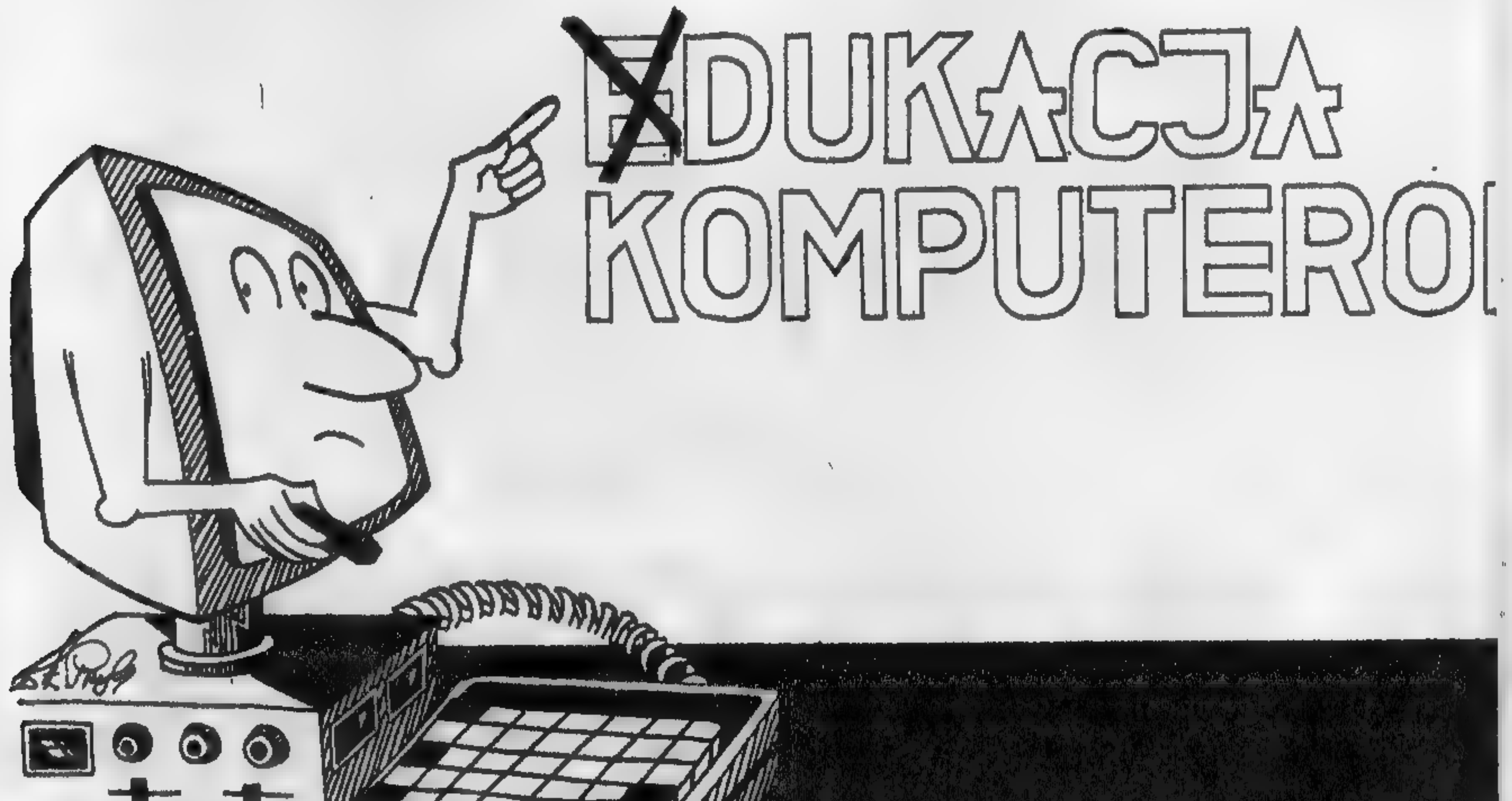
CIEKAWIE prezentują się programy dla nauczycieli Są takie, które pomagają w przygotowaniu lekcji, układaniu różnorodnych diagramów lub służą jako narzędzia do tworzenia nowych programów. Na komputerze można opracować i drukować zestawy pytań, testy sprawdzające albo powielać interesujące publikacje. Nauczyciel może w krótkim czasie dysponować prawie nieograniczoną ilością materiałów pomocniczych. Przez to jego praca może być łatwiejsza, a nauczanie efektywniejsze. Jeszcze szersze perspektywy roztaczają się przed nauczycielami umiejącymi pisać programy. Indywidualna praca z komputerem rozszerza bowiem zakres jego możliwości o nowe wartości.

Prawdziwa rewolucja komputerowa w szkolnictwie jest dopiero przed nami. Już obecnie dzieci, oprócz książek i piórnika, noszą w tornistrach kalkulatory. Za kilka lat będą to komputery osobiste. Podłączone w klasie do komputera nauczyciela, umożliwią prowadzenie lekcji w nieznany do tej pory sposób. Z ich pomocą będzie można pisać sprawdziany i klasówki, posługiwać się nimi jak encyklopedią, słownikiem czy książką, prowadzić bezpośredni dialog z nauczycielem nawet z domu.

Wszystkich możliwości, którymi zaskoczyć może nas jeszcze komputer, nie sposób dzisiaj przewidzieć. Te, o których wiemy, napawają optymizmem... i najczęściej szokują.

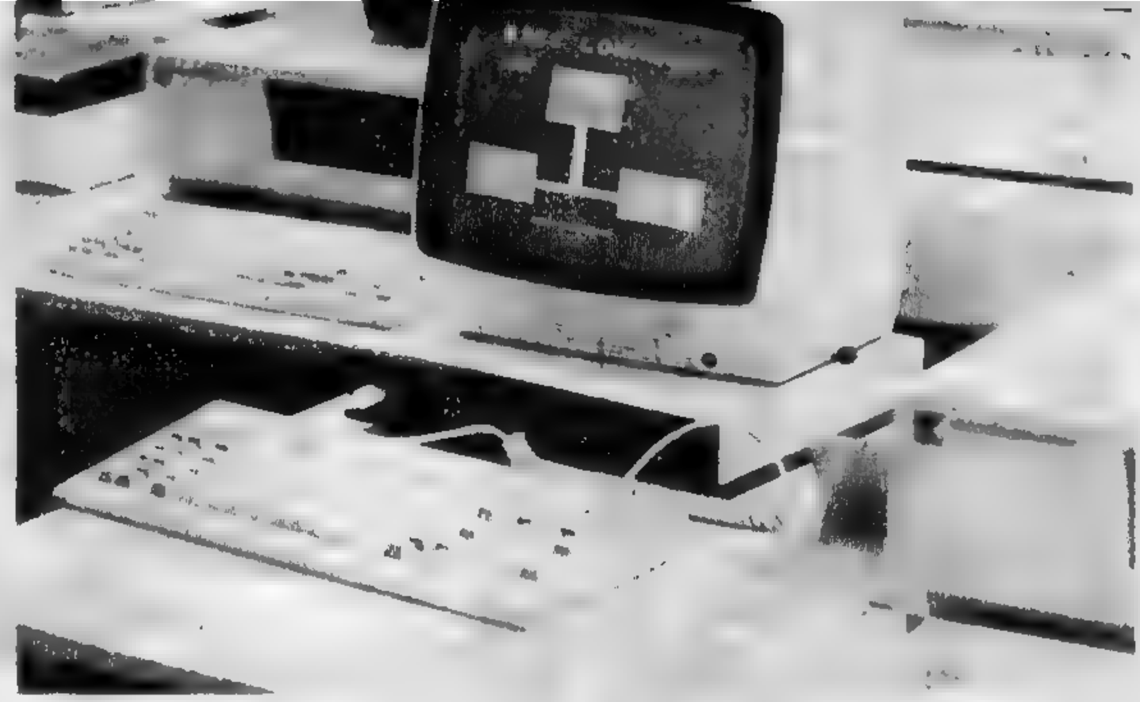
TAK czy inaczej do komputeryzacji życia społeczeństwo musi zostać przygotowane. Szkoła będzie główną instytucją programowo wdrażającą „mikroprocesorową rzeczywistość". Nadciągająca cywilizacja „elektronicznych robotów" już dzisiaj sygnalizuje komputerową dydaktykę

RYSZARD ROGOŃ



Rys. Michał Przybyłowski

Wakacje z komputerem



Tegoroczne lato „obrodziło" w komputerowe imprezy. Tylko pod patronatem ZSMP, a ściślej Federacji Klubów Komputerowych Młodych Mistrzów Techniki utworzono około 30 obozów i zorganizowano kilkanaście kursów, które były najczęściej pierwszym spotkaniem z informatyką.

W Wilkasach koło Giżycka rozgościła się młodzież i miłośnicy komputerów z woj. katowickiego. Patronat nad obozem objął Wojewódzki Zarząd ZSMP w Katowicach przy współudziale władz centralnych związku. Uczestniczyło w nim 40 dziewcząt i chłopców

* * *

Na kurs zapisałam się trochę z ciekawości, ale również dlatego, że w szkole mamy komputer, który chciałabym umieć obsługiwać — zwierza się **Violeta Grstka**, II klasa LO Rybnik — *moim marzeniem jest zbudowanie programu zawierającego określony zbiór roślin, np. kwiatów. A w ogóle to przeżywam fajną przygodę z komputerem.*

— **Artur Zawada** (Technikum Samochodowe): *w naszej szkole jest komputer, ale nie ma kto go obsługiwać. Na kurs namówił mnie wychowawca. Cieszę się, że po przyjeździe do szkoły będę „fachowcem" (jeszcze miernym) „od komputera".*

— **Ewa Idzińska** (Technikum Górnicze Tychy): *w szkole mamy SPECTRUM, ale praktycznie trudno się do niego „dopchać". Trochę jestem przerażona ilością materiału, jaki nam tutaj „serwują". Nie wiem, czy zdam egzamin, ale ta zabawa z komputerem bardzo mi się podoba*

— **Adam Biersk** (LO Wodzisław): *najbardziej denerwujące jest to, że „studio" komputerowe działa tylko od godziny 6.00 do 23.00. Z tego jeszcze od 20.00 do 23.00 dostępne jest dla wszystkich (nie tylko dla kursantów). Mamy tylko sześć stanowisk (komputery „Meritum") i nie zawsze jest dostatecznie dużo czasu, aby teorię „przekuwać" w praktykę. Całe szczęście, że od czasu do czasu udaje się nam zabrać komputery na noc do domków, w których mieszkamy. Wtedy możemy trochę poszaleć.*

WILKASY'86

— **Wojciech Trautsolt:** *wraz z kolegą po 4 dniach tego kursu napisaliśmy własny program. Myślę, że będzie to nasza najlepsza wypowiedź.*

```
1 REM Program „Palenie skraca życie"
2 REM Autorzy: Wojciech Trautsolt i Zowka
3 REM 1986.08.13 Wilkasy
10 CLS
20 PRINT „Czy palisz papierosy"
30 INPUT a$
40 IF a$ = „NIE" THEN GOTO 1000
45 IF a$ <> „TAK" THEN GOTO 20
50 PRINT „Od kiedy palisz"
60 INPUT „Podaj rok"; B
62 IF B>1986 THEN PRINT „Kłamiesz, nie rób tego więcej": GOTO 50
63 IF B<1945 THEN PRINT „Według moich obliczeń jesteś już nieboszczykiem": END
70 PRINT „Napisz w przybliżeniu ile papierosów palisz dziennie"
71 INPUT C: IF C<0 THEN PRINT „Nie ze mną te numery": END
72 IF C>50 THEN PRINT „Może masz kondycję, ale bajać to my, a nie nam": END
90 LET Q=1986 — B
92 IF Q=0 THEN LET Q=0.6
100 LET Z=Q*C*8
110 LET x=Z/60
120 LET y=x/24
130 LET v=y/30
150 PRINT „Jak wiadomo palenie jest szkodliwe czyli skraca życie"
170 PRINT „Żyjesz"; Z: „Minut krócej, to jest"; x: „Godzin"
180 PRINT „Czyli jak łatwo obliczyć umrzesz; v;" miesięcy wcześniej niż mógłbyś"
200 PRINT „Moje obliczenia zostały naukowo udowodnione przez ekspertów": END
1000 PRINT „Przyjdź jak się nauczysz palić": END
```

— **Konrad Ołdakowski** (uczeń 8 klasy Szkoły Podstawowej w Giżycku): *w domu mam Commodore VC-20, ale nie bardzo wiem, jak się nim posługiwać. Jestem wdzięczny kierownictwu obozu za to, że wraz z kolegami zostałem przyjęty na ten kurs. Z Giżycka na zgrupowanie dojeżdżam 5 km na rowerze*

Wnioski:

Piotr Maculewicz — kursant,

- *Najważniejsze, że zajęcia prowadzone są regularnie,*
- *Jest dużo teorii, ale wykładana jest przystępnie,*
- *Doskonalę swoje umiejętności praktyczne i próbuję pisać własny program.*

Adam Cały — instruktor;

- *Przed kursem powinna być prowadzona wstępna kwalifikacja uczestników obozu. W grupie jest zbyt zróżnicowany poziom, (na 40 uczestników kursu 5 osób ma w domu komputer, a w szkole zetknęło się z nim 7).*
- *Nie do końca rozwiązano problem sprzętu, jest go po prostu mało*
- *Brak jest również ogólnodostępnej literatury do prowadzenia szkolenia na kursach.*
- *Na obozach należy zwrócić większą uwagę na właściwą bazę lokalową.*

Andrzej Krowiak — szef i opiekun metodyczny

- *Jest to jedna z form czynnego wypoczynku, łączenie przyjemnego z pożytecznym.*
- *Szkolimy kadry dla szkolnych klubów komputerowych.*
- *Takie kursy to przygotowanie przyszłego społeczeństwa do pracy z komputerem na co dzień i praktycznie w każdej dziedzinie.*

Wakacje z komputerem... hasło, do którego będziemy jeszcze wracać

R. R.

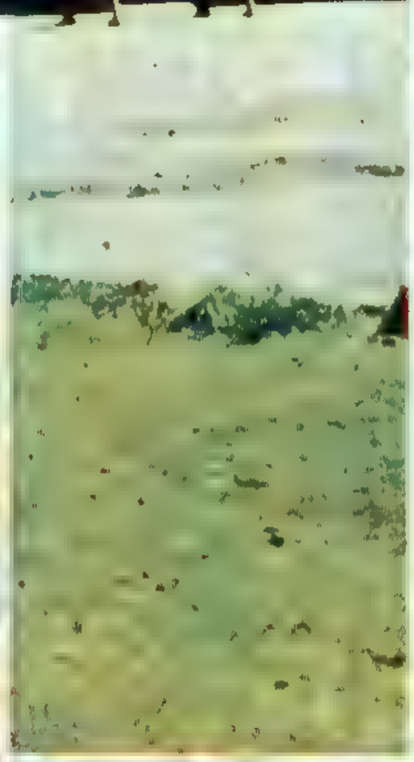




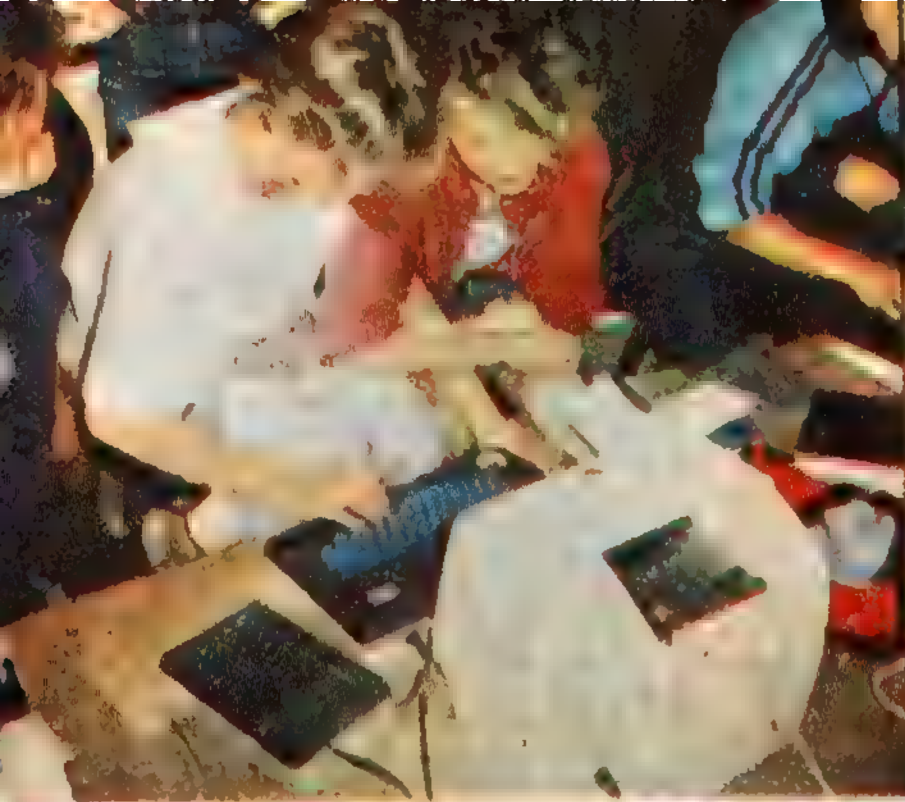
STUDIO
KOMPUTE
ROWE
ATARI













WILKASY'86

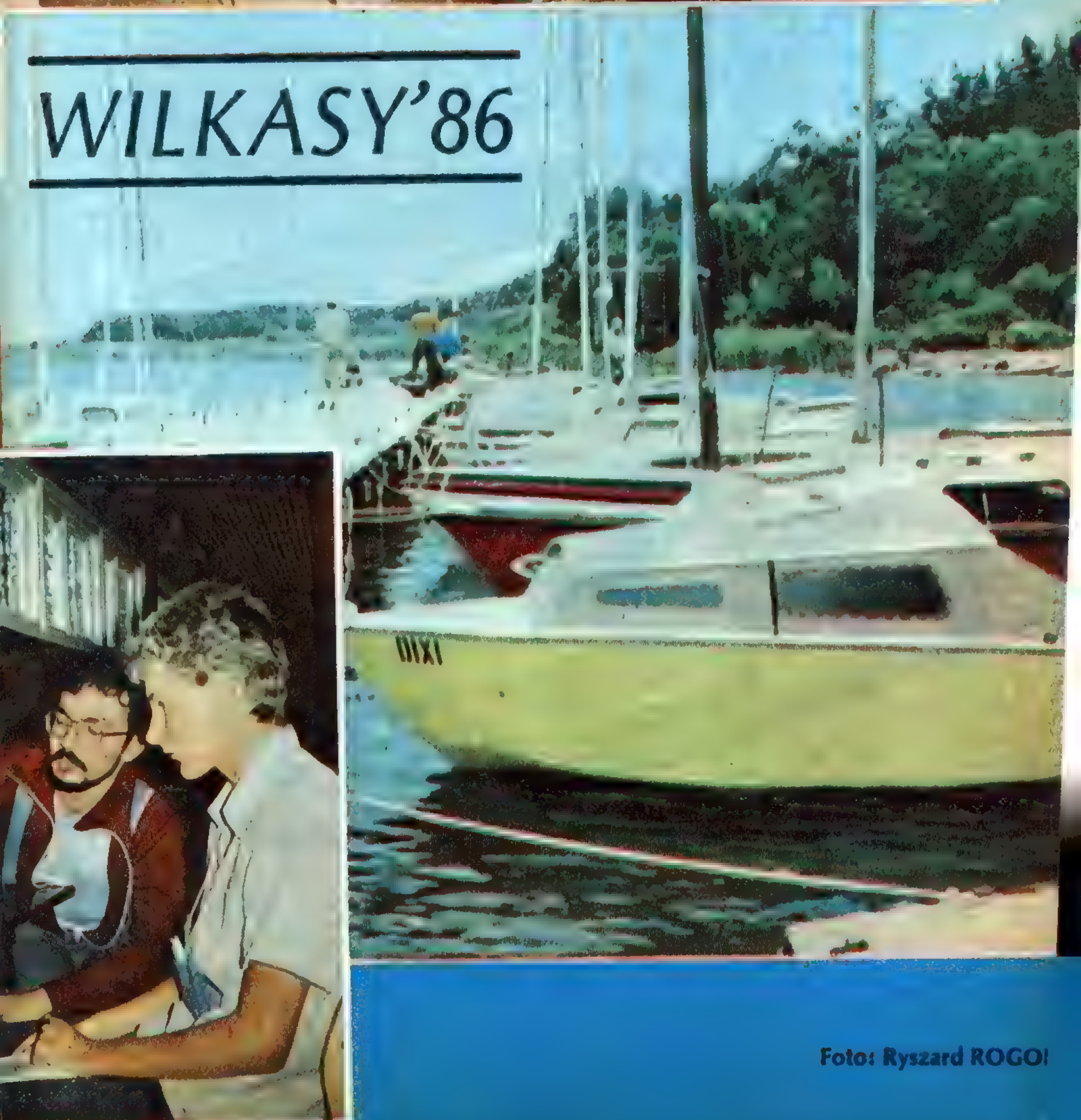


Foto: Ryszard ROGO

O komputerach, przyszłości i interesach...

Rozmowa z Wojciechem Wyszomirskim, sekretarzem generalnym Rady Krajowej Turnieju Młodych Mistrzów Techniki i Andrzejem Maciągowskim — sekretarzem Ogólnopolskiej Federacji Klubów Komputerowych MMT.

Wiesław Cetera: — Mamy już Federację — co dalej?

Wojciech Wyszomirski: — Przed nami dokładnie to, co legło u podstaw jej powołania. W lutym 1986 roku, po spotkaniu przedstawicieli klubów komputerowych postanowiliśmy powołać do życia instytucję, która ułatwiałaby życie wszystkim miłośnikom



raczkującym użytkownikom mikrokomputerów

C.: — Ułatwiać, czyli co?

Andrzej Maciągowski: — Czyli pomagać w zdobyciu sprzętu, mowa oczywiście wyłącznie o klubach działających w strukturze ZSMP, niezbędnej literatury i fachowej pomocy

C.: — Nasz komputerowy rynek informacji rozszalał się ostatnio. Wiele książek, serwisowych instrukcji, programów na kasetach i wreszcie samego sprzętu można znaleźć dosłownie na każdym większym targowisku — barierą jest jednak cena

W. W.: — Komputery są drogim sprzętem, dlatego musimy spojrzeć na nie nie tylko jak na element rozrywki, ale na trwały synonim systemu edukacyjnego. To pierwszy krok, drugi — nieunikniona konieczność uporządkowania cen tego sprzętu, ale ani nasz związek, ani kluby same zrobić tego nie mogą

A. M.: — Sprzęt to nie wszystko. Staramy się, aby szybko dostarczyć naszym członkom rzetelnej wiedzy. Już wkrótce ukaże się [illegible] wydanych przez ośrodek katowicki „Polin", poświęconych mikrokomputerom. Są wśród nich takie między innymi tytuły **„Elementy grafiki komputerowej", „Baza danych na mikrokomputerach", „Pascal dla mikrokomputera Meritum i Spectrum".**

C.: — Gdzie będzie można je kupić?

A. M.: — Niestety, nakład jest niewielki, pytać o nie będzie można w klubach komputerowych

C. — Jedna jaskółka wiosny nie czyni i podejrzewam, że „posucha" na literaturę trwać będzie nadal

A. M.: — Sądzimy, że już wkrótce uda nam się uruchomić bank programów. Jego działanie polegałoby na zbieraniu, powielaniu i przekazywaniu zainteresowanym w zamian za dostarczenie na przykład do banku jakiegoś własnego programu. Pomysł wymaga jeszcze dopracowania, a poza tym bank należałoby dobrze wyposażyć w sprzęt i kadrę, ale mam nadzieję, że już wkrótce rozpocznie działalność.

C.: — Jak na początek, to zamierzenia są ambitne

A. M.: — I realne

W. W.: — Chodzi też nie tylko o sprawdzalność naszego działania, kryterium oceny to nie tylko wiarygodność Rady Federacji, ale przede wszystkim ostateczny efekt naszego działania. Mamy w tym również swój interes, ułatwiamy nie tylko działanie klubom, sądzimy, że ułatwimy rozwój niejednemu młodemu talentowi, a stwarzając odpowiednie warunki, damy szansę rozwoju

C.: — A to z pewnością nam wszystkim się opłaca

Dziękuję za rozmowę

Wiesław CETERA

PROGRAM 25

SPECTRUM

```
 100 PRINT "Uwaga! Tablica obraz
ow   bedzie kasowana!": STOP
 110 DIM z$(10000,3)
 120 DIM a(40): REM - Maksymalna
liczba obrazow
 130 REM - Start kompensacji
 140 LET a(1)=2
 150 CLS : INPUT "Podaj kolejny
numer""kompensowanego obrazka";
p
 160 LET t=a(p-1)
 165 CLS : FLASH 1: PRINT "KOMPE
NSACJA TRWA. PROSZE CZEKAC": FLA
SH 0
 170 GO TO 300
 200 REM - Wczytanie obrazka do
kompensacji
 210 LOAD ""CODE 58000,6912
 220 GO TO 130
 300 LET l=50: LET k=50
 310 LET w=0
 320 LET e=255
 330 GO SUB 500
 340 LET w=2048
 350 LET e=2303
 360 GO SUB 500
 370 LET w=4096
 380 LET e=4352
 390 GO SUB 500
 400 CLS : PRINT "Koniec kompens
acji obrazka.    Dotychczas sko
mpensowane       obrazki  zajmu
ja ";t;" bajtow": LET a(p)=t
 410 STOP
 500 FOR n=w TO e
 510 LET g=PEEK (58000+n)
 520 LET d=PEEK (59280+n)
 530 GO SUB 600
 540 LET k=k+1
 550 IF k<82 THEN GO TO 580
 560 LET l=l+1: LET k=50
 570 IF l>72 THEN GO TO 400
 580 NEXT n
 590 RETURN
 600 IF d=0 AND g=0 THEN RETURN
 610 LET t=t+1
 620 LET c=0
 630 IF g=0 THEN GO TO 670
 640 IF g=15 THEN LET c=1
 650 IF g=240 THEN LET c=2
 660 IF g=255 THEN LET c=3
 670 IF d=0 THEN GO TO 710
 680 IF d=15 THEN LET c=c+4
 690 IF d=240 THEN LET c=c+8
 700 IF d=255 THEN LET c=c+12
 710 LET z$(t,1)=CHR$ (l)
 720 LET z$(t,2)=CHR$ (k)
 730 LET z$(t,3)=CHR$ (c+128)
 740 RETURN
1000 REM - DEKOMPENSACJA
1005 CLS
1010 INPUT "Podaj kolor atrament
u";k2
1015 INPUT "Podaj kolor tla";k:
PAPER k
1020 INPUT "Podaj numer obrazka"
;f
1025 CLS
1030 FOR t=a(f-1) TO a(f)
1040 LET l =CODE (z$(t,1))-50
1050 LET k =CODE (z$(t,2))-50
1060 LET c=CODE z$(t,3)
1070 PRINT AT l,k; INK k2;CHR$ c
1080 NEXT t
1090 STOP
```

100 obrazów w 48K RAM ZX Spectrum

Często podczas, na przykład, prowadzenia zajęć niezbędne jest demonstrowanie rysunków. Także w tym przypadku komputer (tutaj ZX Spectrum) może okazać się bardzo przydatny.

Po wykonaniu szeregu obrazów (na ekranie TV), można je zarejestrować na kasecie lub dysku i w zależności od potrzeby odtwarzać na monitorach. Niestety w praktyce kaseta okazuje się za wolna, a dysk po prostu za drogi...

Prezentowany program KOMPENS znacznie ogranicza te kłopoty. Powoduje bowiem, że obrazy, które dotychczas zajmowały niemal 7 Kbajtów każdy, po pewnej obróbce — kompensacji — zajmują przeciętnie od 100 do 350 bajtów. Zatem na początku zajęć można „ściągnąć" z kasety do komputera na przykład: 100 (sto!) skompensowanych obrazów. Czas wyboru i dekompensacji dowolnego z tych stu obrazów nie przekracza 3 sekund.

Niestety jest pewne ograniczenie — obrazy mogą być zbudowane tylko ze znaków graficznych ZX Spectrum. Praktycznie najlepiej budować obrazy programem M draw (przy największym powiększeniu!) i zapisywać kolejno na kasetę.

Prezentowana wersja KOMPENS pomija atrybuty wyróżnianych pixeli obrazów. Uzupełnienie tego braku jest stosunkowo proste i może dostarczyć dużo satysfakcji.

Uwagi praktyczne:

1. **GO TO 110 — tylko raz na początku wykorzystywania programu!**
2. **GO TO 200 — KOMPENS czyta kolejne obrazki z kasety do kompensacji.**
3. **Po wczytaniu obrazka z kasety należy podać numer kompensowanego obrazka (komputer poprosi o to). Pierwszy obrazek powinien mieć numer 2.**
4. **Wykonuje się kompensacja (kilka minut).**
5. **Koniec kompensacji program sygnalizuje podając ile bajtów zajmują wszystkie dotychczas skompensowane obrazy.**
6. **GO TO 1000 — dekompensacja i wyświetlenie wybranego obrazu. Także kolor tła i atramentu jest wybierany przez podanie jego numeru.**

POWODZENIA

WŁODZIMIERZ GOGOLEK

Wakacje z komputerem

PIECZARKI'86

Foto: Jan Zelman













BASIC

Jako uzupełnienie tego, co zostało powiedziane o algorytmach, zalecamy uważne przeczytanie wspomnianego artykułu z pierwszego numeru „IKS-a" („Co trzeba wiedzieć?").

W przykładzie 3.2 obliczany jest pierwiastek kwadratowy z wyrażenia p (p — a) (p — b) (p — c). Zajmiemy się teraz obliczaniem wartości, np. funkcji sinus, cosinus, pierwiastka kwadratowego w języku BASIC

Jak zawsze zaczynamy od przykładu. Włączamy komputer i do pracy!

Przykład 3.4

Zaprogramować w BASICu obliczenie wartości y danej wzorem

$$y = \sqrt{3}\sin x + ab\cos(x+b) + \frac{\lg x + |a-b|}{\ln x}\, e^{(x-b)}$$

dla a = 1,2   b = 4,25   x = 2,3

Algorytm obliczeń jest prosty więc piszemy program

```
  10 REM Przyklad 3.4
  15 LET a=1.2
  20 LET b=4.25
  25 LET x=2.3
  30 LET z=SQR 3*SIN x+a*b*COS (
x+b)
  35 LET w=((LN x/LN 10)+ABS (a-
b))/LN x
  40 LET y=z+w*EXP (x-b)
  45 REM z,w - wyniki posrednie
obliczen
  50 PRINT "SIN ,COS ,SQR ,LN ,A
BS ,EXP sa  nazwami funkcji stan
dardowych"
  55 PRINT '"argument funkcji SI
N ,COS i TAN nalezy podawac w ra
dianach (uwa-ga: PI/2 radianow =
 90 stopni)"
  60 PRINT '"funkcja LN  oblicza
 wartosc logarytmu przy podstawi
e e (logarytm naturalny - ln)"
  65 PRINT '"wartosc logarytmu p
rzy dowolnej podstawie c oblicza
my wg.wzoru :log przy podst.c z
x =LN x/LN 10"
  70 PRINT '"funkcja SQR oblicza
 wartosc     pierwiastka kwadrat
owego"
  75 PRINT '"funkcja ABS -wartos
c bezwzgledna"
  80 PRINT '"funkcja wykladnicza
 EXP oblicza wartosc e↑x"
  85 PRINT '"  wartosc wyrazeni
a dla"'"a= ";a;"     b= ";b;"
x= ";x'"      wynosi : ";y
```

Wprowadźmy teraz program i wykonajmy. Sprawdźmy wynik z wynikiem podanym niżej

```
SIN ,COS ,SQR ,LN ,ABS ,EXP sa
nazwami funkcji standardowych

argument funkcji SIN ,COS i TAN
nalezy podawac w radianach (uwa-
ga: PI/2 radianow = 90 stopni)

funkcja LN  oblicza wartosc loga
rytmu przy podstawie e (logarytm
 naturalny - ln)

wartosc logarytmu przy dowolnej
podstawie c obliczamy wg.wzoru :
log przy podst.c z x =LN x/LN 10

funkcja SQR oblicza wartosc
pierwiastka kwadratowego

funkcja ABS -wartosc bezwzgledna

funkcja wykladnicza EXP oblicza
wartosc e↑x

   wartosc wyrazenia dla
a= 1.2      b= 4.25     x= 2.3
      wynosi : 6.7939162
```

W podanym przykładzie pokazaliśmy, w jaki sposób język BASIC umożliwia obliczanie wartości podstawowych funkcji matematycznych. Zestaw tych funkcji i innych nazywa się funkcjami standardowymi. Funkcje standardowe wchodzące w skład zestawu zamieszczonego w BASICu dla ZX Spectrum są opisane w książce ZX Spectrum — BASIC programming.

Nazwy funkcji standardowych składają się z 3 liter i poprzedzają argument ujęty (lub nie) w nawiasy okrągłe. W ZX Spectrum, w wypadku gdy argument funkcji jest liczbą lub nazwą zmiennej prostej, nie musi być umieszczony w nawiasach okrągłych. Argumentem każdej funkcji standardowej może być wyrażenie arytmetyczne (może ono zawierać też funkcje), piszemy go wtedy w nawiasach.

Funkcje standardowe stosujemy w wyrażeniach arytmetycznych na tych samych zasadach co zmienne i stałe, posługujemy się wtedy nazwami symbolicznymi funkcji standardowych, rozumiejąc każdą z nich jako symbol wartości funkcji odpowiadającej aktualnym wartościom argumentu.

np. napis **SQR 3** oznacza obliczenie wartości $\sqrt{3}$

napis **SIN X** oznacza obliczenie wartości funkcji sinus dla argumentu x.

W podanych programach w poprzednich wykładach wszystkie instrukcje były wykonywane sekwencyjnie, jedna po drugiej, poczynając od linii o najniższym numerze. W wielu algorytmach obliczeniowych daje się zauważyć potrzebę wszelkiego rodzaju rozgałęzień, *„obejść" i „pętli"*, a więc sytuacji, w których instrukcje nie są wykonywane sekwencyjnie.

Analizując algorytmy podane w postaci schematów blokowych w przykładach 3.1, 3.2, 3.3 znajdujemy przykłady „rozgałęzień", *„obejść"* i *„pętli"*.

Weźmy pod uwagę algorytm przykładu 3.1 i napiszmy program w BASICu.

Przykład 3.1

```
  10 REM Przyklad 3.1
  20 PRINT "        Rownanie a*x+b=0
"
  30 INPUT "wprowadz a=";a,"wpro
wadz b=";b
  40 IF a<>0 THEN GO TO 80
  50 IF b=0 THEN GO TO 70
  55 PRINT '"a= 0 i b<> 0"'"rown
anie sprzeczne -
            brak rozwiazan"
  60 GO TO 75
  70 PRINT '"a= 0 i b= 0"'"kazda
 wartosc x spelnia rownanie"
  75 STOP
  80 LET x=-b/a
  90 PRINT '"a= ";a;"      b= ";b
 100 PRINT '"pierwiastek rownani
a x= ";x
 110 STOP
```

dokończenie na str. 22

dokończenie ze str. 21

Wprowadźmy i wykonajmy ten program. Niżej prezentujemy otrzymane wyniki dla różnych a, b.

```
    Rownanie a*x+b=0

a= 0 i b<> 0
rownanie sprzeczne -
             brak rozwiazan


    Rownanie a*x+b=0

a= 0 i b= 0
kazda wartosc x spelnia rownanie


    Rownanie a*x+b=0

a= 4      b= 10

pierwiastek rownania x= -2.5
```

Sprawdzenie a ≠ 0 oraz b = 0 w schemacie zostało zapisane w postaci bloków warunkowych o dwóch wyjściach:



Kazdy z tych bloków w programie zapisaliśmy instrukcją postaci.

40 **IF** a < > 0 **THEN** **GO TO 80**
i 50 **IF** b = 0 **THEN** **GO TO 70**

Podejmujemy tu pewne decyzje, które powinien wykonać komputer i działać stosownie do nich, piszemy wtedy instrukcję: **IF** co jest prawdą (lub nieprawdą) **THEN** zrób coś

Instrukcja **IF** nazywa się warunkową instrukcją sterującą i ma postać:

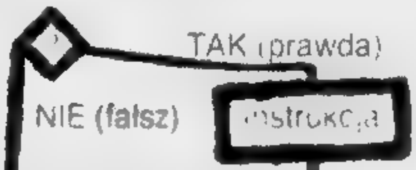
n **IF** wyrażenie logiczne **THEN** instrukcja

n — numer wiersza
wyrażenie logiczne — w najprostszym przypadku relacja.
instrukcja — dowolna instrukcja BASICu (lub ciąg instrukcji oddzielonych między sobą. (dwukropkiem).

W innych reprezentacjach BASICu instrukcja warunkowa ma postać.
IF wyraz. log. **THEN** numer wiersza znaczy to samo co.
IF wyraż. log. **THEN GO TO** numer wiersza.

Wyrazenie logiczne jest to takie wyrazenie, którego wartość może być albo prawdą, albo fałszem. Wykonanie instrukcji IF rozpoczyna się od obliczenia wartości wyrażenia logicznego. W wypadku otrzymania wartości równej **prawda,** następuje wykonanie instrukcji napisanej po słowie **THEN**, następnie przejście do wykonywania instrukcji, która jest napisana w kolejnym wierszu po instrukcji **IF**. Natomiast dla wartości **fałsz** będzie wykonywana ta instrukcja, która jest umieszczona w kolejnym wierszu po instrukcji **IF**

Wykonanie instrukcji **IF** można zapisać w bloku warunkowym postaci:



Relacje przeznaczone są do porównywania wartości dwóch wyrażeń arytmetycznych (lub tekstowych).

W BASICu relacje mają następującą postać ogólną.

$$e_1 \quad R \quad e_2$$

w której e_1 i e_2 są dowolnymi wyrażeniami arytmetycznymi lub tekstowymi, a R jest jednym z operatorów relacji podanych niżej

Operator relacji	Znaczenie	Klawisz
=	równa	L
< >	różne (nierówne, ≠)	W
<	mniejsze	R
< =	mniejsze lub równe (≤)	P
>	większe	T
> =	większe lub równe (≥)	E

Każdy znak operatora relacji otrzymamy, naciskając jednocześnie klawisz SYMBOL SHIFT i klawisz z odpowiednią literą

Teraz omówimy wykonanie instrukcji warunkowych w programie 3.1.

Realizacja instrukcji

40 **IF** a < > 0 **THEN GO TO** 80

przebiega w następujący sposób

jeżeli wartość zmiennej a jest różna od zera (nierówność spełniona) (tzn. wartością relacji a < > 0 jest prawda, to przejdź do instrukcji w wierszu o numerze 80, w przeciwnym przypadku (tzn. gdy jest fałsz), przejdź do instrukcji wypisanej w kolejnym wierszu po instrukcji IF tzn..

50 **IF** b = 0 **THEN GO TO** 70

Realizacja tej instrukcji IF przebiega w identyczny sposób: jeżeli wartością relacji b = 0 jest prawdą, to przejdź do instrukcji w wierszu o numerze 70, w przeciwnym przypadku (tzn. gdy jest fałsz) przejdź do instrukcji wypisanej w kolejnym wierszu po instrukcji IF tzn :

55 **PRINT** „Równanie sprzeczne — brak rozwiązań"

W obu tych instrukcjach po słowie THEN występują instrukcje **GO TO 80** i **GO TO 70.**

Instrukcja **GO TO** nazywa się bezwarunkową instrukcją skoku i ma postać:

n **GO TO** n_1

n — numer wiersza
n_1 — numer wiersza, do którego wykonanie programu „przenosi" instrukcja skoku

Wykonanie instrukcji **GO TO** n_1 powoduje bezwarunkowe przejście do wykonania instrukcji umieszczonej w wierszu o numerze n_1. Wiersz o numerze n_1 musi istnieć w programie, a w wypadku jego braku sygnalizowany jest błąd Jeżeli w wierszu tym (n_1) znajduje się instrukcja bierna (instrukcją bierną jest np. instrukcja **REM** instrukcje bierne nie wykonują się w czasie działania programu, lecz służą np do opisania programu jak instr **REM**), to wykonanie rozpoczyna się od pierwszej instrukcji czynnej, umieszczonej w programie po wskazanej instrukcji biernej.

Wykonanie tego programu przez komputer zakończyć się może wypisaniem na dole ekranu jednego z dwu komunikatów.

9 **STOP** statement, 75:1
lub 9 **STOP** statement, 110:1

które powiadają, że pierwsza instrukcja w wierszu 75 lub 110 spowodowała zatrzymanie programu.

Instrukcja **STOP** jest bezwarunkową instrukcją sterującą i ma postać

n **STOP**

n — numer wiersza

Instrukcja STOP służy do zakończenia obliczeń lub chwilowego przerwania programu. Naciśnięcie klawisza CONTINUE (CONT) spowoduje wykonywanie programu od instrukcji wypisanej w kolejności po instrukcji STOP (o ile taka w programie istnieje).

W BASICu ZX Spectrum nie jest konieczne stosowanie instrukcji STOP jako instrukcji kończącej wykonywanie programu. W schematach blokowych instrukcja STOP odpowiada blokowi granicznemu postaci:

STOP

W programie 3.1 instrukcja.

75 **STOP**

jest konieczna, gdyż wykonywanie programu dla tych danych kończy się w tym miejscu, a następne instrukcje nie mogą być wykonane

Natomiast instrukcję

110 **STOP**

można pominąć, gdyż jest ona ostatnią w tym programie.

Aby zapamiętać to, co zostało omówione w tym wykładzie, przerobimy kilka przykładów

Zapiszmy w BASICu algorytm podany i omówiony w przykładzie 3.2.

```
  10 REM Przyklad 3.2
  20 INPUT "podaj dlugosci bokow
 trojkata:"'"a= ";a;"      b= ";b
;"      c= ";c
  25 CLS
  30 PRINT '"Dany jest trojkat o
 bokach :"'"a= ";a;"      b= ";b;
"      c= ";c
  40 LET p=(a+b+c)/2
  50 LET z=p*(p-a)*(p-b)*(p-c)
  60 PRINT "p= ";p;"      z= ";z
  70 IF z<=0 THEN GO TO 150
  80 LET F=SQR (z)
  90 LET R=a*b*c/(4*F)
 100 LET rm=F/p
 110 PRINT '"Pole trojkata F= ";
F
 120 PRINT '"Promien okregu opis
anego          na trojkacie  R= ";
R
 130 PRINT '"Promien okregu wpis
anego          w trojkat      r= ";
rm
 140 GO TO 160
 150 PRINT '"Z tych bokow nie mo
zna zbudowac trojkata"
 160 PRINT '"chcesz liczy dalej
? ";
 170 INPUT "tak/nie ";x$
 175 PRINT x$
 180 IF x$="tak" THEN GO TO 20
 190 IF x$<>"nie" THEN GO TO 170
 200 PRINT '"Dziekuje -koniec pr
acy"
 210 STOP
```

Wprowadźmy i wykonajmy ten program. Niżej prezentujemy otrzymane wyniki dla różnych a, b, c.

```
Dany jest trojkat o bokach
a= 5      b= 6      c= 7
p= 9      z= 216

Pole trojkata F= 14.696938

Promien okregu opisanego
na trojkacie  R= 3.5721725

Promien okregu wpisanego
w trojkat       r= 1.6329932

chcesz liczy dalej ? nie

Dziekuje -koniec pracy


Dany jest trojkat o bokach
a= 5      b= 8      c= 3
p= 8      z= 0

Z tych bokow nie mozna zbudowac
trojkata

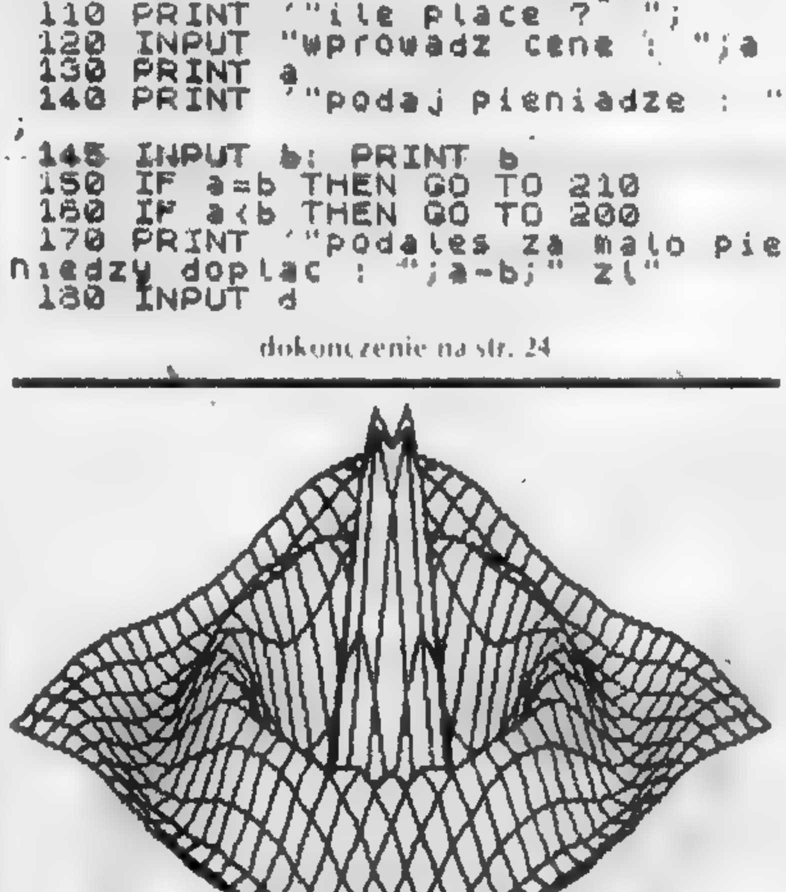
chcesz liczy dalej ? nie

Dziekuje -koniec pracy
```

Zapiszmy w BASICu algorytm podany i omówiony w przykładzie 3.3.

```
  10 REM Przyklad 3.3
  20 PRINT "Kupujemy IKS-a w kio
sku RUCHU"
  30 PRINT '"masz pieniadze ? ";
  40 INPUT "tak/nie ";x$
  45 PRINT x$
  50 IF x$="nie" THEN STOP
  55 IF x$<>"tak" THEN GO TO 40
  60 PRINT '"podejdz do kiosku"
  70 PRINT '"czy jest IKS ? ";
  80 INPUT "tak/nie  ";x$
  85 PRINT x$
  90 IF x$="nie" THEN GO TO 250
  95 IF x$<>"tak" THEN GO TO 80
 100 PRINT '"prosze o jeden egze
mplarz"
 105 REM instrukcji 110,120,130
brak na schemacie blokowym
 110 PRINT '"ile place ?  ";
 120 INPUT "wprowadz cene : ";a
 130 PRINT a
 140 PRINT '"podaj pieniadze : "
;
 145 INPUT b: PRINT b
 150 IF a=b THEN GO TO 210
 160 IF a<b THEN GO TO 200
 170 PRINT '"podales za malo pie
niedzy doplac : ";a-b;" zl"
 180 INPUT d
```

dokończenie na str. 24



Grafika MAZOVII

dokończenie ze str. 23

```
 190 LET b=b+d
 195 GO TO 150
 200 PRINT '"oto reszta  ";b-a
 210 PRINT '"wziales gazete ? ";
 220 INPUT "tak/nie  ";x$
 225 PRINT x$
 230 IF x$="tak" THEN GO TO 250
 240 GO TO 220
 250 PRINT '"odejdz od kiosku !"
 255 PRINT '"kupiles IKS-a ? ";
 260 INPUT "tak/nie  ";x$
 265 PRINT x$
 270 IF x$="tak" THEN STOP
 280 IF x$<>"nie" THEN GO TO 280
 285 PRINT '"szukasz dalej ? ";
 290 INPUT "tak/nie  ";x$
 295 PRINT x$
 300 IF x$="nie" THEN STOP
 305 IF x$<>"tak" THEN GO TO 290
 310 PRINT '"idz do nastepnego k
iosku !"
 320 GO TO 60
```

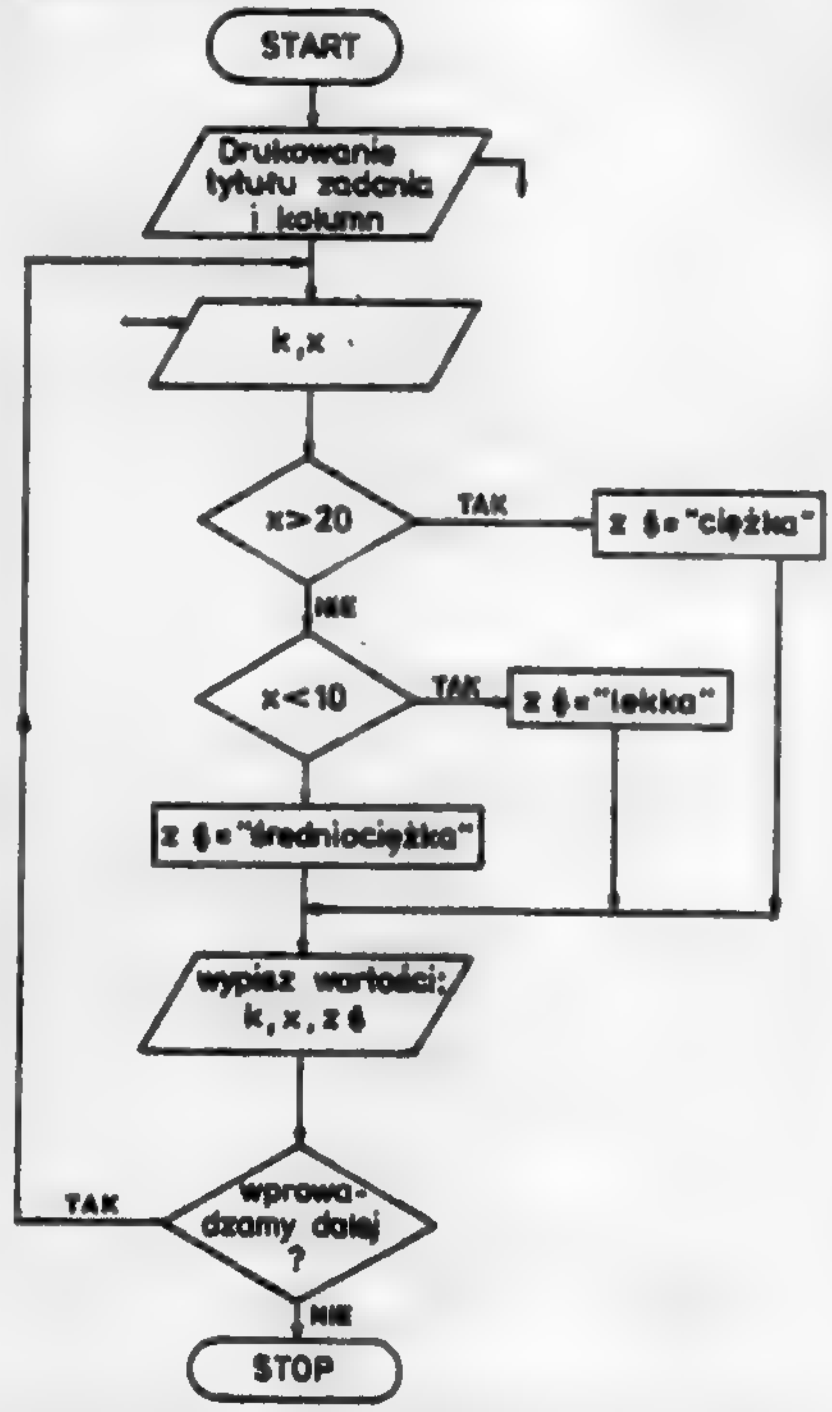
Zachęcam czytelników do napisania nowej na pewno lepszej wersji programu z przykładu 3.3 i przysłania tego programu do redakcji.

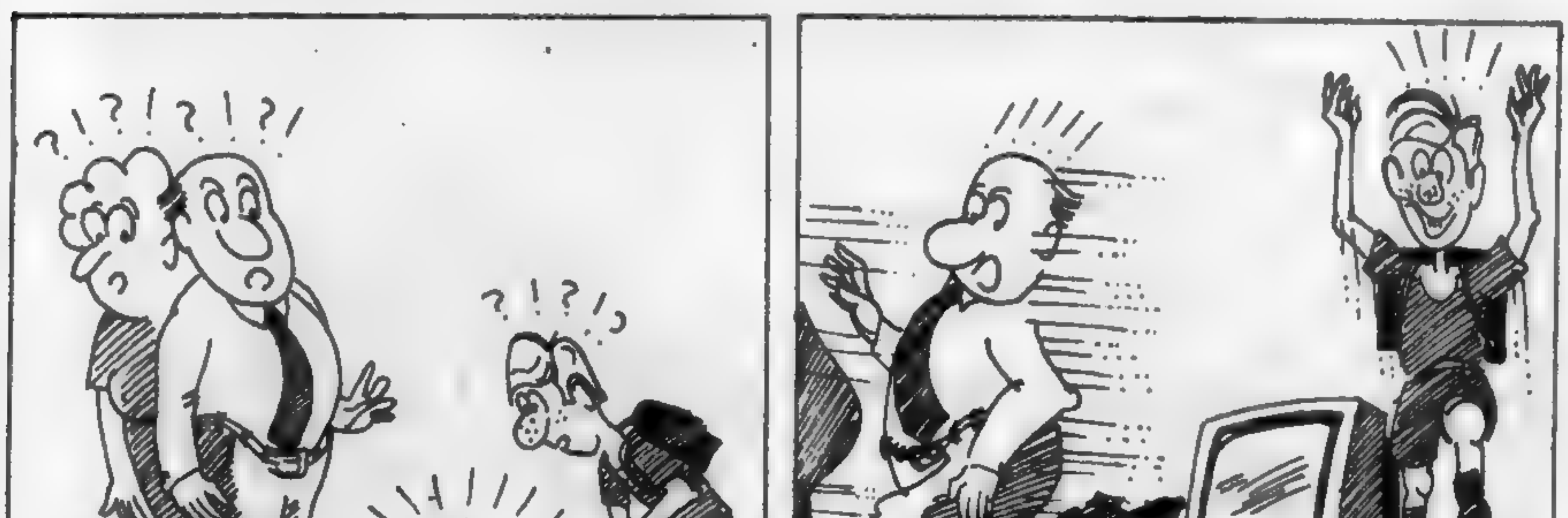
Przykład 3.5

W pudełku znajdują się ponumerowane kule. Należy każdą z nich zważyć i zapisać jej numer i wagę. Następnie napisać program, który dla dowolnej wartości zmiennych:
k — reprezentującej numer kuli
x — reprezentującej wagę kuli
wprowadzanych z klawiatury drukuje następujące komunikaty:

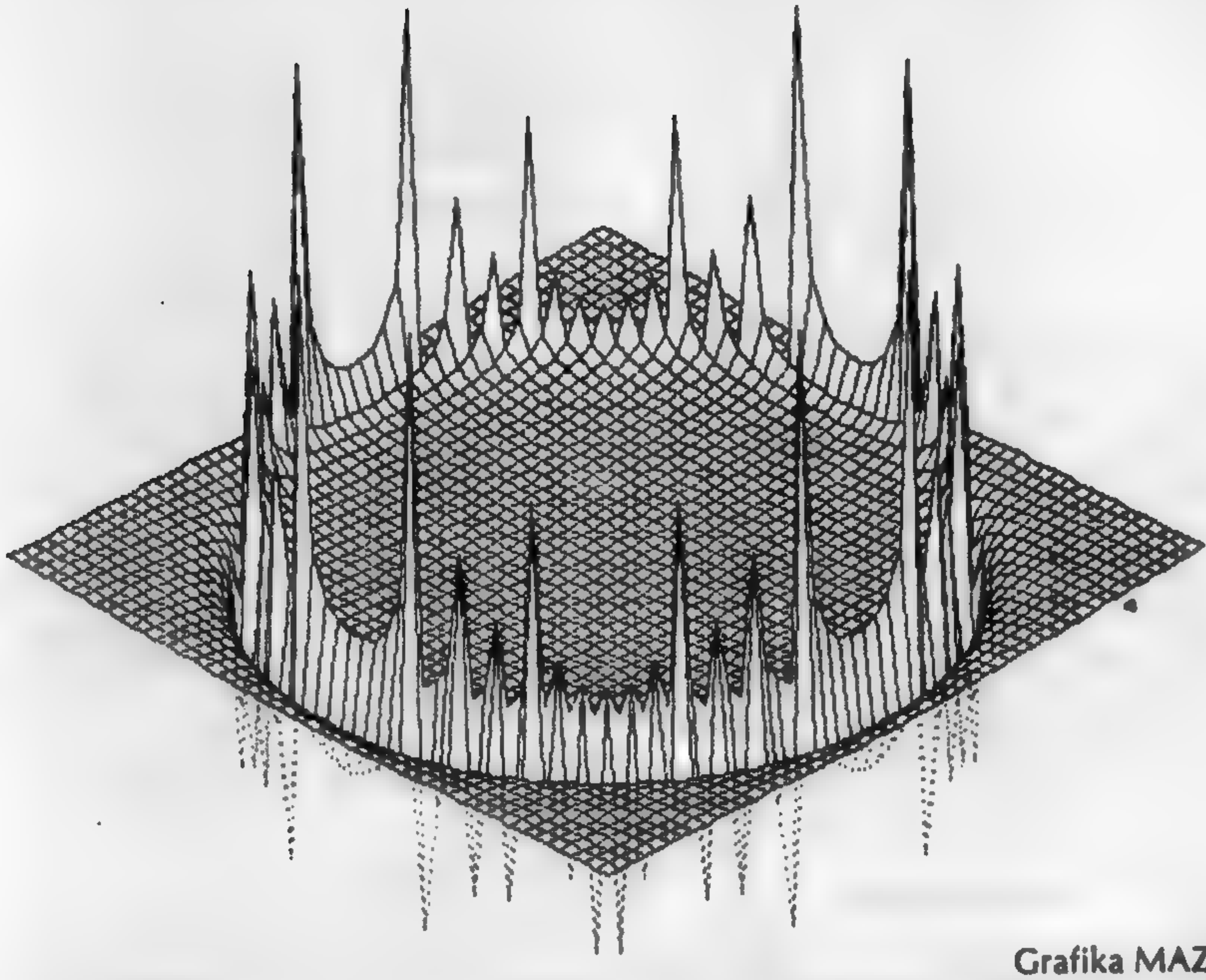
„ciężka"	gdy $x > 20$ kg
„średniociężka"	gdy $10 \text{ kg} \leq x \leq 20$ kg
„lekka"	gdy $x < 10$ kg

a) schemat blokowy





Rys. Michał Przybyłowski



Grafika MAZOVII

Oznaczenia:

k — numer kuli
x — waga kuli
z $— zmienna alfanumeryczna, której „wartością" jest odpowiedni komunikat (tekst)

b) program

```
  10 REM Przyklad 3.5
  20 PRINT "Klasyfikacja kul"
  25 PRINT '"Nr.kuli ";"Waga kul
i ";" Komunikat"
  30 INPUT "podaj nr kuli : ";k'
"waga kuli : ";x
  40 IF x>20 THEN LET z$="ciezka
": GO TO 70
  50 IF x<10 THEN LET z$="lekka".
: GO TO 70
  60 LET z$="sredniociezka"
  70 PRINT "    ";k;"        ";x;"
    ";z$
  80 INPUT "wprowadzamy dalej ?
tak/nie ";y$
  90 IF y$="tak" THEN GO TO 30
 100 IF y$<>"nie" THEN GO TO 80
 110 PRINT '"koniec programu"
 120 STOP
```

Wprowadźmy i wykonajmy ten program. Niżej prezentujemy otrzymane wyniki dla przykładowych danych:

```
Klasyfikacja kul

Nr.kuli Waga kuli  Komunikat
   1        2     lekka
   2       15      sredniociezka
   3       20      sredniociezka
   4       25      ciezka
   5       10      sredniociezka
   6        9     lekka

koniec programu
```

Analizę algorytmu programu pozostawiamy czytelnikowi, natomiast omówimy dwie instrukcje.

a) 40 IF x > 20 THEN LET z $ = „ciężka": GO TO 70
b) 50 IF x < 10 THEN LET z $ „lekka": GO TO 70

Po słowie THEN napisane są instrukcje:

a) LET z $ = „ciężka": GO TO 70
b) LET z $ = „lekka": GO TO 70

BASIC na ZX Spectrum dopuszcza pisanie kilku instrukcji w jednym wierszu oraz po słowie THEN. Instrukcje te oddzielamy od siebie znakiem : (dwukropek). Po wypisaniu znaku : ukazuje się nam kursor K.

Zadanie:

3.1. W czasie realizacji programu trzy osoby mają podać swoje imiona i liczbę lat. Należy podać, która z nich jest najstarsza. Na wynikach należy wypisać imię tej osoby i liczbę lat.

3.2. Napisać program, który dla dowolnej wartości zmiennej x wprowadzanej z urządzenia wejściowego, wyznacza i drukuje wartość y zdefiniowaną następującym wzorem:

$$y = \begin{cases} 2x + 5 & x \leq 0 \\ \sqrt{x} + 2\sin x & 0 < x \leq 2 \\ x^2 - 4x + 4 & x > 2 \end{cases}$$

Z.Z

3

LOGO

W lipcu ukazała się w sprzedaży kaseta magnetofonowa zawierająca translator polskiego Logo na mikrokomputer ZX Spectrum. Wersja języka Logo z tej kasety jest całkowicie zgodna z PTI Logo. Dodatkowo dołączono program, pozwalający zmienić działanie instrukcji ZAMALUJ na sprawniejsze. Przypominam, że strategia zamalowywania figur w polskim Logo jest następująca: z bieżącej pozycji żółwia prowadzone są kreski w górę i w dół do brzegów figury i zaznaczony w ten sposób obszar wypełnia się kolorem. Po wgraniu z kasety krótkiego programu następującego po translatorze, ZAMALUJ działa lepiej — figura o dowolnym kształcie jest całkowicie wypełniana kolorem. Malowanie wygląda tak, jak gdyby malarz szedł wewnątrz figury dookoła, trzymając się lewą ręką ściany, czyli już zamalowanego obszaru. Należy pamiętać, że sprawność działania ZAMALUJ została uzyskana kosztem pamięci. Gdy chcemy pisać większe programy w Logo, lepiej wrócić z powrotem do uboższej wersji ZAMALUJ, bo po prostu szybciej może zabraknąć pamięci.

Aby nauczyć się zapisywania tych, większych programów musimy dalej systematycznie śledzić tajniki języka Logo W poprzednim odcinku była mowa o tworzeniu procedur. Procedura rysująca kwadrat o boku wygląda tak:

OTO kwadrat 1
POWTÓRZ 4 (NAPRZÓD 20 LEWO 90)
JUŻ

Gdy chcemy narysować kwadrat o innej długości boku, możemy napisać inną procedurę, która w komendzie NAPRZÓD będzie miała podaną inną liczbę kroków. Ale można też zrobić to inaczej, napisać procedurę, która będzie rysowała kwadrat o boku dowolnej długości. Wystarczy dodać procedurze parametr. W nagłówku procedury umieszczamy symbol, który zastąpimy konkretną liczbą w chwili wywołania tej procedury. Pamiętajmy, aby zawsze nazwę parametru poprzedzić dwukropkiem dla odróżnienia od nazwy procedury

OTO kwadrat 2 : dł. boku
POWTÓRZ 4 (NAPRZÓD :dł. boku LEWO 90)
JUŻ

Gdy wywołamy kwadrat 20, narysowany będzie taki sam kwadrat, jak przy wywołaniu kwadrat 1. Ale możemy też wywołać kwadrat 2 z każdą inną liczbą np. 5 lub 31 i otrzymać rysunek mniejszego lub większego kwadratu. Procedura może mieć więcej parametrów niż 1. Na przykład prostokąt o dowolnych wymiarach można narysować, wywołując z odpowiednimi wartościami a i b taką procedurę.

OTO prostokąt :a :b
POWTÓRZ 2 (NAPRZÓD :a LEWO 90 NAPRZÓD :b LEWO 90)
JUŻ

Procedurę można wywołać nie tylko wtedy, gdy na ekranie jest znak ? Nazwę procedury można umieścić wewnątrz treści innej procedury. Na przykład rozetę z kwadratów można narysować tak:

OTO rozeta :dłboku :ilekw
POWTÓRZ :ilekw (kwadrat2 :dłboku LEWO 360/:ilekw)
JUŻ

Rozmiar rozety (kwadratów, z których się składa) i liczbę kwadratów określają parametry procedury.

W poprzednim odcinku podałem program rysujący koło o określonym promieniu. Korzystając z możliwości przekazywania procedurom parametrów, możemy zapisać program rysujący koło o dowolnym promieniu. Oto on:

```
OTO KOŁO :r
POWTÓRZ 36 [naprzód 2 * pi† * :
 / 36 lewo 10]
JUŻ
```

A jeśli chcemy, by środkiem okręgu był punkt, w którym w danej chwili znajduje się żółw, możemy to osiągnąć w ten sposób:

OTO koło1 :r
POD NAPRZÓD :r
LEWO 90 OPU
koło :r
POD PRAWO 90
WSTECZ :r OPU
JUŻ

Komendy zapisane po wywołaniu koło :r powodują, że żółw wraca do pozycji wyjściowej. Jest znów w tym samym punkcie i ustawiony pod tym samym kątem, co w momencie rozpoczęcia działania procedury koło1. Jest to dobry zwyczaj programistyczny Gdy zaczniemy pisać długie i naprawdę skomplikowane programy w Logo, przestrzeganie zasady wracania żółwiem do znanego punktu wyjścia będzie po prostu koniecznością. Jeśli zostawialibyśmy żółwia w poszczególnych procedurach w dowolnym punkcie ekranu zamiast spodziewanego końcowego rysunku otrzymalibyśmy plątaninę w niespodziewany sposób poprzesuwanych jego części

Z dotychczasowych uwag mógłby ktoś wyciągnąć wniosek, że wartością przekazywaną do procedury może być tylko liczba Tymczasem wartością parametru może być dowolny obiekt Logo np. słowo albo lista (pamiętamy, że lista to ciąg ujęty w nawiasy kwadratowe) Jak zwykle najlepiej jest poeksperymentować przy komputerze i samemu sprawdzić efekty różnych podstawień Spróbujmy na przykład tak

OTO kartka :tekst1 :tekst2
PISZ :tekst1
PISZ (Przesyłam serdeczne pozdrowienia)
PISZ :tekst2
JUŻ

i wywołanie:
kartka (Droga Jolu) (z Zakopanego)
Otrzymamy tekst.
Droga Jolu Przesyłam serdeczne pozdrowienia z Zakopanego
Posługując się tą samą procedurą, możemy pozdrowić szefa
kartka (Szanowny Panie) (z kursokonferencji w Serbinowie)
Wystarczy tylko tekst wypisać na drukarkę zamiast na ekran. Ale o tym, jak to się robi, będzie mowa dalej, w następnych odcinkach naszego kursu.

Napiszmy POLE i narysujmy kilka współśrodkowych okręgów, posługując się procedurą koło1 i zwiększając coraz bardziej promień — parametr procedury Po pewnym czasie rysowany okrąg będzie tak duży, że nie zmieści się na ekranie. Żółw przestanie rysować, a na dole ekranu pojawi się komunikat: „Żółw poza polem" (w wersji angielskiej „Turtle out of field"). Ekran to zamknięty, ogrodzony obszar, poza który żółw nie może wykraczać Możemy jednak zmienić tę właściwość ekranu i sprawić, by żółw nie zauważał jego brzegów Wystarczy podać komendę OKNO (ang. WINDOW). Teraz żółw porusza się dowolnie w przestrzeni. Może na przykład zakreślić okrąg o promieniu 1000, tyle że na ekranie komputera nie zobaczymy żadnego rysunku Ekran jest oknem, przez które oglądamy przestrzeń Jeśli żółw wejdzie w obszar widoczny z okna, będziemy widzieli, co rysuje i jak się przemieszcza Oprócz trybów ekran POLE i OKNO mamy do dyspozycji jeszcze jeden dający bodaj najciekawsze efekty graficzne tryb SKLEJ (ang WRAP). Po podaniu tej komendy żółw na ekranie zachowuje się tak, jak gdyby chodził po torusie. Dolny brzeg ekranu skleja się z górnym, a lewy z prawym. Żółw wychodzący poza ekran na górze pojawia się natychmiast w odpowiednim miejscu na dole ekranu i kontynuuje rysowanie, w razie potrzeby schodząc w dół poza ekran, aby znów pojawić się na górze Czytelnikom siedzącym przy komputerze (a jeszcze raz podkreślam, że jest to jedyny sensowny sposób nauki języka programowania) radzę napisać SKLEJ, a potem kilka lub lepiej kilkanaście razy wywołać procedurę koło1 od wartości r zwiększającej się kolejno na przykład o 10. Prawda, że na ekranie uzyskaliśmy ciekawy efekt? Warto poeksperymentować także z innymi figurami, choćby ze współśrodkowymi kwadratami o przekątnych leżących na osiach ekranu.

Wróćmy jeszcze raz do definiowania procedur. Wiemy już, jak nauczyć Logo rozumienia nowego słowa — nazwy nowej procedury. Ale co zrobić, gdy w trakcie wpisywania treści procedury pomylimy się i wprowadzimy błędną linię? Otóż teksty procedur można modyfikować. Do tego celu służy wbudowany w system Logo edytor. Do trybu redagowania tekstów przechodzi się po po-

daniu komendy RED (ang. EDIT i skrót ED). Jeśli chcemy poprawić treść jednej procedury należy napisać RED "nazwa, gdzie nazwa jest nazwą danej procedury. Podanie komendy RED bez nazwy spowoduje, że dostępna będzie zawartość ostatniego edytora. Jest to wygodne, gdy wielokrotnie przekształcamy treść tego samego fragmentu programu. RED" lub RED [] tworzy pusty edytor. Wpisywanie tekstu następuje tak samo jak w normalnym trybie pracy, tyle że można na bieżąco korygować błędy. Można jednocześnie modyfikować treść kilku procedur. Należy wtedy ich nazwy ująć w nawiasy kwadratowe i wymienić po słowie RED . RED [nazwa1 nazwa2 .. nazwan]. Do poruszania się po ekranie służą w edytorze strzałki. W zwykłym Spectrum należy nacisnąć jednocześnie klawisz CAPS SHIFT i klawisz z odpowiednią strzałką. Spectrum + ma oddzielne przyciski ze strzałkami. Kasowanie liter następuje po wciśnięciu CAPS SHIFT i O lub w Spectrum + oddzielnego klawisza z napisem DELETE. Aby skasować całą bieżącą linię, trzeba przejść do trybu EXTEND MODE (wcisnąć naraz klawisze CAPS SHIFT i SYMBOL SHIFT) — literka w dolnym prawym rogu ekranu zmienia się na E. Wtedy trzeba przycisnąć klawisz Y. Powielenie ostatnio kasowanej lub wprowadzanej linii to naciśnięcie R w trybie EXTEND MODE. W trybie tym strzałki też mają swoje znaczenie: ← — skok kursorem na początek linii, → — skok na koniec linii, ↑ — skok na początek ekranu, ↓ — skok na koniec ekranu. Gdy poprawiany tekst jest długi, nie mieści się na ekranie. Jeśli chcemy znaleźć się na początku tekstu, musimy podać B w trybie EXTEND MODE, a gdy na końcu — E w tym samym trybie. Wprowadzone podczas pracy z edytorem zmiany są zapamiętywane po przejściu do EXTEND MODE i wciśnięciu C. W tym momencie Logo uznaje, że zaakceptowaliśmy ostateczną postać redagowanego tekstu, kończy pracę z edytorem i na ekranie w pierwszej kolumnie znów wyświetla się znak ?. Podczas pracy z edytorem możemy zrezygnować z zapisywania jakichkolwiek zmian, uznając że poprzednia postać procedury była jednak prawidłowa. Wyjście z edytora bez dokonywania zmian to naciśnięcie jednocześnie klawiszy CAPS SHIFT i SPACE lub pojedynczego klawisza BREAK w Spectrum +. Wszystkie te informacje o trybach, klawiszach i ich jednoczesnym naciskaniu mogą się wydać na pierwszy rzut oka nieco skomplikowane, ale jak zwykle najlepiej jest samemu wypróbować działanie edytora Logo, a po krótkim czasie wszystko stanie się jasne i oczywiste.

Czasem uznajemy, że jakaś procedura jest tak zła, że nie warto jej poprawić. Lepiej napisać od początku. Taką procedurę, podobnie jak każdą inną, która przestała już być potrzebna, należy usunąć z pamięci. Robi się to za pomocą komendy USUŃ (skrót US, ang. ERASE i skrót ER), po której trzeba napisać nazwę usuwanej procedury. Jeśli chcemy wyrzucić kilka procedur jednocześnie, ujmujemy ciąg ich nazw w nawiasy kwadratowe i umieszczamy całość po słowie USUŃ. Gdy trzeba pozbyć się wszystkich procedur z pamięci, podajemy komendę USWP — od słów: usuń wszystkie procedury. W angielskiej wersji Logo USWP to komenda ERPS. Natomiast komenda USW (usuń wszystko) czyści dokładnie całą pamięć Logo. Jej angielski odpowiednik to ERALL. O tym, co jeszcze oprócz treści procedur może znajdować się w pamięci, będzie mowa w następnych odcinkach.

Często się zdarza, że po dłuższej pracy nie pamiętamy dokładnie, co Logo przechowuje w swojej pamięci. Można to sprawdzić bez konieczności korzystania z edytora. Komenda POTP (ang. POTS) wyświetla na ekranie tytuły wszystkich procedur. Gdy chcemy obejrzeć treść którejś z nich podajemy komendę PO (ang. PO) a po niej nazwę procedury poprzedzoną cudzysłowem. Kilka procedur można wyświetlić wymieniając po PO kilka nazw ujętych w nawiasy kwadratowe. Treść wszystkich procedur jest wyświetlana na żądanie POWP (ang. POPS), a zawartość całej pamięci Logo — po podaniu komendy POW (ang. POALL).

Prawidłową gospodarkę pamięcią interpretera Logo ułatwiają jeszcze dwa słowa kluczowe. Możemy zapytać Logo, ile ma jeszcze wolnej pamięci. Piszemy PISZ WOLNE (ang. NODES), a w odpowiedzi wyświetla się liczba wolnych jednostek pamięci. Logo w czasie pracy nie zawsze oszczędnie gospodaruje swoją pamięcią. Zdarza się, że po wykonaniu wielu operacji są w pamięci nie zajęte, ale i niedostępne jednostki. Aby uporządkować pamięć i odzyskać nieużytki, trzeba wydać rozkaz ODŚMIEĆ (ang. RECYCLE). Jeśli teraz znów sprawdzimy, ile jest dostępnej pamięci, pisząc PISZ WOLNE okaże się, że jej przybyło.

Do podanego miesiąc temu słowniczka poznanych komend polskiego Logo dołączamy dziś następne:

POLE	**FENCE**	ekran graficzny to zamknięte pole
OKNO	**WINDOW**	ekran to okienko w przestrzeni
SKLEJ	**WRAP**	sklej ekran graficzny w torus
RED	**EDIT,ED**	przejście do redagowania
USUŃ,US	**ERASE,ER**	usuń procedurę z pamięci
USWP	**ERPS**	usuń wszystkie procedury
USW	**ERALL**	usuń wszystko
POTP	**POTS**	pokaż tytuły procedur
PO	**PO**	pokaż treść danych procedur
POWP	**POPS**	pokaż wszystkie procedury
POW	**POALL**	pokaż wszystko
WOLNE	**NODES**	liczba wolnych jednostek pamięci
ODŚMIEĆ	**RECYCLE**	uporządkuj pamięć

Zadania do samodzielnego zaprogramowania:

1) Napisać procedurę rysującą dowolny wielokąt foremny.
2) Napisać procedurę obliczającą i wyświetlającą na ekranie obwód i pole okręgu o dowolnym promieniu. Zrobić to samo dla kuli — obliczyć jej powierzchnię i objętość. Napisać słowami na ekranie, co która wartość oznacza.

Za miesiąc między innymi o tym, co to jest rekursja i jak się nią sprawnie posługiwać w Logo.

SIS

Rys. Michał Przybyłowski



PROGRAM 26

AMSTRAD SCHNEIDER CPC—6128

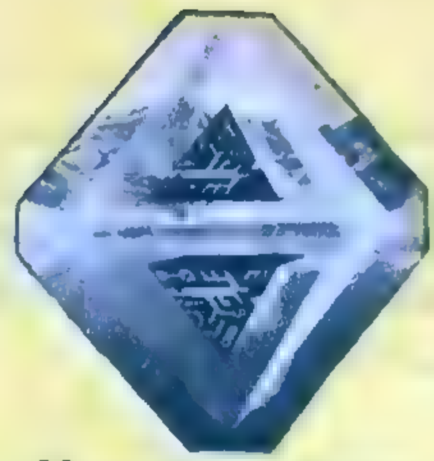
Jest to prosta, kolorowa gra zręcznościowa, polegająca na uzyskaniu jak największej liczby punktów poprzez odbijanie „piłki", rozkruszającej mur ułożony z klocków. „Piłkę" odbijamy poruszając dwoma klawiszami kursora (klawisze oznaczone strzałką w lewo i strzałką w prawo).

Czy uda się Wam uzyskać ponad 1000 punktów?

Życzymy powodzenia.

```
40 MODE 1:BORDER 1:INK 0,1:INK 1,26:INK 2,24:INK 3,6
50 SPEED KEY 15,2
60 ENV 1,1,18,0,11,0,10
70 ENT 1,10,2,2
80 ENV 3,1,0,16,5,-3,2
90 ENV 2,5,3,3,1,-21,22,9,-3,2
100 ENT -2,10,2,2,5,-7,1,2,11,3,2,-4,8
130 MOVE 30,32:DRAWR 0,400,1:MOVE 610,32:DRAWR 0,400,1
140 PEN 3:LOCATE 3,1:PRINT STRING$(36,143)
150 PEN 2:LOCATE 3,2:PRINT STRING$(36,143)
160 PEN 1:FOR r=5 TO 6:LOCATE 3,r:PRINT STRING$(36,143):NEXT r
170 bx=9
180 lives=5:score=0
190 PEN 1:GOSUB 680:CLEAR INPUT
200 IF INKEY$<>CHR$(32) AND JOY(0)<16 THEN 200
210 LOCATE 11,23:PRINT SPACE$(20);LOCATE 1,24:PRINT SPACE$(40);
220 GOSUB 690:GOSUB 660:GOTO 280
250 LOCATE bx,24:PRINT"  ";STRING$(4,131);"  ":RETURN
280 xa=1:ya=1:IF INT(RND*2)=1 THEN xa=-xa
290 PEN 1:GOSUB 250
300 ORIGIN 0,400
310 x=bx+4:y=11:x1=x:y1=y
340 x1=x+xa:y1=y+ya
350 IF x1=3 OR x1=38 THEN xa=-xa
360 GOSUB 540
370 IF y1=24 AND x1>bx+1 AND x1<bx+6 THEN ya=-ya:y1=y1-2:SOUND 130,44,8,7,1,1:a=
((x>bx+5)OR(x<bx+2)):IF a=-1 THEN xa=xa*a:x1=x1+xa:y
1=y1+1
380 IF y1=25 THEN LOCATE x,y:PRINT" ":GOTO 500
390 GOSUB 250
400 t=TEST((16*x1)-1,-(16*y1)-1)
410 IF t<>0 THEN ya=-ya:xz=x1:yz=y1:y1=y1+ya:GOSUB 590:IF t=2 THEN score=score+1
0:GOSUB 660
420 IF t=3 THEN score=score+20:GOSUB 660
430 IF t=1 THEN score=score+5:GOSUB 660
440 IF y1=1 THEN ya=1
450 LOCATE x,y:PRINT " ":LOCATE x1,y1:PRINT CHR$(233):x=x1:y=y1
460 IF y=1 OR x=3 OR x=38 THEN SOUND 129,78,8,7,1,1
470 GOTO 340
500 lives=lives-1:SOUND 132,19,46,12,2,2:IF lives=0 THEN GOTO 620
510 GOSUB 660:GOTO 280
540 IF (INKEY(8)=0 OR INKEY(74)=0) AND bx>2 THEN bx=bx-2:RETURN
550 IF (INKEY(1)=0 OR INKEY(75)=0) AND bx<=32 THEN bx=bx+2:RETURN
560 RETURN
590 LOCATE xz,yz:PRINT " ":RETURN
620 IF score>=hiscore THEN hiscore=score
630 GOSUB 660:score=0:lives=5:GOTO 130
660 SOUND 130,0,20,13,3,0,31:LOCATE 1,25:PRINT TAB(4)"REKORD";hiscore;
670 LOCATE 18,25:PRINT"PUNKTY";score:LOCATE 30,25:PRINT"NR GRY";lives:RETURN
680 LOCATE 11,23:PRINT" START to SPACJA":RETURN
690 LOCATE 1,25:PRINT SPACE$(40);:RETURN
```

PAMIĘCI OPTYCZNE



Współczesne systemy komputerowe wymagają coraz większych pamięci o dużej pojemności i krótkim czasie dostępu. Na przykład satelita wyposażony w teleskop przekazuje w ciągu roku kilka terabajtów informacji. Dla jej zmagazynowania potrzeba około 100 000 taśm magnetycznych o pojemności 150 megabajtów każda. Stacje kosmiczne, które będą budowane w latach dziewięćdziesiątych, będą przekazywać dziennie 1 terabajt informacji z szybkością większą niż 1 gigabit na sekundę. Istniejące obecnie urządzenia pamięciowe są niewystarczające dla gromadzenia tak wielu informacji. Nową nadzieją, która pozwoli przełamać istniejące bariery technologiczne, są pamięci optyczne. Ich pojemność przekracza 1 terabajt, a szybkość dostępu jest większa niż 1 gigabit na sekundę.

„Grająca szafa" dysków optycznych

Urządzenie to, skonstruowane na wzór tradycyjnej „szafy grającej", zawiera zestaw dysków optycznych, umieszczonych w oddzielnych kasetach, z których są wybierane za pomocą precyzyjnych mechanizmów, dokonujących zapisu lub odczytu promieniem laserowym. Pojemność pamięci tego urządzenia wynosi 10^{13} bitów (1250 gigabajtów), a dostęp do dowolnej informacji można osiągnąć w przeciągu 6 sekund. Dyski optyczne pozwalają na jednokrotne zapisanie informacji i wielokrotne jej odczytanie. Stosowane są również dyski magneto-optyczne, umożliwiające wielokrotny zapis informacji. Dyski te pokryte są magnetoczułą warstwą, na której zapisuje promień lasera w obecności pola magnetycznego. Naświetlony punkt zmienia swoją strukturę powierzchniową — kąt odbicia promienia laserowego jest w takich miejscach inny. Pozwala to jednoznacznie odczytać zapisaną informację. Zapis kasuje się przez naświetlanie laserem miejsc o zmienionej strukturze powierzchniowej w obecności odwrotnie niż poprzednio skierowanego pola magnetycznego. Pamięci tego typu są stosowane w amerykańskiej Krajowej Agencji Badania Przestrzeni Kosmicznej (NASA).

Pamięć buforowa na dyskach optycznych

Pamięć ta charakteryzuje się dużą pojemnością (większą niż 1 terabajt), dużą szybkością zapisu i odczytywania informacji (większą niż 1.6 gigabita na sekundę) oraz krótkim czasem dostępu do dowolnej informacji (mniejszym niż 100 milisekund. Urządzenie to składa się z 12 magneto-optycznych dysków, umieszczonych na wspólnym wałku. Każdy dysk obsługiwany jest przez elektrooptyczny moduł laserowy. W porównaniu z najnowszymi pamięciami buforowymi wykorzystującymi dyski magnetyczne, pamięć ta charakteryzuje się 60-krotnie większą szybkością dostępu do danych i 25-krotnie większą pojemnością!

Magneto-optyczny rejestrator taśmowy

W urządzeniu tym do zapisu informacji zastosowana jest taśma o właściwościach identycznych jak dysk magneto-optyczny. Rejestrator składa się z 16 magneto-optycznych modułów, z których każdy zapisuje jednocześnie 8 kanałów na taśmie. Pojemność pamięci wynosi 1250 gigabajtów, a szybkość dostępu do danych 1.6 gigabita na sekundę.

Pamięci optyczne rozpoczynają dopiero swoją karierę. Ich podstawową wadą są duże gabaryty i precyzyjna i zarazem wrażliwa mechanika, co ogranicza możliwości ich stosowania w urządzeniach mobilnych. Oczekuje się, że prawdziwy rozkwit tego rodzaju pamięci nastąpi w latach dziewięćdziesiątych, kiedy będą wykorzystywane nie tylko w badaniach kosmicznych, jak obecnie, ale również w innych dziedzinach nauki i techniki.

W.O.

Chochlik zamieszał...

Dokończenie programu nr 18 (Commodore) z „IKS-a" nr
Przepraszamy.

```
2110 GETZ$
2120 IF(Z$>="0"ANDZ$<="9")ORZ$="."ORZ$="
-"ORZ$="+"ORZ$="E"THEN2150
2130 IFZ$=CHR$(13)THEN2200
2140 IFZ$=CHR$(20)THEN2250
2145 GOTO2110
2150 IFC>MTHEN2110
2151 IFZ$="E"THENPOKEC,5:GOTO2155
2152 POKEC,ASC(Z$)
2155 L$=L$+Z$:C=C+1:GOTO2110
2200 L=VAL(L$):RETURN
2250 IFC=MOTHEN2110
2255 C=C-1:L$=LEFT$(L$,C-MO):POKEC,32:GO
TO2110
2300 FORL=PTOKSTEP40:FORLL=0TO24:POKEL+L
L,32:NEXT:NEXT:RETURN
```

Liga Myślących

Zadanie nr 1

Dwóch robotników wykopało nakazany rów melioracyjny w ciągu 12 godzin. Gdyby pierwszy wykonał sam połowę pracy, a następnie drugi resztę, to zużyłby na to 25 godzin. W ciągu ilu godzin każdy robotnik, pracując oddzielnie może wykopać rów melioracyjny?

Zadanie nr 2

Według legendy na płycie Diofantosa był taki napis ułożony przez Eutropiusza: „Przechodniu! Pod tym kamieniem spoczywają prochy Diofantosa, który umarł w głębokiej starości. Przez szóstą część swego życia był dzieckiem, przez dwunastą część — młodzieńcem. Następnie upłynęła siódma część jego życia, zanim się ożenił. W pięć lat po zawarciu związku małżeńskiego urodził mu się syn, który żył dwa razy krócej od niego. W cztery lata po śmierci swego syna Diofantos opłakiwany przez swych najbliższych, zasnął snem wiecznym. Powiedz, jeśli umiesz obliczyć, ile on miał lat, kiedy zmarł".

Proszę podać rozwiązanie tego zadania.

Rozwiązania zadań prosimy przesyłać do redakcji do końca listopada br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Odpowiedzi na zadania zamieszczone w trzecim numerze „IKS-a".

Zadanie 1

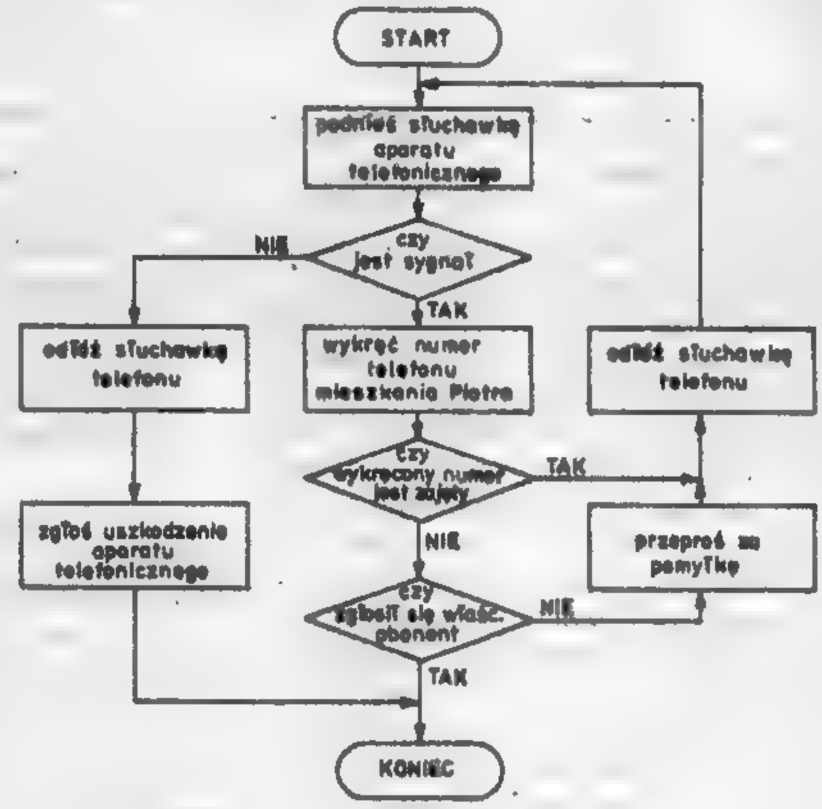
Z chwilą gdy jeden z autobusów zaczął się w godzinach szczytu z powodu przypadkowych zakłóceń opóźniać w stosunku do rozkładu jazdy, zwiększa się liczba pasażerów wsiadających do autobusu, a co za tym idzie zwiększa się również stopień napełnienia autobusu. Zwiększa się także liczba pasażerów wysiadających z autobusu. Im bardziej jest opóźniony autobus tym więcej pasażerów spotyka się na każdym przystanku. Z powodu dużej liczby pasażerów wsiadanie zajmuje więcej czasu, co powoduje dalsze opóźnienie autobusu. Z kolei następny wyjeżdżający na trasę autobus jedzie szybciej bo między innymi na przystankach oczekuje mniej ludzi. W końcu dochodzi do połączenia jazdy autobusów i mamy do czynienia z tzw. jazdą autobusów parami.

Zadanie 2

Do prawidłowego sporządzenia administracyjnej mapy Polski tak, aby sąsiadujące ze sobą województwa miały inne kolory wystarczy użyć czterech barw. Tzw. zagadnienie czterech kolorów było jednym z najsłynniejszych zagadnień teorii grafów. Tezy, iż do pokolorowania dowolnego atlasu wystarczą cztery kolory nie mogło udowodnić wielu wybitnych matematyków świata. Dokonano wielu ważnych odkryć w teorii grafów, ponieważ wydawało się, że będą pomocne przy rozwiązaniu zagadnienia czterech barw. Do 1976 roku hipoteza o czterech kolorach była na pewno najsłynniejszym nierozwiązanym zagadnieniem w teorii grafów. W 1976 roku hipoteza ta została udowodniona przez Kennetha Appel'a i Wolfgana Hakon'a. Przedstawili oni niekonwencjonalny dowód, wykorzystując szybki komputer.

Zadanie 3

Czynności Andrzeja przy próbie telefonowania do Piotra przy założeniu, że Piotr jet w domu w kolejności ich wykonania zostały przedstawione poniżej w postaci tzw. schematu blokowego. Inaczej mówiąc, poniższy rysunek przedstawia algorytm próby telefonowania Andrzeja do Piotra:



Zadanie 4

Przystępując do rozwiązania tego zadania należy założyć, że pociąg mija zawiadowcę i peron ze stałą prędkością v. W ciągu 26 sekund pociąg ze stałą prędkością v mija peron, pokonując drogę równą długości peronu oraz drogę równą swojej długości x, stąd:

$$v = \frac{120 + x}{26}$$

Z tą samą prędkością v pociąg mija zawiadowcę pokonując drogę równą swojej długości x w ciągu 11 sekund, stąd

$$v = \frac{x}{11}$$

zatem

$$\frac{120 + x}{26} = \frac{x}{11}$$

$$x = 88 \text{ m}$$

Długość pociągu wynosi 88 metrów

Zadanie 5

Aby rozstrzygnąć, która z dwunastu drużyn, grając każda z każdą jest najlepsza, należy rozegrać 66 meczy. Wielkość tę można obliczyć wg wzoru

$$\frac{n(n-1)}{2}$$

gdzie n — ilość drużyn biorąca udział w turnieju.

ODPOWIEDZI NA ZADANIA nr 5, 6 z nr. 2 „IKS-a"

Zadanie 5

Różnica między godziną trzecią i szóstą wynosi 180 minut. Ilość minut pozostającą do godziny szóstej można znaleźć, jeśli 180 — 50 = 130 minut podzielimy na takie dwie części, z których jedna jest cztery razy większa od drugiej. To znaczy znajdujemy piątą część liczby 130. Tą liczbą jest 26. A zatem do odejścia pociągu pozostało jeszcze 26 minut. Sprawdźmy powyższe rozumowanie. Przed 50 minutami pozostawało do odjeżdżającego o godzinie szóstej pociągu 26 + 50 = 76 minut czyli licząc od godziny trzeciej, wynosiło to 180 — 76 = 104 minuty po trzeciej. Czas 104 minut jest rzeczywiście czterokrotnie dłuższy od 26 minut.

dokończenie na str. 31

Pocztowa giełda

Rozwiązanie krzyżówki nr 3

Hasło brzmi: „KOMPUTER TWOIM PRZYJACIELEM". Bony pieniężne (1000 zł) wylosowali: Marian Misiaszek — Toruń, Marian Persona — Będzin, Stanisława Sroka — Chrzanów, Tomasz Bednarek — Dęblin, Witold Szydłowski — Sochaczew.

Nagrody książkowe otrzymują; Zdzisław Tarczyński Legnica, Tomasz Wiatr Poznań, Dariusz Sierzkiewicz Siedlce, Robert Gałuszka Jaworzno, Mariusz Petelicki Chrzanów, Ryszard Puskarz Kraków, Antoni Goryl Kraków, Stanisław Cyganik Duszniki-Zdrój, Andrzej Boczek Police, Tomasz Rostecki Leszno.

Przypominamy naszym Czytelnikom, że w losowaniu biorą udział wyłącznie odpowiedzi przysyłane na kartach pocztowych.

Warszawskie notowania „komputerowe"

Spectrum 48K — 70—80 tys.
Spectrum Plus — 110 tys.
Spectrum 128 K — 180 tys.

Commodore 64 + magnetofon + 2 joysticki — 280 tys.
Commodore 116 — 55—65 tys.

Atari 130 XE — 120 tys.
Atari 800 XL + magnetofon — 120 tys.

Amstrad 6128 + kolorowy monitor — 450 tys.
Amstrad CPC464 + kolorowy monitor — 320 tys.

Sharp 700 — 210 tys.

Timex — 95 tys.

Zadanie 6

Gdy zegar wybija godzinę trzecią czas trwania wybijania godzin składa się z dwóch okresów
pierwszego — pomiędzy pierwszym i drugim uderzeniem,
drugiego — pomiędzy drugim i trzecim uderzeniem
Oba okresy trwają 3 sekundy, tzn., że każdy z nich trwa po 1 1/2 sekundy. Gdy zegar wybił godzinę czternastą, okres taki jest jeden i wynosi 1 1/2 sekundy
Wszystkie odpowiedzi oparte na powyższym rozumowaniu zostały przyjęte jako prawdziwe

Za zadania opublikowane w trzecim numerze „IKS-a" można było uzyskać następujące ilości punktów
Zadanie nr 1 — 2 pkt
Zadanie nr 2 — 1 pkt
Zadanie nr 3 — 1 pkt
Zadanie nr 4 — 1 pkt
Zadanie nr 5 — 1 pkt

Oto nazwiska czytelników, którzy wylosowali książki za poprawne rozwiązanie zadań zamieszczonych w pierwszym numerze „IKS-a": **Jarosław Lubczyński** z Łodzi, **Ania Jałczak** z Ozorkowa, **Beata Więcek** z Chorzowa, **Piotr Łabuda** z Bochni, **Joachim Lach** z Nowej Rudy

A oto nazwiska czytelników, którzy wylosowali książki za prawidłowe rozwiązanie zadań zamieszczonych w drugim numerze „IKS-a": **Ryszard Wilkocki** z Łodzi, **Jacek Oryga** z Bytomia, **Marian Misiaczek** z Torunia, **Antoni Goryl** z Krakowa, **Hubert Nowaczyk** z Grudziądza.

Za poprawne rozwiązanie zadań umieszczonych w trzecim numerze „IKS-a" książki otrzymują **Leopold Gulcz** z Rumii, **Andrzej Wrześniowski** z Krakowa, **Józef Kurowski** z Tykocina, **Weronika Kowalska** z Sosnowca, **Agnieszka Para** z Płocka.

W naszym komputerlandzie

Wracając z zasłużonego wakacyjno-urlopowego wypoczynku pracowity robot Spektruś przeżył w podróży nie lada emocje. Widząc na peronie tłumy podróżnych, Spektruś ustawił się czujnie. Kiedy podjechał pociąg, wskoczył przez okno, otrzepał się ze szkła, bo okno było zamknięte, i zaczął ładować do środka, co mu podawano: walizkę, psa, starą komputerzycę Odrę, robocik z komputerkiem, tobół, narty, skrzynię, wózek z niemowlęciem.. Było tego niemało

Pociąg wyjechał z trzygodzinnym opóźnieniem i w złym kierunku, bo dzielorób nim kierujący nie rozróżniał stron komputerlandu. Kiedy wśród wrzasku, przekleństw, śpiewów, wycia i modlitw wszyscy ułożyli się w czterech warstwach, staruszka Odra zaczęła gaworzyć, że ona tylko odprowadzała i niechętnie jedzie. Ponadto w tłoku zamieniono jej przewodnik na półprzewodnik, a ona nie lubi roboty wykonywanej połowicznie. Zaklinowano ją wreszcie między półki i jakoś ucichła

Mały robocik, bawiąc się komputerkiem, zadawał mu pytanie co oznacza słowo, którego najczęściej używano w przedziale, bo krótkie i warczące — chciałby spopularyzować wśród kolegów. Na szczęście, ktoś mu przydeptał to urządzenie i się nie dowiedział.

Pociąg nie zatrzymywał się na żadnej stacji, bo prowadzący go dzielorób włączył całą parę i się zdrzemnął. W końcu dojechał na miejsce z częścią odważnych pasażerów. Nikt nie wnosił pretensji, bo wprawdzie z kilkudniowym opóźnieniem, ale przybył o właściwej godzinie.

Zresztą, jak się komuś bardzo spieszyło, wyskakiwał w biegu i pogotowiem ratunkowym dojeżdżał na miejsce raz dwa

Sporadyczne wnioski o zwrot kosztów podróży odrzucono, bowiem po pierwsze bilety były już przeterminowane, a po drugie opóźnienie wynikło z przyczyn obiektywnych

Zmęczony, ale uszczęśliwiony pomyślnym zakończeniem podróży Spektruś wrócił do pracy...

Podglądał: Eugeniusz MLECZAK

Lista „przebojów"

Tym razem bez większych zmian. Na czele nadal **„Łamigłówka"**. Na drugim miejscu zdecydowanie **„Odwracanka"** — oba programy na Atari. Czytelnicy domagają się podania tych programów w innych wersjach komputerowych. Pokrzywdzeni czują się przede wszystkim posiadacze Spectrum — nie wszyscy potrafią dokonać niezbędnych przeróbek. Na trzecim miejscu nadal utrzymuje się program na C-64 — **„Projektant — znaki"**

Uwaga! Przypominamy, że na łamach sobotnio-niedzielnego wydania „Żołnierza Wolności" — Horyzont drukujemy krótkie programy na różne mikrokomputery. Czekamy na Wasze propozycje

Ogłoszenia

KRZYŻÓWKA nr 5



POZIOMO: 1. stała wartość, wielkość 6. alotropowa odmiana tlenu 7. bakteria, drobnoustrój 8. artretyzm 9. stolica państwa europejskiego 10. inwencja twórcza 14. wprowadza dane do komputera 16. drzewo iglaste 17. pierwszy polski król 18. poprawia jakość zdjęcia 20. organizacja walcząca z narkomanią 23. spisek 26. defensywa 27. malec

PIONOWO: 1. elektryczny lub rzeczny 2. Rzym dla Włocha 3. układ scalony o wielkiej skali integracji 4. maszyna zastępująca człowieka 5. specjalista od przetwarzania danych 11. popularny minikomputer 12. pas ortopedyczny 13. człowiek, postać 15. zabrzańskie zakłady produkujące komputery 19. podłużne zagłębienie w ziemi 21. harcerskie kolonie 22. galaretka z morskich wodorostów 24. zawiera nasiona 25. kamień ozdobny 26. wielka rzeka syberyjska

Hasło: (1k, 3e, 6a) (11d, 9e, 2i, 9k) (16f, 10e, 13i, 7h) (1a, 8i, 13a, 10g, 14k, 16e)

Rozwiązania (tylko hasło) prosimy przesyłać pod adresem redakcji do końca listopada. Wśród Czytelników, którzy nadeślą prawidłowe rozwiązania rozlosujemy bony pieniężne i nagrody książkowe.



OSBORNE 3 jest średnich rozmiarów komputerem przenośnym w pełni kompatybilnym z komputerem IBM. Produkowany jest przez firmę Osborne Computer Co., a o jego konkurencyjności w stosunku do innych komputerów tej klasy decyduje stosunkowo niska cena — 1895 dolarów. Pracuje w oparciu o 16-bitowy procesor 80C86, pamięć 256 KB RAM z możliwością rozszerzenia do 512 KB przy zastosowaniu układu 256 KB DRAMs. Pojemność pamięci dyskowej wynosi 2 x 360 KB przy wykorzystaniu 2 stacji 5,25-calowych dysków elastycznych. Osborne 3 wyposażony jest w ekran ciekłokrystaliczny z możliwością wyświetlania 80 znaków w 16 wierszach.



„IKS" — dodatek „Żołnierza Wolności". Redagują Wiesław Cetera; Ryszard Rogoń. Adres redakcji: 00-950 Warszawa ul Grzybowska 77, telefon centrali 20-12-61 w. 486. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8059. Nr ind. 38280 P-93

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 6/1986 ISSN 0860-2794



Uważni Czytelnicy z pewnością dostrzegają zachodzące na łamach naszego pisma zmiany; coraz więcej miejsca poświęcamy na prezentowanie programów na wszystkie najpopularniejsze w Polsce mikrokomputery. Staramy się, aby w większości były to programy użytkowe — takie, które mogą przydać się na co dzień. Chcemy, aby nasze publikacje zaspokajały nie tylko „pierwszy głód" informatycznej wiedzy, ale i satysfakcjonowały bardziej wymagających sympatyków mikrokomputerów.

Chroniczny brak literatury fachowej sprawia, iż użytkownicy Atari czy Commodore skazani są niemal na obcojęzyczne instrukcje. Dlatego też prezentujemy ciekawsze rozwiązanie programowe dotyczące poszczególnych typów mikrokomputerów.

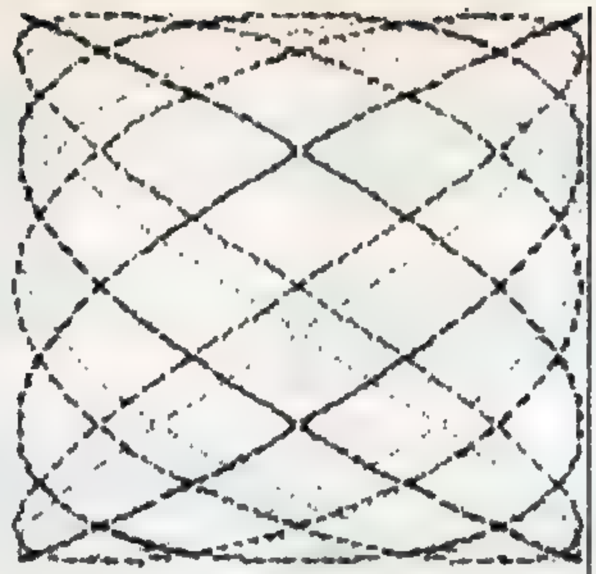
Liczymy oczywiście zawsze na pomoc Czytelników. Dobrych pomysłów z pewnością Wam nie brakuje. Nie brakowało ich nigdy — gorzej było z ich rozpowszechnianiem. Wiedza jest towarem najcenniejszym — szukajmy zatem talentów potrafiących robić właściwy użytek ze swych szarych komórek.

PROGRAM 87 — to hasło konkursu ogłoszonego przez Ogólnopolską Federację Klubów Komputerowych Młodych Mistrzów Techniki, któremu nasza redakcja wiernie sekunduje. Konkurs odbędzie się w trzech etapach: listopad — grudzień 1986, marzec — maj 1987 i wrzesień — listopad 1987 r. Programy oceniane będą w pięciu kategoriach: programy dydaktyczne, graficzne, gry komputerowe, programy sterujące robotami i automatami oraz programy rozwiązujące zagadnienia techniczne.

Federacja Klubów Komputerowych ostro wzięła się do roboty. Jeszcze w tym roku rozstrzygnięty będzie kolejny konkurs — tym razem na najlepiej pracujący klub komputerowy — a jest ich już ponad 200 (to tylko te, które należą do Federacji).

Komputery tanieją, ale jeszcze nieprędko osiągną odpowiednią cenę dla „popularnej pomocy szkolnej" — stąd też często właśnie klub komputerowy jest jedyną formą kontaktu z tym sprzętem. Być może problem zostanie rozwiązany, kiedy ruszy pełną parą produkcja elwrowskiego Juniora, ale tymczasem wiedza informatyczna młodzieży zależy często, na szczęście, od licznych zapaleńców, którzy uważają, że bez komputera nie ma przyszłości; skupiają wokół siebie podobnych pasjonatów, zdobywają środki i organizują klub komputerowy. Czasami o powodzeniu przedsięwzięcia decyduje przypadek. Tymczasem sukces powinien być regułą i do tego dążymy.

REDAKCJA

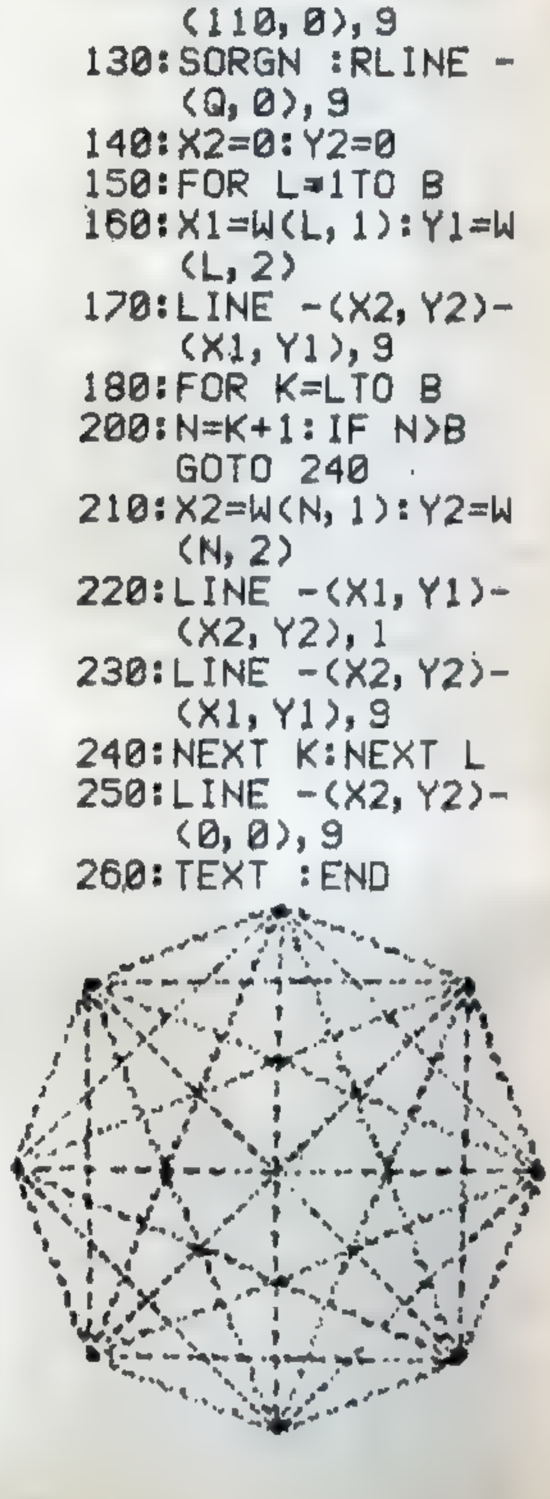


SHARP PC-1500
K R Z Y W E
LISSAJOUS

```
 10:BEEP 10,100,10
    0
 20:"A":CLEAR :
    USING :GRAPH :
    DEGREE :WAIT 0
 30:INPUT "POZIOM
    X=";X:X=ABS X:
    X=INT X
 40:INPUT "PION Y=
    ";Y:Y=ABS Y:Y=
    INT Y
 50:IF (X*Y)=0GOTO
    10
 60:INPUT "FAZA F=
    ";F
 70:LINE -(0,0)-(1
    00,0),9:SORGN
 80:R=75:T=0:X2=0:
    Y2=0
 90:N=36*(X+Y):S=3
    60/N
100:X2=R*SIN (X*T)
    :Y2=R*SIN (Y*(
    T+F))
110:LINE -(0,0)-(X
    2,Y2),9
130:FOR K=0TO N
140:T=T+S:X1=X2:Y1
    =Y2
150:X2=R*SIN (X*T)
    :Y2=R*SIN (Y*(
    T+F))
160:LINE -(X1,Y1)-
    (X2,Y2),1
170:NEXT K
180:LINE -(X2,Y2)-
    (0,0),9
190:TEXT :END
```

SHARP PC-1500
KRYSZTAL

```
 10:BEEP 10,100,10
    0
 20:"A":CLEAR :
    USING :WAIT 0
 30:INPUT "ILE BOK
    OW B=";B:B=ABS
    B:B=INT B
 40:IF B<4GOTO 10
 50:INPUT "WIELKOS
    C Q=";Q:Q=ABS
    Q:Q=INT Q
 60:IF Q=0GOTO 10
 70:A=2*π/B:DIM W(
    B,2)
 80:FOR K=0TO (B-1
    )
 90:M=K+1:W(M,1)=Q
    *COS (K*A)
100:W(M,2)=Q*SIN (
    K*A)
110:NEXT K
120:GRAPH :RLINE -
    (110,0),9
130:SORGN :RLINE -
    (Q,0),9
140:X2=0:Y2=0
150:FOR L=1TO B
160:X1=W(L,1):Y1=W
    (L,2)
170:LINE -(X2,Y2)-
    (X1,Y1),9
180:FOR K=LTO B
200:N=K+1:IF N>B
    GOTO 240
210:X2=W(N,1):Y2=W
    (N,2)
220:LINE -(X1,Y1)-
    (X2,Y2),1
230:LINE -(X2,Y2)-
    (X1,Y1),9
240:NEXT K:NEXT L
250:LINE -(X2,Y2)-
    (0,0),9
260:TEXT :END
```



Na początku był ENIAC ...

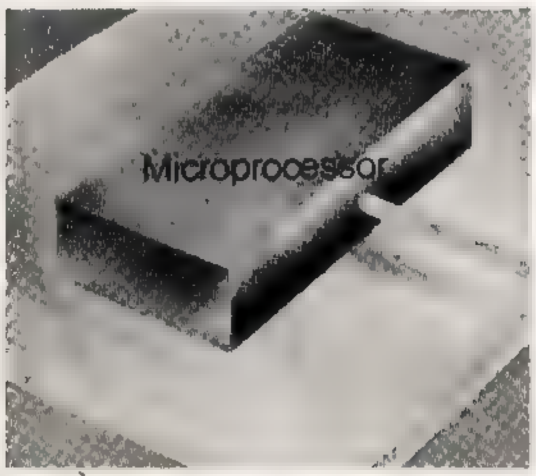
Wyprodukowanie pierwszego mikroprocesora stało się początkiem przewrotu, kojarzonego z drugą rewolucją przemysłową, zwaną rewolucją mikroprocesorową. Mimo dopiero piętnastoletniego istnienia, układ elektroniczny, nazwany w związku z technologią jego wykonania mikroprocesorem, nabrał historycznego znaczenia, znalazł wielostronne i niebanalne zastosowania.

Postęp w dziedzinie technologii elementów elektronicznych powodował kolejne kroki w rozwoju techniki komputerowej. Powstawały kolejne generacje komputerów. Oczywiście, warunkiem koniecznym ich rozwoju, poza doskonaleniem technologii, był rozwój koncepcji działania systemów obliczeniowych i oprogramowania.

Elementy elektroniczne wkroczyły do techniki obliczeniowej w 1946 r. Na Uniwersytecie Pensylwania skonstruowano wówczas pierwszy działający komputer ENIAC (Electronic Numerical Integrator and Computer). Ważył 30 ton (!), zużywał 150 kW energii elektrycznej (!), a zbudowany był z 18 000 lamp próżniowych i 1500 przekaźników elektromagnetycznych. Na wykonanie operacji dodawania dwóch liczb dziesięciocyfrowych ENIAC potrzebował 0,2 milisekundy (1 ms = 0,001 s), a na wykonanie operacji mnożenia 2,8 ms. Komputer posiadał: **procesor**, który tworzyły: **jednostka sterująca** i **arytmometr**; **pamięć wewnętrzną**, składającą się z 20 rejestrów mieszczących liczby dziesięciocyfrowe; **pamięć zewnętrzną,** na pliku kart dziurkowanych.

Komputer ENIAC działał w następujący sposób. Jednostka sterująca wprowadzała na wejście arytmometru daną liczbową, zapisaną w rejestrze pamięci lub na karcie dziurkowanej, a następnie wysterowywała arytmometr, co oznaczało wykonanie odpowiedniej operacji arytmetycznej. Wynik operacji wyprowadzany był do jednego z rejestrów. Kolejne działania procesora zaprogramowane były w tzw. tablicy punktów. Rozwiązanie problemu obliczeniowego oznaczało wprowadzenie właśnie tablicy punktów, których kolejność odpowiednich połączeń interpretowana była przez jednostkę sterującą oraz przygotowanie danych na kartach dziurkowanych.

Czasochłonność oraz niedogodność programowania oraz mała pojemność pamięci wewnętrznej, którymi to charakteryzował się ENIAC, „poprawione" zostały w koncepcji J. Eckerta i J. von Neumanna. Koncepcja ta ukazywała ogromną rolę **pamięci wewnętrznej (operacyjnej)** i jej wpływ na architekturę komputera, a polegała na tym, że jednostka sterująca mogła wykorzystywać, zapisane w tej samej pamięci, dane i **instrukcje programu (rozkazy)** — operujące na tych danych



Architektura komputera, zgodnego z tą koncepcją, zawierać miała procesor, tor przesyłania danych i pamięć. Procesor tworzyły arytmometr i jednostka sterująca, która zawierała m.in. licznik rozkazów i rejestr rozkazów. Pamięć operacyjna zawierała, oprócz komórek pamięci, rejestr adresowy pamięci.

Cykl pobrania rozkazu przez komputer o takiej architekturze wyglądał następująco. W chwili początkowej licznik rozkazów ma wartość zerową. Zawartość licznika przesyłana jest do rejestru adresowego pamięci. Po czasie, zwanym czasem dostępu do pamięci, zawartość komórki o adresie zerowym przesyłana jest do rejestru rozkazów. Rozkaz po pobraniu jest interpretowany przez procesor, a następnie wykonywany.

Rozkaz procesora składa się z **części operacyjnej**, wskazującej, jaką operację ma wykonać procesor oraz z **części adresowej**, wskazującej adres komórki pamięci (adresy komórek pamięci), w których, zawarte są argumenty tej operacji.

Do momentu sformułowania koncepcji Eckerta i von Neumanna do budowy komputerów wykorzystywano **próżniowe lampy elektronowe** z trzema elektrodami: katodą, anodą i siatką, lampy zwane triodami, o zdolnościach wzmacniających. Jako elementy logiki wykorzystywano **przekaźniki elektromagnetyczne**. Pamięć zewnętrzną stanowiły wspomniane wcześniej **karty dziurkowane (perforowane)**, na których informacje kodowane są za pomocą tzw. kodu Holleritha. Konieczne stało się konstrukcyjne rozwiązanie „dużej" pamięci operacyjnej, o pojemności kilku tysięcy komórek.

W komputerze EDVAC, uruchomionym w 1950 r., do budowy pamięci wykorzystano **rtęciowe linie opóźniające**. Odznaczały się dużą szybkością działania, ale małą pojemnością i były szkodliwe dla ludzi, zostały więc szybko wyparte.

Rozpoczęła się era panowania techniki komputerowej. Opracowywano nowe technologie i konstrukcje, poszukiwano nowych rozwiązań funkcjonalnych.

Powstały trzy podstawowe typy **pamięci elektromechanicznych:** pamięć bębnowa, pamięć taśmowa i pamięć dyskowa.

W **pamięci bębnowej** głowice magnetyczne zapisują (przez lokalne namagnesowanie) lub odczytują informacje z wirującego bębna, na który naniesiony jest materiał magnetyczny. Z uwagi na konieczność ustawienia głowic na odpowiedni obszar bębna pamięć bębnowa jest powolna, ale ma ważną zaletę — dużą pojemność. Czas dostępu do informacji zazwyczaj wynosi kilkadziesiąt ms. Pamięć umożliwia zapisanie na jednym bębnie do miliona bitów (bit — binary digit — najmniejsza jednostka do reprezentacji danych dwuwartościowych, czyli zer lub jedynek liczbowego systemu dwójkowego).

dokończenie na str. 4

dokończenie ze str. 3

Pamięci na taśmie magnetycznej stosowane są do dziś, ze względu na znaczną pojemność, mimo stosunkowo długiego czasu dostępu do potrzebnej informacji.

Pamięci dyskowe, wykorzystujące lokalny zapis magnetyczny, znalazły trwałe zastosowanie w komputerach, ze względu na dużą pojemność, rzędu miliardów bitów w konstrukcjach wielodyskowych. Czas dostępu do informacji wynosi kilkanaście ms.

Wymienione pamięci elektromechaniczne stosowane są jako masowe pamięci zewnętrzne

Bardzo duży wpływ na rozwój techniki komputerowej wywarły **rdzeniowe pamięci ferrytowe.** Charakteryzowały się one dużą szybkością odczytu, rzędu kilku mikrosekund (1 µs = 0,000001 s). Były one wykorzystywane jako pamięci wewnętrzne, wyparte zostały dopiero przez **pamięci półprzewodnikowe.**

Odkrycie w 1948 r. przez W. Shokleya, J. Bardeena i W. Brattaina nowego, półprzewodnikowego elementu elektronicznego, **tranzystora,** który zastąpił jedyny dotąd element wzmacniający — lampę elektronową, to początek ery tranzystorowej w elektronice i początek II generacji techniki komputerowej.

Zanim przejdziemy dalej, kilka zdań o innych konstrukcjach komputerowych I generacji.

W 1951 r. firma UNIVAC polecała swój komputer UNIVAC—1, zbudowany z 5000 lamp i z 112 rtęciowych linii opóźniających, mieszczących 1000 liczb dwunastocyfrowych. Wykonywał on 45 rozkazów z wyróżnioną częścią operacyjną i częścią adresową (trzycyfrową). Do komunikacji operatora z systemem po raz pierwszy wykorzystano elektryczną maszynę z klawiaturą, natomiast do zapamiętywania dużych zbiorów danych wykorzystywano taśmę magnetyczną. Operację dodawania dwóch liczb dwunastocyfrowych UNIVAC-1 wykonywał w 0,5 ms, operację mnożenia w 2,5 ms. Już w rok później w komputerze UNIVAC 1103 jako pamięć operacyjną zastosowano rdzenie magnetyczne. Zwiększyło to pięćdziesięciokrotnie szybkość UNIVAC 1103 w stosunku do UNIVAC-1.

W 1952 r. firma IBM weszła na rynek komputerowy z udaną (nie pierwszą) konstrukcją, komputerem 701. Zawierał on 4000 lamp i 12 000 diod germanowych. Jako pamięć operacyjną zastosowano 72 pamiętające lampy katodowe, o czasie dostępu 12 µs. W komputerze zastosowano kilka urządzeń wejścia — wyjścia systemu: czytnik i dziurkarkę kart, drukarkę o szybkości 150 wierszy/minutę, taśmę magnetyczną o gęstości zapisu 100 bitów/cal. IBM 701 wykonywał operacje na danych 36-bitowych, operacja dodania dwóch liczb trwała 62,5 µs, operacja mnożenia 500 µs. Kolejne modele firmy IBM 702, 704, 705 miały pamięci rdzeniowe. W następnych modelach RAMAC 305 i 650 jako masową pamięć zewnętrzną zastosowano pamięci dyskowe, panele złożone z 50 dysków. Na każdym z dysków znajdowało się 100 ścieżek, o pojemności 10 000 znaków każda. Pamięć wyróżniała się więc olbrzymią pojemnością.

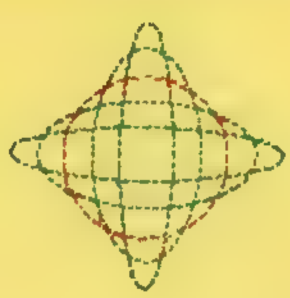
Przedstawiony wyżej etap rozwoju techniki komputerowej charakteryzował się pogonią za parametrami technicznymi. Niewiele uwagi poświęcono systemom operacyjnym i językom programowania, sterowanie pracą komputera oznaczało żmudne pisanie programów w języku procesora. Jako pierwsza tymi problemami zajęła się firma UNIVAC. Komputer UNIVAC 1103 wyposażony został w **interpreter**, program tłumaczący na język procesora kolejne wiersze programu. Po przetłumaczeniu danego wiersza procesor komputera wykonywał odpowiednie rozkazy, następnie czekał na przetłumaczenie kolejnego wiersza programu.

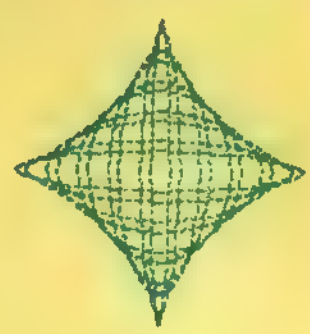
W firmie UNIVAC uruchomiono również program **translatora,** tłumaczącego na język procesora od razu cały program. Po przetłumaczeniu procesor wykonywał wszystkie rozkazy. Taka metoda nazywana jest **kompilacją**, a program **kompilatorem.**

Bardzo ważną sprawą jest organizacja pracy systemu komputerowego czyli współdziałania wszystkich elementów. Zbiór programów obejmujących te zagadnienia nazywa się **systemem operacyjnym.**

Kolejnym problemem są **języki programowania,** umożliwiające pisanie programów w języku zbliżonym do ludzkiego, możliwie dalekim od języka procesora. Dany język jest zazwyczaj przeznaczony do rozwiązywania problemów z pewnej dziedziny. Wśród języków programowania współcześnie „panujących" najważniejsze to: **ALGOL**, do opisu algorytmów obliczeń; **FORTRAN**, do zastosowań naukowo-technicznych; **COBOL,** do przetwarzania dużej liczby danych; **BASIC**, do rozwiązywania problemów naukowo-technicznych; **C,** do prac nad systemami operacyjnymi; **PASCAL,** do operacji na złożonych strukturach danych; **LOGO,** do nauki podstaw informatyki; **PROLOG,** do programowania logicznego i oprogramowania robotów.

JANUSZ MILLER
SHARP PC-1500
E L I P S Y





```
10:BEEP 10,100,100
20:"A":CLEAR :USING :RADIAN
30:INPUT "LICZBA ELIPS ? N=";N:N=
   ABS N:N=INT N:IF N<3GOTO 10
40:INPUT "A+B=M ? M=";M:M=ABS M:IF
   M=0OR M>90GOTO 10
50:F=π/20:D=M/(N+1):S=π/20
70:FOR K=1TO N
80:C=C+1:IF C>3LET C=C-3
90:A=M-K*D:B=K*D:GRAPH
100:COLOR C:LINE -(0,0)-(110,0),9:
    SORGN
110:LINE -(0,0)-(A,0),9:X2=A:Y2=0:T=
    0
120:FOR L=0TO 39
130:X1=X2:Y1=Y2:T=T+S
140:X2=A*COS T:Y2=B*SIN T
150:LINE -(X1,Y1)-(X2,Y2),1
160:NEXT L
170:LINE -(X2,Y2)-(0,0),9
180:TEXT :NEXT K
190:END
```

Wspomniane problemy oprogramowania większe odzwierciedlenie znalazły w komputerze UNIVAC III.

Opanowanie technologii tranzystorów, zwłaszcza tranzystorów bipolarnych, to podstawa konstrukcji komputerów II generacji. Tranzystory unipolarne, inaczej polowe, konstruowane na strukturze MOS (Metal — Oxide — Semiconductor), wykorzystywane były tylko w **układach peryferyjnych** systemów komputerowych. Technologia MOS nabrała znaczenia dopiero w latach siedemdziesiątych.

W latach sześćdziesiątych poczyniono kolejny krok, na jednym kryształku półprzewodnika umieszczono więcej niż jeden element. Zaczęto konstruować **układy scalone**, zawierające wiele elementów — tranzystorów, diod, rezystorów itp.

Układy scalone, w zależności od liczby elementów w strukturze półprzewodnika, można podzielić m.in. na: małej skali integracji SSI (Small Scale of Integration) — zawierające do 100 tranzystorów w swojej strukturze i średniej skali integracji MSI (Medium Scale of Integration) — zawierające do 1000 tranzystorów.

Układy scalone SSI i MSI posłużyły do konstrukcji komputerów III generacji. Wymienione układy scalone wykonywane były głównie technologiami wykorzystującymi tranzystory bipolarne. Z technologii tych największe znaczenie osiągnęła TTL (Transistor — Transistor Logic), która króluje na rynku elektronicznym do dziś.

W związku z wprowadzeniem **elementów półprzewodnikowych**, zapewniających wzrost szybkości działania i niezawodność, zmniejszenie poboru mocy oraz spadek kosztów wykonania oraz w związku z pracami nad oprogramowaniem, sprzęt komputerowy stał się bardziej dostępny, powszechniejszy. Bardzo popularnym był komputer IBM 1360 oraz najszybszy do 1979 r. komputer ILLIAC IV, wykonujący 200 mln rozkazów/sekundę(!).

Na rynku komputerowym zaczęły panować dwie tendencje. Pierwsza z nich to upowszechnienie szybkich i dużych komputerów. Druga to miniaturyzacja systemów komputerowych, przy zachowaniu parametrów technicznych dużych systemów.

W 1970 r. firma Digital Equipment Corporation wprowadziła PDP-11, komputer o bardzo oryginalnej architekturze. Wszystkie elementy tego komputera: procesor, pamięć i urządzenia wejścia-wyjścia, dołączone były do jednej wspólnej szyny.

W tym czasie opanowano technologie wytwarzania układów scalonych dużej skali integracji LSI (Large Scale of Integration), zawierających do 10 000 tranzystorów w jednej strukturze półprzewodnikowej. Rozpoczęła się era dominacji **technologii MOS**, zwłaszcza PMOS, NMOS, CMOS a ostatnio CHMOS. W 1971 r. zaprojektowano i wykonano pierwszy **mikroprocesor**. Rozpoczęła się IV generacja komputerów. Układy LSI to na pewno największe osiągnięcie elektroniki, od czasu wynalezienia tranzystora.

Lata osiemdziesiąte to nowe układy scalone, VLSI (Very Large Scale of Integration) o olbrzymich możliwościach. Przykładowo koprocesor arytmetyczny firmy MOTOROLA MC 68881, wykonany w technologii CHMOS, przeznaczony do współpracy z 32-bitowym mikroprocesorem M 68020, zawiera 155 000 tranzystorów w strukturze o wymiarach 27 na 33 mm.

Tak w skrócie przeszliśmy do komputerów nam współczesnych. Reasumując — komputer to: architektura i jej elementy składowe, czyli hardware; system operacyjny i języki programowania, interpretery i kompilatory, czyli software. W następnych numerach zajmiemy się tym pierwszym, czyli elementami elektronicznymi, z których zbudowane są komputery.

Jacek WOJTALA

PROGRAM 27

SPECTRUM

POSZERZANIE TABLICY ZMIENNYCH — Spectrum 48K

Jak 2-wymiarową tablicę zmiennych tekstowych nS, w której przechowujesz np. pewną liczbę nazwisk, powiększyć o kilka zmiennych? Program wczyta zmienną (linia 9900), ustali jej wymiary (linie 9902—9910), zapyta o liczbę dodatkowych nazwisk (linie 9912—9916), powiększy odpowiednio tablicę (linie 9918—9932) oraz nagra nową tablicę na taśmę (linia 9936). Aby przyspieszyć te operacje nie używamy pętli FOR—NEXT. Poniższą sekwencję możesz stosować jako część własnych, większych baz danych.

```
9900 LOAD "nazwiska" DATA n$()
9902 LET n$(1)=n$(1)
9904 RANDOMIZE FN p(23629)
9906 LET wymiar1= FN p( FN p(23670)-4)
9908 LET wymiar2= FN p( FN p(23670)-2)
9910 INPUT "ile nazwisk extra? ";ile
9912 IF ile<0 OR INT ile<>ile THEN GOTO
 9910
9914 IF NOT ile THEN STOP
9916 DIM r$(wymiar1,wymiar2)
9918 LET a=1
9920 IF a<=wymiar1 THEN LET r$(a)=n$(a)
: LET a=a+1: GOTO 9920
9922 DIM n$(wymiar1+ile,wymiar2)
9924 LET a=1
9926 IF a<=wymiar1 THEN LET n$(a)=r$(a)
: LET a=a+1: GOTO 9926
9928 LET a=1
9930 IF a<=ile THEN INPUT "nazwisko ";(
wymiar1+a), LINE n$(wymiar1+a): LET a=a
+1: GOTO 9930
9932 DIM r$(1)
9934 SAVE "nazwiska" DATA n$()
9940 DEF FN p(a)=PEEK a+256* PEEK (a+1)
```

PROGRAM 28

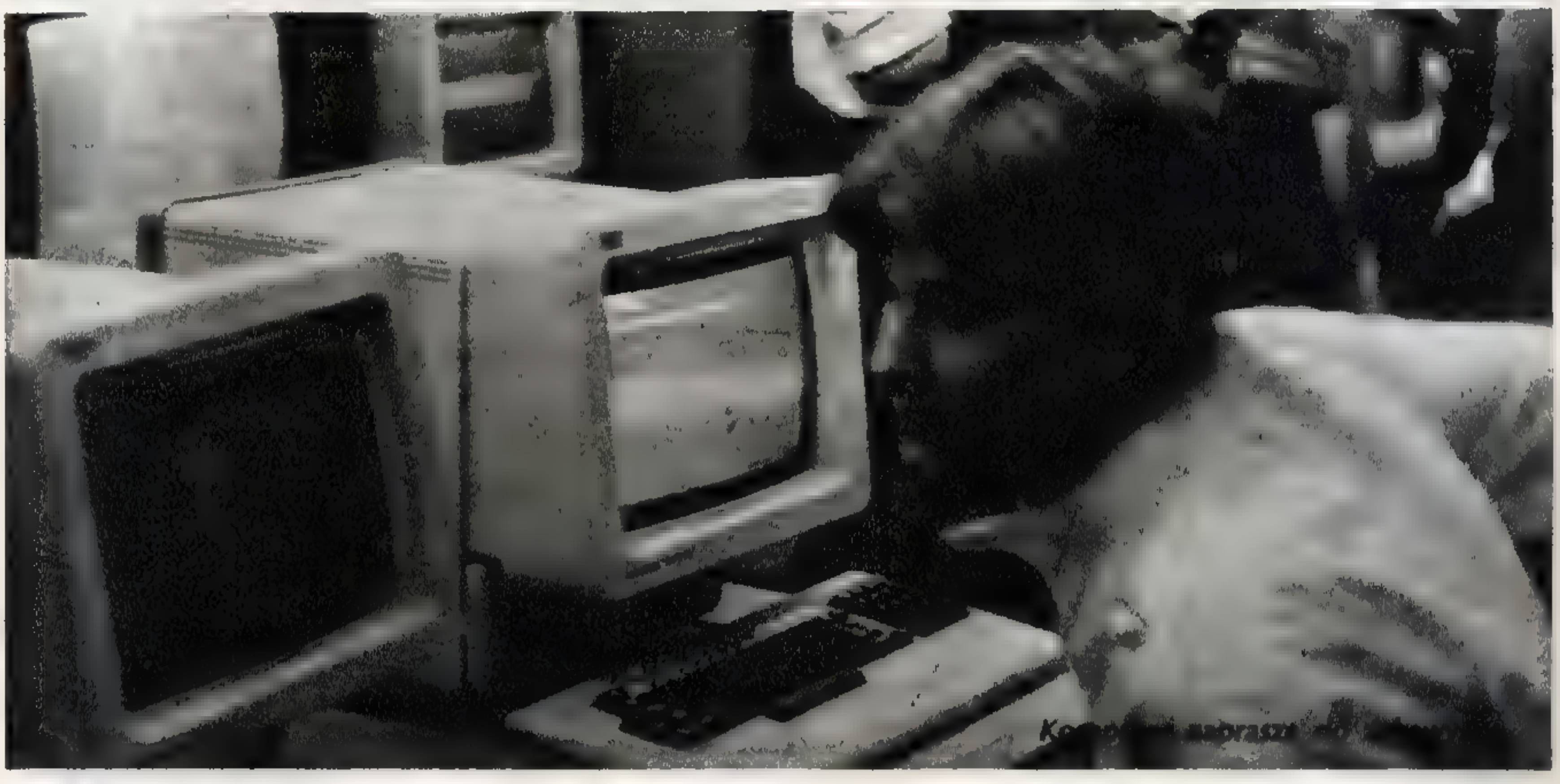
SPECTRUM

„*Kto zabierze ostatnie jabłko ten wygra!*" Twoim przeciwnikiem jest SPECTRUM. Znajdujecie jabłka i bierzecie je na zmianę. Ostatnie jabłko jest najsmaczniejsze. Czy komputer zawsze musi je zabrać? Spróbuj odkryć „system" i zacznij wygrywać. W trakcie zastanawiania się słuchasz muzyki aleatorycznej (przypadkowej — linia 260). Jeżeli chcesz zawsze zaczynać grę jako pierwszy skasuj linię 230.

Powodzenia i... smacznego!

```
  0>REM Jerzy Wozniak

 10 BORDER 0: PAPER 0: INK 7
 20 CLS
 30 PRINT "  KTO ZABIERZE OSTAT
NIE JABŁKO"
 40 PRINT
 50 PRINT "          TEN WYGRA !"
 60 PRINT
 70 PRINT "  WOLNO BRAC 1,2 LUB
 3 JABŁKA."
 80 PRINT "______________________
_______________"
 90
100 REM LOSOWANIE JABŁEK
110 LET p=INT (RND*14)+11
120
130 PRINT AT 8,18;"JABŁEK ";p
140
150 PRINT AT 11,0;"................
..............."( TO p)
160
170 PRINT AT 12,0;"0000000000000
0000000000000"( TO p)
180
190 REM MIGACZ
200 POKE 22912,143
210
220 REM LOSOWANIE=KTO ZACZYNA
230 IF RND<0.5 THEN GO TO 360
240 PRINT AT 16,0,
    "ILE JABŁEK ZABIERASZ ? ";
250 LET r=CODE INKEY$-48
260 PAUSE 12: BEEP 0.01,RND*30
270 IF r<1 OR r>3
    THEN  GO TO 250
280 PRINT r
290 PRINT AT 18,23;" "
300 LET p=p-r
310 PRINT AT 9,2;
    "ZOSTAŁO ";p;" "
320 GO SUB 540
330 IF p<1 THEN GO TO 510
340
350 REM KOMPUTER MYSLI
360 LET n=INT (p/4)*4
370 IF p=n THEN LET
    r=INT (RND*3+1)
380 IF p<>n THEN LET r=p-n
390 LET p=p-r
400 PAUSE 60
410 BEEP 1,10*r
420 PRINT AT 18,5;
    "SPECTRUM ZABIERA  ";r
430 PRINT AT 16,23;" "
440 GO SUB 540
450 PRINT AT 9,2;
    "ZOSTAŁO ";p;" "
460 IF p<1 THEN GO TO 480
470 GO TO 240
480 PRINT FLASH 1;AT 20,2;
    "    WYGRAŁ KOMPUTER !!!
490 BEEP 1,20: BEEP 2,8
500 PAUSE 100: GO TO 10
510 PRINT FLASH 1;AT 20,2;
    "      TY WYGRAŁES !!!
520 BEEP .1,20: BEEP 2,20
530 PAUSE 200: GO TO 10
540 FOR l=1 TO r
550 PRINT AT 12,p;"XXX"( TO r)
560 BEEP .1,10*r
570 PRINT AT 12,p;"   "( TO r)
580 PAUSE 10
590 NEXT l
600 RETURN
610 SAVE "ZABIERANKA" LINE 10
```



PROGRAM 29

SHARP

Nasz program nazwaliśmy *„Czytanie myśli"*, gdyż może on sprawiać takie wrażenie.

Aby posługiwać się „odczytywaczem myśli", wystarczy znajomość... alfabetu: ściślej znajomość KOLEJNOŚCI LITER ALFABETU ŁACIŃSKIEGO.

Przykładowo na pytanie:
PRZED C ? T=1 N=0
należy odpowiedzieć „TAK", czyli wypalcować 1, gdy pomyślaną literą jest A lub B.

Jeżeli zaś pomyślaną literą jest C,D,E,...,Z, to należy odpowiedzieć „NIE", czyli 0. Uwaga: litera C NIE JEST PRZED C.

Po załadowaniu programu pomyślimy jakiś wyraz. Na początek krótki: UL, LAS, KOT.

Program wywołuje RUN „X". Od tego momentu bardzo uważnie odpowiadajmy na pytania. (Nasz jeden błąd psuje zabawę).

Gdy na pytanie
OSTATNIA T=1 N=0
czyli „czy to była OSTATNIA litera pomyślanego wyrazu" — gdy na pytanie odpowiemy „TAK" (1), wówczas komputer „odczyta pomyślany wyraz".

CZYTANIE... MYŚLI. Wykorzystuje się tu zasadę DYCHOTOMII (podziału na dwie przeciwstawne części). Najpierw całość alfabetu dzielimy na pół. Wskazaną przez odpowiedź połówkę dzielimy na ćwiartki. Potem na „ósemki, szesnastki i trzydziestkidwójki".

Ponieważ 2 × 2 × 2 × 2 × 2 = 32, zaś liter jest 26, zatem większość liter „odczytujemy" po PIĘCIU odpowiedziach, niektóre zaś po CZTERECH.

Życzymy dobrej zabawy!

JANUSZ MILLER
SHARP PC-1500

```
CZYTANIE MYSLI

600:"C":CLS
610:WAIT 0:PRINT "
    PRZED ";Z$;"?
    T=1 N=0 "
620:CURSOR 16:
    INPUT " ";D:
    CLS :RETURN
630:"D":USING
640:Z$="N":GOSUB "
    C":IF D=0GOTO
    820
650:Z$="H":GOSUB "
    C":IF D=0GOTO
    750
660:Z$="E":GOSUB "
    C":IF D=0GOTO
    720
670:Z$="C":GOSUB "
    C":IF D=0GOTO
    700
680:Z$="B":GOSUB "
    C":IF D=0LET X
    $="B":RETURN
690:X$="A":RETURN
700:Z$="D":GOSUB "
    C":IF D=1LET X
    $="C":RETURN
710:X$="D":RETURN
720:Z$="G":GOSUB "
    C":IF D=0LET X
    $="G":RETURN
730:Z$="F":GOSUB "
    C":IF D=0LET X
    $="F":RETURN
740:X$="E":RETURN
750:Z$="K":GOSUB "
    C":IF D=0GOTO
    790
760:Z$="J":GOSUB "
    C":IF D=0LET X
    $="J":RETURN
770:Z$="I":GOSUB "
    C":IF D=0LET X
    $="I":RETURN
780:X$="H":RETURN
790:Z$="M":GOSUB "
    C":IF D=0LET X
    $="M":RETURN
800:Z$="L":GOSUB "
    C":IF D=0LET X
    $="L":RETURN
810:X$="K":RETURN
820:Z$="T":GOSUB "
    C":IF D=0GOTO
    900
830:Z$="Q":GOSUB "
    C":IF D=0GOTO
    870
840:Z$="P":GOSUB "
    C":IF D=0LET X
    $="P":RETURN
850:Z$="O":GOSUB "
    C":IF D=0LET X
    $="O":RETURN
860:X$="N":RETURN
870:Z$="S":GOSUB "
    C":IF D=0LET X
    $="S":RETURN
880:Z$="R":GOSUB "
    C":IF D=0LET X
    $="R":RETURN
890:X$="Q":RETURN
900:Z$="W":GOSUB "
    C":IF D=0GOTO
    940
910:Z$="V":GOSUB "
    C":IF D=0LET X
    $="V":RETURN
920:Z$="U":GOSUB "
    C":IF D=0LET X
    $="U":RETURN
930:X$="T":RETURN
940:Z$="Y":GOSUB "
    C":IF D=0GOTO
    970
950:Z$="X":GOSUB "
    C":IF D=0LET X
    $="X":RETURN
960:X$="W":RETURN
970:Z$="Z":GOSUB "
    C":IF D=0LET X
    $="Z":RETURN
980:X$="Y":RETURN
 1000:"X":CLEAR :
      WAIT :CLS
 1010:K=K+1:WAIT :
      PRINT "LITER
      A";K
 1020:GOSUB "D":Y$
      =Y$+X$
 1030:INPUT "OSTAT
      NIA? T=1 N=0
       ";D:IF D=0
      GOTO 1010
 1040:WAIT :PRINT
      "KRYPTONIM "
      ;Y$:END
```

Zmienne systemowe ROM SPECTRUM

Dla tych, którzy poznali już BASIC, proponujemy spojrzenie w te sfery pamięci, które pozwolą wszechstronniej wykorzystać możliwości SPECTRUM. Dobra znajomość zmiennych systemowych jest wygodna dla programisty; ułatwia pracę i pozwala głębiej poznać system. Większość *sztuk i sztuczek*, jakie można wydobyć z tego obszaru, postaramy się przedstawić w kilku najbliższych numerach.

Klawiatura

KSTATE	23552..9
LAST—K	23560
REPDEL	23561
REPPER	23562
PIP	23009

Co 20 ms procedura przerwań maskowalnych o adresie 56 przeszukuje klawiaturę z tym samym celem — odszukania przyciśniętego klawisza. W przypadku pozytywnym dodatkowo następuje dekodowanie na system ASCII. Aktualny stan klawiatury zostaje umieszczony w grupie bajtów KSTATE, którą można podzielić na dwie części po 4 bajty. Oznaczmy adresy 23552...5, jako *B1, B2, B3, B4* zaś adresy 23556..9, jako *A1, A2, A3, A4*.

Wciskamy klawisz. Reaguje grupa druga *(A1...A4)*. Bajt *A1*, który miał dotąd wartość 255, co oznacza, że żaden klawisz nie był wciśnięty, szybko reaguje kodem naszego guzika. Przy czym jest to kod cyfry albo dużej litery. Nie ma więc znaczenia dla *A1*, czy przyciśnięto coś z CAPS SHIFT, czy SYMBOL SHIFT. Wyjątkiem jest reakcja na oba shifty (kod 14). Z chwilą przyciśnięcia pary innych klawiszy (oprócz obu shiftów) ładowany jest kod pierwszego wykrytego.

Natomiast A4 pokazuje już bardziej sprecyzowany kod ASCII. Uwzględnia, czy naciśnięto guzik z CAPS, czy SYMBOL SHIFT. Teraz wystarczy już tylko skupić się nad długością trwania nacisku. Z chwilą wykrycia klawisza w *A3* ładuje się wartość zmiennej systemowej REPDEL. Jest to czas mierzony w 50. częściach sekundy, po tym czasie klawisz zostanie uznany za przyciśnięty powtórnie. Standardowo ma ona wartość 35, czyli ok. 0,5 s. Z każdym przerwaniem *A3* jest dekrementowane (zmniejszane o 1), aż do zera. Komputer wie wtedy, że dalej trzymamy palec na klawiszu. Jeśli jesteśmy w edytorze, nastąpi drugi wydruk znaku o kodzie *A4* dla trybu małych liter (kursor L) lub o kodzie *A1* dla trybu dużych liter (kursor C). O tym, że komputer „czuje" nacisk palca, niech świadczy poniższy test:

```
10 POKE 23562,1
20 IF NOT LEN INKEY$ THEN GOTO 20
30 IF LEN INKEY$ THEN GOTO 30
40 IF PEEK 23558 +3 >= PEEK 23561 T
HEN PRINT "krotko": STOP
50 IF PEEK 23558 +11 >= PEEK 23561
THEN PRINT "srednio": STOP
60 PRINT "dlugo"
```

Kontynuując, dalsze samopowtarzanie zależy już od zawartości zmiennej systemowej REPPER. Decyduje ona o odstępie czasowym, w jakim ma powtarzać się uznawanie trzymanego klawisza za „ponownie wciśnięty" po wyzerowaniu zawartości REPDEL w A3. Normalnie jest tam wartość 5, co oznacza 0.1 s. Samopowtarzanie odbywa się na tej samej zasadzie co wcześniej, czyli na dekrementacji. Jeśli teraz A3 wyzeruje się, zawartość REPPER ładowana jest tam znowu itd. Gdy podczas tego procesu przyciśniemy inny klawisz, system zignoruje go, chyba że puścimy ten pierwszy. Wtedy nastąpi przerzut na komórki B1... B4 i cały omówiony proces powtarza się. W A1 pojawi się 255, zaś precyzyjny kod ostatnio przyciskanego klawisza zostaje umieszczony w zmiennej LAST—K i nie zmieniany, póki nie zostanie wykryty nowy klawisz.

Upraszczając, pierwsze powtórzenie guzika nastąpi po czasie zawartym w REPDEL, a każde następne po czasie zawartym w REPPER.

Należy podkreślić, że guzik naciśnięty z CAPS SHIFT lub SYMBOL SHIFT jest traktowany jako pojedynczy. Pozostaje do wyjaśnienia zawartość A2 lub B2. Jeśli nie naruszamy klawiatury, są tam zera, lecz po wykryciu nacisku pojawia się tam wartość 5 dekrementowana za każdym następnym przerwaniem. W praktyce oznacza to, że komputer 5 razy musi mieć potwierdzenie nacisku (0,1 s), i dopiero wówczas zareaguje dekodowaniem klawisza. Służbista? Nie tylko. Otóż zwarcie styków klawiatury powoduje ich nieznaczne drgania, co mogłoby spowodować błędy w wykryciu klawisza. Spokojne odczekanie przez system daje większą pewność użytkownikowi.

Poza otrzymywaniem szeregu informacji o stanie klawiatury, w zasadzie zmienna KSTATE nie jest wykorzystywana do innych celów. Zmiana wartości któregokolwiek bajtu nic nie da, gdyż uaktualnianie danych następuje piorunująco szybko. Zmieniać możemy wartości REPPER i REPDEL, modyfikując w ten sposób parametry samopowtarzania. Dobre efekty dla zaprzyjaźnionego już dobrze z klawiaturą daje:

POKE 23562,1
POKE 23561,30
POKE 23609,40

Ten ostatni adres (23609), to zmienna systemowa PIP. Decyduje ona o długości dźwięku wydanego z głośnika przy nacisku na klawisz. Początkowo zawiera 0, co objawia się cichym trzaskiem (ang. click). Podwyższenie o 1 powoduje zwiększenie czasu trwania dźwięku o ok. 1/765 s.

Na zakończenie kilka sztuczek. Poniższy program przełączający wydruk tekstu z ekranu na drukarkę i odwrotnie jest pretekstem do rezygnacji z dwuznacznej funkcji INKEY$:

```
10 PRINT "Drukarka / Ekran"
15 PAUSE 1: PAUSE 0
20 LET a= PEEK 23556
30 IF a= CODE "D" THEN LET s=3
40 IF a= CODE "E" THEN LET s=2
50 PRINT #s;"tekst"
```

Dla programistów zniechęconych niedostatkiem instrukcji PAUSE 0 mamy dobrą radę. Sekwencja:

10 PAUSE 0
20 IF PEEK 23558 < 35 THEN GOTO 10

działa znacznie lepiej.

Brak reakcji pojedynczych klawiszy SHIFT również można ominąć. Poniższy program zakończy się, jeśli rzeczywiście przyciśniemy dowolny klawisz:

```
10 PRINT #1;AT 1,8;"dowolny klawisz
"
20 LET a= IN 254
30 IF a=255 OR a=191 THEN GOTO 20
40 STOP
```

W najbliższym numerze trochę szczegółów o pamięci RAM.

Krzysztof MAMCARZ

PROGRAM 30

SPECTRUM

MOŻNA INACZEJ

Chciałbym podzielić się z Czytelnikami „IKS-a" kilkoma uwagami dotyczącymi przedstawionych w numerze 1/86 programów „Wykresy" i „Mapa Polski".

Wartość dydaktyczna programu „Wykresy" jest znacznie obniżona przez obarczenie użytkownika koniecznością podania wartości maksymalnej i minimalnej funkcji w rozpatrywanym przedziale. Wartości te — niezbędne do prawidłowego wyrysowania układu współrzędnych i opisu osi — mogą i muszą być obliczone przez komputer, na podstawie przepisu funkcji i przedziału zmienności zmiennej niezależnej x. Obliczenie tych wartości dla wielu niebanalnych funkcji nie jest rzeczą prostą, dlatego też zadanie to powinna wykonać maszyna. Podanie błędnego zakresu zmienności wartości funkcji powoduje całkowite zniekształcenie wykresu (dla przykładu proszę sprawdzić: y = SIN X, XMIN = 0, XMAX = 5, YMIN = 0, YMAX = 2).

Przedstawiony program nie sporządzi wykresu funkcji stałej y(x) = a. Funkcja stała, aczkolwiek bardzo prosta, musi być przez program akceptowana. Program nie poradzi sobie również z funkcjami nieciągłymi w zadanym przedziale, np. y(x) = 1/x czy y(x) = (SIN X).

W przypadku podania przedziału zmienności argumentu x mniejszego od Ø.5 np. (Ø.1, Ø.2) wygenerowany opis osi będzie błędny. Prawidłowe działanie programu dla małych długości przedziału jest niezbędne, ponieważ umożliwia oglądanie wycinków wykresu funkcji.

W programie nie są sprawdzane dane wejściowe XMIN, XMAX, YMIN, YMAX (powinny to być liczby). Bardzo łatwo można więc doprowadzić do wystąpienia błędu, podając — omyłkowo lub świadomie — wielkości nie będące liczbami.

Sprawdzenie warunku w instrukcji 460

IF CODE Z$ = 84 OR CODE Z$ = 116...

można uprościć pisząc:

IF Z$ = „T" OR Z$ = „t"...

Ponadto należałoby najpierw sprawdzić, czy wprowadzony znak jest jednym ze znaków T, t, N, n. Milczące założenie, że wszystkie znaki różne od T i t są równoznaczne z N/n jest błędem, gdyż prawdopodobieństwo błędnego napisania znaku N lub n jest równe prawdopodobieństwu błędnego wpisania znaku T lub t.

W programie „Mapa Polski" do wykreślenia konturów mapy wielokrotnie użyto instrukcji DRAW z trzema parametrami (łuki), co znacznie wydłuża czas rysowania choć program czyni czytelnym. Poniżej przedstawiam swoją wersję rysowania mapy, która — choć trochę dłuższa — jest kilkadziesiąt razy szybsza (czas rysowania mniejszy od 0.5 s).

Ponadto w obydwu programach po zakończeniu obliczeń nie przywrócono „warunków normalnych", tj. BORDER 7 : PAPER 7 : INK0 : CLS. Pozostawienie po wyjściu z programu takich warunków powinno być regułą, aby nie powodować ewentualnych zakłóceń w pracy programu następnego użytkownika.

Janusz MORBITZER



```
5000 REM rysowanie mapy Polski
5010 CLS : PLOT 138,160: DRAW 27
,-3: DRAW 3,3: DRAW 9,-5: DRAW 8
,-26: DRAW 1,-16: DRAW -4,-2: DR
AW -6,-8: DRAW 8,-5
5020 DRAW -2,-8: DRAW 3,-10: DRA
W 3,-2: DRAW 0,-2: DRAW 5,-5: DR
AW -3,-2: DRAW 3,-4: DRAW -1,-5:
 DRAW -6,-1: DRAW -15,-17: DRAW
2,-10: DRAW 3,-2: DRAW -2,-2
5030 DRAW -12,4: DRAW -1,2: DRAW
 -12,1: DRAW -3,-3: DRAW -3,2: D
RAW -5,0: DRAW -3,-2: DRAW -1,-4
: DRAW -2,3: DRAW -2,-2: DRAW 0,
5: DRAW -4,0: DRAW -2,3
5040 DRAW -3,-2: DRAW -1,-3: DRA
W -6,0: DRAW -8,11: DRAW -8,4: D
RAW -4,-2: DRAW -4,5: DRAW 3,2:
DRAW -1,2: DRAW -8,1
5050 DRAW -4,4: DRAW -2,-1: DRAW
 3,-7: DRAW -6,-3: DRAW -6,10: D
RAW 4,3: DRAW -1,2: DRAW -8,-1:
DRAW -1,2: DRAW -3,0: DRAW -2,2:
 DRAW -3,-1: DRAW -1,7
5060 DRAW -3,0: DRAW -1,-4: DRAW
 -6,1: DRAW 2,3: DRAW 2,11: DRAW
 -5,3: DRAW 1,1: DRAW -3,4: DRAW
 3,4: DRAW 0,8: DRAW -3,1: DRAW
0,6: DRAW 1,0: DRAW -2,8
5070 DRAW -7,6: DRAW 5,6: DRAW -
2,20: DRAW 28,9: DRAW 8,5: DRAW
6,1: DRAW 6,4: DRAW 16,3: DRAW 8
,-1
5080 DRAW 8,-6: DRAW -2,-2: DRAW
 -4,4: DRAW 1,-6: DRAW 5,-3: DRA
W 5,1: DRAW 5,3: DRAW 12,-1
5090 RETURN
```

Program „Wykresy"

na stronie 10

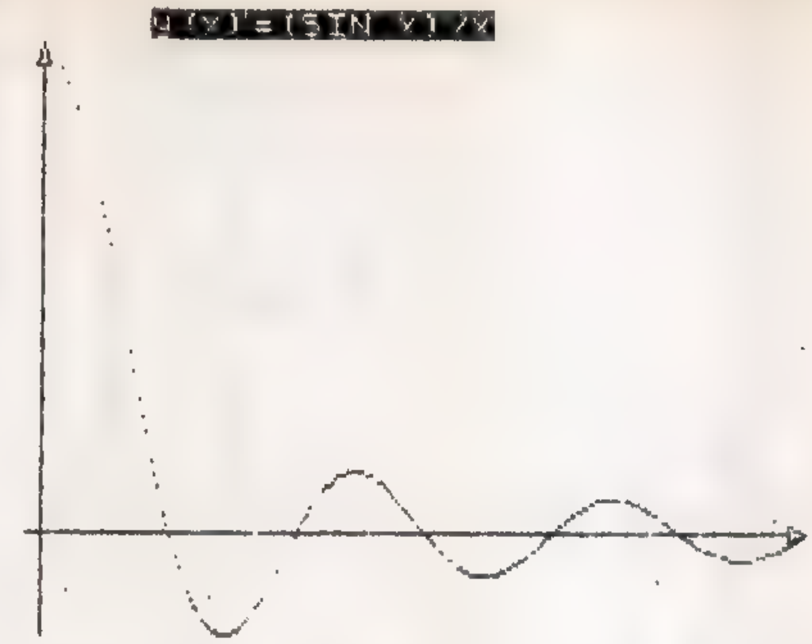
```
   1 REM program "Funkcja"
   2 REM Autor-Janusz MORBITZER
   5 BEEP .2,15: PAUSE 30: BORDE
R 7: PAPER 7: INK 0: CLS
   7 POKE 23609,50
  15 CLS : PRINT BRIGHT 1;AT 8,8
;"PROGRAM ""Funkcja""": PRINT BR
IGHT 1;AT 13,5;"Autor: Janusz MO
RBITZER": PRINT AT 16,11; BRIGHT
 1;"WSP-KRAKÓW"
  20 DIM x(250): DIM f(250): DIM
 p(2)
  22 GO SUB 600: CLS : PRINT AT
8,3;"Program ""Funkcja"" realizu
je": PRINT AT 9,6;"wydruk funkcj
i y=f(x)": PRINT AT 10,8;"w prze
dziale <a,b>": PRINT AT 11,6;"or
az oblicza maksimum ": PRINT AT
12,8;"i minimum funkcji": PRINT
AT 13,8;"w tym przedziale": GO S
UB 550
  25 LET p(1)=1000000: LET p(2)=
1000000
  27 CLS : PRINT AT 14,5; BRIGHT
 1;"Podaj przepis funkcji:"
  30 INPUT "f(x)= "; LINE k$: IF
 LEN k$=0 THEN GO TO 30
  35 LET dlugosc=5+LEN k$: LET p
ozy=(31-dlugosc)/2
  40 CLS : PRINT BRIGHT 1;AT 14,
8;"Podaj przedzial:"
  41 INPUT "  a="; LINE a$;"
     b="; LINE b$: IF LEN a$=0
OR LEN b$=0 THEN GO TO 41
  43 FOR i=1 TO LEN a$: IF a$(i)
>="0" AND a$(i)<="9" OR a$(i)=".
" OR a$(i)="-" OR a$(i)="+" OR a
$(i)="PI" OR a$(i)="*" OR a$(i)=
"" THEN NEXT i: GO TO 50
  44 GO TO 40
  51 FOR i=1 TO LEN b$: IF b$(i)
>="0" AND b$(i)<="9" OR b$(i)=".
" OR b$(i)="-" OR b$(i)="+" OR b
$(i)="PI" OR b$(i)="*" OR b$(i)=
"/" THEN NEXT i: GO TO 55
  52 GO TO 40
  55 LET a=VAL a$: LET b=VAL b$
  60 IF a>=b THEN GO TO 40
  61 LET przedzial=0
  62 LET dt=b-a: LET r=SGN (a*b)
  64 IF r=1 AND a>0 THEN LET dt=
b: GO TO 68
  66 IF r=1 AND a<0 THEN LET dt=
ABS (a)
  68 IF dt<>b-a THEN LET przedzi
al=1
  70 LET z=ABS (a): LET d=dt/250
: LET d4=4*d
  80 CLS : PRINT BRIGHT 1;AT 14,
6;"Podaj liczbe punktów": PRINT
AT 16,6; BRIGHT 1;"nieciagłości
 (0-2):"
  82 LET w$=INKEY$: IF LEN w$<>1
 THEN GO TO 82
  90 IF w$<"0" OR w$>"2" THEN GO
 TO 82
 100 LET w=VAL w$: IF w=0 THEN G
O TO 140
 105 FOR i=1 TO w
 110 PRINT AT 21,3;"Podaj ";i;".
 pkt nieciagłości:"
 115 INPUT LINE c$: IF LEN c$=0
THEN GO TO 115
 120 FOR m=1 TO LEN c$: IF c$(m)
>="0" AND c$(m)<="9" OR c$(m)=".
" OR c$(m)="-" OR c$(m)="*" OR c
$(m)="+" OR c$(m)="PI" THEN NEXT
 m: LET p(i)=VAL c$: GO TO 130
 125 GO TO 115
 130 IF NOT (p(i)>=a AND p(i)<=b
) THEN LET w=w-1
 135 NEXT i
 137 LET pn1=p(1): LET pn2=p(2)
 140 LET n=249: LET l=0
 145 BORDER 0: PAPER 0: INK 7: L
ET dk=(28-LEN a$-LEN b$)/2: CLS
: PRINT AT 4,9; FLASH 1;"PROSZE
CZEKAC": PRINT AT 8,8;"Obecnie f
unkcja:": PRINT INVERSE 1;AT 10,
pozy;"y(x)=";k$: PRINT AT 12,0;"
jest tabelaryzowana w przedziale
": PRINT INVERSE 1;AT 14,dk;"[";
a$;",";b$;"]": PRINT AT 16,0;"Kr
ok tabelaryzacji dx=";d
 150 FOR i=1 TO n+1
 160 LET x=a+(i-1)*d: IF x>b THE
N GO TO 210
 170 IF w=0 THEN GO TO 190
 180 IF ABS (x-pn1)<d4 OR ABS (x
-pn2)<d4 THEN GO TO 200
 190 LET l=l+1: LET x(l)=x: LET
f(l)=VAL k$
 200 NEXT i
 210 LET fmin=f(1): LET fmax=f(1
)
 220 LET k1=1: LET k2=1
 230 FOR i=2 TO l
 240 IF (f(i)-fmin)>=0 THEN GO T
O 260
 250 LET fmin=f(i): LET k1=i: GO
 TO 280
 260 IF (f(i)-fmax)<=0 THEN GO T
O 280
 270 LET fmax=f(i): LET k2=i
 280 NEXT i
 290 LET xmin=x(k1): LET xmax=x(
k2)
 295 LET m$=STR$ fmax: IF LEN m$
>9 THEN LET m$=m$( TO 9)
 296 LET n$=STR$ fmin: IF LEN n$
>9 THEN LET n$=n$( TO 9)
 297 LET x$=STR$ xmin: IF LEN x$
>9 THEN LET x$=x$( TO 9)
 298 LET y$=STR$ xmax: IF LEN y$
>9 THEN LET y$=y$( TO 9)
 300 LET dy=fmax-fmin: BEEP .8,1
5: BORDER 7: PAPER 7: INK 0: CLS

 310 PRINT BRIGHT 1;AT 9,0;"FMIN
=";n$;" dla x=";x$
 320 PRINT BRIGHT 1;AT 11,0;"FMA
X=";m$;" dla x=";y$
 330 GO SUB 550
 340 CLS : LET s=SGN (fmin*fmax)
 350 IF s=1 AND fmin>0 THEN LET
y=5: GO TO 400
 360 IF s=1 AND fmin<0 THEN LET
y=160: GO TO 400
 370 IF s=0 AND fmax>0 THEN LET
y=5: GO TO 400
 380 IF s=0 AND fmax=0 THEN LET
y=160: GO TO 400
 390 LET y=INT ((ABS (fmin)/dy)*
165): IF y<5 THEN LET y=5
 395 IF y>160 THEN LET y=160
 400 LET t=SGN (a*b)
 410 IF t>=0 AND a>=0 THEN LET x
=5: GO TO 440
 420 IF t>=0 AND a<0 THEN LET x=
249: GO TO 440
 430 LET x=INT ((z/dt)*255)
 440 IF x<250 THEN PLOT 0,y: DRA
W 249,0: PLOT 249,y+2: DRAW 0,-4
: DRAW 6,2: DRAW -6,2: GO TO 460
 450 PLOT 6,y: DRAW 249,0: PLOT
6,y+2: DRAW 0,-4: DRAW -6,2: DRA
W 6,2
 460 IF y<160 THEN PLOT x,0: DRA
W 0,159: PLOT x-2,159: DRAW 4,0:
 DRAW -2,6: DRAW -2,-6: GO TO 47
8
 470 PLOT x,6: DRAW 0,159: PLOT
x-2,6: DRAW 4,0: DRAW -2,-6: DRA
W -2,6
 478 IF dy<>0 THEN LET z3=160/dy
 480 LET poprx=0: LET popry=0: L
ET z1=250/dt: IF fmin<>0 THEN LE
T z4=160/fmin
 481 IF x=5 THEN LET poprx=5
 482 IF y=5 THEN LET popry=5
 483 IF fmax=0 THEN GO TO 485
 484 LET z2=160/fmax
```

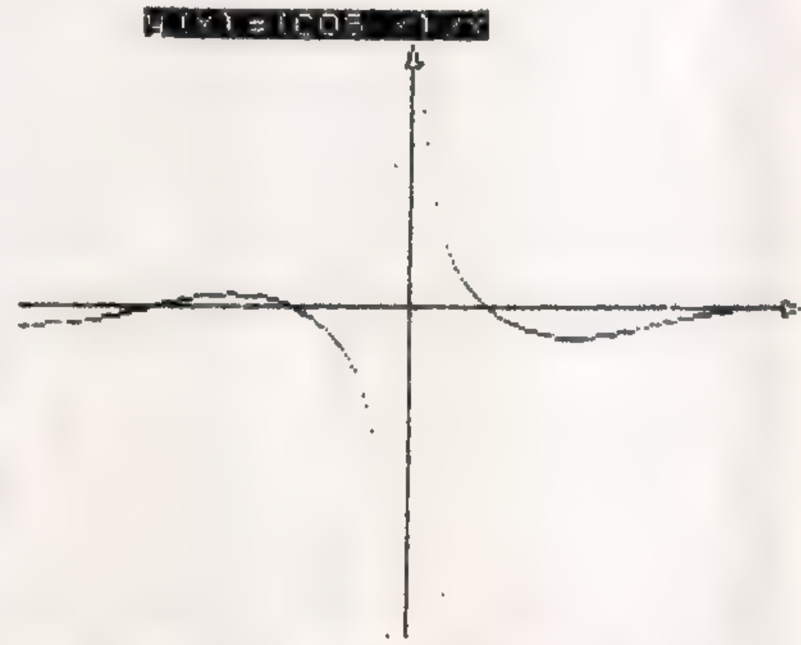
```
 485 IF przedzial=1 AND a>0 THEN
LET z=0
 487 IF s>=0 AND fmax>0 THEN LET
f$="f(i)*z2+popry": GO TO 490
 488 IF s>=0 AND fmin<0 THEN LET
f$="160-ABS (f(i)*z4)": GO TO 4
90
 489 LET f$="ABS ((f(i)-fmin)*z3
)+popry"
 490 PRINT AT 0,5; INVERSE 1;"y(
x)=";k$
 495 FOR i=1 TO l
 500 LET x=((z+x(i))*z1+poprx)
 510 LET y=VAL f$
 515 IF ABS x>255 OR ABS y>175 T
HEN GO TO 530
 520 INK 0: PLOT x,y
 530 NEXT i
 531 PRINT #1;AT 0,2; BRIGHT 1;"
Żądany wydruk(t-tak,n-nie):": LE
T w$=INKEY$: IF LEN w$=0 THEN GO
 TO 531
 532 IF w$<>"t" AND w$<>"n" THEN
 GO TO 531
 533 IF w$="t" THEN COPY : LPRIN
T : LPRINT "Przedział: [";a$;","
;b$;"]": LPRINT "Fmin=";n$," dla
 x=";x$: LPRINT "Fmax=";m$;" dla
 x=",y$
 535 PRINT #1;AT 0,0;"
                   ": PRINT #
1;AT 0,1; BRIGHT 1;"Podaj(0-koni
ec,1-kontynuacja):"
 536 LET t$=INKEY$: IF LEN t$=0
THEN GO TO 536
 538 IF t$<>"0" AND t$<>"1" THEN
 GO TO 536
 539 IF t$="1" THEN GO TO 25
 540 CLS : PRINT AT 8,10;"******
******": PRINT AT 9,10;"*
 *": PRINT AT 10,10;"*  KONIEC
 *": PRINT AT 11,10;"* PROGRAMU
 *": PRINT AT 12,10;"*
*": PRINT AT 13,10;"************
": GO SUB 600: GO TO 630
 545 REM podprogram
 550 PRINT #1; BRIGHT 1; FLASH
1;AT 0,4;"Naciśnij dowolny klawi
sz"
 560 PAUSE 0: RETURN
 600 REM podprogram
 610 LET t=.25: BEEP t,0: BEEP t
,2: BEEP t,4: BEEP t,5: BEEP t,7
: BEEP t,9: BEEP t,11: BEEP t,12
: PAUSE 10: BEEP t,12: BEEP t,11
: BEEP t,9: BEEP t,7: BEEP t,5:
BEEP t,4: BEEP t,2: BEEP t,0
 620 RETURN
 630 PAUSE 0: BORDER 7: PAPER 7:
 INK 0: CLS : GO TO 15 REM koni
ec programu
```

UWAGA !

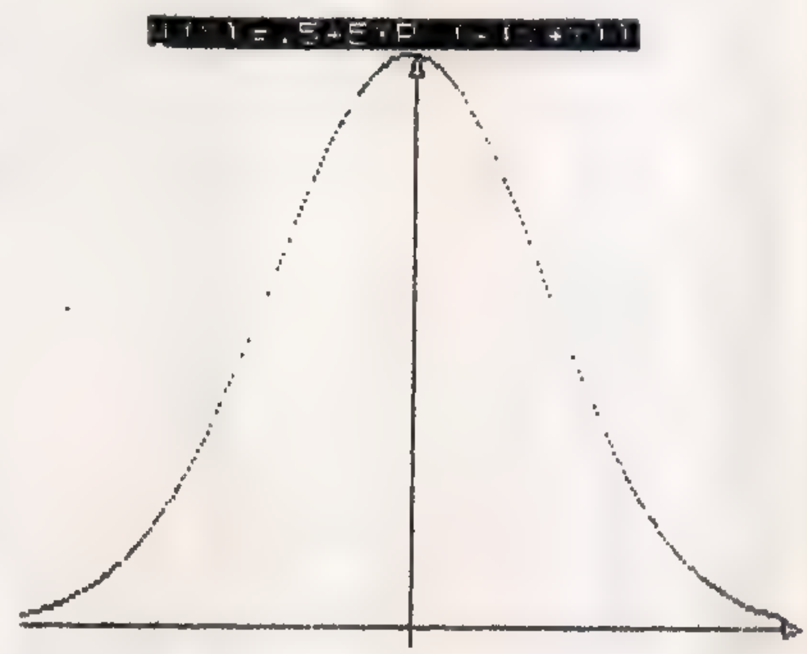
„IKS" w prenumeracie! Prenumeratę przyjmują oddziały RSW „Prasa-Książka-Ruch", a w miejscowościach, gdzie ich nie ma urzędy pocztowe i doręczyciele.



```
Przedział: [0,6*PI]
Fmin=-0.217133 dla x=4.5238934
Fmax=0.3849090 dla x=0.3015928
```



```
Przedział: [-2*PI,2*PI]
Fmin=-4.873399 dla x=-0.201061
Fmax=3.8538699 dla x=0.2513274
```



```
Przedział: [-2,2]
Fmin=.00915781 dla x=-2
Fmax=0.5 dla x=0
```

METODA TRAPEZÓW

Znalezienie wartości całki oznaczonej zadanej funkcji w określonym przedziale na ogół nie jest trudne. Pierwszym, najtrudniejszym krokiem jest znalezienie funkcji pierwotnej — czynności następne są już działaniami arytmetycznymi, nie sprawiają większych kłopotów. Prawdziwe zmartwienie zaczyna się wtedy, kiedy funkcja podcałkowa nie posiada funkcji pierwotnej. Taki problem mamy z funkcją $\exp(-x^2)$, której całkę oznaczoną możemy obliczyć jedynie metodami numerycznymi.

Możemy to zrobić metodą Newtona-Cotesa, Simpsona albo metodą trapezów. Wybierzmy tę ostatnią. Przedział całkowania dzielimy na n równych części, w konsekwencji otrzymujemy węzły:

$x_0 = a < x_1 < x_2 < x_3 < \ldots < x_{n-1} < x_n = b$

Jako przybliżenie całki w metodzie trapezów przyjmuje się wartość następującego wyrażenia:

$S = h(\frac{1}{2}y_0 + y_1 + y_2 + \ldots y_{n-1} + \frac{1}{2}y_n)$

gdzie:

$y_i = f(x_i)$ $(i = 0, 1, 2, \ldots n)$

f — funkcja podcałkowa

h — wysokość elementarnego trapezu (jednakowa dla wszystkich trapezów, co wynika z podziału przedziału całkowania na n równych części).

Metoda ta nie jest wolna od błędu, jest on określony wzorem:

$$R = \frac{-(b-a)h^2\,y''\,t}{12},\ \text{dla } t \in \langle a, b \rangle$$

Przystępując do obliczania całki metodą trapezów nie wiemy na ile części podzielić całkowania < a, b >, tak, aby wynik mieścił się w zadanej dokładności eps. Musimy to uwzględnić w realizowanym algorytmie.

Krok pierwszy. Jako pierwsze przybliżenie (początkowe) całki przyjmujemy wartość całki obliczoną dla n = 1. Jest ono równe polu trapezu o bokach f(a), f(b) i wysokości h = b — a.

Krok drugi. Następne przybliżenie otrzymujemy przez podwojenie liczby podziałów przedziału < a, b > z poprzedniego przybliżenia. Przedział całkowania dzielimy oczywiście na równe części. Liczba podziałów jest podwajana, a wysokość h zmniejsza się o połowę.

Krok trzeci. Z otrzymanych podziałów obliczamy kolejne przybliżenie całki. Sprawdzamy, czy moduł różnicy pomiędzy tym i poprzednim przybliżeniem jest mniejszy od zadanej wielkości eps Jeżeli nie, to cofamy się do kroku drugiego, jeżeli tak, to w tym miejscu następuje koniec obliczeń i wydruk wyników

```
comment Obliczanie całki metodą trapezów
begin
    real h, a, b, eps, x; x0, x1, y, s;
    integer i, n;
    line (1);
    print ('WARTOŚĆ CAŁKI W
                PRZEDZIALE < a, b >');
    line (1);
P: read (a, b, eps);
    h = b — a; n = 1;
    s = (exp(—a↑2) + exp(—b↑2))/2;
    x0 = s * h;
    y = 10 ↑ 2;
    for n = 2 * n while y > eps do
    begin
h = h/2; x1 = s;
x = a;
for i = 1 step 1 until n — 1 do
begin
    x = x + h;
    x1 = x1 + exp(—x↑2)
end i;
x1 = x1 * h;
y = abs(x1 — x0);
x0 = x1
    end n;
    format (' —12.346 '); line (2);
    print (' a = ', a, ' b = ', b, ' eps = ',
            eps ' wartość całki = ', x1);
    wait (' nowe dane ');
    goto P
end programu
```

Algorytm zakodowany został w dwóch językach: Algolu i w dobrze nam znanym Basicu. Łatwo wyróżnić w nich sekwencje wykonujące te same obliczenia. Algol jest językiem, w którym zapis programu pozostaje w dużej zgodności z „kształtem" algorytmu — bardzo często zamiast rysować schemat działania rozwiązanie problemu zapisuje się wprost w Algolu.

Różnice? Program w tym języku wymaga wcześniejszych deklaracji zmiennych i stałych. **Real** to liczby rzeczywiste **integer** — całkowite. Komenda line (1) powoduje wysunięcie papieru drukarki o jeden wiersz. Inne są oczywiście komendy i formaty instrukcji we/wy.

Właściwości Algolu pozwoliły na warunkowe zorganizowanie pętli:

for n = 2 * n **while** y > eps **do**

instrukcje z jej wnętrza wykonywane są w przypadku gdy y > eps. Aby zapewnić sobie przynajmniej jednokrotne ich wykonanie przed pętlą dokonano podstawienia y = 100 — eps z pewnością jest mniejsze.

W Basicu natomiast warunek ten sprawdzany jest w wierszu — 200 — w ten sposób, bez uprzednich podstawień, obliczenia pierwszego przybliżenia wartości całki będą zawsze wykonane.

```
10 REM Obliczanie całki metodą trapezów
15 PRINT „WARTOŚĆ CAŁKI W PRZEDZIALE < a, b >"
20 INPUT „podaj dolną granicę całkowania"; a
30 INPUT „podaj górną granicę całkowania"; b
35 INPUT „podaj dokładność"; eps
40 LET h = b — a
50 LET n = 1
60 LET s = (EXP(—a↑2) + EXP(—b↑2))/2
70 LET x0 = s * h
90 LET n = 2 * n
100 LET h = h/2
110 LET x1 = s
120 LET x = a
125 REM Początek pętli warunkowej
130 FOR i = 1 TO n — 1
140 LET x = x + h
150 LET x1 = x1 + EXP(—x↑2)
160 NEXT 1
170 LET = x1 = x1 * h
180 LET y = ABS(x1 — x0)
190 LET x0 = x1
200 IF y > = eps THEN GOTO 90
210 PRINT "a = "; a, "b = "; b, "eps = "; eps, "wartość całki = "; x1
220 LET b$ = "TAK"
230 INPUT "czy nowe dane TAK/NIE"; a$
240 IF b$ = a$ THEN GOTO 20
250 STOP
```

A.S.

Jedno z komputerowych pism angielskich nie bez złośliwości stwierdza, że trudno jest podziwiać francuski mikrokomputer bez świadomości, że wytwarzane we Francji produkty są niepowtarzalne: sery, wina, samochody i... komputery.

Tylko Francuzi mogli stworzyć wehikuł (CITROEN 2CV) o uroku, który niesie sobą brak elegancji. Tylko oni mogli rozwinąć mikro typu GOUPIL G4. G4 pracuje pod systemem MS DOS i praktycznie jest kompatybilny z IBM. Oceniając ten mikro, należy mieć pewien pogląd o genealogii maszyny i firmie, która ten komputer zbudowała. SOCIETE DE MICRO — INFORMATIQUE lub prościej SMT International. Firma ta ma zaledwie pięć lat, ale już od ponad dwóch lat stara się zdobyć rynek angielski. Oczywiście, towarem jest mikro typu GOUPIL. Słowo to według średniowiecznej francuskiej legendy oznacza sympatycznego drapieżnika — LISA

Do końca 1983 roku SMT sprzedał nieco poniżej 10 000 tych maszyn. Potem firma zdecydowała się na wspomnianą próbę zdobycia rynku angielskiego. Proponowano tam komputery w trzech wersjach różniących się procesorami: 6809, Z-80, 8088. Oferowane maszyny umożliwiały między innymi wieloprogramowość.

Niestety, wynik był mizerny. Starano się więc wykorzystać bardziej konwencjonalny komputer G4. Wykorzystuje on standardową (nieco zmodyfikowaną) klawiaturę IBM. Klawisze funkcyjne zostały przesunięte z lewej strony nad klawisze alfanumeryczne.

Warto w tym miejscu podkreślić olbrzymie znaczenie usytuowania klawiszy, szczególnie funkcyjnych. Nawet pozornie drobne zmiany przyjętych standardów (praktycznie narzuca je IBM) mogą znacznie utrudnić wykorzystanie komputera. Tak jest w wypadku G4, bowiem część klawiszy jest źle usytuowana; trudno na przykład znaleźć klawisz BREAK.

W mikro G4 zainstalowano dyski typu IBM, a w skład jego wyposażenia wchodzi „myszka" i długo oczekiwany system Microsofta OKNA (WINDOWS). Serce G4 to 16-bitowy mikroprocesor 80186.

80186 jest zastanawiającym wyborem, jest różny bowiem od 8088 wykorzystanym w IBM PC, a jednocześnie ma liczne braki wobec 80286, wykorzystanego w IBM PC AT

G4 realizuje oprogramowanie IBMa, Electric Disck, Chit-Chat i Framework w większości wypadków robi to szybciej niż konwencjonalne maszyny IBM.

Oceniany model ma pamięć 256 KB RAM, twardy dysk 10 MB i kompatybilne do IBM floppy. Typowe uzupełnienie to monitor oraz klawiatura. Jednak o atrakcyjności G4 decydują OKNA Microsofta i doskonale oprogramowana „myszka"

Daje się odczuć, że jakość konstrukcji komputera, jego oprogramowanie wiąże się z odpowiednim przygotowaniem rynku, zaspokojeniem najbardziej wyszukanych potrzeb potencjalnych klientów, łącznie z poradnictwem np. w zakresie instalacji komputera w wybranej firmie.

SMT zdaje sobie sprawę, że nie zdobędzie rynku jedynie niskimi cenami. Jest tu zbyt wielka konkurencja w postaci takich firm jak TANDY, COMMODORE. Goupil oferuje trochę większą szybkość, nieco lepszy styl oraz oczekiwane, bardzo praktyczne OKNA

OKNA są realizacją i jednocześnie zwiastunem nowej, niewątpliwie trwałej tendencji rozwoju i doskonalenia *„interfejsu użytkownika"*. Tych możliwości komputera, które decydują o wzroście *„życzliwości komputera"*, prostszym i bardziej naturalnym sposobie komunikowania się z komputerem.

Komputer G4 jest przyjemny w eksploatacji i nie zaskakuje doświadczonych użytkowników PC, aczkolwiek podczas pracy zestawu pojawia się czasem błąd parzystości.

Jedynie monochromatyczny ekran nieco zaskakuje zbyt małą rozdzielczością, jakiej należałoby oczekiwać w maszynie, która ma się stać popularna właśnie dzięki grafice użytkowej.

Interfejsowe możliwości G4 wydają się standardowe. Oba szeregowe i równoległe porty są wbudowane w tylnej części. Pracują poprawnie z modemami i równoległą drukarką. System twardego dysku jest cichy i szybki — sprawia wrażenie, że to dodatkowa pamięć RAM.

Pamięć 640 KB powinna być wystarczająca dla większości zastosowań. Należy jednak sprawdzić, ile OKNA zostawiają pamięci dla programów użytkowych, przed określeniem ostatecznej konfiguracji G4 (cena 1395 funtów 128 KB lub 4420 funtów 512 KB).

Część przedstawionych powyżej uwag na temat jednego z ostatnich francuskich komputerów tylko pozornie jest nieco różna od informatycznych zainteresowań czytelników „IKS-a". Jutro bowiem istotną kwestią stanie się wybór komputera dla naszego zakładu pracy, wyróżnienie jego podstawowych cech, które decydują o jakości i komforcie pracy maszyny. Wydaje się, że zasadniczym wnioskiem jest uznanie kompatybilności z IBM jako podstawowej cechy profesjonalnego mikrokomputera. Kolejną cechą, którą wyróżniono jest „życzliwość komputera" (friendliness). Mieści się w tym zarówno mechaniczna konstrukcja, jak i oprogramowanie maszyny.

Na przykładzie G4, „życzliwość" swą mikro uzyskuje dzięki bardzo komunikatywnemu systemowi OKNA. Uzupełnieniem tej własności miała być nieudana próba poprawienia konstrukcji klawiatury (rozmieszczenia klawiszy).

Wymowne jest to, że profesjonalne pismo komputerowe bardziej eksponuje składowe „życzliwości komputera", na przykład kiepską jakość monitora, niż szybkość G4, która przekracza możliwości IBM.

Można sądzić, że wkraczamy w nowy etap rozwoju mikrokomputerów, etap dążeń ku poprawie jakości współpracy z komputerami w pracy i w domu.

W.G.



Giełda na Grzybowskiej... wrzesień'86

Foto: Jan Zelman

Jak to zrobić (1)

Zapewne wszystkich fascynuje barwny, ruszający się świat gier. Docieklīwi, zanim wykonają pierwszy ruch joystick'iem, zastanawiają się, JAK TO można ZROBIĆ. Artykuł ten wyjaśnia podstawy najprostszej techniki animacji na ATARI 800, jaką dostarcza grafika PLAYER-MISSILE, odpowiadająca SPRIT'om na innych komputerach.

W związku z tym, że w dalszej części używane będą operacje POKE i PEEK, a nie chcę osobom rozpoczynającym swoją przygodę z komputerem odbierać szansy zrozumienia, rozpocząć muszę od kilku uwag z nimi związanych.

PEEK jest funkcją, która odczytuje wartość bajtu spod wskazanego adresu. Na przykład instrukcje

A = PEEK (704)
B = PEEK (53248)

przypisują zmiennym A i B wartości zapamiętane odpowiednio w komórkach o adresach 704 i 53248.

Odwrotne role odgrywa procedura POKE. Pod adresem występującym w roli jednego z jej parametrów umieszcza ona dowolną wartość z przedziału 0—255, podana jako drugi parametr. Na przykład

POKE 704,160
POKE 53248,100

Przechodząc już do właściwego tematu, należy odpowiedzieć na pytanie, czym jest *player*?

Wyobraźmy sobie pionowy pasek na ekranie. Podobnie jak wszystko, co jest wyświetlane, pasek nasz jest obrazem pewnego kawałka pamięci komputera. Jeżeli teraz (... na razie) uwierzymy, że może się on przesuwać wzdłuż ekranu, zmieniać swoją szerokość i barwę, to będziemy znali najprostszą odpowiedź na to pytanie.

Należy dodać, że dysponujemy czterema *playerami* i czterema *misselami*, z których każdy jest także „paskiem", ale o szerokości 1/4 rozmiaru *playera*.

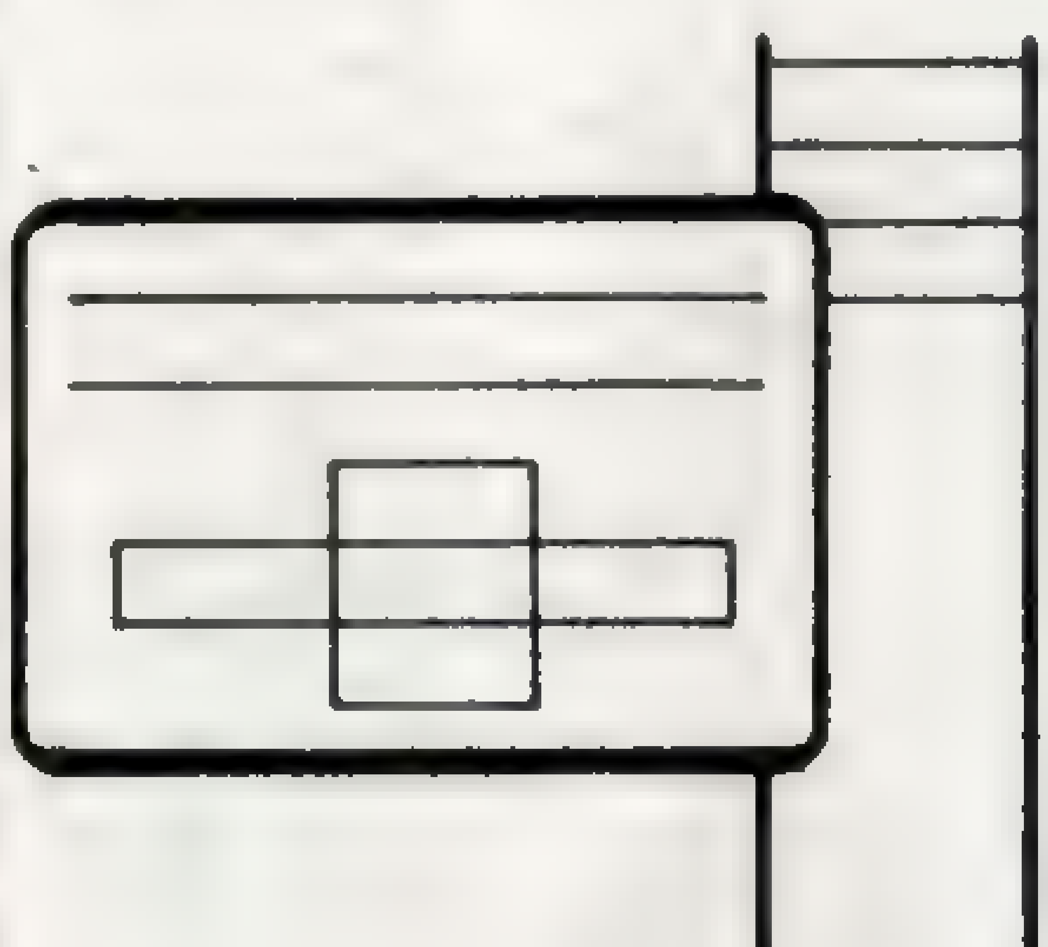
Wielu zapalonych amatorów programowania już wyobraża sobie bajeczne obrazy własnego autorstwa, przesuwające się po ekranie. My jednak postępować powinniśmy systematycznie, a w fazie poznawania możliwości *playerów* powoli, ale dokładnie analizować każdy krok.

Rozszyfrowywanie tajemnic grafiki PLAYER-MISSILE rozpoczniemy od następującego stwierdzenia:

player jest obrazem ciągłego obszaru pamięci.

Fakt ten ma ogromne znaczenie przy definiowaniu kształtu *playera*, co zobaczymy dalej. Rola, jaką on odgrywa, i cała „filozofia" *playerów* szczególnie widoczna jest w momencie analizy sposobu, w jaki komputer wyświetla zadane informacje na ekranie.

Wiemy, że obraz widoczny na monitorze jest odwzorowaniem pewnej części pamięci. W zależności od wybranej grafiki, różne jej komórki mogą być wyświetlane w różnych miejscach ekranu (zależy to od liczby bajtów tworzących linię dla danej grafiki). Różny jest także sposób liczenia adresów bajtów wyświetlanych bezpośrednio wyżej i niżej wybranej pozycji ekranu. Dla grafiki „0" adresy pamięci, tworzącej obraz ekranu, przedstawić można następująco:



Realia te powodują, że stworzenie pionowego obrazu, który mógłby się poruszać, jest, może nietrudne, ale dość skomplikowane i pracochłonne. Ponadto zauważmy, że każdorazowa chęć zmiany grafiki niesie ze sobą niebezpieczeństwo zniszczenia rysunku.

Rozwiązanie tego problemu dostarcza grafika *playerów*. Ciągłość obszaru pamięci przeznaczonej dla ich reprezentacji i jego niezależność od pamięci ekranu zapewniają:

— łatwość dostępu do kolejnych bajtów, odpowiadających następującym po sobie liniom obrazu,
— uniezależnienie *playera* od trybu graficznego tła.

Ponadto komputer dostarcza nam możliwość wyboru jednego z dwóch sposobów odwzorowania pamięci *playera* na ekran różniących się rezolucją, tj. liczbą linii opisanych jednym bajtem. Możemy zażądać, by każdy bajt przeniesiony został na dwie linie ekranu — rezolucja podwójna, ale wyraźniejszy rysunek uzyskujemy przy rezolucji pojedynczej — każdy bajt tworzy jedną linię obrazu.

Do wskazania wybranej rezolucji służy czwarty bit komórki o adresie 559. Brzmi to przerażająco, dlatego należy natychmiast zapomnieć to zdanie, a zapamiętać dwie liczby 62 i 46, które wpisane instrukcją

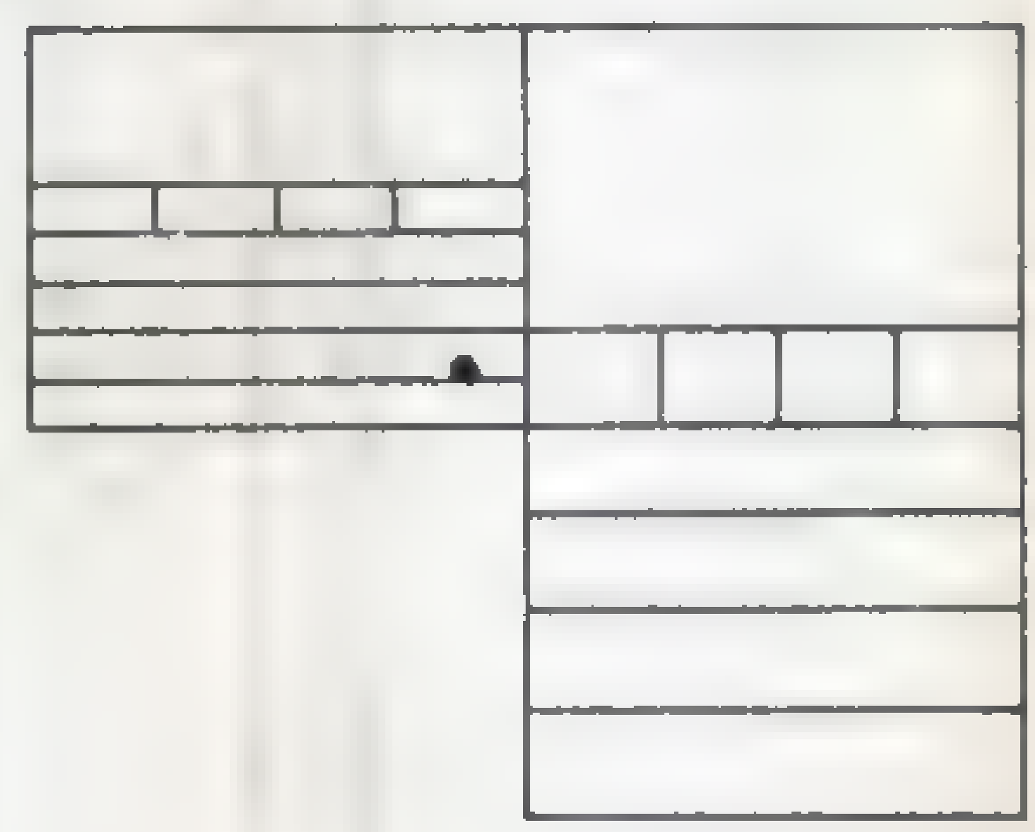
POKE 559,...

zapewniają

62 — rezolucja pojedyncza,
46 — rezolucja podwójna.

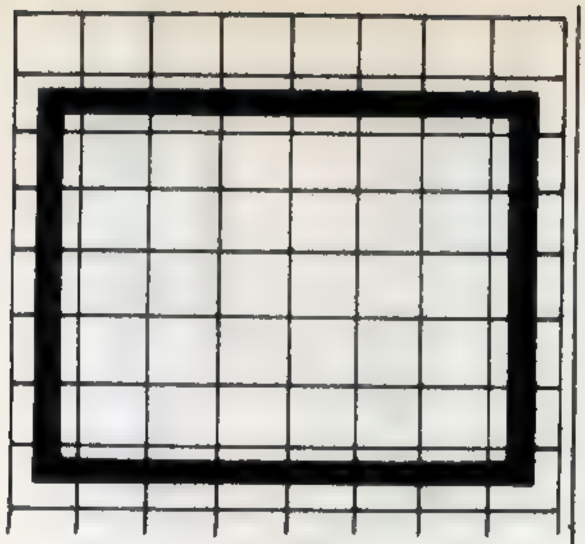
Z rezolucją *playerów* ściśle związana jest wielkość pamięci, potrzebna do ich reprezentacji, dla pojedyńczej wynosi aż 256 bajtów, dla podwójnej tylko połowę, czyli 128 bajtów. Dla poprawnej obsługi *playerów* adres tego pola znajdować się musi w komórce o adresie 54279. Może to być oczywiście liczba nie przekraczająca 255 (dlaczego?), dlatego właściwy adres wyraża ona w postaci bloków po 256 bajtów każdy.

Poniższy rysunek prezentuje schemat omawianej pamięci.



Po tych niezbędnych wyjaśnieniach przystąpić możemy do nadania *playerowi* kształtu. Metoda jest powszechnie znana — rysujemy planowany obraz na siatce o szerokości ośmiu pól, w której kolumny, począwszy od prawej strony, ponumerowane są kolejnymi potęgami dwójki 1,2,4,... Następnie pola, przez które przechodzą linie rysunku, zamieniamy na 1, pozostałe na 0. Dla każdej linii rysunku obliczamy wartość będącą sumą iloczynów wartości pola (1 lub 0) przez wartość kolumny, w której się znajduje (1,2,4,...).

Zasadę ilustruje poniższy przykład:



Szczególną ostrożność należy zachować przy tworzeniu rysunków *misseli*, bowiem jeden bajt pamięci zawiera informację o postaci pojedynczej linii każdego z nich. I tak na powyższym rysunku kolumny (bity) z numerami

1,2 — zawierają informacje dotyczące *missela* 0,
4,8 — dotyczą *missela* 1,
16,32 — *missela* 2, zaś
64 i 128 — *missela* 3.

Nie zmienia się sposób liczenia odpowiednich wartości.

Liczby, które w opisany sposób utworzyliśmy, musimy teraz umieścić we właściwych miejscach pamięci. Przyjrzyjmy się następującemu programowi, umieszczającemu pokazany wyżej prostokąt w *playerze* 0.

```
10 REM DEFINIOWANIE RYSUNKU
   PLAYERA — PROSTOKĄT
11 REM MAMY NADZIEJĘ, ŻE
   PAMIĘĆ PLAYERA JEST
   „CZYSTA",
12 REM W PRZECIWNYM
   PRZYPADKU NALEŻY UMIEŚCIĆ
   W NIEJ STOSOWNĄ
13 REM LICZBĘ ZER
20 AD = PEEK (106)—16
30 POKE 106, AD—16:REM
   REZERWUJEMY PAMIĘĆ DLA
   PLAYERÓW
40 POKE 54279, AD:REM
   INFORMUJEMY SYSTEM GDZIE
   ZNAJDUJĄ SIĘ
   OPISY PLAYERÓW
41 REM
50 POKE 53277,3:REM „WŁĄCZAMY"
   GRAFIKĘ P—M
60 POKE 559,62: REM WYBIERAMY
   REZOLUCJE OBRAZU
70 START = AD 256 + 1024 + 100
80 FOR I = START TD START + 5:
   REM PĘTLA WPISUJĄCA DANE
90 READ B
100 POKE I,B
110 NEXT I
120 DATA 255,129,129,129,129,255
```

Wyjaśnienia wymagają instrukcje w następujących liniach:

50 — „włączenie" polega na wskazaniu systemowi, jakie elementy mają być uwidocznione na ekranie; wstawiając do komórki 53277 wartość 1, pozwalamy na przeniesienie na ekran obrazu misseli, — wartość 2 — obrazu playerów; w celu wykorzystania wszystkich możliwości, należy posłużyć się sumą tych wartości — stąd liczba 3.

70 — obliczanie adresu pierwszego bajtu, od którego należy rozpocząć wpisywanie definicji linii playera. Liczba 100 określa na ekranie linię, od której pokaże się nasz prostokąt. Poniżej przedstawione zostały numery linii zapewniające widzialność obrazka na monitorze. Podane liczby dotyczą rezolucji pojedynczej, w nawiasach znalazły się ich odpowiedniki dla rezolucji podwójnej.



Z powyższego rysunku wiemy, jakimi wartościami możemy opisywać poziome ustawienie *playera*. Zobaczymy teraz, jak to zrobić. Dla ułatwienia zapoznajmy się od razu z następującymi komórkami pamięci, które będą nam jeszcze potrzebne. Dla potrzeb naszego programu są to adresy dotyczące playera 0.

53248 — przechowuje wartość określającą poziome pozycje *playera*,
704 — definiuje jego kolor, jest to liczba postaci 16 (kod koloru) + (jasność)
53256 — opisuje jego szerokość, w sposób następujący:
0,2 — normalna, to znaczy jednemu bitowi z każdego bajtu *playera* odpowiada jeden punkt w linii ekranu, ma on szerokość 8 punktów (uff, to było trudne wyjaśnienie),
1 — podwójna, rozumieć należy w sposób oczywisty,
3 — poczwórna.

Natychmiast wykorzystajmy nowo zdobytą wiedzę do rozszerzenia programu o następne instrukcje:

```
130 POKE 53248,100:REM
    USTAWIENIE POZYCJI
    POZIOMEJ
140 POKE 704,64:REM NIECH
    BĘDZIE CZERWONY
150 POKE 53256,0:REM
    SZEROKOŚĆ NORMALNA
```

Dalej wzbogacić go możemy instrukcjami powodującymi poziomy przesuw, zgodny z ruchami joysticka. Oto one

```
160 HPOZ = 100:REM USTALENIE
    POZYCJI POCZĄTKOWEJ
170 IF STICK(0) = 11 THEN
    HPOZ = HPOZ—1:REM W
    LEWO
180 IF STICK(0) = 7 THEN
    HPOZ = HPOZ + 1:REM W
    PRAWO
190 POKE 53248,HPOZ
200 GOTO 170
```

Kryją one jednak poważny błąd (oprócz tego, że tworzą nieskończoną pętlę). Proszę spróbować przesunąć prostokąt poza ekran (np. w prawo), zaobserwować efekt i wytłumaczyć go. Powinienem przy tym wytłumaczyć się, że nie zapomniałem o *misselach*, ani o pozostałych *playerach*. Po prostu uprościłem sobie problem, wiedząc, że operacje dotyczące wszystkich elementów grafiki P—M są jednakowe i przedstawiłem je na przykładzie *playera* 0. Jest to jednak moim obowiązkiem podać adresy wszystkich komórek, które mogą mieć znaczenie w trakcie programowania z wykorzystaniem przedstawionej techniki. Oto one:

704 kolor *playera* i *missela* 0,
705 kolor *playera* i *missela* 1,
706 kolor *playera* i *missela* 2,
707 kolor *playera* i *missela* 3,
53248 pozioma pozycja *playera* 0,
53249 pozioma pozycja *playera* 1,
53250 pozioma pozycja *playera* 2,
53251 pozioma pozycja *playera* 3,
53252 pozioma pozycja *missela* 0,
53253 pozioma pozycja *missela* 1,
53254 pozioma pozycja *missela* 2,
53255 pozioma pozycja *missela* 3,
53256 rozmiar *playera* 0,
53257 rozmiar *playera* 1,
53258 rozmiar *playera* 2,
53259 rozmiar *playera* 3,
53260 rozmiar wszystkich *misseli*, jest to liczba tworzona jako suma wartości opisujących szerokość każdego z nich.

Na tym chciałbym zakończyć rozważania dotyczące playerów. Zachęcam wszystkich do wpisania tych kilku zaprezentowanych instrukcji i przeprowadzenia eksperymentów. Życzę przyjemnej zabawy i powodzenia przy tworzeniu swoich własnych ruszających się obrazków.

GRZEŚ

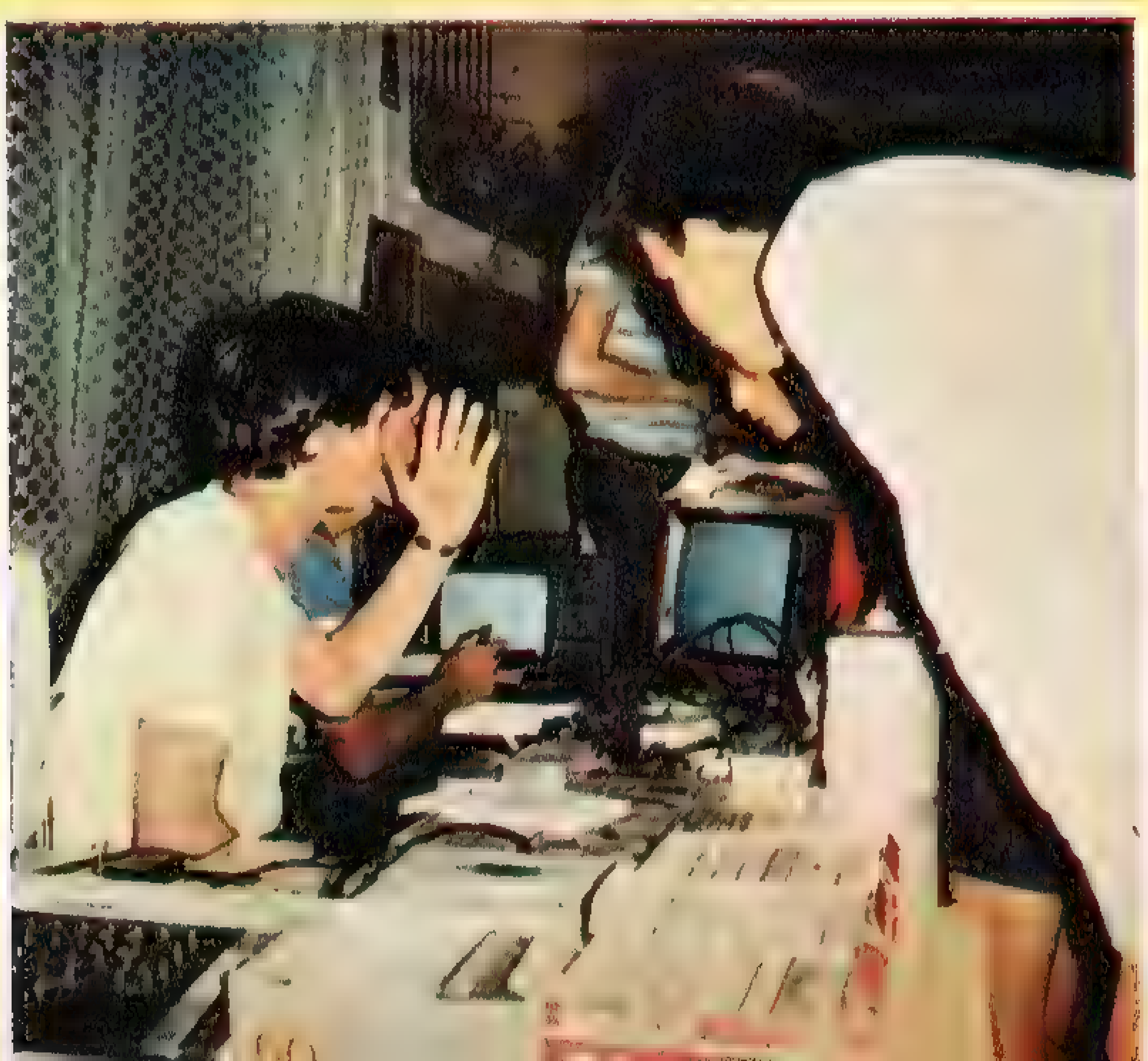


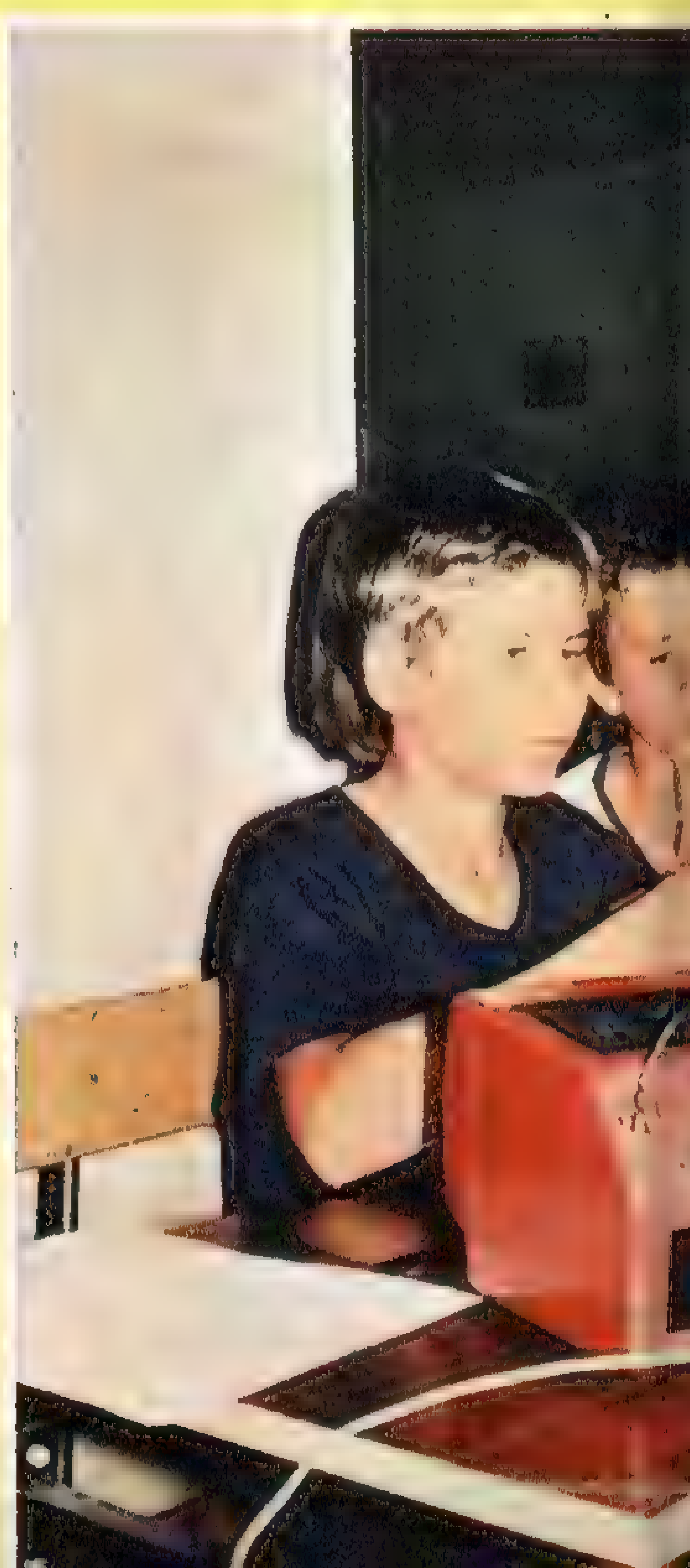


„SZKOŁA" E KOM

(czyli giełda n









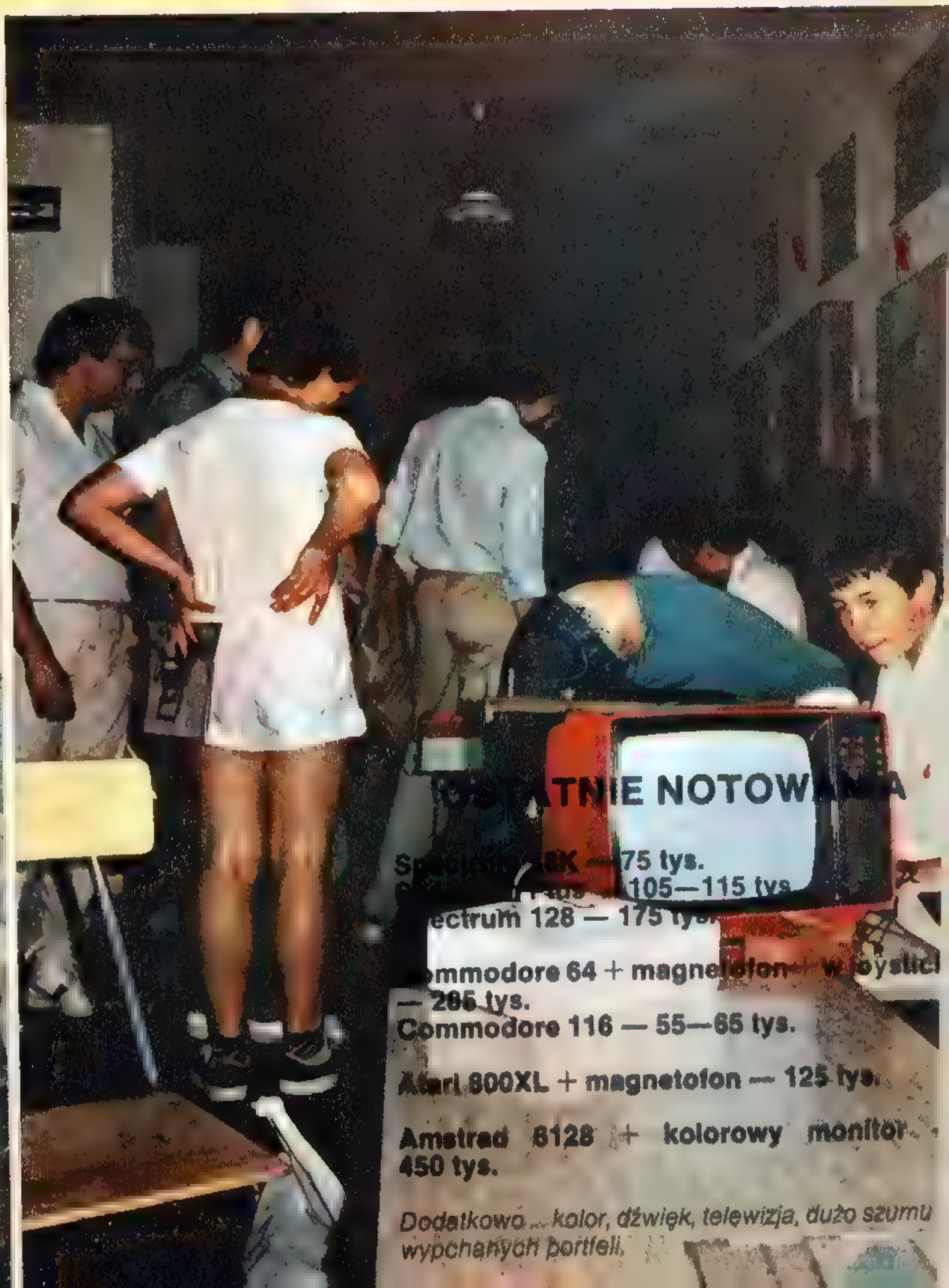


...DUKACJI... ...MPUTEROWEJ

...na Grzybowskiej)

Foto: Ryszard Rogoń





...ATNIE NOTOW...

Sp... ...K — ...75 tys.
... — 105—115 tys.
...ectrum 128 — 175 tys.

...ommodore 64 + magne... + ...oystick — 2[illegible]5 tys.
Commodore 116 — 55—65 tys.

Atari 800XL + magnetofon — 125 tys.

Amstrad 6128 + kolorowy monitor — 450 tys.

Dodatkowo ... kolor, dźwięk, telewizja, dużo szumu wypchanych portfeli.

BASIC

(Podstawy programowania)

Kontynuujemy wykład z programowania w języku BASIC na mikrokomputer ZX Spectrum.

W wykładzie tym podamy podstawowe wiadomości i zasady tworzenia cykli obliczeniowych, zwanych także pętlami. Omówimy odpowiednie instrukcje BASIC-u, za pomocą których zapisujemy cykle obliczeniowe (pętle). Wiadomości te będą uzupełniane w następnych wykładach.

Wykład czwarty: cykle obliczenowe (pętle) zapisywanie pętli w programach w BASICu z wykorzystaniem — instrukcji **IF** — instrukcji cyklu **FOR I NEXT**.

W poprzednim wykładzie (wykład 3) zapoznaliśmy się między innymi z instrukcją IF. Stosowaliśmy ją np. w celu sprawdzenia odpowiedzi na pytanie: *„czy wprowadzamy dalej?"*. Jeżeli odpowiedzieliśmy *„tak"*, to wykonując instrukcję skoku bezwarunkowego (GO TO n) powracaliśmy do wykonania instrukcji o numerze n i powtarzaliśmy cykl obliczeń zapisany kilkoma instrukcjami. I tak np. w programie z przykładu 3.5 instrukcje o numerach 30—80 są wykonywane tyle razy, ile razy napisaliśmy *„tak"*.

Omówimy teraz zapisywanie za pomocą instrukcji IF cykli obliczeniowych zwanych także pętlami. Jak zwykle zaczynamy od przykładu.

Przykład 4.1

Napisać program na obliczenie sumy 5 liczb (dowolnych) wczytywanych w trakcie realizacji programu. Napisany program ma postać:

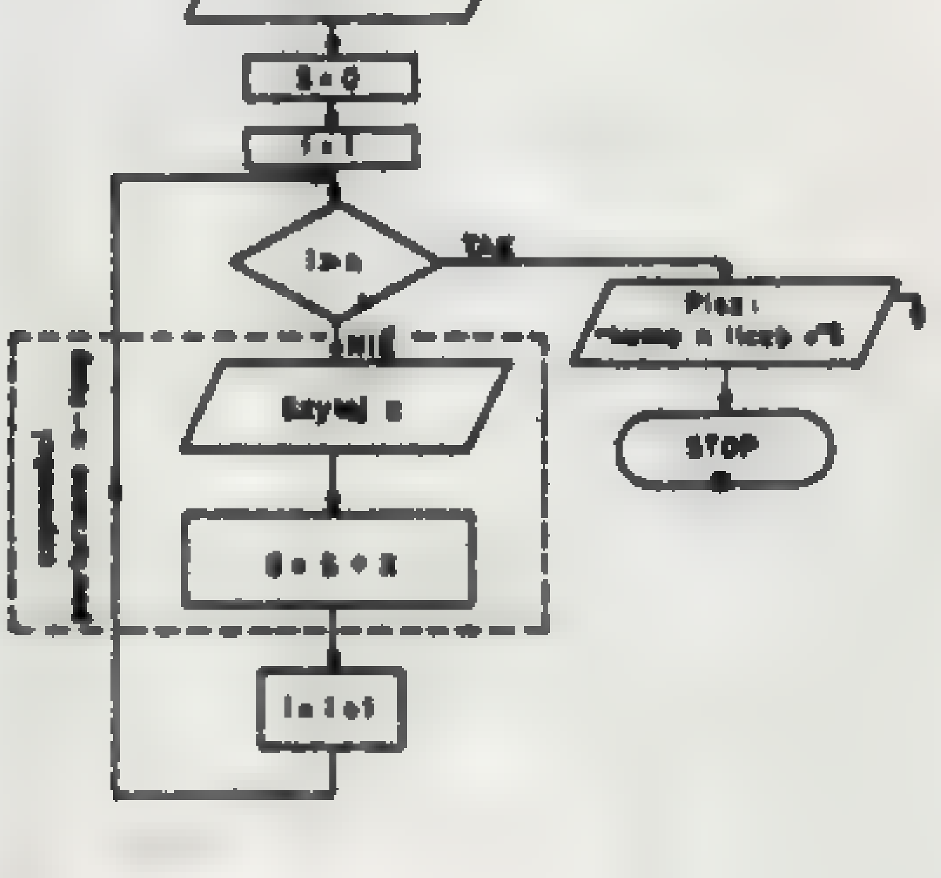
```
10 REM Przyklad 4.1
15 REM s-suma 5 liczb
     x-kolejna czytana liczba
20 LET s=0
25 INPUT "x=";x: PRINT x
30 LET s=s+x
35 INPUT "x=";x: PRINT x
40 LET s=s+x
45 INPUT "x=";x: PRINT x
50 LET s=s+x
55 INPUT "x=";x: PRINT x
60 LET s=s+x
65 INPUT "x=";x: PRINT x
70 LET s=s+x
75 PRINT "suma 5 liczb = ";s
```

Napisanie takiego programu, gdy chcemy zsumować więcej, np. 100 liczb, byłoby bezsensowne. Sumowanie dowolnej ilości liczb można przeprowadzić za pomocą programu, w którym występują instrukcje wielokrotnie wykonujące te same obliczenia dla różnych wartości danych.

Przykład 4.2

Napisać program obliczający sumę *n* liczb (dowolnych) wczytywanych w trakcie realizacji programu. W programie należy skorzystać z instrukcji **IF.**

a) schemat blokowy:



b) program

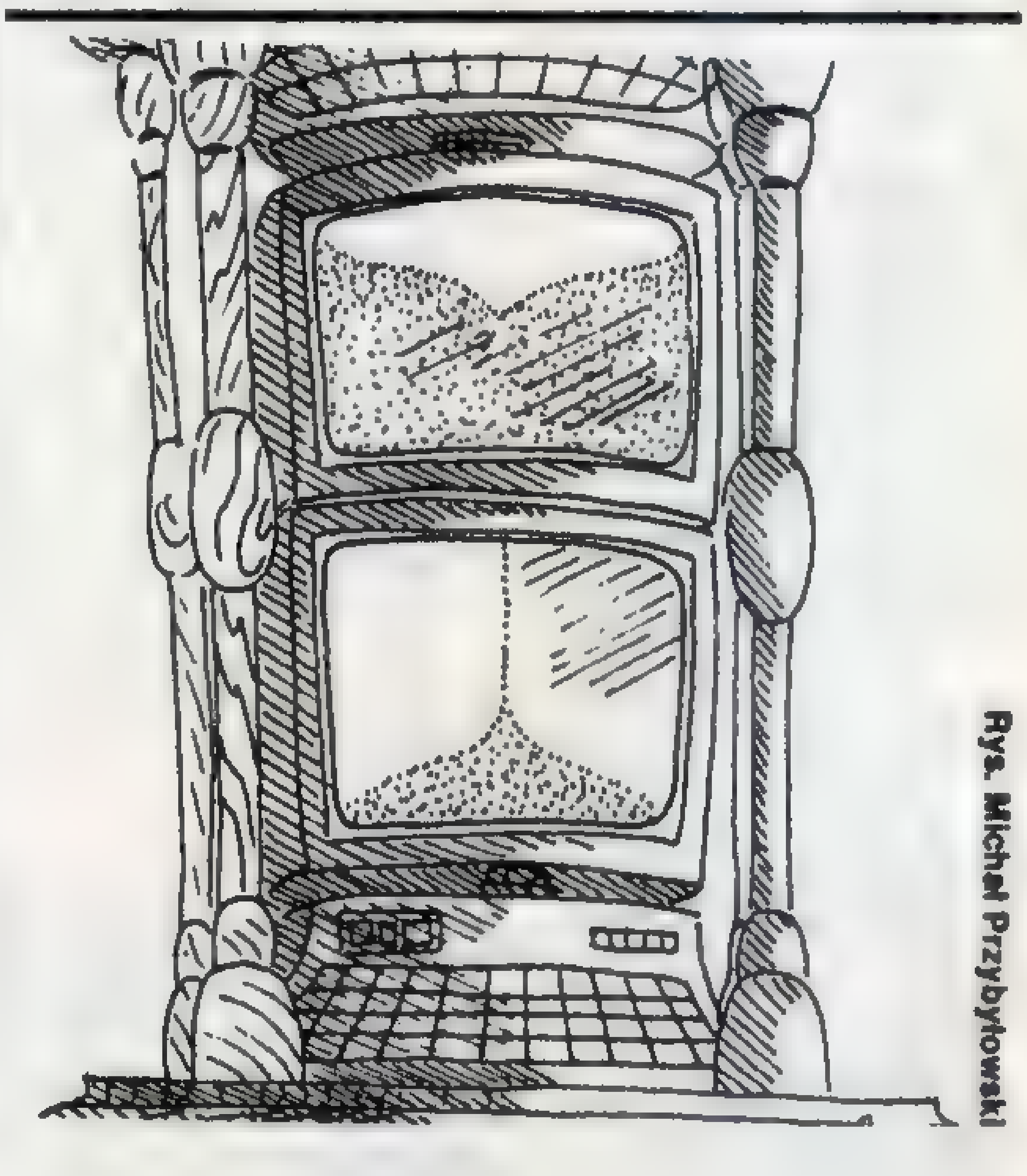
```
10 REM Przyklad 4.2
15 REM n-ilosc wczytywanych
     liczb
     x-kolejna czytana liczba
     i-licznik,s-suma n liczb
20 INPUT "n= ";n
25 PRINT "Obliczanie sumy ";n;
liczb"
30 PRINT '"wczytane liczby :"
35 LET s=0
40 LET i=1
45 IF i>n THEN GO TO 80
50 INPUT x
55 PRINT x;"  ",
60 LET s=s+x
65 LET i=i+1
70 GO TO 45
80 PRINT ''"suma ";n;" liczb =
";s
```

c) wyniki dla przykładowych danych

```
Obliczanie sumy 5 liczb
wczytane liczby :
2  3  5  6  8
suma 5 liczb = 24
```

Zauwazmy, ze jeżeli rozwiązywanie problemu zawiera pewien proces powtarzalny, to w schemacie blokowym pojawia się pętla (cykle obliczeniowe). Pętla musi zawierać określenie sposobu jej zakończenia. Czynność polegająca na wykonaniu każdej operacji w jednym przebiegu pętli nazywa się cyklem pętli.

Zajmiemy się teraz analizą schematu blokowego z przykładu 4.2. Czynności opisane blokami: czytanie *x* i sumowanie ($S = s + x$) mają się wykonać *n*-razy dla różnych wartości *x* (w tym miejscu wartość *n* jest już znana). Wykonanie tych czynności dla jednej wartości *x* to cykl pętli.



Rys. Michał Przybyłowski

Zapis w tym bloku odczytujemy w następujący sposób: jeżeli warunek $i > n$ jest spełniony, to przejdź do wypisania wartości sumy n-liczb; jeżeli jest nie spełniony, to powtórz obliczenia tzn.:

— czytaj x
— dodaj x do sumy ($s = s + x$)
— dodaj do i liczbę 1; obliczoną wartość podstaw pod zmienną i ($i = i + 1$)
— powróć do pytania $i > n$

W tym przykładzie zmienna i służy do zliczania ilości liczb, a zarazem zlicza liczbę wykonanych cykli w pętli, można także powiedzieć, że zmienna i odgrywa również rolę licznika „obrotów" pętli; jej wartość zmienia się o stałą liczbę równą 1. Po zakończeniu obliczeń, tzn. w razie spełnienia nierówności $i > n$, wartość zmiennej i wynosi $n + 1$.

Zapiszemy teraz program z przykładu 4.2 w postaci:

```
10 REM Przyklad 4.2.1
15 REM n-ilosc wczytywanych
       liczb
     x-kolejna czytana liczba
     s-suma n liczb
20 INPUT "n= ";n
25 PRINT "Obliczanie sumy ";n;
liczb"
30 PRINT '"wczytane liczby :"
35 LET s=0
40 FOR i=1 TO n STEP 1
45 REM i-zmienna sterujaca
50 INPUT x
55 PRINT x;"  ";
60 LET s=s+x
70 NEXT i
80 PRINT ''"suma ";n;" liczb =
";s
```

Wprowadźmy i wykonajmy ten program dla tych samych liczb co program z przykładu 4.2. Otrzymane wyniki są takie same jak poprzednio.

Do zapisu pętli w BASICu przeznaczone są dwie specjalne instrukcje zwane instrukcjami cyklu (lub instrukcjami powtórzeń), są to instrukcje czynne FOR i NEXT. Są one zawsze używane razem i mają dwie postacie ogólne:

n_1 FOR $z_p = w_1$ TO w_2 STEP w_3
:
n_2 NEXT z_p

lub

n_1 FOR $z_p = w_1$ TO w_2
:
n_2 NEXT z_p

gdzie: n_1, n_2 — numery wierszy

z_p — zmienna sterująca (lub zmienna kontrolowana, lub wskaźnik pętli) jest zmienną prostą numeryczną, nazwa zmiennej sterującej musi być jednoliterowa,

w_1 — wyrażenie arytmetyczne określające wartość początkową zmiennej sterującej,

w_2 — wyrażenie arytmetyczne określające wartość końcową zmiennej sterującej,

w_3 — wyrażenie arytmetyczne określające wartość kroku zmiany zmiennej sterującej,

w_1, w_2, w_3 — nazywane są parametrami cyklu.

Jeżeli krok zmiany zmiennej sterującej ma wartość 1 ($w_3 = 1$) to fragment STEP w_3 może być pominięty. Instrukcja FOR przyjmuje wtedy postać drugą.

Przykłady instrukcji FOR:

a) 100 FOR i = 1 TO n
:
140 NEXT i

b) 60 FOR k = i + k TO 5 * i + 2 * k + 1 STEP 2
:
160 NEXT k

c) 70 FOR j =−25 TO 20
170 NEXT j

d) 50 FOR x = $x1$ TO $x2$ STEP $x3$
100 NEXT x

e) 100 FOR l = 1 TO n STEP k
150 NEXT l

W przykładach b, d, e, przed wejściem do pętli muszą być znane wartości i, $x1$, $x2$, $x3$, oraz l, n, k.

Instrukcje cyklu umożliwiają proste zapisanie w programie tej części schematu blokowego, którą opisywaliśmy za pomocą instrukcji warunkowej IF. W naszym przykładzie zamiast instrukcji:

40 LET i = 1
45 IF i > n THEN GO TO 80

oraz

65 LET i = i + 1
70 GO TO 45

napisaliśmy instrukcje:

40 FOR i = 1 TOW STEP 1

oraz

70 NEXT i

Przy tworzeniu pętli można wyróżnić pięć grup czynności, z których każda została opisana w postaci odpowiedniego bloku na powyższym schemacie.

Wyjaśnień dodatkowych wymaga blok 1.

Parametry cyklu w_1, w_2, w_3 są wyrażeniami arytmetycznymi, a więc mogą być to stałe numeryczne nazwy zmiennych numerycznych lub wyrażenia arytmetyczne. Przed wejściem do pętli muszą być nadane wartości zmiennym numerycznym wchodzącym w skład wyrażeń arytmetycznych, reprezentujących parametry cyklu w_1, w_2, w_3.

Zapamiętajmy pewne reguły obowiązujące przy pisaniu programów z instrukcjami FOR i NEXT:

— instrukcję postaci:

n FOR $z_p = w_1$ TO w_2 STEP w_3

piszemy w miejscu, które odpowiada na schemacie blokom o numerach 2 i 3 (uprzednio należy ustalić wartości wyrażeń w_1, w_2, w_3, blok numer 1).

— instrukcję postaci:

n_1 NEXT z_p

piszemy w miejscu, które odpowiada na schemacie blokowi o numerze 5.

— blok o numerze 4, to ciąg instrukcji, które mają być wykonane tyle razy, ile wskazują parametry cyklu.

Tak więc realizację podanego schematu można w BASICu zapisać ciągiem instrukcji postaci:

```
40 REM instrukcje przed petla,
       blok P1
50 REM nadanie wartosci zmien-
       nym wystepujacym w para
       metrach cyklu (jezeli
       wystepuja)
60 FOR z=w1 TO w2 STEP w3
70 REM ciag instrukcji -
80 REM wnetrze petli
90 NEXT z
100 REM instrukcje po petli,
       blok P2
```

Ten sam ciąg instrukcji zapisany w BASICu za pomocą instrukcji IF wygląda następująco:

```
40 REM instrukcje przed petla,
       blok P1
50 REM nadanie wartosci zmien-
       nym wystepujacym w para
       metrach cyklu (jezeli
       wystepuja)
55 REM G, H - zmienne dostepne
       jedynie translatorowi
       jezyka BASIC
60 LET z=w1
62 LET G=w2
64 LET H=w3
66 IF SGN (H)*(z-G)>0 THEN GO
TO 100
70 REM ciag instrukcji -
80 REM wnetrze petli
90 LET z=z+H
92 GO TO 66
100 REM instrukcje po petli,
       blok P2
```

dokończenie na str. 20



dokończenie ze str. 19

Z podanego ciągu instrukcji wynika, że na wykonanie instrukcji cyklu FOR—NEXT składają się następujące czynności:

1. obliczenie wartości wyrażenia w_1 i nadanie zmiennej sterującej z wartości początkowej (wiersz 60),
2. obliczenie wartości wyrażeń w_2 i w_3 (z uwzględnieniem nowej wartości zmiennej) oraz zapamiętanie wartości końcowej w_2 i kroku w_3 pod zmiennymi G i H niedostępnymi dla programu użytkownika (wiersz 62 i 64),
3. sprawdzenie warunku niewykonywania cyklu (wiersz 66), przy spełnieniu tego warunku następuje przejście do instrukcji występującej bezpośrednio po instrukcji NEXT.(SGN *(x)*) jest to funkcja standardowa — podaje ona znak zmiennej *x*, wartości wykonania tej funkcji wynoszą (—1) jeżeli $x < 0$; 0 jeżeli $x = 0$; 1 jeżeli $x > 0$),
4. wykonanie ciągu instrukcji będącego wnętrzem pętli,
5. nadanie nowej wartości zmiennej sterującej (wiersz 90),
6. przejście do wykonywania następnego cyklu (wiersz 92).

Z podanych czynności wynika następujące działanie instrukcji cyklu:

— jeżeli wartość kroku jest większa od zera ($w_3 > 0$), to wartość początkowa w_1 musi być nie większa od wartości końcowej w_2 ($w_1 \leq w_2$),

— jeżeli wartość kroku jest mniejsza od zera ($w_3 < 0$) to wartość początkowa w_1 musi być mniejsza od wartości końcowej w_2 ($w_1 \geq w_2$)

W przeciwnym przypadku cykl nie jest wykonywany, a zmienna sterująca przyjmuje wartość początkową. Po wykonaniu instrukcji cyklu wartość zmiennej sterującej jest większa od w_2 o krok w_3 tzn. $z = w_2 + w_3$ (patrz: wydruk do przykładu 4.3).

Podaliśmy obszerną porcję informacji teoretycznych, postaramy się je zapamiętać, wykonując kilka przykładów.

Przykład 4.3

Danych jest 15 liczb. Należy wczytywać po jednej liczbie w czasie realizacji programu oraz drukować tę liczbę, jej kwadrat i sześcian.

Program jest prosty i podajemy go bez schematu blokowego.

```
 10 REM Przyklad 4.3
 20 REM x-czytana liczba
 30 PRINT "    x     ";"    x↑2
";"   x↑3   "
 40 FOR i=1 TO 15 STEP 1
 50 INPUT "x= ";x
 60 PRINT "    ";x;"      ";"     "
;x*x;"     ";"     ";x*x*x;"   "
 65 REM liczby moga byc dowolne
        ujemne,rowne zero lub
        dodatnie,a wiec nie moze
        my zastosowac operatora
        potegowania ( ↑ )

 70 NEXT i
 80 PRINT "i=";i
```

Wprowadzmy i wykonajmy ten program. Niżej prezentujemy otrzymane wyniki dla przykładowych danych:

```
     x          x↑2         x↑3
    -1           1         -1
    -2           4         -8
    -3           9         -27
     0           0          0
     2           4          8
     3           9          27
     4           16         64
     5           25         125
     6           36         216
     7           49         343
     8           64         512
     9           81         729
     6           36         216
     5           25         125
     4           16         64
i=16
```

Patrząc na programy z przykładów 4.2.1 i 4.3 **zauważamy, że instrukcje cyklu zostały zastosowane do zliczenia powtórzeń wykonywania ciągu instrukcji.** Można powiedzieć, że jest to podstawowa funkcja tych instrukcji.

W przykładzie 4.2 instrukcje cyklu spowodują wykonanie ciągu instrukcji (wiersze 50, 55, 60) *n*-razy, (wartość *n* jest już w tym momencie znana), a w przykładzie 4.3 — wykonanie ciągu instrukcji (wiersze 50, 60) — 15 razy.

Przykład 4.4

Napisać program na mnożenie danej liczby *x* przez 10 kolejnych liczb naturalnych, poczynając od zadanej liczby *l*.

Program jest prosty i podajemy go bez schematu blokowego.

```
10 REM Przyklad 4.4
20 INPUT "x= ";x
30 INPUT "l= ";l
40 FOR j=l TO l+10-1 STEP 1
50 PRINT x;" * ";j;" = ";j*x
60 NEXT j
```

Wprowadźmy i wykonajmy ten program. Niżej prezentujemy otrzymane wyniki dla przykładowych danych

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

W programie tym ciąg instrukcji powtarzanych (wnętrze pętli) składa się z jednej instrukcji

70 PRINT x; "*" ; *j* ; "=" ; *j* * x

W instrukcji tej występuje zmienna *j*, która jest zmienną sterującą instrukcji cyklu napisanej w wierszu 40. Wykorzystujemy tu jej wartość. **W ciągu instrukcji powtarzanych (we wnętrzu pętli) można wykorzystywać wartość zmiennej sterującej, ale nie należy zmieniać jej wartości.**

We wnętrzu pętli można zmieniać (ale się nie zaleca) wartość zmiennej sterującej. Nowa wartość nadana tej zmiennej zostanie uwzględniona przy kolejnym obliczaniu nowej wartości zmiennej sterującej i sprawdzaniu z wartością końcową.

Przykład 4.5

Dany jest program. W programie tym w linii 70 następuje zwiększenie wartości zmiennej sterującej o 3:

```
40 REM Przyklad 4.5
50 FOR i=1 TO 18
60 PRINT i,
70 LET i=i+3
80 PRINT i
90 NEXT i
```

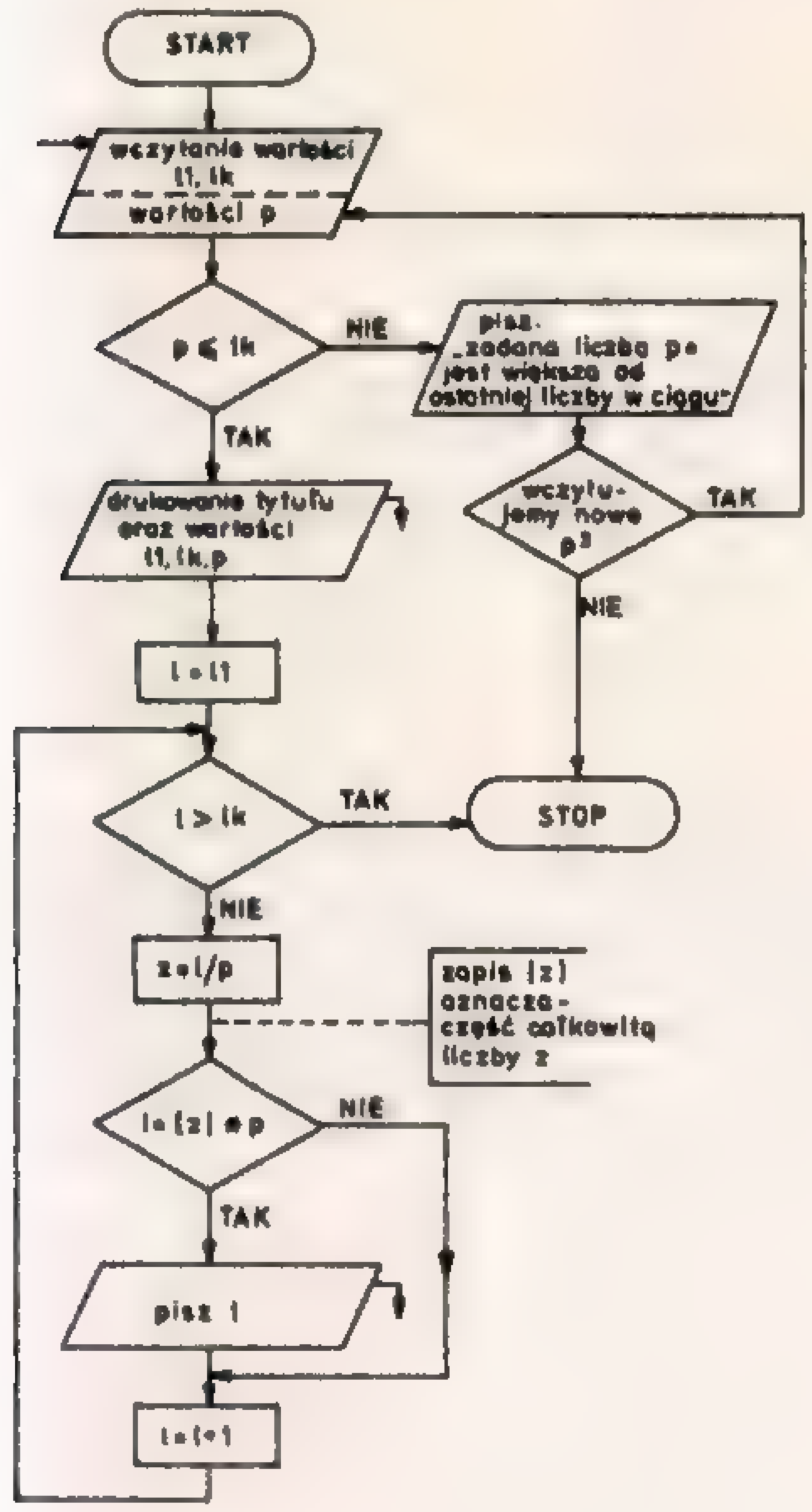
Instrukcje wnętrza pętli wykonane zostaną tylko 5 razy zamiast 18. Wprowadźmy ten program i wykonajmy. Sprawdźmy wynik z wynikiem podanym niżej.

```
1           4
5           8
9           12
13          16
17          20
```

Przykład 4.6

Dany jest ciąg liczb naturalnych $l_1, l_2 ... l_k$. Napisać program na wybieranie spośród tych liczb i drukowanie liczb podzielnych (bez reszty) przez zadaną liczbę p.(p-naturalne, o wartości nie większej od wartości maksymalnej liczby w ciągu tzn. $p \leq l_k$)

a) schemat blokowy



b) program

```
  10 REM Przyklad 4.6
  15 PRINT "wybieranie z danego
ciagu liczb naturalnych liczb po
dzielnych   przez zadana liczbe
p"
  20 REM l1-pierwsza liczba w
         ciagu
         lk-ostatnia liczba w
         ciagu
         p-zadana liczba
  25 INPUT "l1= ";l1,"lk= ";lk
  30 INPUT "p= ";p
  40 IF p<=lk THEN GO TO 100
  50 PRINT '"podana liczba p= ";
p'"jest wieksza od ostatniej
  liczby w ciagu = ";lk
  60 PRINT '"wczytujemy nowe p(t
ak/nie)?";
  70 INPUT x$: PRINT x$
  80 IF x$="tak" THEN GO TO 30
  90 STOP
 100 PRINT '"liczby podzielne pr
zez ",p'"w ciagu ";l1,"......";l
k
 105 PRINT '
 110 FOR l=l1 TO lk STEP 1
 120 LET z=l/p
 130 IF l=INT (z)*p THEN PRINT l
"   ";
 140 NEXT l
 150 STOP
```

c) wyniki dla przykładowych danych

```
wybieranie z danego ciagu liczb
naturalnych liczb podzielnych
przez zadana liczbe p

liczby podzielne przez 3
w ciagu 6......30

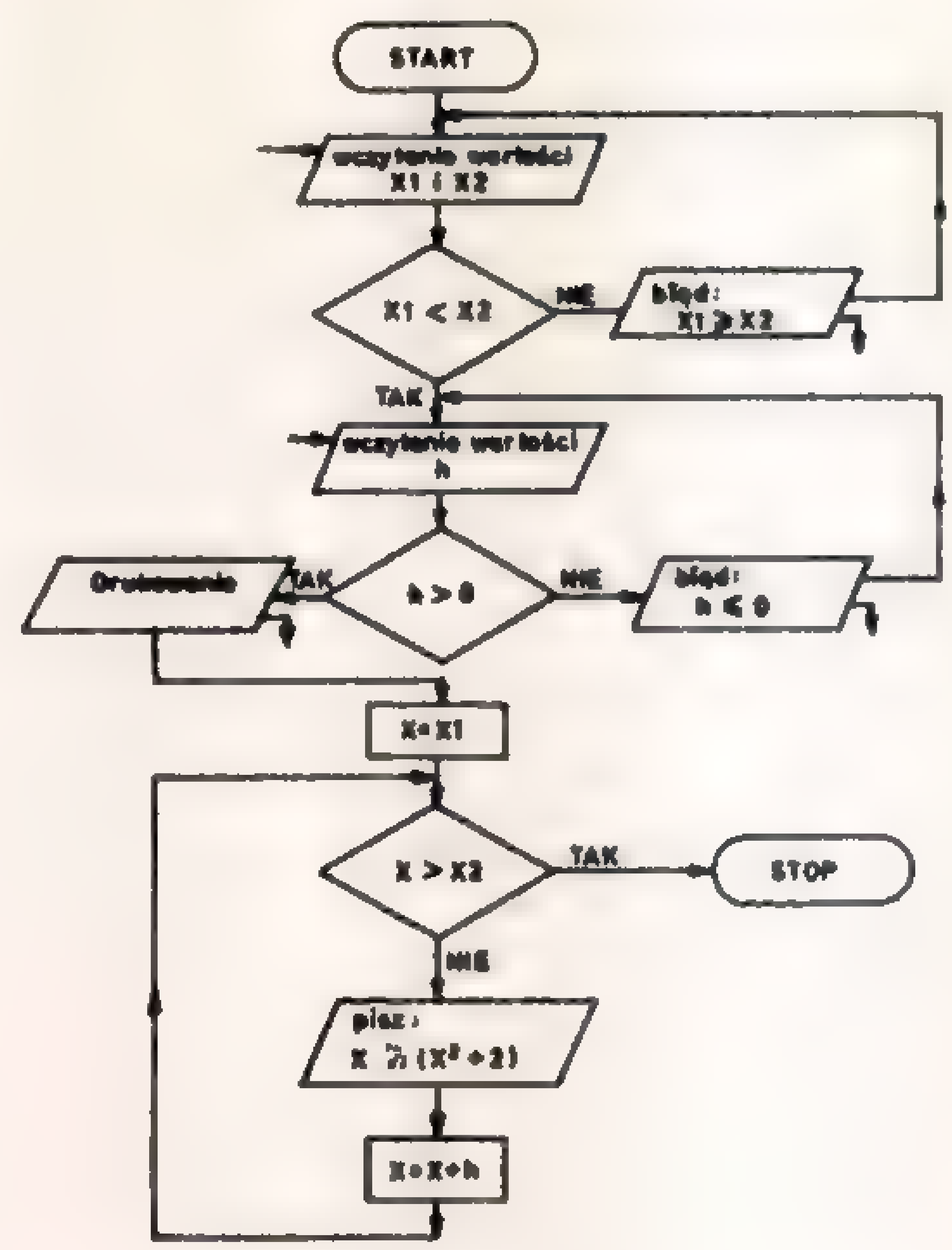
6   9   12   15   18   21   24   27   30
```

Przykład 4.7

Napisać program drukowania tablicy wartości funkcji $ln\ (x^2+2)$ dla argumentu x zmieniającego się w granicach od $x1$ do $x2$ z krokiem h. Wartości $x1$, $x2$, h wczytywane są w czasie realizacji programu. Należy sprawdzić czy spełniają one warunki:

$$x1 < x2 \text{ i } h > 0$$

a) schemat blokowy




```
  10 REM Przyklad 4.7
  15 PRINT "Tablicowanie funkcji
 LN(x↑2+2)  w przedziale <x1,x2>
 z krokiem h"
  20 INPUT "x1= ";x1,"x2= ";x2
  30 IF x1<x2 THEN GO TO 60
  40 PRINT '"zle wartosci x1 i x
2 - x1>=x2"'"x1 = ";x1,"x2 = ";x
2
  50 GO TO 20
  60 INPUT "h= ";h
  70 IF h>0 THEN GO TO 95
  80 PRINT '"zla wartosc kroku h
 - h<=0"'"h = ";h
  90 GO TO 60
  95 PRINT '"Dane: ";"x1 = ";x1;
   x2 = ";x2;"   h = ";h
 100 PRINT '"     x"," Ln(x↑2+2)"
 110 FOR x=x1 TO x2 STEP h
 120 PRINT x,LN (x*x+2)
 130 NEXT x
 140 STOP
```

dokończenie na str. 22

dokończenie ze str. 21

c) wyniki dla przykładowych danych

```
Tablicowanie funkcji LN(x↑2+2)
w przedziale <x1,x2> z krokiem h

Dane: x1 = 1   x2 = 5   h = 1

   x              Ln(x↑2+2)
1                1.0986123
2                1.7917595
3                2.3978953
4                2.8903718
5                3.2958369
```

oraz

```
Tablicowanie funkcji LN(x↑2+2)
w przedziale <x1,x2> z krokiem h

zle wartosci x1 i x2 - x1>=x2
x1 = 5               x2 = 1

zla wartosc kroku h - h<=0
h = -1

Dane. x1 = 1   x2 = 5   h = 1

   x              Ln(x↑2+2)
1                1.0986123
2                1.7917595
3                2.3978953
4                2.8903718
5                3.2958369
```

Przykład 4.8

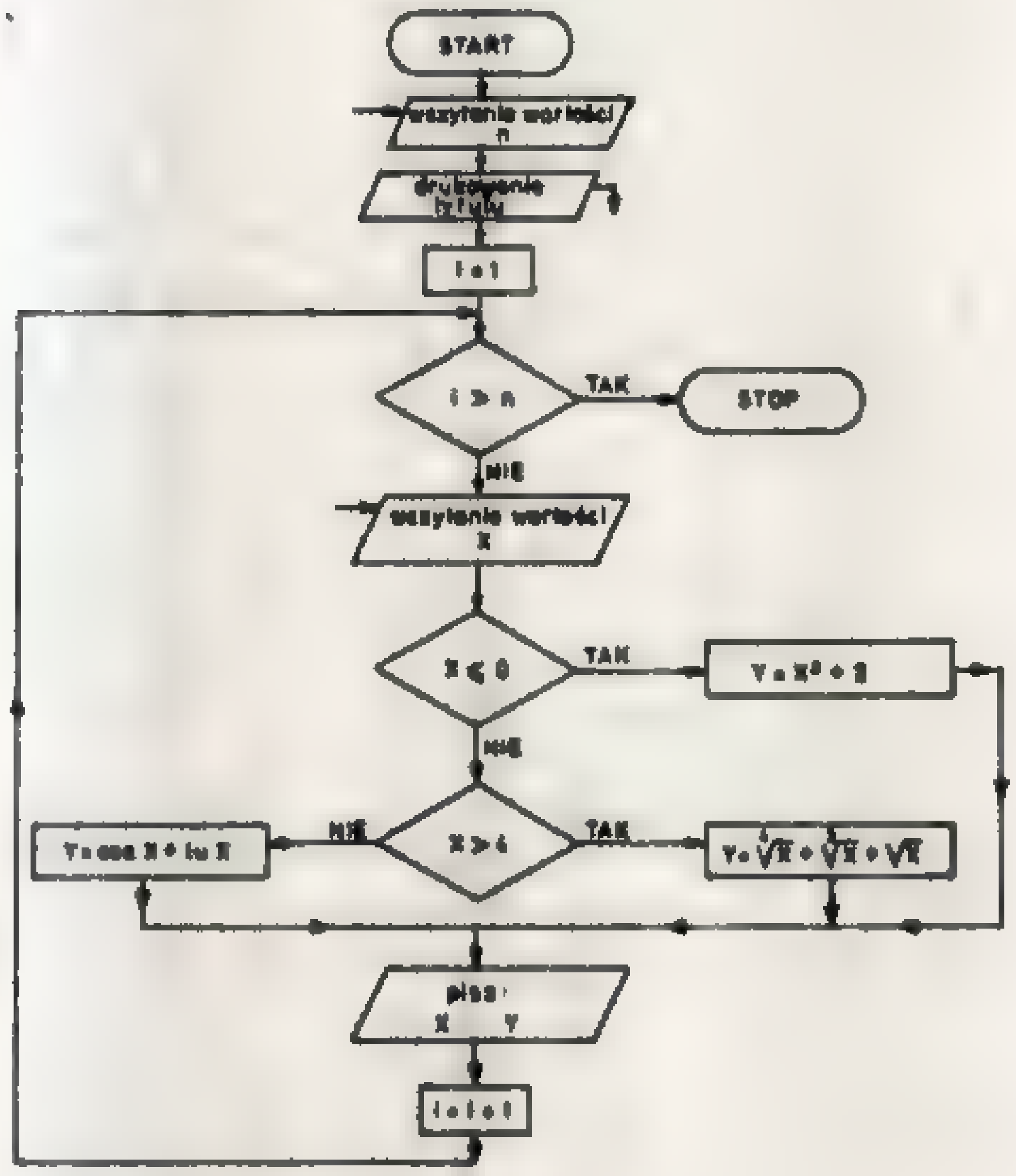
Napisać program drukowania tablicy wartości funkcji y zadanej wzorem:

$$y = \begin{cases} x^2 + 2 & \text{dla} \quad x \le 0 \\ \cos(x) + \ln(x) & \text{dla} \quad 0 < x \le 4 \\ (x)^{1/4} + (x)^{1/3} + (x)^{1/2} & \text{dla} \quad x > 4 \end{cases}$$

dla *n* — wartości zmiennej *x*

Kolejne wartości zmiennej *x* mają być wczytywane w czasie realizacji programu.

) schemat blokowy



b) program

```
 10 REM Przyklad 4.8
 15 PRINT "tablicowanie funkcji
 y dla      n-wartosci zmienne
j x"
 20 INPUT "n = ";n
 30 PRINT '"     x","     y"
 35 PRINT
 40 FOR i=1 TO n
 50 INPUT "x = ";x
 60 IF x<=0 THEN LET y=x*x+2: G
O TO 90
 70 IF x>4 THEN LET y=SQR (SQR
(x))+x↑(1/3)+SQR (x): GO TO 90
 80 LET y=COS x+LN x
 90 PRINT x,y
100 NEXT i
110 STOP
```

c) wyniki dla przykładowych danych

```
tablicowanie funkcji y dla
  n-wartosci zmiennej x

   x                  y

-2                 6
-1                 3
1.5                0.47620231
3                  0.10861979
16                 8.5198421
```

Na tym kończymy omawianie podstawowych informacji o pętlach (cyklach obliczeniowych) oraz o instrukcjach cyklu FOR i NEXT. Wiadomości te będą uzupełniane i rozszerzane w następnych wykładach.

Zadania

4.1. Podać ile razy i jakie wartości zostaną wprowadzone w trakcie wykonywania następujących fragmentów programów

```
10 REM ZAD.4.1.1
20 FOR x=-21 TO -1 STEP 3
30 PRINT "x = ";x+1
40 NEXT x

10 REM ZAD.4.1.2
20 FOR i=0 TO 5
30 LET x=-3+i*0.25
40 PRINT "x = ";x
50 NEXT i

10 REM ZAD.4.1.3
20 LET b=32
30 FOR j=1 TO b STEP 4
40 PRINT "j =";j
50 LET j=j+2
60 LET b=b+2
70 NEXT j
80 PRINT "b = ";b

10 REM ZAD.4.1.4
20 LET i=5
30 FOR i=i-10 TO i+10 STEP ABS
(i/2)
40 PRINT "i = ";i
50 NEXT i
60 PRINT '"i = ";i
```

4.2. Napisać następujące programy, tak, aby nie występowały w nich instrukcje cyklu FOR i NEXT — wykorzystać instrukcje IF:

4.2.1. program z **przykładu 4.7**

4.2.2. program z **przykładu 4.8**

Z.Z.

LOGO

Program w języku Logo składa się z procedur — tyle wiemy już z dotychczasowych rozważań. Wiemy też, że procedury mogą być wywoływane od różnych parametrów, a zatem wywołane w odmienny sposób, mogą obliczać różne wartości — dla innych danych, albo też, jak to się niebawem okaże, wykonywać zupełnie inne obliczenia dla różnych wartości parametrów. Zazwyczaj rozwiązanie problemu wymaga napisania więcej niż jednej procedury. Procedury te muszą mieć możliwość porozumiewania się między sobą, przekazywania sobie nawzajem wyników obliczeń. W tym celu w Logo przyjęto, że każda procedura może zwracać wartość. Wewnątrz treści procedury należy umieścić słowo WYNIK, a następnie wielkość, która powinna być dostępna na zewnątrz procedury. Wielkością tą może być dowolny obiekt Logo, może też być to dowolne wyrażenie, które po obliczeniu „stanie się" obiektem (wyrażenie jest po prostu skomplikowaną nazwą obiektu). Wykonanie komendy WYNIK powoduje, że nazwie procedury nadawana jest wartość i wykonywanie tej procedury jest przerywane. Nazwą takiej procedury posługujemy się tak jak obiektem, który jest jej wartością. Oto proste przykłady:

1) Obliczamy pole i objętość kuli o promieniu r, korzystając z procedury liczącej pole koła o tym samym promieniu (zauważmy, że: $\frac{4}{3}\pi r^3 = \frac{4\pi r^2 r}{3}$).

```
?
OTO KOŁO :r
WYNIK PI* * :r * :r
JUŻ

OTO KULA :r
PISZ [pole kuli :]
PISZ 4 * KOŁO :r
PISZ [objętość kuli :]
PISZ 4 / 3 * :r * KOŁO :r
JUŻ
```

2) Rysujemy wielokąt foremny, korzystając z procedury liczącej, o jaki kąt powinien obrócić się żółw zależnie od liczby boków wielokąta:

```
?
OTO WIELOKĄT :lb :db
POWTÓRZ :lb [np :db lewo ilestop
ni :lb]
JUŻ

OTO ILESTOPNI :lb
WYNIK 360 / :lb
JUŻ
```

Napiszmy teraz procedurę „wielokąt 2" tak:

```
?
OTO SPIRALA :odl :kąt :skok
NP :odl
LW :kąt
SPIRALA :odl + :skok :kąt :skok
JUŻ
```

i wywołajmy ją. Na ekranie nie będzie żadnego rysunku, ale Logo zapyta, co zrobić z wynikiem zwracanym przez procedurę „wielokąt". Komenda WYNIK przekazuje dane na wyższy poziom i jednocześnie zatrzymuje działanie procedury. Jest to logicznie uzasadnione — po co jeszcze coś obliczać skoro wynik jest już ustalony. Gdybyśmy w ostatniej procedurze „wielokąt" zamiast WYNIK napisali STOP, nie otrzymalibyśmy na ekranie w ogóle żadnego efektu — procedura nic nie robi i nie ma żadnej wartości. Komenda STOP po prostu kończy, działanie procedury. Na obecnym etapie znajomości Logo może się wydawać, że używanie STOP jest pozbawione większego sensu. Tymczasem okaże się później, że komenda ta jest bardzo potrzebna i często stosowana. W angielskim Logo WYNIK to OUTPUT, a komenda STOP brzmi tak samo po polsku.

Procedury rozwiązujące ten sam problem komunikują się między sobą, wywołują się nawzajem. Gdy jakąś czynność należy powtórzyć wielokrotnie, najprościej jest, gdy procedura, która tę czynność wykonuje, wywołała samą siebie. Jest to tak zwane wywołanie rekurencyjne. Logo pozwala na posługiwanie się rekursją w programowaniu. Co więcej, w Logo rekursja jest jednym z podstawowych narzędzi programowania. Dla niektórych obliczeń program rekurencyjny jest najprostszym i najłatwiej narzucającym się rozwiązaniem. Tak jest choćby przy liczeniu „silni" liczby naturalnej:

```
?
OTO SILNIA :n
JEŚLI :n = 0 [wy 1]
WY :n * SILNIA :n - 1
JUŻ
```

W procedurze tej pojawiła się nader użyteczna instrukcja warunkowa JEŚLI. Jej zastosowanie w tym programie jest oczywiste, szczegóły JEŚLI nieco niżej. Tymczasem wróćmy do rekursji. Służy ona w Logo dosłownie do wszystkiego. Korzystając z niej można szybko i elegancko programować. Spróbujmy natomiast napisać jakikolwiek większy program, zupełnie nie posługując się rekursją. Jestem pewny, że będzie to niełatwe zadanie, jeśli w ogóle wykonalne. Logo dysponuje tylko jedną instrukcją iteracyjną (powtórzenie) — jest to już omawiana instrukcja POWTÓRZ. Wiemy już, że nie daje ona zbyt wielkich możliwości — trzeba z góry określić, ile razy należy wykonać podane po słowie POWTÓRZ instrukcje. W Logo nie ma instrukcji skoku w rodzaju choćby Basic'owego GO TO. Zatem zakładanie pętli z użyciem skoku jest niemożliwe. Pozostaje rekursja, z której należy jak najwięcej korzystać. Przy pewnej wprawie programowanie rekurencyjne jest bardzo naturalne i oczywiste. Trzeba się tylko przestawić na ten sposób myślenia. Łatwiej z pewnością będzie tym, dla których Logo jest pierwszym poznawanym językiem programowania, trudniej tym, którzy przyzwyczaili się do programowania w języku bez rekursji np. w Basic'u. Mają oni już utrwalone schematy myślenia programistycznego i trudniej im przyjdzie znajdować rozwiązania rekurencyjne, bo mimo woli najpierw będą się głowić nad rozwiązaniem iteracyjnym (za pomocą pętli) danego problemu. Tacy fani programowania najlepiej zrozumieją to, co fachowcy wciąż powtarzają: że ważne jest nie tylko to, by jak najszybciej zetknąć się z komputerem i programowaniem, ale także, a może nawet głównie to, by zrobić to w sposób właściwy — tak, aby od początku „zakodować" sobie odpowiednie odruchy myślenia programistycznego. A oto przykład nadzwyczaj prostej a dającej niespodziewanie ciekawe efekty procedury rekurencyjnej:

```
?
OTO WIELOKĄT2 :lb :db
WYNIK ILESTOPNI :lb
POWTÓRZ :lb [np :db lw ilestopni
 :lb]
JUŻ
```

Jak wskazuje to jej nazwa, procedura rysuje spiralę — jej kształt jest zależny od podanych parametrów: długości dodawanego odcinka i kąta, o jaki za każdym razem obraca się żółw. Czytelnikom siedzącym przy komputerze radzę wypróbować działanie procedury „spirala" dla różnych wartości parametrów. Prawda, że rysunki są interesujące? Ci, którzy testowali procedurę zauważyli już, że procedura „spirala" raz uruchomiona działa bez końca. Trzeba ją

dokończenie na str. 24

dokończenie ze str. 23

przerywać naciskając klawisz BREAK. Aby procedura kończyła swe działanie bez brutalnej interwencji z zewnątrz, trzeba w jej wnętrzu umieścić instrukcję stopu, obwarowaną jakimś sensownym warunkiem. Instrukcja warunkowa w polskim Logo zaczyna się od słowa JEŚLI, dalej następuje warunek (na przykład: n=0) i dwie listy. Pierwsza z nich zawiera instrukcje, które są wykonywane, gdy warunek jest spełniony, druga — instrukcje do wykonania przy niespełnionym warunku. Pierwsza lista musi być podana, drugą można pominąć. Instrukcja warunkowa w procedurze „silnia" jest zapisana właśnie z pominięciem drugiej listy. W tej sytuacji Logo stwierdzając, że zadany warunek nie jest spełniony, po prostu przechodzi do wykonywania następnej po JEŚLI instrukcji. Zauważmy, że procedurę „silnia 2" możemy równoważnie zapisać tak:

```
?
OTO SILNIA2 :n
JEŚLI :n = 0 [wy 1] [wy :n * sil
nia2 :n - 1]
JUŻ
```

W procedurze „spirala" instrukcję warunkową wykorzystamy dla zatrzymania rysowania spirali w odpowiednim momencie. Przyjmijmy, że na przykład procedura ma przerwać rysowanie, gdy jednorazowo kreślony odcinek będzie miał długość większą niż 100:

```
OTO SPIRALA2 :odl :kąt :skok
JEŚLI :odl > 100 [stop]
NP :odl
LW :kąt
SPIRALA2 :odl + :skok :kąt :skok
JUŻ
```

W angielskim Logo zamiast JEŚLI należy pisać IF. Inne zasady dotyczące zapisu instrukcji warunkowej są takie same, jak w polskiej wersji języka. A oto jeszcze jeden program w istotny sposób wykorzystujący instrukcję warunkową i możliwość przekazywania wyników obliczeń przez procedury. Podany niżej komplet procedur oblicza i wypisuje na ekranie pierwiastki równania kwadratowego oraz współrzędne wierzchołka paraboli, będącej wykresem funkcji kwadratowej o zadanych współczynnikach. Program doskonale nadaje się do weryfikacji rozwiązań szkolnych zadań „do odrobienia" w domu!

```
?
OTO RÓWKWAD :a :b :c
MIEJSCAZEROWE :a :b :c
WIERZCHOŁEK :a :b :c
JUŻ

OTO MIEJSCAZEROWE :a :b :c
JEŚLI 0 > DELTA :a :b :c [stop]
JEŚLI 0 = DELTA :a :b :c [pisz [
X0 =] PISZ - :b / ( 2 * :a ) STO
P]
PISZ [x1 =] PISZ X1 :a :b :c
PISZ [x2 =] PISZ X2 :a :b :c
JUŻ

OTO DELTA :a :b :c
WYNIK :b * :b - 4 * :a * :c
JUŻ

OTO X1 :a :b :c
WY ( - :b - PIERWIASTEK DELTA :a
 :b :c ) / ( 2 * :a )
JUŻ

OTO X2 :a :b :c
WY ( - :b + PKW DELTA :a :b :c )
 / ( 2 * :a )
JUŻ

OTO WIERZCHOŁEK :a :b :c
PISZ [współrzędne wierzchołka]
PISZ [x =]
PISZ - :b / ( 2 * :a )
PISZ [y =]
PISZ ( - DELTA :a :b :c ) / ( 4
* :a )
JUŻ
```

Nic dotychczas nie mówiliśmy o zmiennych w Logo. A zatem zmienną w tym języku można utożsamiać z nazwą. Jest to jak gdyby etykietka przyporządkowana obiektowi Logo, a nie tak jak w niektórych innych językach pudełko, do którego wkłada się wartości zmiennej. Obiekty opatrywane są etykietkami za pomocą instrukcji przypisania. Po słowie PRZYPISZ należy podać nazwę zmiennej, poprzedzoną cudzysłowem, a następnie wartość nadawaną tej zmiennej. Wartość ta powinna mieć normalną dla Logo postać: liczby podaje się zwyczajnie, wartości zmiennych powinny być poprzedzone dwukropkiem (zapis taki jak dla parametrów procedur), nazwy funkcji (procedur zwracających wynik) wymienione normalnie, listy jako ciągi obiektów ujęte w nawiasy kwadratowe (o listach i operacjach na nich będzie szczegółowo w następnych odcinkach), słowa (napisy) poprzedzone cudzysłowem. Na przykład:
PRZYPISZ "y 89
PRZYPISZ "ala "kot
PRZYPISZ "x :y — etykietka dla x została przypisana także wartości 89. Przypisywać można także skomplikowane wyrażenia np:
PRZYPISZ "X1 (—:b — PIERWIASTEK delta :a :b :c) /2 + :a

W angielskim Logo odpowiednikiem słowa PRZYPISZ jest MAKE. Instrukcję przypisywania można wykorzystać choćby do zaprogramowania iteracyjnej wersji procedury „silnia":

```
?
OTO SILNIA3 :n
PRZYPISZ "m 1
PRZYP "i 0
POWTÓRZ :n [przyp "i :i + 1 przy
p "m :m * :i]
WY :m
JUŻ
```

Na zakończenie, jak zwykle, przypomnijmy sobie nowe słowa poznane w tym odcinku. A więc:

PRZYPISZ	(PRZYP)	przypisuje wartości nazwę: np. przyp "a11 to nadanie nazwy a liczbie 11 (ang. MAKE)
WYNIK	(WY) n	przypisuje procedurze wartość i kończy jej działanie (ang. OUTPUT (OP))
STOP		kończy działanie procedury (przerywa — powoduje „wyjście" z procedury) (ang. STOP)
Jeśli war akcja1 akcja2		jeśli warunek jest prawdą wykonuje się lista — akcja1, w przeciwnym przypadku akcja 2 (może jej zresztą nie być)
PIERWIASTEK	(PKW) n	daje (zwraca) pierwiastek kwadratowy n (ang.SQRT)

SIS

PROGRAM 31

COMMODORE C-64

W bardzo wielu zastosowaniach matematyki korzysta się z funkcji zwanych wielomianami. Wynika to ze szczególnych własności tych funkcji:

- łatwość obliczenia wartości wielomianu za pomocą skończonej liczby mnożeń i dodawań;
- łatwość wykonywania podstawowych działań arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia, obliczania pochodnej i całkowania;
- możliwość wykorzystania wielomianów do aproksymacji funkcji ciągłych z dowolną dokładnością.

Wykonanie takich podstawowych działań arytmetycznych, jak dodawanie i odejmowanie wielomianów, z reguły nie sprawia nam trudności, gdyż sprowadza się do redukcji wyrazów podobnych. Natomiast mnożenie i dzielenie wielomianów wymaga nie tylko znajomości odpowiednich metod, ale cierpliwości w wykonywaniu żmudnych obliczeń. W takich przypadkach okazuje się pomocny mikrokomputer. Zamieszczony poniżej program WIELOMIANY wspomaga wykonanie tych operacji. Jest on tak skonstruowany, że pozwala w prosty sposób rozbudować swe możliwości o dodatkowe operacje na wielomianach.

Program WIELOMIANY opracowany został na mikrokomputery: Commodore 64, Plus/4 i 16. Dla mikrokomputerów Commodore 16 i Plus/4 należy wiersz 10 zmienić według wzoru (patrz 10 str. 26).

Program WIELOMIANY umożliwia wykonanie działań (mnożenia i dzielenia) na wielomianach postaci:

$$A(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$$
$$B(x) = b_m x^m + b_{m-1} x^{m-1} + \ldots + b_0$$

gdzie $a_n \neq 0$; $b_m \neq 0$.

Zapisując wielomiany (np. $A(x) = 2x^4 + x^3 + x - 1$) podajemy:

- stopień wielomianu (w naszym przykładzie jest równy 4)
- dla współczynników różnych od zera: numer współczynnika i jego wartość (nr 0, wartość —1; nr 3, wartość 1; nr 1, wartość 1; nr 4, wartość 2).

Zapisywanie wielomianu kończy się po podaniu wartości współczynnika o numerze równym stopniowi wielomianu. Istnieje możliwość modyfikacji postaci wielomianów przy ponownym wykorzystaniu operacji zapisu.

W wyniku operacji mnożenia wielomianów A(x) i B(x) uzyskujemy wielomian $W(x) = A(x) \cdot B(x)$. W wyniku operacji dzielenia wielomianu A(x) przez wielomian B(x) otrzymamy iloraz W(x) i resztę z dzielenia R(x) spełniające równość:

$$A(x) = W(x) \cdot B(x) + R(x)$$

Wyniki wyprowadzane są według konwencji zapisu, tzn. numery i wartości współczynników różnych od zera.

Algorytmy mnożenia i dzielenia wielomianów wykorzystywane w programie WIELOMIANY są odmienne od tych, jakimi posługujemy się w obliczeniach ręcznych. Zainteresowanych namawiam do prześledzenia tych algorytmów i na ich podstawie opracowania szczególnych algorytmów dzielenia wielomianu przez dwumian $x - b_0$ lub trójmian kwadratowy $x^2 + b_1 x + b_0$.

Dla osób, które będą chciały rozbudować program WIELOMIANY, podaję krótki opis kodu programu:

1. wiersze 20—70: wyświetlenie informacji o operacjach aktualnie możliwych do realizacji; wybór operacji;

   wiersze 90—160: zapis wielomianu A(x)
   wiersze 170—230: zapis wielomianu B(x)
   wiersze 240—300: wykonanie dzielenia wielomianów A(x) przez B(x)
   wiersze 310—350: wykonanie mnożenia wielomianów

   wiersze 360—560: wyprowadzenie wyników w zależności od ostatnio wykonanej operacji
   wiersze 600—620: „czyszczenie" ekranu po obejrzeniu partii wyników.

2. zmienne sterujące:
   T — określa aktualnie dostępne operacje;
   D — określa matrycę, według której należy wyprowadzać ostatnio otrzymane wyniki.

Uwaga: w tekście programu zamiast znaku $ użyty jest znak ×.

```
0 REM ***WIELOMIANY***
10 T=1:DIMW(39):C=1064:POKE53280,0:
POKE53281,0
20 PRINTCHR×(147)"OPCJE:":ONTGOTO60
,50,40,30
30 PRINT"6 - WYNIKI"
40 PRINT"5 - MNOZENIE WIELOMIANOW A
(X)*B(X)":PRINT"4 - DZIELENIE WIELO
MIANOW A(X)/B(X)"
50 PRINT"3 - ZAPIS WIELOMIANU B(X)"
60 PRINT"2 - ZAPIS WIELOMIANU A(X)"
:PRINT"1 - KONIEC":PRINT:INPUT"WYBI
ERAM";O%
70 PRINTCHR×(147):ON O% GOSUB 80,90
,170,240,310,360:GOTO20
80 END
90 IFT>1THEN120
100 INPUT"MAKSYMALNY STOPIEN WIELOM
IANOW";MS%:IFMS%<0THEN100
110 DIMA(MS%),B(MS%),C(MS%+MS%):T=2
120 INPUT"AKTUALNY STOPIEN WIEL. A(
X)";SA%:IFSA%>MS%ORSA%<1THEN120
130 INPUT"A(X):  NR WSP.";NW%:IFNW%
>SA%ORNW%<0THEN130
140 INPUT"A(X):  WARTOSC";A(SA%-NW%
):IFNW%<>SA%THEN130
150 IFA(0)=0THEN140
160 RETURN
170 IFT<2THENRETURN
```

dokończenie na str. 26

dokończenie ze str. 25

```
180 INPUT"AKTUALNY STOPIEN WIEL. B(
X)";SB%:IFSB%>MS%ORSB%<1THEN180
190 INPUT"B(X):  NR WSP.";NW%:IFNW%
>SB%ORNW%<0THEN190
200 INPUT"B(X):  WARTOSC";B(SB%-NW%
):IFNW%<>SB%THEN190
210 IFB(0)=0THEN200
220 IFT=2THENT=3
230 RETURN
240 IFT<3THENRETURN
250 IFSB%>SA%THENFORK=0TOSA%:C(K)=A
(K):NEXT:D=2:RETURN
260 S=SA%-SB%:FORK=0TOSA%:JP=K-SB%:
IFJP<0THENJP=0
265 JK=K-1:IFK>STHENJK=S
270 C(K)=0:FORJ=JPTOJK:C(K)=C(K)+C(
J)*B(K-J):NEXT:C(K)=A(K)-C(K)
280 IFK<=STHENC(K)=C(K)/B(0)
290 NEXT:IFT=3THENT=4
300 D=1:RETURN
310 IFT<3THENRETURN
320 FORK=0TOSA%+SB%:JP=K-SA%:IFJP<0
THENJP=0
330 JK=SB%:IFJK>KTHENJK=K
340 C(K)=0:FORJ=JPTOJK:C(K)=C(K)+B(
J)*A(K-J):NEXT:NEXT:IFT=3THENT=4
350 D=3:RETURN
360 IFT<4THENRETURN
370 K=0:ONDGOSUB380,470,520:PRINT:I
NPUT"OBEJRZALES";Z$:RETURN

380 PRINT"WSPOLCZYNNIKI ILORAZU ROZ
NE OD ZERA:":PRINT:W=1
390 IFC(K)=0THEN410
400 GOSUB600:PRINT"WSP. NR"S-K;TAB(
12)"WARTOSC:"C(K)
410 K=K+1:IFK<=SA%-SB%THEN390
415 PRINT:INPUT"OBEJRZALES";Z$:PRIN
TCHR$(147)
420 PRINT"WSPOLCZYNNIKI RESZTY ROZN
E OD ZERA:":PRINT:W=1
430 IFC(K)=0THEN450
440 GOSUB600:PRINT"WSP. NR:"SA%-K;T
AB(12)"WARTOSC:"C(K)
450 K=K+1:IFK<=SA%THEN430
460 RETURN
470 PRINT"WSPOLCZYNNIKI RESZTY ROZN
E OD ZERA:":PRINT:W=1
480 IFC(K)=0THEN500
490 GOSUB600:PRINT"WSP. NR"SA%-K;TA
B(12)"WARTOSC:"C(K)
500 K=K+1:IFK<=SA%THEN480
510 RETURN
520 PRINT"WSPOLCZYNNIKI ILOCZYNU RO
ZNE OD ZERA:":PRINT:W=1
530 IFC(K)=0THEN550
540 GOSUB600:PRINT"WSP. NR:"SA%+SB%
-K;TAB(12)"WARTOSC:"C(K)
550 K=K+1:IFK<=SA%+SB%THEN530
560 RETURN
600 W=W+1:IFW<20THENRETURN
610 W=0:PRINT:INPUT"OBEJRZALES";Z$:
FORI=0TO39:W(I)=PEEK(C+I):NEXT
620 PRINTCHR$(147):FORI=0TO39:POKEC
+I,W(I):NEXT:PRINT:PRINT:RETURN


10 T=1:DIMW(39):C=3112:COLOR0,1,1:
COLOR4,1,1
```

PROGRAM 32 ATARI

Program może służyć do nauki alfabetu Morse'a z dźwiękowym i wizualnym jego odwzorowaniem lub po podłączeniu mikrokomputera do nadajnika jako moduł usprawniający pracę operatora.

Korzystając z tego programu można nadawać klawiaturą lub kluczem. W obu przypadkach jest możliwe otrzymanie na ekranie znaków Morse'a lub alfabetu łacińskiego. Ponadto takie kody jak: nazwa stacji, wywołanie ogólne itp., nadawane są w całości przez naciśnięcie określonego klawisza. W tym celu należy powrócić do MENU wciskając klawisz „↑", a następnie wcisnąć klawisz z literą występującą przy danym kodzie.

W miejsce nazwy stacji SP5IVJ należy wpisać swoją nazwę. Dokonuje się tego w L.136 i L.4000 (na końcu ciągu należy pozostawić gwiazdkę). Gdyby użytkownik zechciał rozszerzyć zestaw znaków alfabetu Morse'a, wystarczy wprowadzić do programu dane pomiędzy L.20050 i L.20060 np. 20055 DATA (znak), (kod znaku).

Szybkość emitowania znaków za pomocą klucza zależy od sprawności manualnej użytkownika oraz jakości klucza. Stąd też w przypadku znaków dekodowanych i otrzymywanych w postaci alfabetu łacińskiego na ekranie monitora wprowadzono pętlę L.2120,2140. W pętli tej zmienną CZ ustala się czas, w jakim jest dekodowany wprowadzony znak. Zmienna CZ obliczana jest w L.2103 na podstawie danej Z wprowadzonej w L.2102.

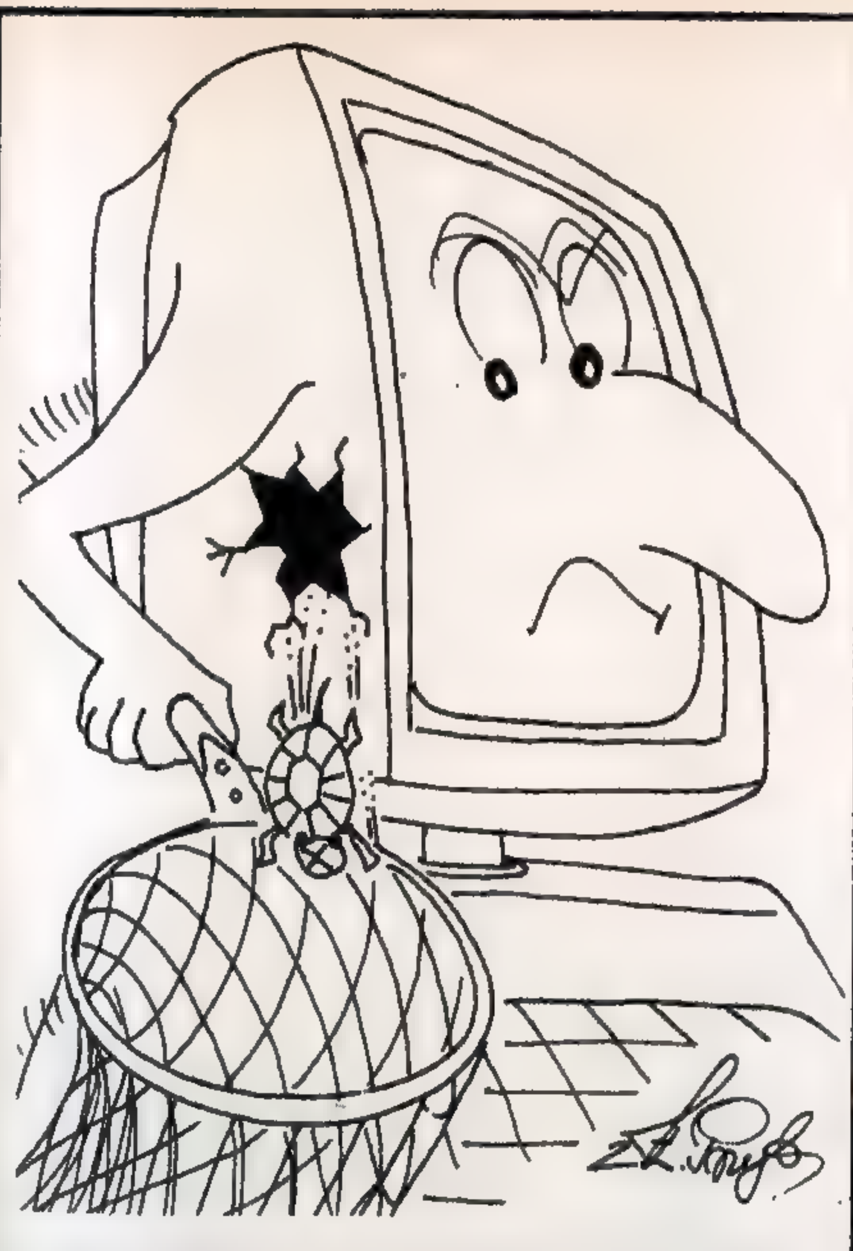
Jako klucz nadaje się najbardziej przedstawiony na rysunku manipulator klucza elektronowego. Przyłącza się go do wyprowadzeń nr 3, 4 i 8 (GND) gniazda joystick'a oznaczonego na obudowie mikrokomputera jako CONTRLLERS 1. Zamiast klucza można stosować również joystick, jednak powoduje to znacznie wolniejsze nadawanie. Przekazywanie sygnału z mikrokomputera do nadajnika dokonuje się z końcówek nr 3 i 2 (GND) gniazda MONITOR.

```
5 REM **Alfabet Morse'a**
6 REM (C) Janusz J.
7 REM Warszawa 1986,7
10 OPEN #1,4,0,"K:"
12 ? "}":POKE 752,1
14 GRAPHICS 2
16 POSITION 2,1:? #6;"-.-.-.-.-.-.-.-"
18 POSITION 2,3:? #6;"alfabet morse'a"
20 POSITION 2,5:? #6;"(C)      1986,7"
22 POSITION 2,7:? #6;"-.-.-.-.-.-.-.-"
25 DIM B(12,9),SP$(7),L$(10)
26 DIM WL$(1),DL$(1),A$(6)
28 DIM KL$(6),AL$(1),K$(6)
29 L$="0123456789"
30 FOR E=0 TO 4
35 FOR F=0 TO 4
40 EF=E+F:FE=INT(0.3*F*E)
45 B(F,EF)=45
50 B(F+6,EF+1)=46
55 B(F+6,FE)=45
60 B(F+1,FE)=46
65 NEXT F:NEXT E
66 B(10,0)=45:B(10,1)=46:B(10,2)=46
B(10,3)=46:B(10,4)=45
67 B(11,0)=45:B(11,1)=46:B(11,2)=46
B(11,3)=45:B(11,4)=46
70 ? :? "      Wcisnij dowolny klawisz"
75 GET #1,P
80 GRAPHICS 0+16
82 POKE 752,1
84 POKE 764,147
100 ? "}"
110 POSITION 16,5:? " MENU "
120 POSITION 7,7:? "A - KORZYSTANIE
Z KLAWIATURY "
130 POSITION 7,8:? "B - KORZYSTANIE
Z KLUCZA"
135 POSITION 15,11:? "  KODY: "
136 POSITION 14,13:? "D - SP5IVJ"
137 POSITION 14,14:? "E - QRZ"
138 POSITION 14,15:? "F - QRA"
139 POSITION 14,16:? "G - QRT"
140 POSITION 14,17:? "H - QRX"
141 POSITION 14,18:? "I - QTH"
160 GET #1,AM
165 IF AM<65 OR AM>73 THEN 160
170 MA=AM-64
180 GOTO 1000*MA
1000 REM WYKORZYSTANIE KLAWIATURY
1010 GOSUB 1500
1020 ? "}"
1025 IF AK=42 THEN 100


1030 GET #1,X
1032 IF X=42 THEN 1010
1033 IF AK=65 THEN ? CHR$(X);
1035 IF X>46 AND X<58 OR X=61 THEN 1400
1040 WL$=CHR$(X)
1050 RESTORE
1060 READ DL$,A$
1065 L=LEN(A$)
1070 IF DL$="$" THEN ?"BRAK DANYCH "
;:GOTO 1030
1080 IF DL$=WL$ THEN 1200
1090 GOTO 1060
1200 FOR D=1 TO L
1210 IF AK=65 THEN 1230
1220 ? A$(D,D);
1230 IF A$(D,D)="-" THEN GOSUB 1300
:GOTO 1245
1240 FOR F=1 TO 8:SOUND 0,20,14,15:
NEXT F
1242 FOR F=1 TO 10:NEXT F
1245 SOUND 0,0,0,0
1250 NEXT D
1255 ? " ";
1260 GOTO 1030
1300 FOR F=1 TO 30
1310 SOUND 0,20,14,15
1320 NEXT F
1330 RETURN
1400 Z=X-48
1402 IF Z=13 THEN Z=10
1404 IF Z=-1 THEN Z=11
1410 FOR FF=0 TO 4
1415 IF AK=65 THEN 1425
1420 ? CHR$(B(Z,FF));
1425 IF B(Z,FF)=45 THEN GOSUB 1300:
GOTO 1432
1430 FOR U=1 TO 8:SOUND 0,20,14,15:
NEXT U
1431 FOR U=1 TO 10:NEXT U
1432 SOUND 0,0,0,0
1435 NEXT FF
1440 ? " ";
1450 GOTO 1030
1500 ? "}"
POSITON 7,5:? "CO CHCESZ NA
EKRANIE ?"
1520 POSITION 10,7:? "A - alfabet"
1530 POSITION 10,8:? "B - kod Mors'
1540 POSITION 10,10:? "* - powrot
 do menu"

1550 GET #1,AK
1560 RETURN
2000 REM WYKORZYSTANIE KLUCZA
2004 POKE 764,255
2005 GOSUB 1500
2007 IF AK=65 THEN 2100
2008 IF AK=42 THEN 100
2010 ? "}"
2015 ST=STICK(0)
2020 IF ST=11 THEN ? ".";:FOR F=1 T
O 8:SOUND 0,20,14,15:NEXT F
2030 IF ST=7 THEN ? "-";:FOR F=1 TO
30:SOUND 0,20,14,15:NEXT F
2040 SOUND 0,0,0,0
2050 IF PEEK(764)=7 THEN 2004
2060 GOTO 2015
2100 POSITION 10,14:? "CZAS DEKODOW
ANIA ? "
2101 POSITION 17,15:? "(10 - 60)"
2102 POSITION 27,14:? ;:INPUT Z
2103 CZ=INT(120/Z)
2105 ? "}"
2110 DK=1
2115 KL$=""
2120 FOR R=1 TO CZ
2121 SOUND 0,0,0,0
2123 ST=STICK(0)
2125 IF ST=11 THEN 2210
2130 IF ST=7 THEN 2230
2133 L=LEN(KL$)
2134 IF L=6 THEN 2175
2135 IF PEEK(764)=7 THEN 2004
2136 IF L=5 THEN 2300
2140 NEXT R
2170 IF L=0 THEN 2120
2175 RESTORE
2180 READ AL$,K$
2185 IF AL$="$" THEN 2400
2190 IF K$=KL$ THEN ? AL$;:GOTO 2110
2195 GOTO 2180
2210 IF DK>6 THEN 2400
2212 KL$(DK,DK)="."
2215 FOR F=1 TO 8:SOUND 0,20,14,15
:NEXT F
2220 DK=DK+1
2225 GOTO 2120
2230 IF DK>6 THEN 2400
2232 KL$(DK,DK)="-"
2235 FOR F=1 TO 30:SOUND 0,20,14,15
:NEXT F
2240 DK=DK+1
2245 GOTO 2120
2300 FOR G=0 TO 11
2310 FOR E=0 TO 4
2315 N=E+1
```

dokończenie na str. 28



```
2317 IF G=11 AND ASC(KL$(N,N))<>B(11
,E) THEN 2400
2320 IF ASC(KL$(N,N))<>B(G,E)THEN 2
330
2323 IF E=4 AND ASC(KL$(5,5))=B(G,4
) THEN 2333
2325 NEXT E
2330 NEXT G
2333 IF G=10 THEN ? "=";:GOTO 2110
2335 IF G=11 THEN ? "/";:GOTO 2110
2350 ? G;
2360 GOTO 2110
2400 ? "BRAK OKRESLENIA ZNAKU",
2410 GOTO 2110
4000 SP$="SP5IVJ*"
4005 LS=LEN(SP$)
4010 GOSUB 4500
4020 GOTO 160
4500 N=0
4510 N=N+1
4515 IF LS>4 AND N=3 THEN 4800
4520 RESTORE
4530 READ DL$,A$
4540 IF A$="*" THEN RETURN
4550 IF DL$=SP$(N,N) THEN 4570
4560 GOTO 4530
4570 L=LEN(A$)
4580 FOR F=1 TO L
4590 IF A$(F,F)="." THEN GOSUB 4930
:GOTO 4690
4610 GOSUB 4900
4690 NEXT F
4700 FOR E=1 TO 12
4710 NEXT E
4720 GOTO 4510
4800 FOR M=1 TO 10
4810 IF L$(M,M)=SP$(N,N) THEN 4820
4815 NEXT M
4820 MM=M-1
4830 FOR F=0 TO 4
4840 IF B(MM,F)=46 THEN GOSUB 4930
:GOTO 4860
4850 GOSUB 4900
4860 NEXT F
4890 GOTO 4510
4900 FOR E=1 TO 19
4910 SOUND 0,20,14,15
4920 NEXT E
4930 FOR E=1 TO 5
4940 SOUND 0,20,14,15
4950 NEXT E
4960 SOUND 0,0,0,0
4970 FOR E=1 TO 5
4975 NEXT E
4980 RETURN
```

```
5000 SP$="QRZ*"
5010 GOSUB 4500
5020 GOTO 160
6000 SP$="QRA*"
6010 GOSUB 4500
6020 GOTO 160
7000 SP$="QRT*"
7010 GOSUB 4500
7020 GOTO 160
8000 SP$="QRX*"
8010 GOSUB 4500
8020 GOTO 160
9000 SP$="QTH*"
9010 GOSUB 4500
9020 GOTO 160
20000 DATA A,.-,B,-...,S,...,T,-,G,--.
20010 DATA J,.---,N,-.,O,---,K,-.-,Y,-.--
20020 DATA M,--,F,..-.,Z,--..,I,..,X,-..-
20030 DATA D,-..,R,.-.,H,....,E,.,W,.--
20040 DATA L,.-..,Q,--.-,P,.--.,V,...-,C,-.-.
20050 DATA U,..-,?,..--..,',--..--
20060 DATA $,*
```



Wideo — straszak

Cokolwiek zrobilibyśmy — będzie źle. Nie było w Polsce komputerów — źle. Są — też źle. Taka ocena sytuacji przypomina starą anegdotę. Znacie? No to, posłuchajcie! Otóż w knajpce siedzi kilku „podciętych" facetów. Nudzi im się. Spostrzegają przy sąsiednim stoliku chudego, bladego faceta w okularach. Rozmowa „zmęczonych" nabiera rozmachu.

— Trzeba by mu dać w ucho!

— Ale za co?

— Spytaj się go czy był kiedyś w Gródku...

— A jak był?

— Dasz mu w ucho.

— A jak nie był?

— Też dasz mu w ucho!

Ludzie nam niechętni zawsze znajdą powód, aby próbować wbić przysłowiową szpilę. Tym razem okazją stały się minikomputery i videokasety. Niedawno londyński dziennik „The Times" opublikował artykuł pod frapującym tytułem: *„Początek innych wojen gwiezdnych"*. Autor, mister Roger Boyes rozwodził się długo i boleśnie na temat zagrożenia, jakie rzekomo dla Polski i w ogóle krajów socjalistycznych stanowią owe minikomputery i videokasety. Dlaczego?

Ponieważ można je wykorzystać do antypaństwowej roboty, np. w zaciszu domowym, wyświetlać dla rodziny, a co gorsze dla sąsiadów, taki np. film jak „Amadeusz" Milosa Formana, który z pewnością uważany jest „tam", to znaczy — u nas, za „niebezpieczny ładunek". I właśnie, dlatego, mieliśmy okazję zobaczyć go przed paroma miesiącami na „Konfrontacjach'86".

Henryk KAWKA

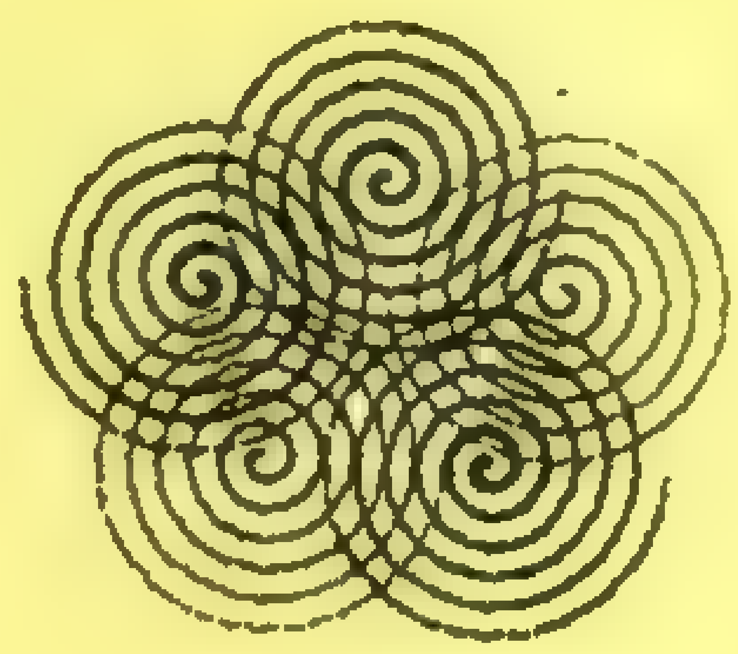


Giełda... październik'86

Foto: J. Zelman

JANUSZ MILLER
SHARP PC-1500
S P I R A L A

```
 10:BEEP 10,100,10
    0
 20:"A":CLEAR :
    USING :WAIT 0:
    RADIAN :GRAPH
 30:INPUT "ILE SPI
    RALI ? N=";N:N
    =ABS N:N=INT N
    :IF N=0GOTO 10
 40:INPUT "ILE ZWI
    TEK ? Z=";Z:Z=
    ABS Z:Z=INT Z:
    IF Z=0GOTO 10
 50:A=2*π/N:DIM W(
    N,2)
 60:S=π/40:Q=1/(Z*
    2):PRINT "N";N
    ;" Z";Z
 70:FOR K=0TO (N-1
    )
 80:M=M+1:W(M,1)=6
    0*COS (K*A)
 90:W(M,2)=50*SIN
    (K*A)
100:NEXT K
110:RLINE -(80,0),
    9:SORGN
120:FOR K=1TO N
130:X=W(K,1):Y=W(K
    ,2)
140:LINE -(0,0)-(X
    ,Y),9:SORGN
150:T=0:R=0:X2=0:Y
    2=0:U=U+1:IF U
    =4LET U=1
160:COLOR U
170:FOR J=1TO Z
180:FOR L=1TO 80
190:X1=X2:Y1=Y2:T=
    T+S:R=R+Q:X2=R
    *COS T
200:Y2=R*SIN T:IF
    (K-INT (K/2)*2
    )=0LET X2=-X2
210:LINE -(X1,Y1)-
    (X2,Y2),1
220:NEXT L:NEXT J
230:LINE -(X2,Y2)-
    (0,0),9
240:NEXT K:TEXT
250:END
```



LIGA MYŚLĄCYCH

Zadanie nr 1

Na drodze o długości 720 m przednie koło wozu wykonało o 96 obrotów więcej niż tylne. Gdyby obwód każdego koła był o pół milimetra mniejszy, to przednie koło na tej samej drodze wykonałoby o 128 obrotów więcej niż tylne.
Obliczyć długość obwodu każdego koła.

Zadanie nr 2

W szkole przeprowadzono ankietę wśród uczniów na temat nowo wprowadzonego programu nauczania z matematyki. Jeden z uczniów, miłośnik łamigłówek w rubryce wiek ankietowanego wpisał: jeśli mój wiek, który będę miał za trzy lata, trzykrotnie zwiększycie i odejmiecie od tego powiększony trzykrotnie mój wiek, który miałem trzy lata temu, to dowiecie się ile mam obecnie lat.

Zadanie nr 3

Na przebycie drogi o długości 520 km, jeden podróżny zużyje o 3 dni więcej czasu niż drugi, który przebywa dziennie o 12 km więcej niż pierwszy. Ile dni potrzebuje każdy z nich na przebycie tej drogi?

Zadanie nr 4

W trójwymiarowej przestrzeni danych jest n-punktów, z których żadne cztery nie leżą na jednej płaszczyźnie. Ile prostych i ile płaszczyzn wyznaczają te punkty?

Rozwiązania zadań prosimy przesyłać do redakcji do końca listopada br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Odpowiedzi do zadań z 4 nr. „IKS-a"

Zadanie 1

Z warunków zadania wiemy, że z każdą godziną różnica wskazań zegarów rośnie o 3 minuty. Zatem, od momentu wyregulowania i nakręcenia, 1 godzinę różnicy wskazań, zegary osiągnęły po 20 godzinach. Stąd, do chwili zatrzymania, budzik pośpieszył się o 20 minut. Z tego wynika, że w momencie zatrzymania zegarów była godzina 7.40 (od chwili, w której stanął budzik tj. od godziny 8.00 odejmujemy 20 minut). Wyregulowanie i nakręcenie zegarów odbyło się 20 godzin wcześniej, tj. o godzinie 11.40 dnia poprzedniego.

Zadanie 2

Wszystkich możliwości wylosowania trzech różnych cyfr z dziesięciu, a więc wszystkich kombinacji 3-elementowych z 10 jest:

$$\binom{3}{10} = \frac{10!}{3!\,(10-3)!}\text{, tj. } 120.$$

Sprzyjających kombinacji czyli takich, kiedy suma cyfr równa się 9 lub 18 jest tylko 14 (018, 027, 036, 045, 126, 135, 234, 378, 468, 567, 918, 927, 936, 945). A więc prawdopodobieństwo wyciągnięcia „na chybił trafił" spośród 10 cyfr od 0 do 9 trzech takich, aby można z nich ułożyć liczbę podzielną przez 9 wynosi:

$$P = \frac{14}{120} = \frac{7}{60}$$

Zadanie 3

Jeżeli n-2 zawodników rozegrało między sobą partię każdy z każdym, to rozegrano tyle partii ile jest 2-elementowych kombinacji ze zbioru n-2-elementowego:

$$\binom{n-2}{2} = \frac{(n-3)(n-2)}{2}$$

Gdy założymy, że pozostali dwaj gracze nie rozegrali partii między sobą, to z warunków zadania wynika, że:

$$\frac{(n-3)(n-2)}{2} + 10 + 1 = 55,$$

czyli $n^2 - 5n - 82 = 0$.

Równanie to nie posiada rozwiązań w zbiorze liczb naturalnych a zatem założenie, że dwaj gracze nie rozegrali między sobą partii było błędne.
Załóżmy więc, że pozostali dwaj gracze rozegrali między sobą partię. Wtedy otrzymamy następujące równanie:

$$\frac{(n-3)(n-2)}{2} + 10 = 55$$

czyli $n^2 - 5n - 84 = 0$.

Rozwiązaniem tego równania są $n_1 = 7$, które odrzucamy i $n_2 = 12$.
Zatem w turnieju szachowym brało udział 12 zawodników, a wspomniani dwaj gracze rozegrali między sobą partię.

NAZWISKA OSÓB, które wylosowały książki za poprawne rozwiązanie zadań z 4 nr. „IKS-a".

1. Krzysztof PAROL z Mińska Mazowieckiego (05-300 ul. Obrońców Stalingradu 27/11)
2. Bartosz BOBER z Rzeszowa (35-106 ul. Wnukowskiego 47/44)
3. Witold ŻYŁA z Warszawy (02-123 ul. Korotyńskiego 19a/32)
4. Beata BIESZK ze Szczecina (70-136, ul. 9 Maja 8/21

Pocztowa giełda

Rozwiązanie krzyżówki z „IKS-a" nr 4

Hasło brzmi: „Z komputerem do szkoły". Bony pieniężne (1000 zł) wylosowali: Maciej Janik — Częstochowa, Grażyna Kolbusz — Brzeg, Adam Kardasz — Gdańsk, Jarosław Piąstka — Lucynowo/Ostrowite, Sławomir Rydel — Kolno.

Nagrody książkowe otrzymują: Monika Andrzejewska — Ostrołęka, Robert Kalisz — Wrocław, Tomasz Łapiński — Pruszków, Michał Sułkowski — Tarnobrzeg, Dariusz Domański — Żarów, Adela Kopyciok — Świętochłowice, Sylwester Bąk — Lublin, Mirosława Orłowska — Kłodzko, Beata Bobowska — Wrocław, Barbara Siarkiewicz — Sosnowiec.

UWAGA — KONKURS!

Miłośnicy literatury science fiction z oryginalnymi pomysłami i fantazją! Każdy z Was ma szansę spróbować swoich sił w naszym konkursie na nowelę lub opowiadanie pod hasłem „Przed nami XXI wiek".

„Najfantastyczniejsze" utwory zostaną uhonorowane przez ziemskie jury, które przyzna nagrody:

I — 15 tys. złotych
II — 10 tys. złotych
III — 5 tys. złotych

Nad ich sprawiedliwym i obiektywnym podziałem czuwać będzie komputer. Autorzy materiałów zakwalifikowanych do druku zostaną ponadto usatysfakcjonowani godziwym honorarium zgodnie z obowiązującymi stawkami.

Entuzjastów informatyki i komputerów nie musimy przekonywać, że i fantazja musi mieć swoje ścisłe parametry. A więc mile widziane będą prace o objętości nie przekraczającej 15 stron maszynopisu bądź czytelnego rękopisu.

Jak będzie wyglądał nasz glob opanowany przez komputery n-tej generacji, w jaki sposób informatyka zmieni życie człowieka XXI wieku, jak homo sapiens ułoży sobie koegzystencję z rozumnymi maszynami? To tylko niektóre sugestie organizatorów dotyczące tematyki nadsyłanych utworów.

Prace opatrzone godłem należy nadsyłać do końca grudnia, dołączając kopertę podpisaną godłem, zawierającą dane o autorze, pod adresem:

„IKS" — ul. Grzybowska 77
00-950 W-wa

Przychylności weny, zdyscyplinowanej fantazji — życzy **REDAKCJA**

PS Zastrzegamy sobie prawo pierwszeństwa publikacji prac nagrodzonych oraz wybranych spośród nie nagrodzonych.

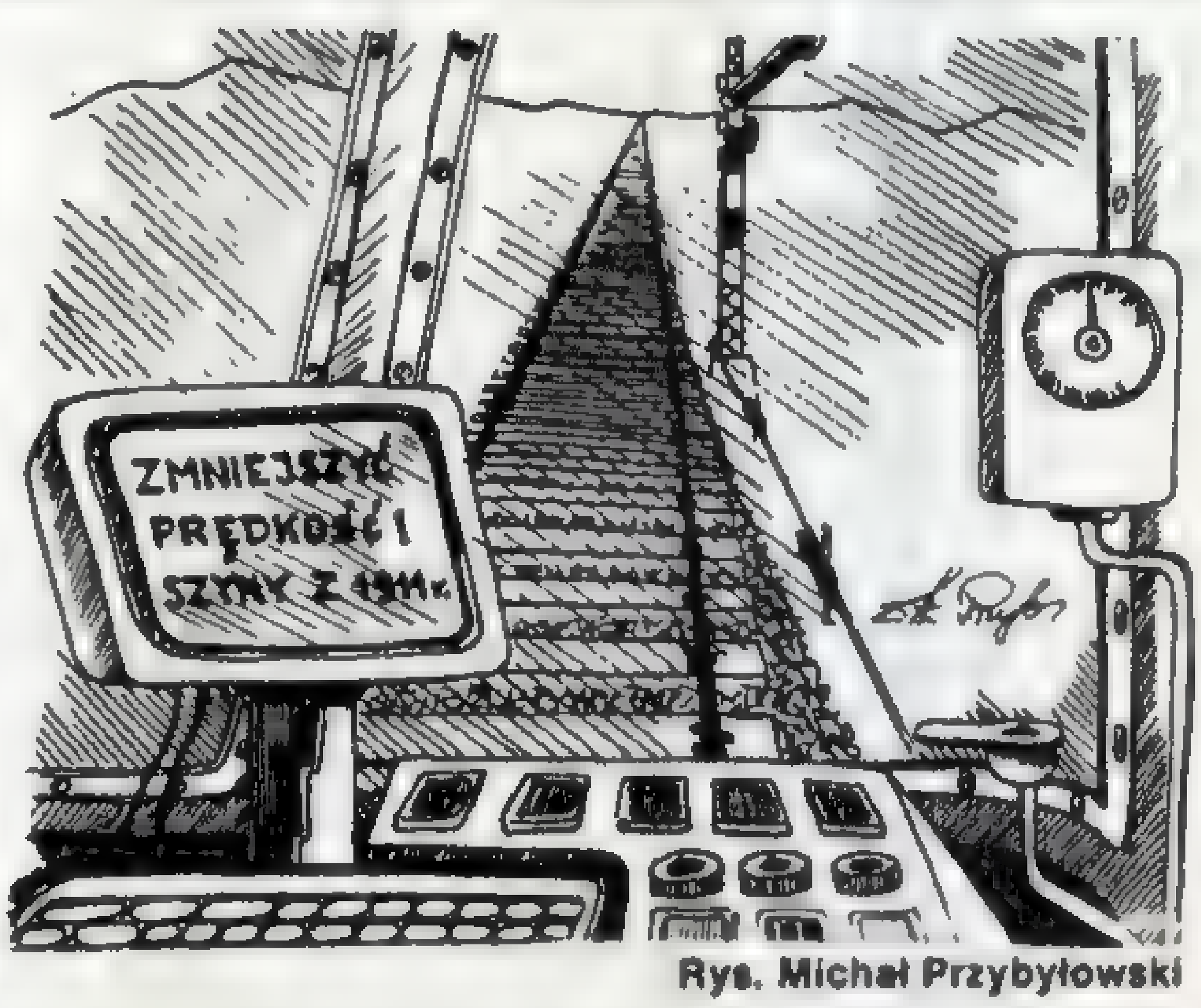
W naszym komputerlandzie

W zakresie kształcenia intelektualnego pracowity Spektruś „zabezpieczył" dla kierowanego przez siebie zespołu robotów uznanego lektora. Przygotował on wykład oraz dyskusję panelową na temat: „Dlaczego w komputerlandzie jest tyle znaków nierówności, a tak mało równania do przodu?".

Zaplanowano trzy spotkania, w których wziąć miały udział wszystkie roboty a także sam Spektruś. Prelegent jednak najzwyklej przynudzał i po dwudziestu minutach pierwszego wykładu okazało się, że połowa miejsc na sali jest pusta bo niesforne roboty wycofały się po angielsku, żeby nie zwracać uwagi. W połowie drugiego spotkania wyłączono światło i zanim znaleziono świeczki lektor oberwał po głowie i mógł jedynie bredzić. Tym samym prowadził wykład na takim samym poziomie, jak w pierwszej części. Do trzeciego spotkania nie doszło, bo wcześniej ktoś zamknął lektora w ustronnym miejscu i zgubił klucz od skobelka. Kiedy go wyswobodzono sala była już pusta.

Aby uniknąć takich sytuacji w przyszłości i zapewnić wysoki poziom prelekcji zapobiegliwy Spektruś opracował kompetentną i wszechstronną ściągę dla prelegentów na każdą okazję. Jego zdaniem najlepsze efekty można uzyskać jeśli jedną trzecią odczytu stanowią stare, odwieczne prawdy, żeby słuchacze doszli do wniosku, że wcale nie są głupi — a przeciwnie. Środkową część należy poświęcić zagadnieniom nowym lecz podanym w przystępnej formie, by słuchaczom zdało się, iż z odczytu odnoszą korzyść. Ostatnia część powinna traktować o takich czy innych zjawiskach językiem niezrozumiałym, aby słuchacze nie stracili szacunku dla nauki. No i osoby prelegenta...

podglądał: Eugeniusz MLECZAK



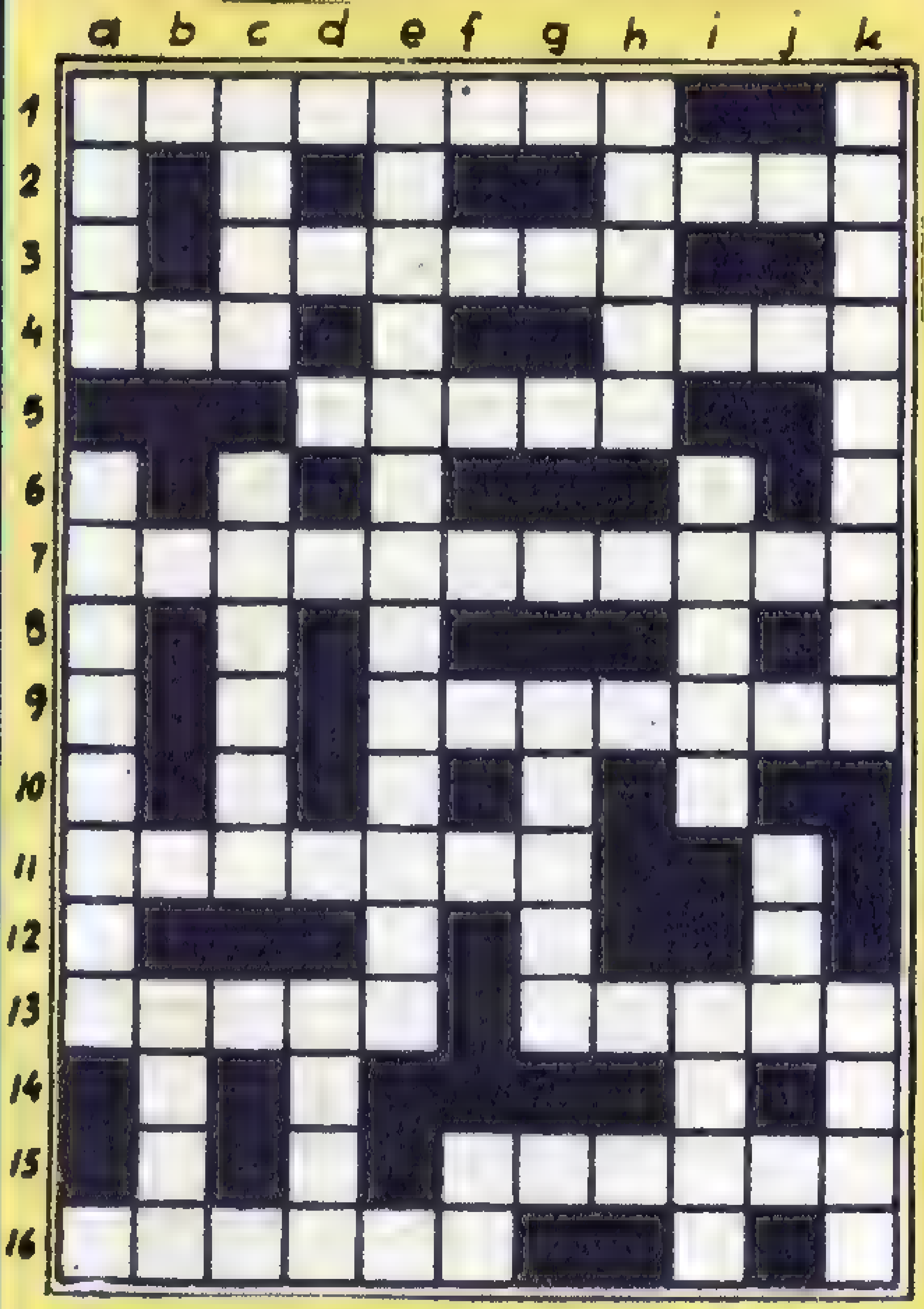
Rys. Michał Przybyłowski

Przepraszamy...

„IKS" nr 5, program „Wykres funkcji" — powtarzamy nieczytelne fragmenty

```
500 REM WYZNACZENIE WSP. SKALI
510 XZ=XK-XP:YZ=YG-YD
520 DX=XZ/239:DY=YZ/159
530 CY=INT(ABS(YD*(YD<0))/DY):CY=160-CY
540 CX=INT(ABS(XP*(XP<0))/DX)+40
```

Krzyżówka nr 6



POZIOMO: 1-a) Kapelusz lub część silnika, 2-h) Praca w polu, 3-c) „Wypadek" na ringu, 4-a) Koleżanka Oli, 4-h) Jeden z gromady stawonogów, 5-d) Tkanina runowa lub miasto w Indiach, 7-a) Światłoczuły barwnik w pręcikach siatkówki oka, 9-e) Rodzaj kaktusa, 11-a) Miasto w ZSRR, 13-a) Rój meteorów, 13-g) Port nad Niemnem, 15-f) Gryzoń z komputera, 16-a) Wygnaniec.

PIONOWO: 1-a) Ważna dla kobiety, 1-c) Rzeka lub imię żeńskie, 1-e) Występuje w jądrze i cytoplaźmie każdej komórki, 1-h) Wirnik, 1-k) Kapłan w przedchrześcijańskiej Litwie, 6-a) Mieszkaniec Nilu, 6-c) Imię męskie, 6-i) W piosence nauczycielka..., 9-g) Deformacja tektoniczna warstw skalnych, 11-j) Autobus lub rzeka, 13-b) Tańcowała z nitką, 13-d) Tajemnica Tybetu, 13-i) Zezwolenie na pobyt za granicą, 13-k) Państwo na Płw. Arabskim, 15-f) Jedna ze stron konta.

Hasło: 9-f, 7-a, 15-l, 13-d, 9-f, 2-h, 15-f, 9-c, 13-j, 2-k, 15-f, 13-d, 3-d, 9-f, 1-h, 5-e, 9-h, 5-g, 14-k, 1-g, 5-h, 12-j, 9-i, 13-b, 14-d, 16-d, 11-g, 15-h, 4-c.

Rozwiązania (tylko hasło) prosimy przesyłać pod adresem redakcji do końca listopada. Wśród Czytelników, którzy nadeślą prawidłowe rozwiązania rozlosujemy bony pieniężne i nagrody książkowe.



Oto najnowszy produkt firmy ICL — DRS 300. Najważniejszą jego zaletą jest modułowa konstrukcja; podstawową konfigurację można łatwo rozbudowywać dołączając kolejne urządzenia. Nie ma przy tym plątaniny kabli; foremne moduły idealnie pasują do siebie, a można je ustawić nawet w podręcznej biblioteczce.



Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera, Ryszard Rogoń. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8112. Nr ind. 382809

P-8 5

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 7/1986 ISSN 0860—2794



Foto: Jan Zelman

INFORMATYKA KOMPUTERY SYSTEMY

Mamy kolejny jubileusz, minęło dwadzieścia pięć lat wojskowej informatyki, stąd okazja do sentymentalnych wspomnień. Przyznam się, że z tych pierwszych lat niewiele pamiętam. Kiedy w Wojskowej Akademii Technicznej powstawały konstrukcje pierwszych maszyn liczących przygotowywałem się do konkursowego egzaminu na pierwszy rok studiów w szkole podstawowej. „Jaga" była dla mnie czarnym charakterem z przedszkolnej literatury — dopiero parę dziesiątek lat później dowiedziałem się, że „Jaga" to również udana konstrukcja dwóch młodych naukowców z Katedry Automatyki i Telesterowania WAT — Jarosińskiego i Gajewskiego.

Kto dziś pamięta BINUZa — szeregową, jednoadresową maszynę liczącą z arytmetyką binarną i uzupełnieniem stałoprzecinkowym? I jeszcze jedno wspomnienie: dodawanie i odejmowanie trwało 7 sekund, mnożenie 35, a dzielenie aż 42 sekundy! — to parametry przekaźnikowej maszyny cyfrowej skonstruowanej przez kapitana Stanisława Jarosińskiego w 1959 roku w WAT. Na tej maszynie zawodowe ostrogi zdobywali pierwsi informatycy w mundurach. To oczywiście nie wszystko: były jeszcze niezwykle udane konstrukcje maszyn analogowych zespołu, wiedy również kapitana, Józefa Kapicy. Pierwszy był ELWAT-1 przeznaczony do syntezy i analizy układów dynamicznych, później konstrukcję tę wielokrotnie modyfikowano. Powstała również rodzina komputerów — łączących cechy maszyn cyfrowych i analogowych.

Tyle wspomnienia. Jubilatom zaś życzymy jeszcze wielu, wielu udanych pomysłów, a przede wszystkim krótkiej i łatwej ich realizacji.

Nie krzepmy się jednak wspomnieniami, informatyka jest bowiem dyscypliną, która nie znosi bezczynności. To, co wiemy dzisiaj już jutro może nie wystarczać. Zapowiadamy zatem kilka zmian; przede wszystkim od pierwszego numeru przyszłego roku zmieniamy winietę. Nowa szata graficzna, z pewnością ładniejsza, i rozszerzenie tematyki to zapowiedź na najbliższe miesiące — zgodnie z życzeniem Czytelników rejestrować będziemy na naszych łamach wszelkie nowości.

Mamy nadzieję, że poprawi się również kolportaż „IKS-a", znikną kłopoty z jego zakupem.

To oczywiście nie wszystko, inne zmiany już wewnątrz niniejszego numeru.

WIESŁAW CETERA

Infogryf '86

Planowy, i niestety rzeczywisty, przyjazd pociągu relacji Warszawa — Kołobrzeg godz. 5.05! (bo niby dlaczego pasażerowie mieliby opuszczać wygodny wagon sypialny o 7 lub 8 godzinie...). Mały spacerek „Kołobrzeg nocą" i już o godz. 10 dnia 21.10.86 r. rozpoczął się VIII INFOGRYF.

Jest to chyba największe krajowe forum informatyków związanych z teorią i praktyką wdrażania komputerów w przemyśle, i nie tylko.

Czternaście lat temu przewodniczący Sekcji Informatyki TNOiK Szczecin — Edward Kolbusz — rozpoczął organizację pierwszego Infogryfu. Nazwę tę zaproponował (na trasie Warszawa — Szczecin), ówczesny dyrektor Zeto Szczecin, Zdzisław Bogdanowicz. Współorganizatorami tej szacownej już dzisiaj imprezy były szczecińskie instytucje: Politechnika, TNOiK i ZETO. Od V Infogryfu do grupy tej dołączyło Zeto Koszalin.

Po raz pierwszy w tym roku zamiast Politechniki, współorganizatorem Konferencji został Uniwersytet Szczeciński. Jest to wynikiem przejścia jej duchowego przywódcy profesora Tadeusza Wierzbickiego wraz z Wydziałem Ekonomicznym z Politechniki do Uniwersytetu Szczecińskiego.

INFOGRYF'86 otworzył prof. Wierzbicki witając serdecznie uczestników konferencji, w tym gości z NRD i Francji. Jednocześnie dał wyraz szczeremu ubolewaniu, że nie może powitać przedstawicieli władz decydujących o kształcie polskiej informatyki, ponieważ nie zaszczyciły swoją obecnością konferencji.

Wykład inauguracyjny, który okazał się felietonem, wygłosił sekretarz naukowy Infogryfu dr hab. Wojciech Olejniczak. Wzbudzał on śmiech, smutek i inne bardzo kontrowersyjne odczucia. Felieton ten dotyczył bowiem EFEKTU SKALI, który, definiując przez przykład, polega na tym, iż nagłe zniknięcie długopisów mogło by doprowadzić do katastrofalnych następstw, natomiast gdyby to (zniknięcie) dotyczyło komputerów (wszystkich w kraju) negatywne skutki byłyby znacznie mniejsze — nie ta skala?



Obrady toczyły się w trzech sekcjach:

I. Systemy, metody i techniki tworzenia systemów informatycznych.

II. Komputerowe systemy wspomagania decyzji.

III. MIKROSKALA'86 — zastosowanie mikrokomputerów w przedsiębiorstwie.

Ponadto odbywały się wykłady WSZECHNICY.

Równoległy przebieg obrad we wszystkich sekcjach był jednym z istotnych potknięć organizacyjnych Infogryfu. Okazało się bowiem, co można było przewidzieć, że Mikroskala była praktycznie bezkonkurencyjną sekcją. Oznaczało to minimalną frekwencję na pozostałych dwóch salach obrad, gdzie były wygłaszane nie mniej wartościowe referaty, jednak nie tak atrakcyjne (modne jak mikrokomputery).

Tematem jednego z wyróżniających się wykładów były systemy operacyjne mikrokomputerów. Autor — Jan MADEY z UW — w sposób niezmiernie interesujący zaprezentował podstawowe systemy operacyjne CP/M i jego pochodne, MS-DOS i PC-DOS ze szczególnym podkreśleniem architektury zestawu mikro, oraz czynności użytkownika. UNIX został określony przez autora wykładu jako fenomen systemów operacyjnych, podobnie jak Pascal wśród języków programowania.

Bardzo ciekawy był także wykład pani profesor Chislaine DUFOURD z Universite Louis Pasteur w Strasburgu — „Informatyka dla wszystkich: doświadczenia francuskie". Wykład dotyczył problemów edukacji w kontekście rozszerzania się sfery zastosowań informatyki z punktu widzenia sprzętu, oprogramowania, a także przygotowania kadr nauczycielskich. W trakcie wykładu autorka wprowadziła oryginalny termin — nanokomputery.

Cenne uzupełnienie konferencji stanowią cztery tomy referatów i wykładów (brak nazw firm, autorów, adresów do korespondencji jest istotnym mankamentem).

Podczas INFOGRYFU'86 odbywała się wystawa sprzętu komputerowego. Przez kilka godzin można było oglądać ELWRO 800 — mikrokomputer firmy ELWRO, którego kilkadziesiąt egzemplarzy prawdopodobnie będzie wykonane w najbliższej przyszłości... Natomiast „od ręki" można było kupić prezentowany zestaw, kompatybilny z IBM PC/AT, oferowany przez firmę MIKROKOMPUTERY za 14,9 mln zł. Podobną ofertę złożył COMPUTEX (CS-88PC i CS-286PC, odpowiednio XT i AT). Firma DSE ze Sztokholmu proponowała komputer DSE TURBO 1 (kompat. z XT) i DSE TURBO 2 (kompat. z AT) odpowiednio za 2300 i 3300 dolarów. Reprezentowane były także firmy HIROSS i SELIN proponujące kompletne wyposażenie sal komputerowych (w tym poszukiwana podwójna podłoga!), klimatory, przetwornice. Stoisko swoje miało także COMPUTER STUDIO KAJKOWSCY.

Organizatorzy konferencji twierdzili, iż Infogryf jest taki jak jego uczestnicy. W ocenie wielu osób był chyba trochę zbyt teoretyczny, za co zapewne nie można winić samych referentów. Faktem jest natomiast, że korzyści z osobistych kontaktów przedstawicieli różnych firm, giełda sprzętu i oprogramowania stanowiły wystarczające uzupełnienie INFOGRYFU'86 — był nadal miarodajnym źródłem danych i opinii o stanie informatyki w kraju. Do zobaczenia za dwa lata.

UCZESTNIK



Dzięki uprzejmości firmy DSE można ocenić drobny przykład pomocy mikro — wykonanie prezentowanych rycin trwało kilkanaście minut.

UNIX

Są dwie historie UNIXa. Jedna mówi, iż UNIX jest znany na całym świecie, druga, że nie. Każda z nich ma swych zwolenników, których liczba co roku się zmienia.

Obecnie zakres potencjalnych maszyn tego systemu operacyjnego, takich jak IBM PC AT i Big Blue oferujących UNIX jako swój podstawowy system odnowił związane z nim spekulacje. Czy tylko dlatego, że to co IBM myśli dzisiaj, reszta świata próbuje powielać jutro?

Jak wygląda UNIX z bliska? Co to jest? Skąd pochodzi? Dlaczego ludzie nad nim pracują? Jakie są jego perspektywy? Czy stanie się standardem systemu operacyjnego następnej generacji mikrokomputerów?

Dla wielu, UNIX jest sposobem życia, religią lub obiektem miłości lub nienawiści. Dla pozostałych jest pewnym systemem operacyjnym — to znaczy programem, który czyni, iż części składowe (hardware) komputera pracują. Bez systemu operacyjnego komputer może służyć tylko jako kosztowny mebel

System operacyjny jest oprogramowaniem, które mówi jak wprowadzić i wyprowadzić informacje, jak sterować pracą dysków, zbiorów i programów i jak dzielić moc obliczeniową komputera na wielu użytkowników oraz urządzenia zewnętrzne

W systemach mikroprocesorowych, tylko mała część systemu operacyjnego jest zależna od specyfiki komputera; jego większa część jest niezależna od sprzętowej konfiguracji maszyny. Oznacza to, że system operacyjny może być łatwo dostosowany do różnych maszyn, których jedyną wspólną charakterystyką jest to, że wykorzystują one ten sam mikroprocesor (CPU). W każdym przypadku tylko specyficzne dla danej maszyny kody powinny być uwzględnione, reszta może pracować bez zmian.

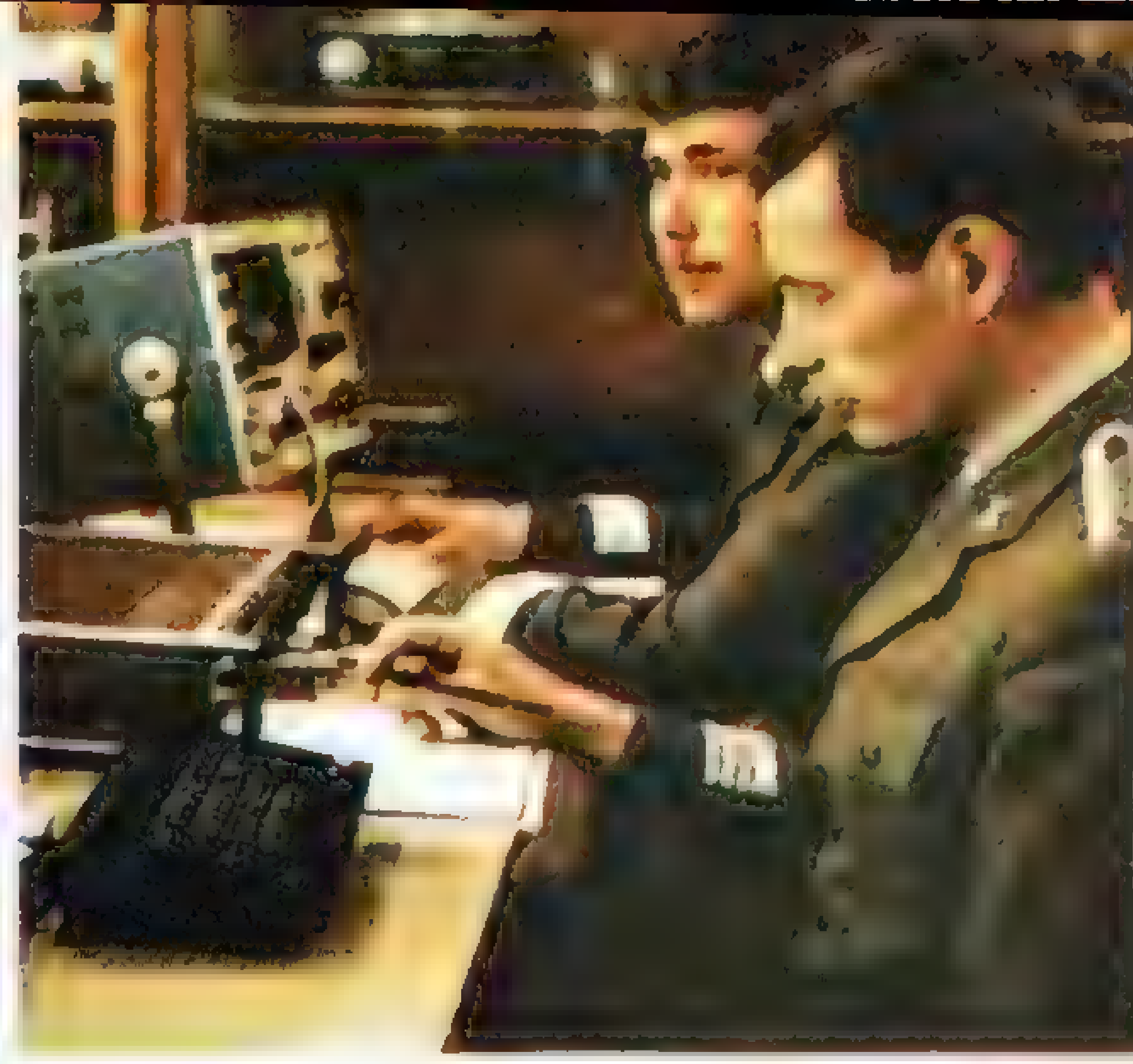
Z innego punktu widzenia oznacza to, że jeśli programiści piszą programy przeznaczone do określonego typu systemu operacyjnego, mają oni duży potencjalny rynek dla swojej produkcji w istocie tworzą go wszystkie maszyny, w których pracuje ten szczególny typ systemu operacyjnego

Praktycznie mikrokomputery miały dwa znaczące systemy operacyjne. Pierwszym był **CP/M** (Control Program for Micros) napisany przez Gary Kidalla dla procesorów Z80 i 8080. CP/M stał

dokończenie na str. 5

29 listopada – Dzień Podchorążego







Wszystkim studentom w żołnierskich mundurach z okazji ich święta składamy najlepsze życzenia.

REDAKCJA

Foto: S. Iwan



dokończenie ze str. 3

się nie tylko podstawą firmy **Kidalla — Digital Research,** ale standardem dla komputerów 8-bitowych. Rezultatem tego była znaczna liczba programów pracujących pod CP/M

Kiedy IBM zaprojektował swój PC Digital Research opracować miał system operacyjny dla procesora 8088. Stworzono wiele różnych wersji, ale ostatecznie oprogramowanie wykonała mała firma — **MICROSOFT.** Efektem był **MS DOS** — rozbudowane wersje CP/M.

CP/M i MS DOS są przeznaczone dla indywidualnych użytkowników, niemozliwa jest praca w tak zwanym wielodostępie. Stan ten był zadowalający do chwili, gdy mikrokomputery miały ograniczony maksymalny adres pamięci równy 640 KB i szybkość przetwarzania nie przekraczającą 4,77 MHz.

Powstanie procesorów o większej mocy obliczeniowej, znacznie przekraczającej potrzeby jednego użytkownika takich jak: INTEL 80286, MOTOROLA 68020 i ZILOG Z8000 oraz olbrzymi spadek cen pamięci, spowodowało pilną potrzebę posiadania systemu, który może obsłużyć wielu użytkowników i realizować wiele zadań jednocześnie.

Wykorzystanie do tego celu dotychczasowych systemów napotykało na wiele trudności. Dwie z nich są szczególnie ważne. Pierwsza — większość dużych systemów operacyjnych jest specyficzna dla danej maszyny. Na przykład IBM OS — 360 pracuje tylko na komputerach z rodziny IBM 360. VMS Digital'a pracuje tylko na procesorach rodziny VMS. Druga trudność także wynika ze specyfiki maszyn, a w zasadzie właścicieli oprogramowania, którzy zazwyczaj niechętnie umożliwiają dopasowanie systemu do podobnej maszyny, ale produkowanej w innej firmie

UNIX natomiast jest wyjątkiem wśród większości wspomnianych trudności. Po pierwsze jest to system wielodostępny, wieloprogramowy bazujący na olbrzymich doświadczeniach zdobytych na całym świecie. Praktycznie każdy uniwersytet i komputerowy zakład naukowy na Zachodzie ma conajmniej jedną maszynę pracującą pod UNIXem. UNIX jest jednocześnie „stołowym" systemem, **ponieważ w odróżnieniu od innych systemów nie jest napisany w specyficznym dla danej maszyny assemblerze, ale zazwyczaj w wysokopoziomowym języku C.**

UNIX z powodzeniem został wykorzystany w dominujących na rynku rodzinach procesorów 16/32-bitowych włączając 68000, 80x86 i Z8000. Niezadowoleni użytkownicy MS DOS chętnie i efektywnie wykorzystują UNIXa

Właścicielem UNIXa jest korporacja **AT&T,** która dotychczas wcale nie była znana.

W zasadzie UNIX jest nie tylko systemem operacyjnym. Lepszym jego określeniem jest system operacyjny plus komputerowe otoczenie. Zatem, dodatkowo do zabezpieczenia podstawowych zadań pozwalających na pracę w wielodostępie użytkowników wykorzystujących różne programy UNIX oferuje obsługę zaskakującej liczby urządzeń i mozliwości — języki, kompilatory, detektory błędów, dodatki dokumentacyjne, edytory i możliwości przetwarzania tekstów, pocztę elektroniczną, nadzór zbiorów, możliwości dokumentacji itp.

Inne systemy operacyjne posiadają podobne możliwości, ale wyłącznie jako pojedyncze dodatki — jedynie UNIX oferuje je jako standard, integralną część pakietu.

Część własności UNIXa ma na celu ułatwienie pracy programistom. Wiele wersji tego systemu posiada pakiet, który zawiera SOURCE CODE CONTROL SYSTEM (SCCS). Jest to pakiet programów, który automatycznie śledzi i dokumentuje każdą nową lub poprawioną wersję programu, zaznaczając datę, nazwisko autora itp.

Znaczna część pomysłów, rozwiązań może być wykorzystana w innych systemach, a sam UNIX dzięki oryginalnym pomysłom stał się dostępny także nie-programistom.

Można również zauważyć pewien wpływ tego nowego standardu na inne systemy? Na przykład hierarchiczna struktura zbioru w MS DOS v2.0 jest niemal wierną kopią konstrukcji z UNIXa.

Jeśli chcesz zrozumieć UNIXa to wyobraź sobie go jako owoc. W środku, gdzie zazwyczaj bywa pestka, jest jądro UNIXa — podstawowe kody, które pozwalają na pracę systemu. Na zewnątrz pestki jest muszla — interfejs między jądrem, a Tobą.

Jedną z przyczyn, z powodu które, ludzie się ekscytują UNIXem jest siła i elastyczność wspomnianej muszli. Ponadto UNIX oferuje ponad 120 samodzielnych komend (instrukcji). Część z nich jest bardzo podobna do innych prostszych systemów. Na przykład „cp" w UNIXie jest podobne do „copy" w MS DOS. Użytkownicy mogą decydować o priorytetach realizowanych przez komputer zadań.

Już w ubiegłym roku sprzedano 100 000 UNIXów, ale MS DOS aż trzy miliony. Jednak proporcję tę należy szacować mając na uwadze, że każdy UNIX dotyczy około 10 użytkowników (wielodostęp).

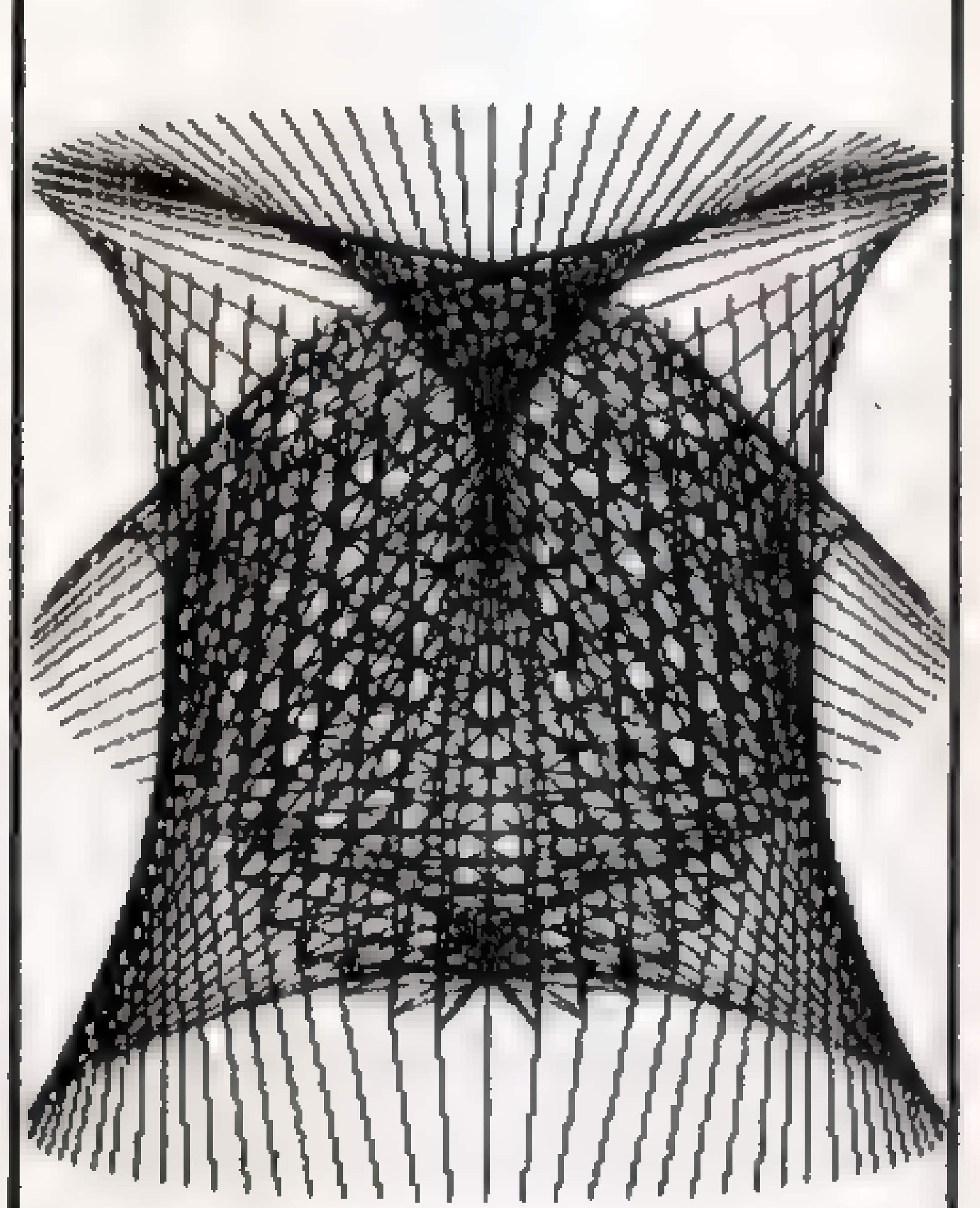
Istnieje wiele przyczyn stosunkowo niewielkiej powszechności UNIXa. Do niedawna jedną z nich było małe zainteresowanie producenta, firmy AT&T w dystrybucji tego systemu. Inną przyczyną jest obecnie zbyt ubogi sprzęt — aby UNIX poprawnie pracował potrzebne jest — 1MB RAM i 20 MB twardy dysk. Po trzecie dotychczas rynek był na tyle nasycony 16-bitowymi komputerami dla jednego użytkownika, ze zainteresowanie systemem wielodostępu było nikłe.

Stan ten zmienił się radykalnie. Firma AT&T rozszerzyła swoje komercyjne zainteresowania rynkiem komputerowym. Ponadto IBM wydaje się być poważnie zainteresowany UNIXem. Dowodzą tego starania zlikwidowania technicznych ograniczeń, tym bardziej że użytkownicy IBM PC dostrzegają pewne ograniczenia, które narzuca 8088/86 MS DOS. Maksymalna dostępna pod tym systemem pamięć wynosi zaledwie 640 KB, a np. pakiet typu SYMPHONY, tylko dla celów systemowych, wymaga 320 KB. Oznacza to po prostu głód pamięci RAM do celów użytkowych

Procesor 80286, który pracuje w IBM AT może operować na pamięci ponad 4 MB, ale nie pod systemem MS DOS. Dla pojedynczych użytkowników całą tak dużą pamięć nadzoruje DOS 286.

Zalety UNIXa dostrzegło wiele firm deklarując ścisłą współpracę w rozwoju tego systemu np.: ICL, NIXDORF, PHILIPS, SIEMENS, BULL, OLIVETTI i oczywiście Japończycy.

Włodzimierz GOGOŁEK



Grafika — AMSTRAD

ARCHITEKTURA INTEGRACJI

W 1971 r. firma INTEL skonstruowała pierwszy na świecie mikroprocesor, oznaczony symbolem 4004, o architekturze czterobitowej a wykonany w technologii PMOS. Dało to początek rywalizacji kilkudziesięciu firm i tysięcy inżynierów o najkorzystniejsze rozwiązania architektury mikroprocesorów i technologii ich wytwarzania. Wśród firm obecnie dominujących na świecie należy wymienić: INTEL, ZILOG, MOTOROLA, MOS TECHNOLOGY, RCA, FAIRCHILD, NATIONAL SEMICONDUCTOR, TEXAS INSTRUMENTS, TOSHIBA.

Mikroprocesor poza przewrotem, którego dokonał w przemyśle komputerowym, na stałe pojawił się w automatyce przemysłowej, w telekomunikacji, w medycynie, w technice pomiarowej, w sprzęcie powszechnego użytku i w technice wojskowej.

Mikroprocesorem nazywana jest jednostka centralna (CPU — Central Processor Unit) systemu mikrokomputerowego, wykonana w postaci jednego lub kilku układów wielkiej skali integracji. Mikroprocesor przeznaczony jest do wykonania dowolnego ciągu **operacji (rozkazów)** ze zbioru określanego jako **lista rozkazów**, a zawierającego zestaw operacji podstawowych mikroprocesora danego typu.

Ciąg operacji oraz dane, które stanowią **argumenty operacji**, tworzą **program** dla systemu mikrokomputerowego.

W tym miejscu autor chce się przyznać, iż nie może podać precyzyjnych definicji **systemu mikrokomputerowego** i **systemu mikroprocesorowego**. Fizycznie systemy te nie różnią się, gdyż składają się z tych samych elementów. Można przyjąć, że systemem mikrokomputerowym jest system wyposażony w **język programowania wyższego poziomu**, daleki od języka dwójkowego, którym posługuje się procesor a możliwie zbliżony swoją symboliką do języka ludzi.

Wróćmy do elementów, z których zbudowany jest system mikrokomputerowy (system mikroprocesorowy). Zawiera on następujące elementy, przedstawione na rys. 1.:

— mikroprocesor, przeznaczony do przetwarzania informacji, inaczej wykonania operacji na danych i do sterowania pracą pozostałych elementów systemu;

— pamięci: stałą ROM (Read — Only — Memory) i zapisywalną RAM (Random — Access — Memory);

— układy wejścia — wyjścia.

Wymienione elementy systemu połączone są ze sobą za pomocą **magistral**, w tym: **magistrali adresowej** (Address Bus), **magistrali danych** (Data Bus) i **magistrali sterującej** (Control Bus), w prostszych rozwiązaniach określanej jako **linie sterujące**.

W **pamięci stałej ROM** przechowywany jest program oraz dane nie zmieniające się w czasie pracy systemu, znajdujące się na trwałe nawet po wyłączeniu zasilania. Pamięć ta zapisywana jest przez producenta, znajdują się w niej **interpretery**, **system operacyjny** itp. Przykładowo w komputerze AMSTRAD CPC 464 znajduje się pamięć ROM o pojemności 32 Kilobajty (bajt — słowo 8-bitowe, inaczej długość słowa mikroprocesora 8-bitowego, przy czym 1 KB = 1000 bajtów). W pamięci tej 16 KB to system operacyjny i 16 KB to interpreter języka BASIC.

W **pamięci zapisywalnej RAM**, zwanej **pamięcią operacyjną**, zapisywane (i odczytywane) są dane powstałe w czasie pracy systemu oraz programy użytkownika.

Układy wejścia — wyjścia umożliwiają komunikację systemu z otoczeniem czyli wszelkimi urządzeniami zewnętrznymi np. monitorem, klawiaturą, pamięciami masowymi, urządzeniami pomiarowymi itp.

Wymienione zasadnicze elementy systemu wykonane są jako układy LSI technologią MOS, umożliwiającą osiągnięcie dużej **integracji (scalenia)** tranzystorów. Musimy pamiętać, że tranzystory to podstawowy element używany do konstrukcji mikroprocesorów. Tranzystory te noszą nazwę **unipolarnych** typu MOS. Technologia MOS zapewnia zastępowanie innych elementów konstrukcyjnych mikroprocesorów np. rezystorów również tranzystorami. Pozwala ona na osiągnięcie integracji, jak we wspomnianym w poprzednim numerze „IKS-a" koprocesorze arytmetycznym MC 68881.

W systemach mikrokomputerowych do budowy połączeń sprzęgających ich elementy, w dalszym ciągu wykorzystywane są **bipolarne** układy MSI i SSI. Technologiami bipolarnymi wykonywane są również **mikroprocesory segmentowe**.

Wróćmy na chwilę do terminologii. Określenie mikrokomputer pojawiło się przez analogię do mikroprocesora. Obecnie „mikro-" nie świadczy o małych możliwościach, należy to chyba rozumieć z punktu widzenia technologii produkcji. Przykładowo porównajmy komputer IBM 650 z dwoma mikroprocesorami, z Z-80 i MC 68000. Komputer IBM 650 posiadał pamięć bębnową o pojemności 2000 słów, miał listę 60 rozkazów, czas wykonywania rozkazu wynosił 20 milisekund [ms]. W 8-bitowym Z-80 czas wykonania rozkazu najkrótszego/najdłuższego wynosi 1 μs (5,75 μs czyli 0,001 ms) 0,00575 ms, mikroprocesor adresuje 64 KB przestrzeni pamięci i posiada listę zawierającą 158 rozkazów. 16-bitowy MC 68000 posiada listę 61 rozkazów, czas wykonania rozkazu 0,5 μs czyli 0,0005 ms i adresuje przestrzeń adresową 16 megabajtów [MB]!!

Zanim przejdziemy do ogólnej charakterystyki mikroprocesorów jeszcze kilka słów. Obecnie budowane są systemy mikrokomputerowe złożone z wielu mikroprocesorów, o różnym przeznaczeniu, są to tzw. **systemy wieloprocesorowe**, charakteryzujące się dużą efektywnością i niezawodnością pracy. Stosuje się w tych systemach m.in. **koprocesory arytmetyczne**, przeznaczone do wykonywania operacji arytmetycznych (których nie wykonuje już **mikroprocesor centralny**) oraz stosuje się **mikroprocesory wejścia — wyjścia**, przeznaczone do obsługi szybkich transmisji między urządzeniami zewnętrznymi a pamięcią operacyjną. Przykładowo 16-bitowy komputer personalny IBM PCAT zawiera, obok mikroprocesora 80286, koprocesor arytmetyczny 80287.

Mikroprocesory można zgrupować w cztery **generacje**.

Mikroprocesorami **pierwszej generacji** są dwie konstrukcje firmy INTEL, czterobitowy **4004** i ośmiobitowy **8008**, charakteryzujące się ubogimi listami rozkazów i małymi szybkościami działania.

Czterobitowy mikroprocesor INTEL 4004, zaprojektowany i wykonany w 1971 r., posiada listę 46 rozkazów, w tym 16 rozkazów procesora, 14 rozkazów akumulatora i 16 rozkazów wejścia—wyjścia. W skład systemu ozna-

czonego MCS-4 wchodzą, oprócz mikroprocesora, układ pamięci ROM 4001 o pojemności 256 × 4 bity, układ pamięci RAM 4002 o pojemności 80 × 4 bity, układ 4003 do współpracy z wejściem — wyjściem do — z systemu. Cały cykl rozkazowy mikroprocesora trwa około 10,8 μs.

INTEL 4004, wykonany w technologii PMOS, może adresować przestrzeń 4 K czyli 4096 słów 4-bitowych. Rozwinięciem rodziny MCS-4 jest MCS-40. INTEL 4040, adresuje 8192 słowa 4-bitowe. Jego architektura jest rozwinięta w stosunku do poprzednika, umożliwia ona obsługę **przerwań jednopoziomowych**.

Ośmiobitowy mikroprocesor INTEL 8008, posiadający listę 45 rozkazów, stanowi rozwinięcie architektury mikroprocesora 4004. Posiada on możliwość działania na danych ośmiobitowych. Produkcję mikroprocesora rozpoczęto w 1972 r.

Cały cykl rozkazowy 8008 wynosi około 12,5 μs. Mikroprocesor może adresować 16 KB czyli 16 384 słowa 8-bitowe. MCS-8 jest systemem równoległym, wykonanym w technologii N MOS, komunikującym się z pamięcią i układami wejścia — wyjścia przez magistrale 8-bitowe. Mikroprocesor 8008 posiada listę 45 rozkazów, w tym rozkazy 8-, 16- i 24-bitowe.

INTEL 8008 zapoczątkował rozwój najliczniejszej grupy mikroprocesorów ośmiobitowych. W ciągu następnych kilku lat powstało kilkadziesiąt różnych konstrukcji mikroprocesorów ośmiobitowych, wykonanych przede wszystkim technologią N MOS, tworzących **drugą generację**.

Najważniejszymi przedstawicielami drugiej generacji mikroprocesorów są rodziny INTEL MCS-80 i MCS-85, mikroprocesor firmy MOTOROLA MC 6800 i rodzina mikroprocesorów firmy MOS TECHNOLOGY MCS 650X. Mikroprocesory te posiadają dużo większe szybkości działania, bogatsze listy rozkazowe i przestrzeń adresową 64 KB. Wykonano je technologią N MOS.

Największą popularność z wymienionych wyżej mikroprocesorów zyskał 8080. Wprowadzony na rynek w 1974 r. dominował przez kilka lat. Wzięło się to z monopolu firmy INTEL na produkcję pamięci (stałych) programowalnych PROM i reprogramowalnych EPROM, wśród nich standardem światowym stały się EPROM 1702 A o pojemności 256 × 8 bitów i EPROM 2708 o pojemności 1024 × 8 bitów. Charakterystyką mikroprocesorów 8008 i 8085 zajmiemy się w kolejnych numerach „IKS-a", a więc na razie tylko tyle.

Mikroprocesor MC 6800 znacznie przewyższa mikroprocesor 8080 zarówno łatwością organizacji systemu jak również: mocą obliczeniową, szybką obsługą przerwań, elastycznością adresowania i wymaga on tylko jednego napięcia zasilającego +5 V. W skład rodziny MC 6800 wchodzą następujące układy towarzyszące: układy interfejsu MC 6820, MC 6850, MC 6852, MC 6843, MC 6854, MC 68488, układ bezpośredniego dostępu do pamięci (DMA—Direct Memory Access) MC 6844, programowalny zegar czasu rzeczywistego MC 6840, modulator cyfrowy MC 6862, modem MC 6860 i inne.

Podstawowe cechy charakterystyczne mikroprocesora 6800 to: 16-bitowa magistrala adresowa, 8-bitowa magistrala danych, lista złożona z 72 rozkazów, czas wykonania rozkazu najkrótszego/najdłuższego 1/2,5 μs.

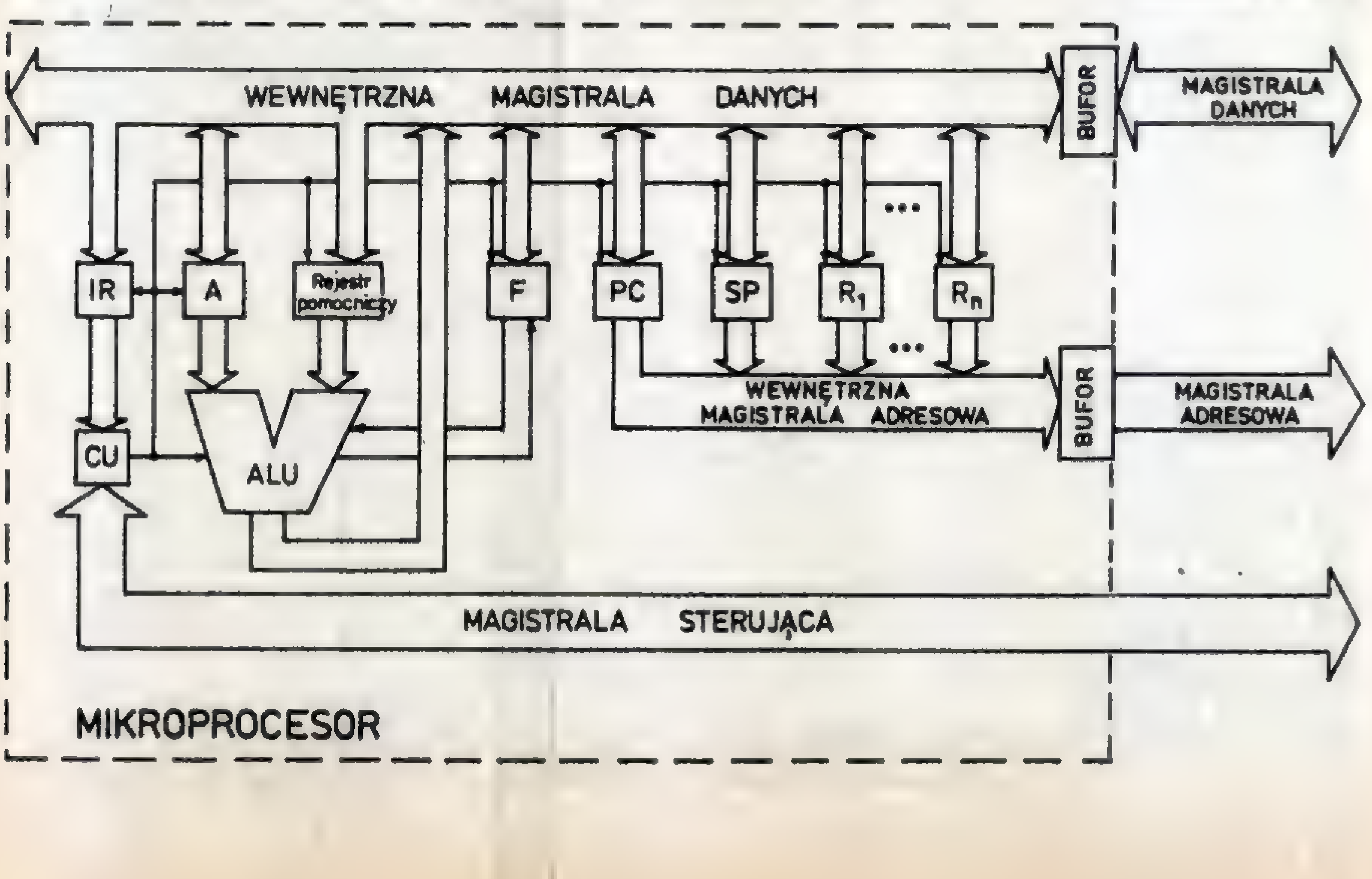
Produkcję układów rodziny mikroprocesorowej MOTOROLA MC 6800 uruchomiono w Bułgarii, oznaczając rodzinę symbolem CM 600. Z uwagi na niewielkie (jak dotąd) zainteresowanie tym mikroprocesorem w naszym kraju, nie będziemy się nim zajmować. Mikroprocesor ten możemy spotkać w nowoczesnej aparaturze kontrolno-pomiarowej firm Hewlett-Packard czy Tektronix.

Udoskonalenie MC 6800 stanowią mikroprocesory rodziny MCS 650X, opracowane w firmie MOS TECHNOLOGY. Najważniejsze cechy tej rodziny to: duża liczba sposobów adresowania (13), elastyczność przerwań oraz zakładkowe wykonywanie instrukcji — wykonanie jednej mikroinstrukcji i interpretacja następnej w czasie jednego cyklu zegarowego. Mikroprocesor MCS 6502, mimo tylko 56-elementowej listy rozkazów uważany jest za najlepszego przedstawiciela drugiej generacji.

Trzecią generację otwierają, w dalszym ciągu 8-bitowe, mikroprocesor ZILOG Z-80, wprowadzony na rynek w 1976 r. i ulepszona wersja MC 6800 oznaczona MC 6809, wprowadzona w 1977 roku. Mikroprocesor Z-80 stanowi daleko idące ulepszenie i rozwinięcie INTEL-a 8080, będzie również dokładnie omawiany w następnych numerach „IKS-a". Zasadniczy trzon generacji tworzą mikroprocesory 16-bitowe. Najważniejszymi ich przedstawicielami są mikroprocesory: INTEL 8086 i INTEL 8088, MOTOROLA 68000 i ZILOG Z-8000. Na pograniczu **czwartej generacji** jest mikroprocesor INTEL 80286. Z uwagi na pewien ciąg rozwojowy zajmiemy się również 16-bitowymi mikroprocesorami firmy INTEL.

Czwarta generacja to mikroprocesory 32-bitowe, jeden firmy INTEL oznaczony 80386 i jeden firmy MOTOROLA

dokończenie na str. 8



dokończenie ze str. 7

MC 68020. Dzięki nim **piąta generacja** komputerów jest już coraz bliżej.

Po ogólnej charakterystyce mikroprocesorów zapoznajmy się z zasadą ich działania i architekturą.

W każdym z mikroprocesorów, zgodnie ze znaną nam koncepcją Von Neumanna, występują następujące podzespoły, przedstawione na rys. 2:

— **jednostka arytmetyczno-logiczna Alu** (Arithmetic and Logic Unit);
— **układ sterowania CU** (Control Unit);
— **akumulator A** (Accumulator);
— **zestaw rejestrów roboczych R_n** (General Purpose Registers);
— **rejestr rozkazów IR** (Instruction Register);
— **licznik rozkazów PC** (Program Counter);
— **wskaźnik stosu SP** (Stack Pointer);
— **rejestr wskaźników SF** (Status Flag).

Cykl rozkazowy mikroprocesora rozpoczyna się od fazy pobrania rozkazu do **rejestru rozkazów IR**, z komórki pamięci adresowanej przez **licznik rozkazów PC**. Po pobraniu przez mikroprocesor słowa z **pamięci programu** licznik zwiększa swoją zawartość o jeden tzn. adresuje następną komórkę pamięci lub w przypadku rozkazu skoku skokowo zmienia swoją zawartość.

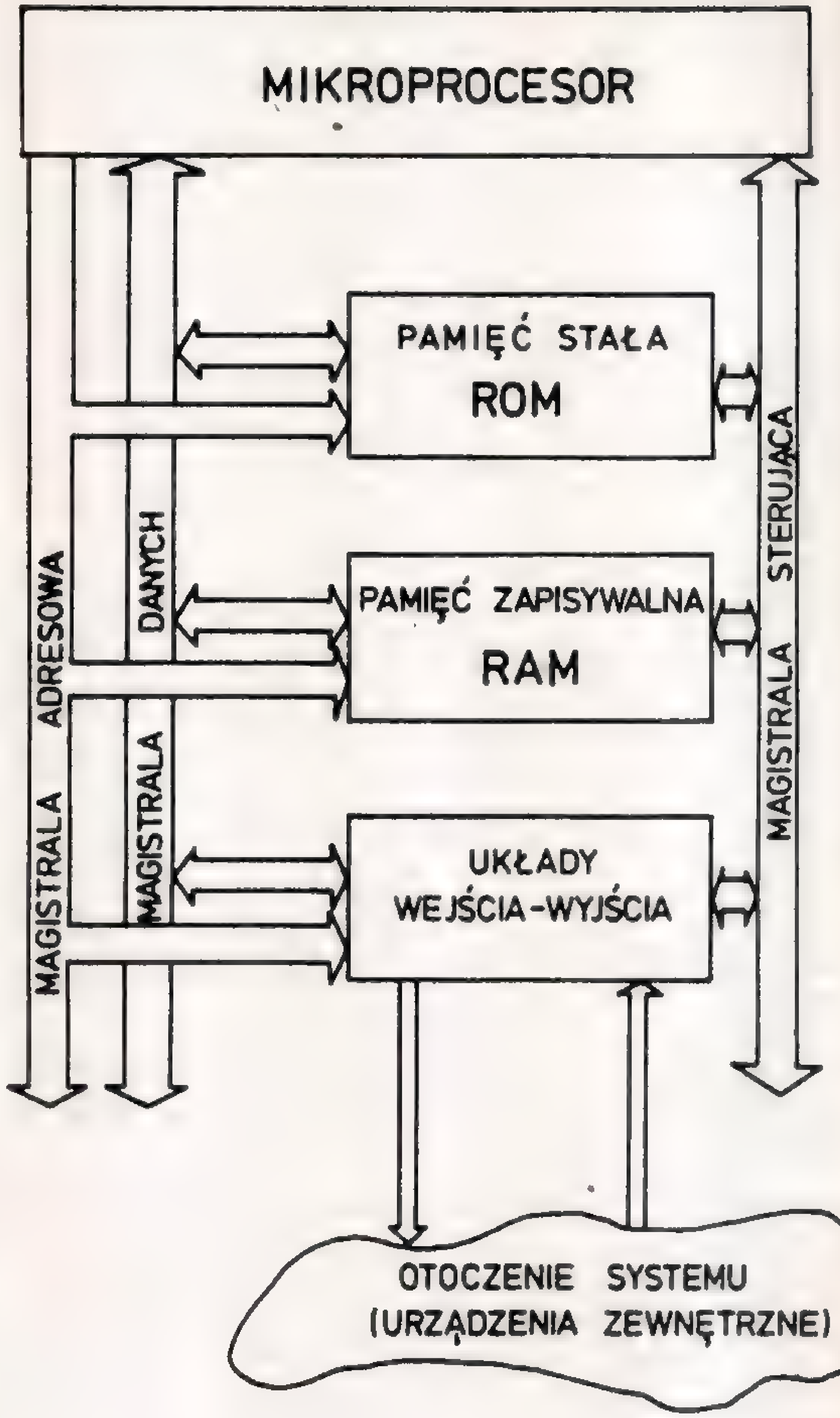
Po pobraniu rozkazu następuje faza wykonania rozkazu. **Układ sterowania CU** generuje sekwencję sygnałów sterujących, zapewniających wykonanie rozkazu.

Każdy cykl rozkazowy składa się z kilku **cykli maszynowych**. Cykl maszynowy składa się z kilku **taktów zegarowych**. Faza pobrania rozkazu posiada tyle cykli maszynowych ile słów zawiera rozkaz, natomiast faza wykonania rozkazu może zawierać różną liczbę cykli maszynowych.

Jednostka arytmetyczno-logiczna ALU, po wysterowaniu przez układ sterowania CU, wykonuje podstawowe działania arytmetyczne i logiczne. Są to zazwyczaj dodawanie arytmetyczne (binarne i dziesiętne), odejmowanie, porównywanie, przesuwanie słowa o jeden bit w lewo lub prawo, suma logiczna, iloczyn logiczny, różnica symetryczna, negacja słowa. Wyniki operacji zapamiętywane są w **rejestrze wskaźników SF**.

Z ALU funkcjonalnie związany jest specjalny rejestr roboczy **akumulator A**, przeznaczony do przechowywania jednego z argumentów (słowa) wykonywanej operacji dwuargumentowej i do przechowywania wyniku. Za pośrednictwem akumulatora dokonywane są najczęściej przesłania danych (słów) między ALU a pamięciami lub urządzeniami zewnętrznymi.

Rejestry robocze R_n służą do przechowywania danych i do określania stanu magistrali adresowej. Rejestry



mogą być łączone w pary tworząc, w przypadku mikroprocesorów 8-bitowych, rejestr 16-bitowy. Taki rejestr może służyć do określania adresu komórki pamięci, nosi on nazwę **licznika danych DC** (**Data Counter**).

Do adresowania wydzielonego obszaru pamięci RAM, przeznaczonego do przechowywania (i odczytywania) danych lub adresów na tzw. **stosie**, służy **wskaźnik stosu SP**. Stos umożliwia zawieszenie wykonywania programu i przejście do **podprogramu**.

Do przesyłania danych między podzespołami mikroprocesora służy dwukierunkowa **wewnętrzna magistrala danych**. Jednokierunkowa **wewnętrzna magistrala adresowa** pozwala na adresowanie zawartościami rejestrów pamięci lub urządzeń zewnętrznych.

W następnym numerze „IKS-a" omówimy logikę i arytmetykę mikroprocesora, sposoby adresowania, przerwania i rozpoczniemy szczegółowe omawianie mikroprocesora ośmiobitowego.

J.W.

PROGRAM SHARP 33

Jesteśmy zdecydowanymi przeciwnikami takich gier, w których OSZUST, posługujący się ZNACZONYMI kartami, bez skrupułów OGRABIA naiwnego.

Ponieważ jednak kuszenie szczęścia zawsze korciło, zatem proponujemy Wam grę UCZCIWĄ, w której przeciwnikiem (niezainteresowanym materialnie) będzie Wasz komputer.

Prezentowana gra jest PROGRAMOWO NIERÓWNA — uprzywilejowany jest gracz: MOŻE on ZMIENIĆ jedną, dwie, a nawet pięć kart. Komputer (programowo) NIE WYMIENIA: „gra" tym, co wylosuje.

Zatem po nabraniu wprawy, gdy rozgromimy komputer, dajmy mu szansę: OGRANICZAJMY ZMIANY. Gdy z nich zrezygnujemy, gra będzie WYRÓWNANA.

Program wywołuje RUN „A.

Karta wymieniona jest oznakowana gwiazdeczką.

Życzymy dobrej zabawy!

Budowa programu

— „wydawaniem kart" zajmuje się generator liczb losowych. Losuje „honor" RND 6, potem „kolor" RND 4,

— wydane karty są rejestrowane w dwuwymiarowej pamięci. To UNIEMOŻLIWIA POWTÓRZENIA. Macierz ma wymiar 6 x 4, zatem operujemy 24 kartami. „Honory" pamięta jednokolumnowa macierz X$ zaś „kolory" macierz Y$.

Janusz MILLER

SHARP PC-1500

KOMPUTER...
...POKERZYSTA

```
100:"A":CLEAR :DIM
    T(5,3):DIM X$(
    5):DIM Y$(3):
    DIM M$(5):DIM
    P$(5):WAIT :K=
    RND 4:W=RND 8
110:X$(0)="A":X$(1
    )="K":X$(2)="D
    ":X$(3)="W":X$
    (4)="10":X$(5)
    ="9"
120:Y$(0)="KIER":Y
    $(1)="KARO":Y$
    (2)="TREFL":Y$
    (3)="PIK"
130:PRINT "MOJE KA
    RTY"
140:FOR L=1TO 5
150:GOSUB "R":W$=
    STR$ L+". "
160:M$(L)=Z$:PRINT
    W$;Z$
170:NEXT L
180:INPUT "ILE ZMI
    ENIC ? ";Z:Z=
    INT Z:Z=ABS Z:
    IF Z=0GOTO 340
190:IF Z<0OR Z>5
    THEN BEEP 10,1
    00,100:GOTO 18
    0
200:IF Z=1THEN
    INPUT "ZMIEN W
    IERSZ ";A:GOTO
    290
210:IF Z=2THEN
    INPUT "ZMIEN D
    WA WIERSZE ",A
    ,B:GOTO 280
220:IF Z=3THEN
    INPUT "ZMIEN T
    RZY WIERSZE ",
    A,B,C:GOTO 270
230:IF Z=4THEN
    INPUT "ZMIEN C
    ZTERY WIERSZE
    ",A,B,C,D:GOTO
    260
240:A=1:B=2:C=3:D=
    4:E=5
250:GOSUB "R":M$(E
    )=Z$+"*"
260:GOSUB "R":M$(D
    )=Z$+"*"
270:GOSUB "R":M$(C
    )=Z$+"*"
280:GOSUB "R":M$(B
    )=Z$+"*"
290:GOSUB "R":M$(A
    )=Z$+"*"
300:CLS :PRINT "ZM
    IENIONE"
310:FOR L=1TO 5
320:W$=STR$ L+". "
    :PRINT W$;M$(L
    )
330:NEXT L
340:PRINT " KOMPUT
    ER "
350:FOR L=1TO 5
360:GOSUB "R":W$=
    STR$ L+". "
370:P$(L)=Z$:PRINT
    W$;Z$
380:NEXT L
390:INPUT "GRASZ D
    ALEJ? T=1 N=0
    ";D
400:IF D=1GOTO 100
410:END
420:"R":W=RND 6:W=
    W-1
430:K=RND 4:K=K-1
440:IF T(W,K)=1
    GOTO 420
450:T(W,K)=1:Z$=X$
    (W)+" "+Y$(K):
    RETURN
```

```
 PRZYKLAD
MOJE KARTY
1. A KIER
2. A TREFL
3. 10 KARO
4. 9 KIER
5. 9 KARO
ZMIENIONE
1. A KIER
2. A TREFL
3. K KIER*
4. 9 KIER
5. 9 KARO
 KOMPUTER
1. A KARO
2. 9 TREFL
3. K KARO
4. 10 KIER
5. K TREFL
```

PROGRAM 34

AMSTRAD

Pełne możliwości programu przedstawia tzw. menu. Program „Książka telefoniczna" umożliwia wprowadzenie i zachowanie na dyskietce (w zbiorze „DANE") dowolnej liczby abonentów. Prezentowany program może zachować informacje o 100 abonentach, ale chcąc zwiększyć tę liczbę do np. 1000 należy w wierszu o nr. 20 zmienić n = 100 na n = 1000. W trakcie korzystania z programu, w celu wyszukania konkretnego abonenta lub przejrzenia całego zbioru abonentów należy najpierw odzyskać tę informację z dyskietki (czynność 5). Zobrazowanie informacji w tzw. oknach oraz sygnalizacja dźwiękowa wyraźnie poprawia komfort korzystania z programu.

UWAGA:

Chcąc wprowadzić do istniejącego zbioru na dyskietce nowych abonentów, należy najpierw odzyskać cały zbiór abonentów (czynność 5 z „menu").

```
10 REM KSIAZKA TELEFONICZNA
20 n=100
30 DIM imie$(n)
40 DIM tel$(n)
50 GOSUB 1260
60 MODE 1
70 WINDOW #1,7,35,8,16
80 WINDOW #2,3,20,22,20
90 PRINT #1,"1.- wprowadzanie danych"
100 PRINT #1
110 PRINT #1,"2.- przegladanie danych"
120 PRINT #1
130 PRINT #1,"3.- wyszukiwanie danych"
140 PRINT #1
150 PRINT #1,"4.- zachowywanie danych"
160 PRINT #1
170 PRINT #1,"5.- odzyskiwanie danych"
180 INK 1,24:PEN #2,0,0;PAPER #2,13
190 LOCATE #2,2,25
200 PRINT #2, "WYBIERZ CZYNNOSC"
210 a$=INKEY$
220 IF a$="" THEN 210
230 IF a$="1" THEN 340
240 IF a$="2" THEN 550
250 IF a$="3" THEN 830
260 IF a$="4" THEN 1020
270 IF a$="5" THEN 1100
280 GOTO 210
290 CLS #0
300 LOCATE 5,12
310 PRINT "POWROT DO MENU - [DOWOLNY KLAWISZ]"
320 IF INKEY$="" THEN 320
330 GOTO 60
340 CLS:INK 1,26:k=0
350 FOR x=1 TO n
360 IF LEN(imie$(x))>0 THEN 540
370 a=1:CLS #0
380 LOCATE 3,25
390 PRINT "KONIEC WPROWADZANIA - [RETURN]"
400 WINDOW #7,3,32,20,20
410 PAPER #7,13:PEN #7,0
420 PRINT #7," WPROWADZONO ";k;" ABONENTOW "
430 LOCATE 1,2
440 PRINT "imie i nazwisko"
450 INPUT imie$(x)
460 IF imie$(x)="" THEN 290
470 PRINT "telefon"
480 INPUT tel$(x)
490 k=k+1
500 ENV 1,=9,2000:ENT -1,6,3,1
510 SOUND 1,200,0,1,1
520 NEXT
530 GOTO 290
540 LOCATE 12,14:PRINT "prosze czekac":GOTO 520
550 CLS
560 WINDOW #3,3,29,4,22
570 WINDOW #4,33,39,4,22
580 PAPER #3,13:PAPER #4,13:PEN #3,0:PEN #4,0:CLS #3:CLS #4
590 PRINT #0," IMIE  I  NAZWISKO";SPC(13);"TELEFON"
600 IF imie$(1)="" THEN 780
610 z=1
620 IF imie$(z)="" THEN 750
630 FOR x=0 TO 18
640 PRINT #3,imie$(z+x)
650 SOUND 1,200*RND,12
660 PRINT #4,tel$(z+x)
670 SOUND 1,400*RND,12
680 IF imie$(z+x)="" THEN 750
690 NEXT x
700 LOCATE 1,25
710 PRINT #0,"NACISNIJ DOWOLNY KLAWISZ"
720 IF INKEY$="" THEN 720
730 CLS #3:CLS #4
740 z=z+19:GOTO 620
750 LOCATE 1,25
760 PRINT "KONIEC PRZEGLADANIA - [DOWOLNY KLAWISZ]
770 GOTO 320
780 LOCATE #3,6,10
790 PRINT #3," BRAK ABONENTOW"
800 FOR a=200 TO 250 STEP 10
810 SOUND 1,a,12:NEXT
820 GOTO 750
830 CLS
840 WINDOW #5,5,30,4,4
850 PAPER #5,13:PEN #5,0:CLS #5
860 PRINT #0, "PODAJ IMIE I NAZWISKO"
```

```
870 INPUT #5,a$
880 FOR x=1 TO n
890 IF INSTR(imie$(x),a$)=0 THEN 950
900 LOCATE 7,6
910 PRINT imie$(x);SPC(5);tel$(x)
920 ENV 3,=9,9000
930 SOUND 1,300,0,13,3
940 IF LEN(imie$(x))>0 THEN 300
950 NEXT
960 WINDOW #7,5,38,6,6
970 PAPER #7,13:PEN #7,0
980 PRINT #7," BRAK ABONENTA W ZBIORZE  "
990 ENV 3,=9,9000
1000 SOUND 1,100,0,13,3
1010 GOTO 300
1020 OPENOUT "dane"
1030 FOR x=1 TO n
1040 WRITE #9,imie$(x),tel$(x)
1050 NEXT
1060 CLOSEOUT
1070 CLS
1080 PRINT "DANE O ABONENTACH - ZACHOWANO"
1090 GOTO 300
1100 OPENIN "dane"
1110 FOR x=1 TO n
1120 INPUT #9,imie$(x),tel$(x)
1130 NEXT
1140 CLOSEIN
1150 CLS
1160 PRINT "DANE ODZYSKANO"
1170 k=0
1180 FOR x=1 TO n
1190 IF imie$(x)="" THEN 1220
1200 k=k+1
1210 NEXT
1220 j=n-k
1230 LOCATE 1,4
1240 PRINT "MOZESZ JESZCZE ZACHOWAC ";j;" ABONENTOW"
1250 GOTO 300
1260 t1$="T E L E F O N Y"
1270 T2$=CHR$(164)+CHR$(32)+CHR$(32)+" CJS "
1280 MODE 0
1290 ENV 1,=9,2000:ENT -1,6,3,1
1300 ENV 2,127,0,0,127,0,0,127,0,0,127,0,0:ENV 3,=9,9000
1310 FOR c=1 TO LEN(t1$)
1320 LOCATE 2+c,10
1330 FOR n2=LEN(t1$) TO c STEP -1
1340 PRINT MID$(t1$,n2,1)
1350 LOCATE 2+c,10
1360 SOUND 135,20*n2,5,12,2,1
1370 NEXT n2,c
1380 SOUND 135,100,0,13,3,1,20
1390 LOCATE 5,15:PRINT t2$
1400 FOR c=1 TO 2000:NEXT
1410 CLS
1420 RETURN
```

PROGRAM 35

Program KONSTRUKTOR LICZB jest grą edukacyjną, przeznaczoną przede wszystkim dla dzieci w wieku szkolnym. Może ona zainteresować również inne osoby. Celem gry jest rozwijanie umiejętności posługiwania się podstawowymi działaniami arytmetycznymi i szacowania wyników tych działań. Program KONSTRUKTOR LICZB opracowany został na mikrokomputer **Commodore 64, Plus/4 i Commodore 16**. W wypadku mikrokomputerów Commodore 16 i Plus/4 zamiast wiersza 10 wprowadzić należy wiersz

```
10 COLOR0,1,1:COLOR4,1,1:Q=205:
K=202:B=239:KL=2648:PT=3670:A¤=
CHR¤(159)
```

W grze może brać udział od jednej do trzech osób. Każdy z graczy jest konstruktorem liczb, dysponującym narzędziami i materiałami. Narzędziami są cztery działania arytmetyczne: +, —, *, /. Materiałami są cyfry generowane przez mikrokomputer. Liczba cyfr jest zależna od tego, jakimi konstruktorami są gracze. Zmienia się od trzech cyfr (konstruktor zaawansowany) do pięciu cyfr (konstruktor początkujący). Dla każdego z graczy określany jest projekt liczby z zakresu od 100 o 999. Zadaniem gracza jest skonstruowanie za pomocą dostępnych narzędzi i materiałów własnej liczby, jak najbliższej zadanemu projektowi, w jak najkrótszym czasie. Liczba operacji, która w końcowym efekcie może być skonstruowana jest ograniczona do siedmiu. Czas tworzenia liczby zmienia się od jednej do trzech minut, zależnie od poziomu reprezentowanego przez konstruktorów.

Gra trwa pięć rund. W każdej rundzie gracze mają do wykonania po jednym projekcie. Liczba uzyskanych punktów jest wypadkową czasu wykonywania konstrukcji liczby oraz dokładności, z jaką nasza liczba odpowiada projektowi.

Grę rozpoczynamy od podania liczby graczy oraz poziomu reprezentowanego przez konstruktorów (początkujący, przeciętny, zaawansowany). Określony gracz (numer jest wyświetlany na ekranie) po pojawieniu się komunikatu GOTÓW! przyciśnięciem dowolnego klawisza powoduje rozpoczęcie gry. Pojawiają się wówczas na ekranie: aktualnie dostępne materiały, projekt liczby a zegar zaczyna odliczać czas. Wymieniony gracz zaczyna konstrukcję liczby. Konstruowana liczba nie może wyjść poza zakres 0 — 9999. Używane materiały i narzędzia są w tej samej kolejności wyświetlane na ekranie, tworząc ciąg operacji arytmetycznych. Wyświetlany jest wynik bieżącego ciągu operacji arytmetycznych. Eliminację dowolnego elementu ciągu (materiału lub narzędzia) uzyskujemy po ustawieniu kursora na tym elemencie i przyciśnięciu klawisza DELETE. Przesuwanie kursora wewnątrz ciągu wykonujemy klawiszami → i ←. Zakończenie gry następuje po upływie określonego czasu lub po przyciśnięciu przez gracza klawisza RETURN, gdy konstrukcję liczby chce zakończyć przed czasem. Następuje wówczas podliczenie punktów uzyskanych przez gracza i do gry zapraszany jest kolejny gracz.

Uwaga: w tekście programu zamiast znaku $ użyty jest znak ¤.

dokończenie na str. 12

dokończenie ze str. 11

```
0 REM ***KONSTRUKTOR LICZB***
10 POKE53280,0:POKE53281,0:Q=214:K=211:B=198:KL=55896:PT=1622:A$=CHR$(150)
20 B$=CHR$(156):C$=CHR$(5):D$=CHR$(98):E$=CHR$(99):F$=CHR$(173)
30 G$=CHR$(174):H$=CHR$(176):I$=CHR$(189):J$=CHR$(147):K$=CHR$(32)
40 L$=CHR$(157):S$="    ":G=1:RN=1
50 PRINTJ$SPC(11)A$H$;:FORI=1TO17:PRINTE$;:NEXT:PRINTG$
60 PRINTSPC(11)D$C$"KONSTRUKTOR LICZB"A$D$
70 PRINTSPC(11)F$;:FORI=1TO17:PRINTE$;:NEXT:PRINTI$
80 PRINT:PRINT:PRINTB$"NARZEDZIA:"C$" +  -  *  /"SPC(12)A$"CZAS"
90 PRINT:PRINTB$"MATERIALY: "C$"0  0  0  0  0"SPC(9)"0:00"
100 PRINT:PRINTSPC(9)B$H$E$E$E$G$
110 PRINT"PROJEKT: "D$C$"000"B$D$
120 PRINTSPC(9)F$E$E$E$I$
130 PRINT:PRINT:PRINTSPC(32)A$H$E$E$E$E$G$
140 PRINTC$"0 ";:FORI=1TO7:PRINT"+0 ";:NEXT:PRINTCHR$(146)"= "A$D$C$"0000"A$D$
150 FORI=1TO15:PRINTCHR$(163)K$;:NEXT:PRINTSPC(2)F$E$E$E$E$E$I$
160 POKEQ,18:PRINT:PRINT"LICZBA GRACZY? (1-3)"
170 GETGR$: IFGR$<"1"ORGR$>"3" THEN170
180 FORI=1TOVAL(GR$):G(I)=0:NEXT:POKEQ,18:PRINT:PRINT"JESTES KONSTRUKTOREM?"
190 PRINT:PRINT"1 - POCZATKUJACYM":PRINT"2 - PRZECIETNYM":PRINT"3 - ZAAWANSOWANYM"
200 GETKR$: IF KR$<"1"ORKR$>"3"THEN200
210 M=3-(KR$<"3")-(KR$<"2"):KR=4-VAL(KR$)
220 POKEQ,17:PRINT:PRINTSPC(26)B$"PUNKTACJA":PRINT:GR=VAL(GR$)
230 FORI=1TOGR:PRINTSPC(24)B$"GRACZ"I;L$": "C$"0000":P(I)=0:NEXT:GOSUB560
240 REM KOLEJNA GRA
250 POKEQ,17:PRINT:PRINTA$"RUNDA"C$RN;S$A$"GRACZ"C$G
260 FORI=1TOM:M(I)=0:NEXT:FORI=1TOM
270 R=INT(RND(0)*9)+1:J=1
280 IFR=M(J)THEN270
290 J=J+1:IFJ<=MTHEN280
300 M(I)=R:NEXT:FORI=M105:M(I)=M(M):NEXT
310 W=INT(RND(0)*900)+100:R$="000000":N=1:O=1:L=0:F=0
320 POKEQ,20:PRINT:PRINTSPC(7)C$"GOTOW!":FORI=0TO29:POKEKL+I,7:NEXT
325 WAITB,1:GETZ$:POKEQ,20:PRINT:PRINTSPC(5)S$S$
330 POKEQ,6:PRINT:PRINTSPC(10);:FORI=1TOM:PRINTM(I);:NEXT
340 PRINTS$S$:PRINT:PRINT:PRINTSPC(9)W:POKEQ,9:PRINT:PRINTSPC(9)B$D$
350 POKEQ,14:PRINT:FORI=1TO7:PRINTS$;:NEXT:PRINTK$SPC(4)S$C$:TI$="000000"
355 POKEQ,6:PRINT:PRINTSPC(32)KR;L$":00"
360 AZ=PT+N*2:ZN=PEEK(AZ):POKEAZ,ZN+128:IFNOT((N=15)AND(O<0))THENO=-O
370 GETZ$:POKEB,0:IFVAL(MID$(TI$,4,1))>=KRTHEN610
380 T$=TI$:POKEQ,6:PRINT:IFT$=R$THEN410
390 R$=T$:POKEK,33:PRINTCHR$(47+KR-VAL(MID$(T$,4,1)))":";
400 PRINTCHR$(53-VAL(MID$(T$,5,1)))CHR$(57-VAL(MID$(T$,6,1)))
410 IFZ$=""THEN370
420 Z=ASC(Z$):C=VAL(Z$):IFZ=13 THEN610
430 P=(N+1)/2:IF(Z=42ORZ=43ORZ=45ORZ=47)ANDO>0ANDN<16THENO(P)=Z:GOTO480
440 IF(C=M(1)ORC=M(2)ORC=M(3)ORC=M(4)ORC=M(5))ANDO<0ANDN<16THENC(P)=C:GOTO480
450 IFZ=20ANDN=LANDN>0 THEN POKEAZ,32:L=L-1:O=-O:C(P)=-C(P)*(N>1):GOTO490
460 IFF=1ORNOT((Z=29 AND(N<L+1)ANDN<15)OR(Z=157ANDN>1))THEN370
470 IF(Z=29 AND(N<L+1)ANDN<15)OR(Z=157ANDN>1)THENPOKEAZ,ZN:N=N-(Z=29OR1):GOTO360
480 PRINTC$:POKEAZ,Z:L=L-(N>L):N=N-(N<15)
490 POKEQ,14:PRINT:PRINTSPC(33)S$:FORI=1TO8:W(I)=C(I):NEXT
500 FORI=1TO7:W(I+1)=(O(I)=45)*W(I+1)-(O(I)<>45)*W(I+1):NEXT:F=0:W%=0:J=L:I=0
510 I=I+1:IFNOT(O(I)=42ORO(I)=47)ORI>=J/2THEN530
520 W(I+1)=-(W(I)*W(I+1))*(O(I)=42)-(W(I)/W(I+1))*(O(I)=47):GOTO510
530 H=W%:IFH+W(I)>9999ORH+W(I)<0THEN570
540 W%=W%+W(I):IFI<J/2THEN510
550 POKEQ,14:PRINT:PRINTSPC(32)W%:F=0:POKEQ,14:PRINT:PRINTSPC(32)A$D$C$:GOTO360
```

```
560 POKEQ,18:PRINT:FORI=0TO4:FORJ=
0TO21:PRINTK$;:NEXT:PRINT:NEXT:RET
URN
570 POKEQ,19:PRINT:PRINT"PRZEKROCZ
ONO ZAKRES":PRINT"LICZB! (0-9999)"
580 POKEAZ,32:N=N+(N<15):L=L+(N=LA
NDL<>15):O=-O
590 POKEQ,14:PRINT:PRINTSPC(33)S$:
WAITB,1
600 GOSUB560:GOTO360
610 POKEAZ,ZN:E=W-W%:E=E+2*E*(E<0)
:E=-E*(E<=100)-100*(E>100)
620 TR=KR*60-(60*VAL(MID$(T$,4,1))
+VAL(RIGHT$(T$,2)))
630 U=INT((100-E)*VAL(KR$)*TR/10):
G(G)=G(G)+U:POKEQ,18+G:PRINT
635 PRINTSPC(33)S$CHR$(145):PRINTS
PC(32)G(G)
640 G=G+1:IFG>VAL(GR$)THENG=1:RN=R
N+1
650 IFRN<6THEN240
660 POKEQ,20:PRINT:PRINT"KOLEJNA G
RA? (T/N)"
670 GETZ$:IFZ$<>"T"ANDZ$<>"N"THEN
670
680 PRINTJ$:IFZ$="T"THENRUN
```

Zaczarowana skrzynka



Racuchy pałaszowały jeden za drugim, a przy tym wbrew upomnieniom pani wychowawczyni, bardzo hałasowały i gestykulowały. Jak się okazało, tyle emocji wśród przedszkolaków wzbudziła „zaczarowana skrzynka" czyli komputer Amstrad 6128.

— Jeszcze na stałe nie posiadamy komputera — powiedziała Marianna Błaszczak, dyrektorka w warszawskim przedszkolu, mieszczącym się przy ulicy Koszykowej, ale nosimy się z zamiarem jego zakupu. Może nie Amstrad, bo co tu ukrywać, nie dysponujemy aż tak dużymi finansami, ale może Commodore lub Spectrum.

Rodzice przyklasnęli inicjatywie dyrekcji przedszkola. „Niech nasze dzieci nie zostają w tyle za nowinkami techniki — powiadają. Komputery w wielu krajach traktowane są jako narzędzie pracy, a u nas wciąż wzbudzają wiele emocji. A wiadomo czym skorupka za młodu nasiąknie..."

Na razie sześciolatki korzystają z uprzejmości zaprzyjaźnionych placówek i... zamożniejszych rodziców.

W ciągu roku grupa zerowa musi poznać 22 litery wielkie i małe. Nauczyć się czytać proste zdania i krótkie teksty. Liczyć od 0 do 10, poznać cyfry, znaki i podstawowe pojęcia matematyczne. Muszą umieć układać wzory z poznanych znaków. W tym wszystkim mógłby pomóc im właśnie komputer - chociaż...

*

Pani wychowawczyni dużo wcześniej zapowiedziała zajęcia z komputerem. Kiedy wreszcie będzie — niecierpliwiły się przedszkolaki. Ja tam już widziałem — chwalił się Krzyś. I co, co — dopytywali się rówieśnicy.

— Wujek grał z komputerem w karty a dla mnie włączył wyścigi. Mówię wam bomba. Emocje doszły zenitu. Nareszcie po śniadaniu spotkanie z „wielką niewiadomą".

Amstrad „zawitał" do świetlicy. Nim pracownik Wojskowego Instytutu Informatyki zdążył cokolwiek omówić, dzieci zarzuciły go mnóstwem pytań. A po co są te klawisze! A co to za ekran? A te blaszki do czego służą? Każde chciało dotknąć skrzynkę, wydającą dziwne dźwięki, nacisnąć chociaż jeden klawisz.

Proszę się nie przepychać — karciła przedszkolaków pani wychowawczyni, Elżbieta Kąca. Wszyscy zobaczycie. Za klawiaturą zasiada Kamil. Dla niego to nie nowina. Komputer ma w domu. Umiejętnie zdobywa punkty, zbijając „Uciekającego Franciszka". Ja bym chciał mieć takie cacko w domu — wzdycha Paweł. Dziewczynki bardziej sceptycznie się wypowiadały. Może jednak wolą Zuzię, lalę niedużą...

— Według mnie — powiedział pracownik Wojskowego Instytutu Informatyki, mjr Romuald Głąb — nie powinniśmy wpadać w przesadną euforię. Jak do tej pory, na rynku nie pojawiły się programy opracowane dla potrzeb szkolnictwa, a zwłaszcza grup zerowych. Programy z prawdziwego zdarzenia, które po konsultacji nauczycieli z zawodowymi informatykami mogłyby pomóc, a nie zaszkodzić w nauczaniu początkowym. Chałupnicze wymysły, to nie to. Zresztą jeden komputer na 30 dzieci, to przysłowiowa kropla w morzu — podsumował.

No cóż, od czegoś należy zacząć. Poprzednia grupa zerowa uwielbiała zajęcia z komputerem (wypożyczonym). Na ekranie ukazywały się proste działania i jeśli je dobrze wykonały, premiowane były punktami. W przeciwnym wypadku rozlegał się sygnał — popraw się. Kilku sześciolatków, w domu, uczyło się języka LOGO, w którym się programuje, tworzy za pomocą prostych instrukcji — obrazy.

— Wszystkie zaczęłyśmy się dokształcać — powiedziała pani Błaszczak. Przecież nie możemy zostać w tyle... za naszymi wychowankami, z których w przyszłości może wyrosną informatycy.

*

Jak na razie dzieci wychodzą na przywitanie dnia jutrzejszego. Uczą się myślenia twórczego, by w 2000 roku, już jako dwudziestolatki nie miały poczucia straconego czasu.

ANNA TOMCZYKOWSKA



Foto: Jan Zelman

Zmienne systemowe ROM SPECTRUM

WSKAŹNIKI RAM

PROG 23635/6
VARS 23627/8
E—LINE23641/2
WORKSP 23649/50
STKBOT 23651/2
STKEND 23653/4
ERR—SP 23613/4
RAMTOP 23730/1
UDG 23675/6
P—RAMT 23732/3

Kiedy piszemy program w BASIC-u, pamięć RAM jest na bieżąco porządkowana. Komputer musi wiedzieć wszystko, a będzie to możliwe wtedy, gdy nasze posunięcia będą kontrolowane. Z tego powodu RAM oznaczona jest wieloma wskaźnikami. Wszystkie zajmują w obszarze zmiennych po 2 bajty, gdyż zawierają adresy.

P—RAMT przechowuje wielkość adresowanej pamięci, a ściślej bajt o najwyższym adresie. Dla Spectrum 48K jest to adres 65535. Jeśli coś nie jest w porządku z naszym przyjacielem, lepiej sprawdzić zawartość tej zmiennej. Uzyskamy informację, czy w porządku są moduły RAM.

Program w BASIC-u lokuje się od adresu przechowywanego przez PROG. Zwykle jest to 23755, jednak po przyłączeniu INTERFACE 1 wynosi on 23812, co wynika z faktu pojawienia się obszaru mapy microdrive'u.

Pozostałe wskaźniki wyznaczają początek następujących obszarów:

VARS	—	zmienne BASIC-u
E—LINE	—	bufor edycyjny, gdzie redaguje się linie BASIC.
WORKSP	—	przestrzeń robocza, gdzie trzymane są dane wczytane w czasie egzekucji komendy INPUT oraz tworzony jest nagłówek po wczytaniu instrukcji SAVE...
STKBOT	—	stos kalkulatora, na którym zachodzą wszystkie obliczenia arytmetyczne,
STKEND	—	wolna pamięć zmniejszająca się w miarę dodawania linii do programu.

Granicą ekspansji programu BASIC jest adres trzymany przez RAMTOP, zmniejszony o 80. Przy przekroczeniu tej wielkości otrzymamy komunikaty *Out of memory* lub *No room for line*. W trakcie powstawania programu użyteczne mogą być następujące informacje:

```
9000 CLS: PRINT '''
9010 PRINT "dlug. programu",FN a(23
627,23635)
9020 PRINT "dlug. zmiennych",FN a(2
3641,23627)-1
9030 PRINT "wolna pamiec",65536- US
R 7962
9040 PRINT "ramtop", PEEK 23730+256
* PEEK 23731
9050 DEF FN a(h,l)=PEEK h- PEEK l+2
56*(PEEK(h+1)-PEEK (l+1))
```

Od RAMTOP w dół pamięci odkłada się stos GOSUB'ów, na którym przechowywane są 3-bajtowe (numer linii i wyrażenia) informacje o powrocie z podprogramu po komendzie RETURN. Oto przykład wymuszania powrotu;

```
10 GOSUB 100: REM stos gosub'ow zaw
iera linie 10 i wyrazenie 2 jako po
wrot
20 PRINT "liczba dodatnia": STOP
100 INPUT "podaj dowolna liczbe";zn
ak
110 IF SGN znak=1 THEN RETURN
120 LET linia=200: LET wyr=1
130 LET stos=2+ PEEK 23613+256*PEEK
 23614
140 POKE stos,linia-256* INT (linia
/256)
150 POKE stos+1, INT (linia/256)
160 POKE stos+2,wyr
170 RETURN
200 PRINT "liczba nie jest dodatnia
": STOP
```

Poniżej stosu podprogramów odkłada się stos maszynowy, a rozdziela je adres obsługi wydruku raportów błędów, który zawsze jest wskazywany przez zmienną ERR—SP.

RAMTOP, jako jedyną zmienną systemową, można ustawiać komendą CLEAR adres. Początkowo wynosi on 65367. Zaraz za nim (65368) rozpoczyna się obszar grafik użytkownika, który wskazuje UDG. Zaletą tego obszaru (168 bajtów) jest możliwość umieszczenia go w innym, rozsądnym miejscu RAM, choćby w buforze drukarki, gdy nie jest przyłączona.

Jeśli nie wystarcza 21 grafik użytkownika, można ich liczbę powiększyć.

```
200 RESTORE 200+obszar*10
202 READ bajt1,bajt2
204 POKE 23675,bajt1
206 POKE 23676,bajt2
208 RETURN
210 DATA 8,254
220 DATA 176,254
230 DATA 88,255
```

Ten podprogram umożliwia korzystanie z 63 znaków graficznych, lecz na każdy z obszarów trzeba się przełączyć. Odbywa się to przez sekwencję np.:

LET obszar = 1 : GOSUB 200

Konieczne jest na początku programu głównego obniżenie RAMTOP przez CLEAR 65031.

Sposobów przechowywania procedur w kodzie maszynowym jest kilka. Najpopularniejsze z nich to bezpośrednie wczytywanie pliku z nośnika (taśmy, dyskietki) do określonego obszaru pamięci, trzymanie bajtów w instrukcjach DATA oraz przechowywanie ich w instrukcjach REM. Ten ostatni sposób jest używany najczęściej. Pierwszą linię programu zaopatrujemy komendą REM i tyloma znakami, ile bajtów zajmuje procedura. Następnie wczytujemy kolejne bajty, korzystając z podprogramu:

```
10 REM ilosc znakow rowna dlugosci kodu
20 INPUT "dlugosc kodu: ";d
30 IF d<1 THEN GOTO 20
40 LET prog= PEEK 23635+256* PEEK 23636
50 FOR a=1 TO d
60 INPUT bajt
70 IF bajt>255 OR bajt<0 THEN GOTO 60
80 POKE prog+4+a,bajt
90 NEXT a
100 PRINT "adres poczatkowy =";prog+5
```

PROGRAM 36

AMSTRAD

Problem jest prosty: rozwiązanie trójkąta o danych trzech bokach. Po wprowadzeniu danych obliczony jest obwód trójkąta, pole (ze wzoru Herona) oraz wszystkie kąty (z twierdzenia Carnota). Linie 360... 430 szkicują trójkąt proporcjonalnie do wymiarów ekranu. Wysokość prowadzona jest do najdłuższego boku. Objaśnienia:

1) Nie ma potrzeby deklarowania zmiennych rzeczywistych, gdyż Amstrad robi to automatycznie.
2) Kolory ustala się poprzez podwójne przyporządkowanie, co może stwarzać nieco kłopotów debiutantom. Najlepiej zacząć od „napełnienia" kałamarzy (INK) odpowiednimi kolorami tuszu (linie 50... 70).
   W trakcie pisania programu wylewamy tusz z kałamarza na tło (PAPER) lub maczamy w nim pióro (PEN). Jeśli definiujemy okna trzeba określić dla nich odpowiednie atrybuty (linie 80... 90).
3) W stosunku do standardu Spectrum uproszczono maksymalnie definiowanie grafik użytkownika (linie 100...110).
4) Wprowadzenie funkcji użytkownika należy umieszczać przed korzystaniem z nich (w Spectrum można je umieszczać w dowolnym miejscu programu).
5) Czyszczenie okien najlepiej przeprowadzać od największego do najmniejszego (linia 170). Należy pamiętać, że CLS niszczy tryb obliczania funkcji trygonometrycznych w stopniach kątowych.
6) Deklaracja tablicy 11-wymiarowej (linia 150) tworzy pamięć dla 11 + 1 zmiennych uwzględniając indeks zerowy (linie 310...330).
7) Przed wprowadzeniem danych z klawiatury dobrze jest wyczyścić bufor, w którym Amstrad skrupulatnie odnotowuje każdy przyciśnięty klawisz (linia 190).
8) Pomysłowa instrukcja MASK pozwala na tworzenie różnych linii. Argumentem jest wartość 8 bitów, które będą logicznie mnożone przez każde kolejne 8 punktów linii. Wartość MASK obowiązuje od momentu określenia do zmiany wartości argumentu.
9) Amstrad nie ma kłopotów z zaokrągleniem liczby do kilku cyfr po przecinku, co realizuje wygodne ROUND (linie 460...470). Jednak zaokrąglenie części ułamkowej kąta nie poda minut kątowych, lecz wartość w systemie dziesiętnym. Przejście na minuty realizuje linia 520.
10) Niefortunny brak funkcji arcsin, arccos czy arcctg zmusza do odwiedzenia biblioteki (linia 130).

K. M.

```
10 ' ** trojkat 86
20 ' deklaracje funkcji i zmiennych
30 ' definicje grafik i kolorow
40 BORDER 6
50 INK 0,21
60 INK 1,0
70 INK 2,22
80 WINDOW #1,30,40,1,15
90 PAPER #1,3:PEN #1,2
100 SYMBOL 255,22,12,24,52,24,12,22,0
110 SYMBOL 254,6,9,9,6,0,0,0,0
120 DEF FN k(a,b,c)=-(a^2-b^2-c^2)/(2
*b*c)
130 DEF FN l(a)=90-ATN(a/SQR(1-a^2))
140 DEF FN p(o,a,b,c)=SQR(o/2*(o/2-a)
*(o/2-b)*(o/2-c))
150 DIM k(3)
160 ' wprowadzanie danych
170 CLS:CLS#1:DEG
180 MASK 255
190 CLEAR INPUT
200 INPUT #1,"bok 1 ";a
210 INPUT #1,"bok 2 ";b
220 INPUT #1,"bok 3 ";c
230 IF ABS(b-c)<a AND a<b+c THEN ELSE
 PRINT"trojkat nie istnieje":GOTO 540
240 ' obliczenia
250 k(1)=FN l(FN k(a,b,c))
260 k(2)=FN l(FN k(b,c,a))
270 k(3)=FN l(FN k(c,a,b))
280 ob=a+b+c
290 m=MAX (a,b,c)
300 n=MIN (a,b,c)
310 k(0)=FN l(FN k(ob-m-n,m,n))
320 x=n*COS(k(0))
330 y=n*SIN(k(0))
340 m1=300/m
350 ' konfiguracja trojkata
360 ORIGIN 20,20
370 PLOT 0,0
380 DRAW m*m1,0
390 DRAW x*m1,y*m1
400 DRAW 0,0
410 MOVE x*m1,y*m1
420 MASK 219
430 DRAW x*m1,0
440 ' wydruk wynikow
450 CLS #1
460 PRINT#1,"obwod=";ROUND(ob,2)
470 PRINT#1,"pole =";ROUND(FN p(ob,a,
b,c),2)
480 FOR a=1 TO 3
490 PRINT#1
500 PRINT#1,CHR$(255);
510 IF a=1 THEN PRINT#1,"(a,b)" ELSE
IF a=2 THEN PRINT#1,"(b,c)" ELSE PRIN
T#1,"(c,a)"
520 PRINT#1,FIX (k(a));CHR$(8);CHR$(2
54);ROUND ((k(a)-FIX(k(a)))*60);CHR$(
8);CHR$(39)
530 NEXT
540 PRINT"inne dane ?  T/N"
550 IF INKEY(51)=-1 THEN 550 ELSE 170
```

dokończenie ze str. 14

Na zakończenie metoda ochraniania programu w języku BASIC, polegająca na zapisie na nośnik sekwencji bajtów symulujących program w języku wewnętrznym.

```
9900 INPUT "nazwa zbioru ",LINE n$
9910 IF LEN n$>8 THEN IF USR 1602 THEN
9920 LET s= PEEK 23653+256* PEEK 23654
9930 SAVE n$ CODE 23552,s-23500
9940 RUN
```

Autostart programu następuje dzięki linii 9940. Proszę zwrócić uwagę, że zachowany zostaje cały obszar zmiennych systemowych, m.in. zmienna przechowująca numer ostatnio wykonanej linii.

W kolejnym numerze kilka sztuczek z grafiki.

Krzysztof MAMCARZ





KLUBY KOM
MŁODYCH MIST

...w których przygoda z komputerem jes
stopnia". Autorzy najlepszych rozwiązań i
łacu Kultury i Nauki.







PUTEROWE
RZÓW TECHNIKI

już „bliskim spotkaniem III
programów spotkali się w Pa-

Foto: Jan Zelman







LOGO

W poprzednim odcinku dowiedzieliśmy się trochę o zmiennych w LOGO. Zmienne te nie są jednak odpowiednikiem zmiennych w językach typu Basic. Reprezentują one bowiem nazwę, a nie wartość. Są bardziej etykietką przyczepioną do czegoś, a nie tym czymś. Zauważyliśmy, że przed zmienną mogą stać znaczki „lub". Znaczki te mówią, jak należy interpretować napis stojący za nimi. I tak, jeśli przed zmienną (słowem, napisem) jest umieszczony cudzysłów (uwaga! bez odstępu — spacji), to napis należy brać dosłownie, jak stałą — łańcuch (string). Jeśli natomiast znaczkiem tym jest dwukropek, wtedy należy w tym miejscu operować na wartości reprezentowanej przez daną nazwę (coś w rodzaju zmiennej Basic'owej). Jednak jak się dalej okaże, to porównanie jest niezbyt szczęśliwe.

Ogólnie rzecz biorąc, w LOGO preferuje się programowanie bez użycia zmiennych. Tylko w ostateczności sięgamy po to narzędzie. Fakt istnienia parametrów procedur i możliwości ich przekazywania, w zupełności wystarcza do większości problemów. Ponieważ LOGO jest językiem REKURENCYJNYM i PROCEDURALNYM, zmienne globalne są tylko czasami użyteczne. Zwróćmy też uwagę, że przed parametrami procedur jest dwukropek. Tutaj oznacza on, że dana nazwa reprezentuje jakąś wartość, lecz zupełnie niewiadomo jaką. Nie tylko ze względu na wartość np. liczbową, ale także na samą istotę tej wartości (np. czy to słowo, czy liczba).

Zmieńmy na kilkanaście zdań temat. W którymś z poprzednich odcinków zasygnalizowałem fakt pojawienia się na rynku kasety z polskim LOGO wersja 2.2. Wersja ta jest w zupełności zgodna z PTI LOGO, a posiada dodatkowo możliwość „dogrania" efektownej procedury zamalowującej rysunki. Słów kilka o sposobie korzystania z tej procedury. Po pierwsze jak ją podłączyć do LOGO. A więc, po załadowaniu LOGO (Polskie LOGO... itd.) zatrzymujemy taśmę. Następnie piszemy:

```
? zajmij 2000
```

co powoduje zarezerwowanie 2000 bajtów na tę właśnie procedurę. Z tych 2000 bajtów ok. 400 przeznaczonych jest na właściwą procedurę, napisaną w języku maszynowym, a ok. 1600 na lokalny tzw. STOS wykorzystywany przy malowaniu. Dalej piszemy:

```
? ładujbin zam 63025
```

i „puszczamy" taśmę dalej. Liczba 63025 oznacza, że pod ten adres pamięci ma wgrać się dodatkowa procedura. LOGO powinno zasygnalizować odnalezieniem na taśmie zbioru „zam" i po chwili znów wesoło migać kursorem. Od tej chwili działanie słowa „zamaluj" („zam") jest inne niż opisywane wcześniej. Kontury, wewnątrz których znajduje się żółw, są dokładnie wypełniane kolorem żółwia, bez względu na ich kształt (mogą to być figury „mocno wklęsłe"). Kontur jest „obrysowywany" po obwodzie i każde zakamarki są oddzielnie zamalowywane. Jednak działanie tego kolorowania jest znacznie wolniejsze niż poprzednio (skuteczność to pewien narzut czasowy). Ale też mamy tutaj pewne usprawnienie, bo gdy się pomylimy, możemy kolorowanie przerwać. Wciskamy po prostu klawisz BREAK i kończymy „na siłę" działanie procedury. W przypadku „zwykłego zamaluj" (PTI LOGO lub 2.2. bez dogrywania), kolorowania nie można przerwać. Spróbujcie, jeśli macie kasetę z tą wersją, pobawić się w wypełnianie kolorem rysunków, a obserwacja drogi, jaką przebiega malująca kreska, na pewno będzie ciekawa.

W poprzednim odcinku poznaliśmy nader ważną konstrukcję języka, a mianowicie instrukcję — operację „jeśli". Umożliwia ona wykonywanie „inteligentnych" obliczeń, gdyż pozwala wybrać jakąś ewentualność. Co to byłby za program, który liczy tylko jedno? I tak, instrukcja „jeśli" sprawdza, czy stojący na początku warunek jest prawdą i jeśli tak, wykonuje jedną sekwencję (LISTE) komend, a w przeciwnym przypadku drugą, która zresztą może być pusta (może jej w ogóle nie być), weźmy banalny przykład:

```
? oto pisz—moduł : par
> jeśli : par < 0 [pisz — :par] [pisz : par]
> już
PISZ MODUŁ zdefiniowane
```

Spróbujmy wywołać tę procedurę dla różnych liczb. Widzimy, że dzięki „jeśli" możemy sprawdzić, czy dana liczba jest ujemna, i w zależności od tego wypisać odpowiednią wartość. Przypominam, że lista instrukcji do wykonania MUSI być w nawiasach KWADRATOWYCH. Na marginesie jeszcze jedna uwaga. Listy, bądź jedna lista, do wykonania, musi być zawarta w jednej linii. Linia ta ma znaczenie linii logicznej, co nie znaczy, że musi być w jednym wierszu na ekranie. Linia logiczna, to linia kończąca się klawiszem ENTER. Jedna linia logiczna może być w kilku „ekranowych" wierszach. Kontynuacja linii w następnym wierszu, sygnalizowana jest wykrzyknikiem w ostatniej kolumnie. Podobna zasada obowiązuje listę instrukcji w komendzie „powtórz".

Proponuję teraz małe ćwiczenie na zrozumienie sensu parametru. Zdefiniujmy proste procedury:

```
? oto w ramce: co
> szlaczek
> pisz : co
> szlaczek
> już
W ramce zdefiniowane
> oto szlaczek
> pisz „'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!
> już
SZLACZEK zdefiniowane
```

Zauważmy najpierw, że szlaczek to napisanie jednego SŁOWA — poprzedzony CUDZYSŁOWEM napis „'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!" Spróbujmy teraz wywołać procedurę „w ramce" z różnymi parametrami. Najpierw:

```
? w ramce 10
'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!
10
'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!
```

teraz jako parametr wstawmy napis (słowo) np.:

```
? w ramce „bumbum i mamy
'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!
bumbum
'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!'/.!
```

wreszcie wywołajmy:

```
? w ramce [ale fajny szlaczek] i na ekranie jest
```

```
/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!
ale fajny szlaczek
'/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!/.!
```

W zależności od tego, co umieściliśmy jako parametr procedury „w ramce", to właśnie było umieszczone między szlaczkami. Najpierw była to liczba, później słowo (napis po cudzysłowie) w końcu LISTA słów ujęta w nawiasy kwadratowe. Parametr określa zatem dowolną pod względem znaczenia rzecz — obiekt, którego sens widać dopiero przy wykonywaniu procedury.

Czas teraz przystąpić do napisania „poważnego" programu (zespołu procedur) w LOGO. Najpierw postaramy się wyspecyfikować problem. Jest to bardzo ważny element (nie ma się co śmiać) projektowania każdego programu, czy całego systemu programów. Wiele problemów pojawiających się w profesjonalnej informatyce (choćby nieprzewidziane błędy w programie), wielkie tzw. pady systemów, są głównie spowodowane zaniedbaniem w początkowej fazie projektowania (zawodowcy dosyć dobrze programują). Specyfikacje zresztą to już cała gałąź informatyki.

Stawiam zatem ogólny problem:

realizacja testu z dowolnej dziedziny (może być z wielu) wiedzy. A ściślej: chcę aby komputer zadawał pytania, sprawdzał poprawność odpowiedzi — wyświetlał np. „Brawo" lub „Niedobrze" oraz liczył punkty — dobrze 1 punkt, źle 0. Na końcu powinien podać liczbę zdobytych punktów. Ocenę możemy sobie na razie darować. Pytania i prawidłowe odpowiedzi podajemy redagując (korzystając z edytora—redaktora) pewną procedurę.

Taka specyfikacja powinna być wystarczająca do zbudowania prostego systemu procedur (z jedną oczywiście główną). Od niej też zaczniemy budowanie naszego programu. Zastanówmy się jak ma przebiegać testowanie. A więc, inicjalizacja (wyświetlenie tytułu itp.), zadawanie pytań i zakończenie, czyli podanie wyniku i pożegnanie. Po prostu:

```
? oto test
> inicjalizacja
> pytania
> zakończenie
> już
   TEST zdefiniowanie
```

Prawda, że proste? Oczywiście pozostaje zdefiniowanie procedur „inicjalizacja", „pytania" i „zakończenie". No więc do roboty. Do pamiętania punktów, dla ułatwienia, użyjmy sobie zmiennej:

```
? oto inicjalizacja „przyp" to skrót od
> przyp „punkty 0" „przypisz"
> pp [Dzień dobry. Powodzenia:] „pp" to skrót od „pisz"
> już
   INICJALIZACJA zdefiniowane
```

Teraz pytania. Musimy wykazać się inwencją (na razie się bawimy) i wymyślić jakieś testowe pytania. Na przykład:

```
? oto pytania
> pytaj [Stolica ZSRR] [Moskwa]
> pytaj [Ile jest 11x11?] [121]
> pytaj [Prószy w zimie?] [śnieg]
> pytaj [Często miauczy?] [kot]
> już
  > PYTANIA zdefiniowane
```

Procedura „pytaj", którą zdefiniujemy, ma dwa parametry. Jeden to pytanie, a drugi to prawidłowa odpowiedź. Oba w konkretnym naszym wywołaniu są LISTAMI słów (w nawiasach KWADRATOWYCH). „Najtrudniejsza" procedura" „pytaj" może wyglądać tak:

```
? oto pytaj : pyt: odp.
  pp : pyt
  jeśli czytaj listę = odp (pp „Brawo!)
  przyp „punkty: punkty + !! (pp „Niedobrze!)
  już
     PYTAJ zdefiniowane
```

Słowo „czytaj listę" to instrukcja WEJŚCIA. W miejsce tego słowa zostanie wstawione to, co testowany delikwent wpisze z klawiatury — odpowiedź. Równoprawnym skrótem jest „cl". „Czytaj listę" to LISTA słów wprowadzona z klawiatury, aż do naciśnięcia ENTER. Jeśli wciśniemy od razu ENTER, lista będzie pusta — J. Może też to być jedno słowo np. kot i ENTER, a wtedy lista ta to (kot). W tym miejscu jeszcze raz przypomnę, że instrukcja „jeśli", razem ze swoimi listami do wykonania, w zależności od warunku, musi być w jednej linii logicznej. Nie można po drodze wcisnąć klawisza ENTER. Tutaj w procedurze „pytaj", będzie się ona mieścić w 3 wierszach na ekranie, a dwa wiersze będą kontynuowane (znaczek „!" w ostatniej kolumnie na ekranie).

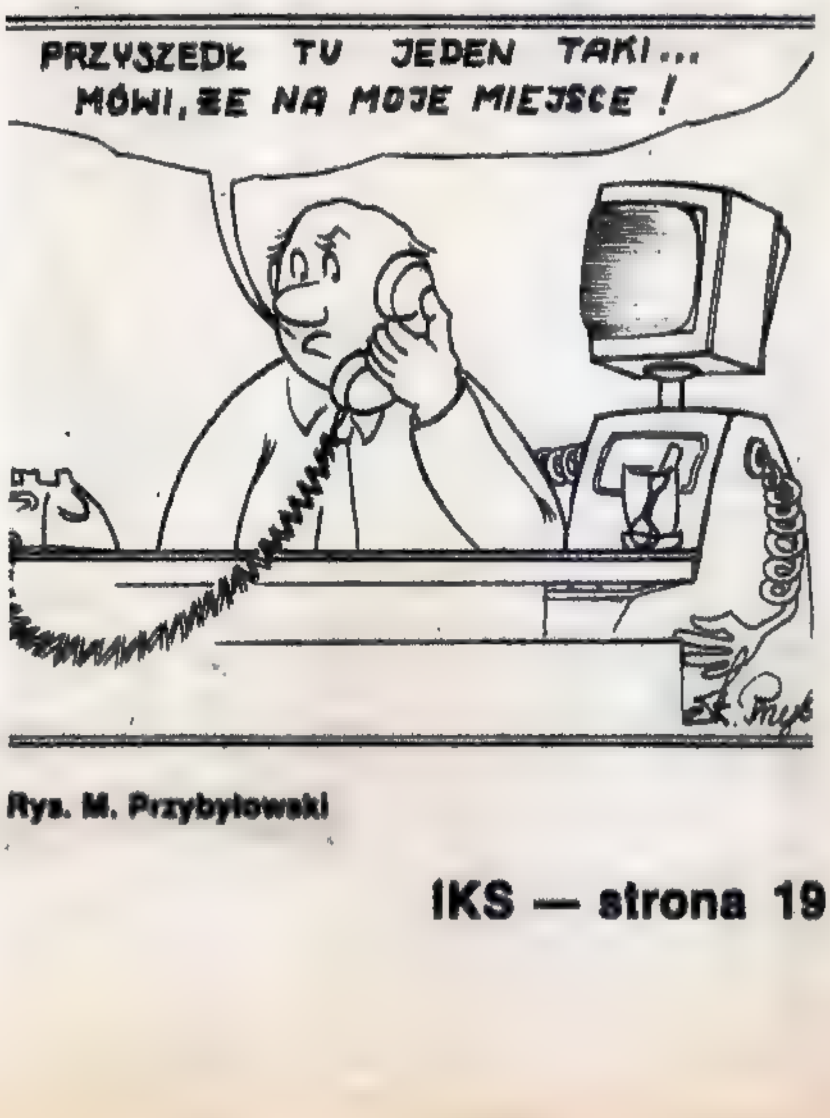
Pozostaje jeszcze problem „zakończenia". Nic prostszego:

```
? oto zakończenie
> pp „koniec!
> pp (liczba zdobytych punktów:)
> pp: punkty
> pp: [Do widzenia!]
> już
   ZAKOŃCZENIE zdefiniowane
```

i koniec. Wywołajmy teraz procedurę „test" i posadźmy kogoś przy klawiaturze — będzie trochę śmiechu! Można rozszerzyć repertuar pytań i prawidłowych odpowiedzi. Trzeba w tym celu przeredagować procedurę „pytania" (umiemy chyba korzystać z edytora — red. „pytania) i dodać np.:

```
pytaj [ile jest 13x13] [169]
pytaj [polskie morze] [Bałtyk]
```

dokończenie na str. 20



Rys. M. Przybyłowski

dokończenie ze str. 19

Zauważmy, że odpowiedź „Bałtyk" będzie zła. Należy wpisać ją dokładnie tak jak jest w nawiasie kwadratowym, czyli zacząć wielką literą. Dla przypomnienia sposobu pracy z edytorem — redaktorem, napiszę jak dodać nowe linie do procedury „pytania". A więc : najpierw piszemy:

? red. pytania

„weszliśmy" w edytor i na dole jest to sygnalizowane: Edytor LOGO... Teraz „najeżdżamy" kursorem (używając strzałek pionowych i poziomych) na koniec linii, po której chcemy wstawić nową i wciskamy ENTER. Mamy nową pustą linię. Wpisujemy tam treść nowej linii. Aby zapamiętać zmiany, wciskamy Extended mode i klawisz „C". Będzie:

PYTANIA zdefiniowane

Dla przypomnienia kilka instrukcji edytora — redaktora:

Extended mode i	
strzałka w prawo	skok na koniec linii
strzałka w lewo	skok na początek linii
strzałka w górę	skok na początek ekranu
strzałka w dół	skok na koniec ekranu
klawisz „B"	skok na początek tekstu (może być na kilku ekranach)
klawisz „E"	skok na koniec tekstu
klawisz „N"	przejście do następnej strony
klawisz „P"	przejście do poprzedniej strony
klawisz „C"	zapamiętanie zmian
klawisz „Y"	skasowanie linii na prawo od kursora
klawisz „R"	powielanie ostatnio skasowanej lub wprowadzonej linii

kasowanie liter — kursor na prawo od kasowanej litery i klawisz DELETE wyjście z edytora bez dokonania zmian — BREAK
Redagowanie (wejście w edytor) to napisanie

? red. „nazwa lub
? red. [nazwa 1 nazwa 2... nazwa]
i wtedy redagujemy więcej niż jedną procedurę
wejście w pusty edytor (tylko po co?) to

? red [ ]

wejście w ostatnio używany edytor to po prostu

? red

Extended mode i klawisz „Y" lub „R" można też używać poza edytorem do usprawnienia naszej konwersacji. Sprawdźcie to.

Na zakończenie przedstawiam cały program (5 procedur — główna „test" — ją wywołujemy) realizujący testowanie. Wydruk został zrobiony na drukarce Seikosha GP50. O tym jak współpracować z drukarką i magnetofonem, będzie jeszcze mowa w następnych odcinkach.

```
?
OTO TEST
INICJALIZACJA
PYTANIA
ZAKONCZENIE
JUŻ
```

```
OTO INICJALIZACJA
PRZYP "punkty 0
PP [Dzień dobry. Powodzenia!]
JUŻ

OTO PYTANIA
PYTAJ [Stolica ZSRR?] [Moskwa]
PYTAJ [Ile jest 11 + 11?] [22]
PYTAJ [Prdszy w ziele?] [Kotka]
PYTAJ [Często miauczy?] [Kot]
JUŻ

OTO ZAKONCZENIE
PP "Koniec!
PP [Liczba zdobytych punktów:]
PP :punkty
PP [Do widzenia!]
JUŻ

OTO PYTAJ :pyt :odp
PP :pyt
JEŚLI CZYTAJLISTĘ = :odp [pp "Brawo! przyp "punkty :punkty + 1]
[pp "Niedobrze!]
JUŻ
```

SIS



Fani mikro mają nowe miejsce „pielgrzymek". W przejściu podziemnym u zbiegu Wolskiej i Młynarskiej na warszawskiej Woli firma System oferuje mikrokomputery, urządzenia peryferyjne i programy. Najwięcej jak zwykle oglądających...
Foto: Jan Zelman



METODA ELIMINACJI ELEMENTÓW GŁÓWNYCH

Jak rozwiązać układ równań liniowych? Metod jest wiele. Jednak każda z nich jest uciążliwa, szczególnie kiedy dysponujemy jedynie kartką papieru, a równań w układzie jest więcej niż 3.

Zacznijmy zatem od początku. Mamy rozwiązać układ równań zapisany ogólnym wzorem:

$$\sum_{j=1}^{n} a_{ij}x_j = a_{i,n+1} \quad (i + 1, 2, ..., n)$$

Do rozwiązania tego układu, czyli znalezienia wszystkich x_i posłużymy się metodą eliminacji elementów głównych, polegających na kolejnym eliminowaniu współczynników układu w porządku zależnym od ich wielkości.

Pierwsze równanie dzielimy stronami przez ten współczynnik a_{1j}, który ma największą wartość bezwzględną. Związaną z nim niewiadomą eliminuje się z pozostałych równań. Z otrzymanym układem n-1 równań z n-1 niewiadomymi postępujemy podobnie: eliminujemy z niego niewiadomą, której w pierwszym równaniu nowego układu towarzyszy współczynnik o największej wartości bezwzględnej. Analogicznie eliminujemy następne niewiadome. W efekcie otrzymujemy układ równań zawierających odpowiednio n,n-1, n-2,... 2,1 niewiadomych. Otrzymujemy również numery $p_1, p_2, ... p_n$ kolejno eliminowanych niewiadomych. Ciąg ten powstaje stopniowo, w miarę eliminacji niewiadomych, z ciągu 1,2... n. Po ustaleniu, że najpierw należy wyeliminować niewiadomą x_j, przestawia się liczby 1 i j:

j,2... j-1, 1, 1-2,... n.

Za pomocą pierwszego równania eliminuje się z pozostałych równań tę niewiadomą, której numer występuje w powyższym ciągu na pozycji pierwszej i oblicza się nowe współczynniki przy pozostałych niewiadomych. W następnym kroku bada się tylko współczynniki przy tych właśnie niewiadomych w drugim równaniu. Numer niewiadomej, której odpowiada współczynnik o największej wartości bezwzględnej, przestawia się z drugą liczbą powyższego ciągu.

Za pomocą pierwszego równania (zredukowanego) eliminuje się z pozostałych równań tę niewiadomą, której numer występuje w nowym ciągu wskaźników na drugiej pozycji. Podobne operacje wykonujemy, aby wyeliminować pozostałe niewiadome.

Równania otrzymanego układu zawierają coraz mniej niewiadomych. Kolejne x_i wyznaczamy w kolejności odwrotnej do porządku eliminacji.

Powyższą metodę najłatwiej sprawdzić oczywiście na niewielkim układzie równań — większe rozwiąże poniższy program.

Pierwszy z nich jest wiernym zapisem przedstawionego algorytmu w języku ALGOL, drugi natomiast jest jego odpowiednikiem BASICu na Spectrum.

```
BEGIN
    COMMENT rozwiązanie układu metodą elementów głównych
    INTEGER h, H, i, j, n, N, ph, pj;
    REAL s, t;
    ininteger (1, n);
    N: =n+1;
    BEGIN
 INTEGER ARRAY p [1 : N];
 ARRAY a [1 : n, 1 : N], x [1 : n];
 inarray (1, a);
 FOR i: = 1 STEP 1 UNTIL n DO
   p[i] : = i;
 FOR h: = 1 STEP 1 UNTIL n DO
  BEGIN
  s: = 0;
  FOR i: = h STEP 1 UNTIL n DO
   BEGIN
   t: = abs (a[h,p [i]]);
   IF t ≥s
    THEN BEGIN
    s: = t;
    j: = i
    END t ≥s
 END i;
 ph : = p [j],
 p [j]: = p [h]:
 p [h]: = ph;
 s: = a [h, ph]:
 H: = h+1
 FOR j: = H STEP 1 UNTIL n Do
 BEGIN
 pj: = [i],
 t: = a [h, pj]: a [h, pj] /s;
 FOR i: = H STEP 1 UNTIL n DO
 a [i, pj]: = a [i, pj] - a  [i, ph]* t
 EDN j
END h;
 FOR h: = n STEP — 1 UNTIL 1 DO
  BEGIN
   s: = a [h, N];
   FOR j: = h+1 STEP 1 UNTIL n
DO
     BEGIN
     pj: = p [j]:
    s: = s-a[h, pj]*[pj]
   END j;
  x [p[h]] : = s
  END h;
 outarray (1, x)
 END
 END
```

```
5 REM ***********************
7 REM
10 REM  Program rozwiazuje n rownan liniowych o n niewiado-mych metoda elementow glow- nych.
13 REM
15 REM ***********************
20 INPUT "Podaj liczbe zmiennych ";n
30 LET z=n+1
40 DIM p(z)
50 DIM x(n)
60 DIM a(n,z)
65 REM
67 REM ***********************
69 REM
70 REM Rozpoczyna sie  wprowa-dzanie kolejnych wspolczynnikow ukladu
72 REM
75 REM ***********************
80 FOR i=1 TO n
90 FOR l=1 TO z
100 INPUT "Podaj wspolczynnik a(";(i);",";(l);") ";a(i,l)
110 NEXT l
120 NEXT i
130 FOR i=1 TO z
```

dokończenie na str. 30

PASCAL (1)

PASCAL jest jednym z najpopularniejszych języków programowania. Opracowany przez N. Wirtha w końcu lat sześćdziesiątych (opublikowany w roku 1971) szybko zdobył sobie niezwykłe powodzenie. Zadecydowały o tym takie jego cechy, jak:
— łatwość zapisu programu,
— możliwość wykorzystania do opisu licznej klasy problemów,
— przystosowanie do wymagań programowania systematycznego.

PASCAL jest obecnie powszechnie stosowany w nauczaniu programowania praktycznie na całym świecie. Nowe języki programowania w wielu wypadkach wzorowane są na języku PASCAL i jego konstrukcjach. PASCAL przyjął też rolę języka publikacyjnego, co oznacza, że większość algorytmów publikowanych w książkach i pismach specjalistycznych zapisywana jest właśnie w tym języku.

Niniejszy kurs programowania w języku PASCAL jest przeznaczony dla licznego kręgu odbiorców, którzy opanowali podstawowe wiadomości z dziedziny programowania. Zakładamy jednak, że PASCAL może być pierwszym językiem programowania, z którym styka się Czytelnik. Z tego powodu nie odwołujemy się do innych języków programowania, ilustrujemy opis języka przykładami i proponujemy zadania do samodzielnego rozwiązania. Wszystkie prezentowane przykłady zostały uruchomione w systemie TURBO PASCAL, zaimplementowanym [1] na komputerze AMSTRAD CPC 6128, pracującym pod systemem operacyjnym CP/M PLUS. Wybierając system TURBO PASCAL, kierowaliśmy się następującymi względami:
— TURBO PASCAL obejmuje (poza nielicznymi wyjątkami) właściwości standardowego języka PASCAL, a ponadto dostarcza wiele możliwości ułatwiających programowanie i zwiększających zakres języka,
— TURBO PASCAL wyposażony jest w efektywny edytor ekranowy [2], ułatwiający tworzenie i poprawianie programów źródłowych,
— TURBO PASCAL umożliwia pełne wykorzystanie systemu operacyjnego.

O wyborze komputera zadecydowało to, że z powodu ceny jest on bardziej dostępny niż inne, na których również zaimplementowano system TURBO PASCAL, np. komputery serii IBM PC.

Całość kursu, ze względu na dość obszerny zakres materiału, została podzielona na kilka części. W każdej z nich oprócz opisu elementów języka, zamieszczamy opisy wybranych własności systemu TURBO PASCAL.

1. Podstawowe zasady korzystania z systemu TURBO PASCAL

System TURBO PASCAL obejmuje edytor, kompilator i zestaw programów pomocniczych. Ponieważ prezentowany TURBO PASCAL pracuje pod kontrolą jednego z systemów operacyjnych, CP/M PLUS lub CP/M-80 2.2, dowolny z nich należy załadować do pamięci komputera przed użyciem TURBO PASCAL. Uzyskuje się to przez wprowadzenie z klawiatury komendy: CPM. Po załadowaniu systemu operacyjnego należy wprowadzić z klawiatury komendę TURBO i wcisnąć klawisz RETURN. Spowoduje to załadowanie kompilatora do pamięci komputera i wyświetlenie komunikatu wstępnego:

```
TURBO PASCAL system Version 3.00E
                        CP/M-80,Z80
Copyright 1985 AGROCOMPUTER
Terminal: AMSTRAD CPC6128
Include error messages (Y/N)?
```

zawierającego pytanie o sposób sygnalizowania błędów kompilacji. Na pytanie należy dać odpowiedź, wciskając klawisz **Y** lub **N**. Wciśnięcie klawisza **N** spowoduje, że błędy kompilacji będą sygnalizowane wyłącznie numerami. Wciśnięcie klawisza **Y** sprawi, że sygnalizacja błędów kompilacji będzie zawierała również słowne opisy błędów. Rezygnacja z opisów błędów przyniesie zwiększenie pamięci dostępnej dla programu o 1,5 Kbajtów, jednak w pierwszym etapie nauki języka opisy błędów będą raczej przydatne. Po wprowadzeniu odpowiedzi zostaje wyświetlone tzw. menu główne, zawierające zbiór dostępnych komend systemu TURBO PASCAL:

```
Logged drive: A
Work file:
Main file: Edit Compile Run Save
execute Dir Quit compiler Options
Text: O bytes
Free: 28909 bytes
```

Wykonanie określonej komendy następuje po wciśnięciu odpowiedniego klawisza (W,M,E,C,R,S,X,D,Q,O). Po wykonaniu komendy ponowne wyświetlenie menu głównego można uzyskać przez wciśnięcie klawisza **RETURN**.
Wyjście z systemu TURBO PASCAL umożliwia komenda Quit. Po wciśnięciu klawisza Q następuje powrót do systemu operacyjnego. Aby wrócić do systemu TURBO PASCAL, należy ponownie wykonać komendę **TURBO**.

2. Jak uruchomić pierwszy program

Po napisaniu programu należy wywołać edytor (komenda Edit), używając komend edytora wprowadzić program źródłowy do zbioru roboczego edytora, skompilować go i uruchomić (komenda Run). Ciąg czynności prowadzących do uruchomienia programu można przedstawić w postaci następującego algorytmu:

a) wcisnąć klawisz E (wywołanie edytora),
b) podać nazwę zbioru roboczego edytora i wcisnąć klawisz **RETURN**,
c) wprowadzić z klawiatury wiersz tekstu programu źródłowego (zostanie on wyświetlony na ekranie monitora),
d) wcisnąć klawisz **RETURN**,
e) czynności c i d wykonać dla kolejnych wierszy tekstu programu źródłowego,
f) po wprowadzeniu ostatniego wiersza tekstu programu źródłowego wcisnąć jednocześnie klawisze **CONTROL** i **K**, a następnie wcisnąć klawisz **D** (wykonanie komendy edytora **Koniec Edycji**),
g) wcisnąć klawisz **R** (kompilacja i uruchomienie programu). Po wciśnięciu klawisza E nastąpi wyświetlenie komunikatu:

Work file name:

po którym należy podać nazwę zbioru w postaci:

nazwa lub nazwa. typ

gdzie:

nazwa — ciąg od jednego do ośmiu znaków, określający nazwę zbioru,
typ — ciąg od jednego do trzech znaków, określający typ zbioru
(pominięcie typu jest równoważne przyjęciu typu domyślnego **PAS**).

W obu wypadkach żaden ze znaków nie powinien być znakiem:

= ? * . , < > : ; [ ] { }

Po podaniu nazwy zbioru (np. **FILENAME.PAS**) i wciśnięciu klawisza RETURN w pierwszym wierszu, u góry ekranu, zostanie wyświetlony opis stanu edytora w postaci:

Line 1 *Col* 1 *Insert Indent A*:FILENAME.PAS

(od tego momentu edytor reaguje tylko na własne komendy, a nie na komendy systemu TURBO PASCAL). W kolejnych wierszach będą wyświetlane wprowadzane wiersze tekstu programu źródłowego. Gdy cały ekran zostanie zapełniony, wprowadzenie nowego wiersza następuje po przesunięciu zawartości ekranu (bez opisu stanu edytora) o jeden wiersz w górę. Nie należy się jednak obawiać utraty „znikających" wierszy. Można je bowiem wyświetlić, wykorzystując odpowiednie komendy edytora.

Wciśnięcie klawisza **R** (po zakończeniu edycji) spowoduje kompilację programu i, jeśli zakończy się ona poprawnie, jego uruchomienie. W razie błędnej kompilacji należy poprawić program źródłowy i ponownie wykonać komendę Run. Poprawienie programu źródłowego dokonuje się za pomocą odpowiednich komend edytora. Opis tych komend oraz sposób ich użycia przedstawiamy w następnej części kursu.

3. Proste programy w języku PASCAL

Każdy program zapisywany w języku PASCAL składa się z trzech części:
— nagłówka (początku programu),
— bloku (utworzonego z części definiującej i części operacyjnej),
— separatora "." kończącego program,

które będziemy stopniowo definiowali, odwołując się do intuicyjnie wykazanych potrzeb.

Zacznijmy od najprostszego wariantu — wyprowadzenia na monitor ekranowy rezultatów wykonania programu, niezbędnego w większości programów. Budowę takiego programu ilustruje przykład 1.

Przykład 1

```
PROGRAM progi:
BEGIN
    WRITE ('pierwszy program w języku pascal')
END.
```

Nagłówek programu składa się z następujących elementów:
— tzw. słowa kluczowego [3] **PROGRAM**,
— nazwy programu, będącej kombinacją liter i cyfr rozpoczynającą się od litery (prog 1),
— separatora ";" oddzielającego nagłówek od bloku programu.

Blok programu, w tym prostym wypadku, zawiera tylko część operacyjną, w której pomiędzy słowami kluczowymi **BEGIN — END** zapisano jedną instrukcję wyjścia **WRITE**. Jej zadaniem jest tutaj wyprowadzenie na ekran monitora tekstu ujętego w apostrofach.

W wyniku programu "program 1" na ekranie monitora został wyświetlony tekst o postaci:

pierwszy program w języku pascal

W ogólnym wypadku cały proces obliczania przez komputer składa się z trzech faz [4]:
— wprowadzenia danych wejściowych,
— przetworzenia danych wejściowych,
— wyprowadzenia wyników.

Każda z tych faz występuje w programie "prog 2" przedstawionym w przykładzie liczb całkowitych

Przykład 2

```
PROGRAM prog 2;
  VAR x, y, suma, różnica, iloczyn: integer;
      iloraz: real; (deklaracje zmiennych)
  BEGIN
      READ (x,y);
      WRITELN;
      WRITELN ('pierwsza liczba = ' x:2, 'druga liczba =
                   y:2);
      suma: = x+y;
      różnica: =x—y;
      iloczyn: =x*y;
      iloraz: =x/y;
      WRITELN (' suma = ', suma: 3,' różnica = ' różnica:2,'
                   iloczyn = ', iloczyn:4,' iloraz = ' iloraz:
                   8:5)
  END
```

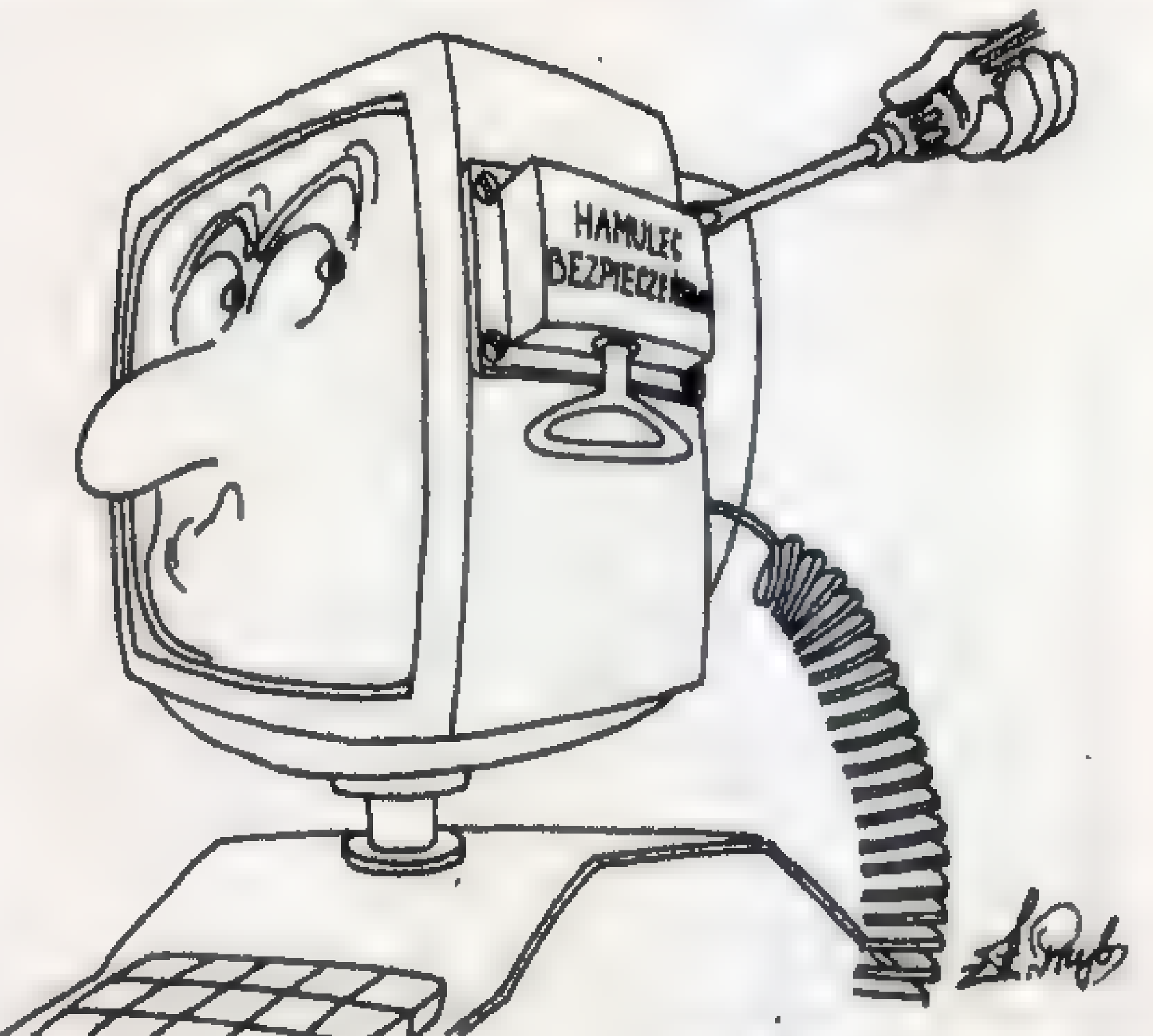
oraz wyznaczenie wyprowadzenie na monitor ekranowy ich sumy, różnicy, iloczynu i ilorazu. Zauważmy, że w części operacyjnej bloku programu występuje pięć zmiennych: X, Y, SUMA, RÓŻNICA, ILOCZYN, które będą przyjmowały wartości całkowite oraz zmienna ILORAZ, która będzie przyjmowała wartości rzeczywiste. Wykorzystanie zmiennych (symbolicznie reprezentujących obiekty rzeczywiste, w tym wypadku liczby (pozwala na wykorzystanie programu dla różnych danych wejściowych bez konieczności wprowadzania jakichkolwiek zmian w zapisie programu. W danym momencie realizacji programu zmienne reprezentują tylko jedną wartość, zwaną bieżącą wartością zmiennej. Wartość zmiennej może być zmieniona na inną jedynie w wyniku wykonania instrukcji przypisania (w której jako lewy argument operatora ":=" występuje nazwa zmiennej) lub instrukcji wejścia (w której nazwa zmiennej występuje jako jeden z argumentów).

Opisy zmiennych programu występują w części definiującej bloku programu w formie tzw. deklaracji zmiennych.

Deklaracja zmiennych składa się ze słowa kluczowego VAR i zapisów deklaracji zmiennych. Każdy zapis deklaracji zmiennych składa się z następujących elementów:
— listy nazw deklarowanych zmiennych danego typu (np. liczby całkowite) oddzielonych od siebie przecinkami,
— znacznika ":" oddzielającego listę zmiennych od nazwy typu,
— nazwy typu (klasy wartości, przyjmowanej przez deklarowane zmienne, np. integer, dla liczb całkowitych, real — dla liczb rzeczywistych),
— separatora ";" oddzielającego zapis danej deklaracji zmiennej od innych zapisów.

Występująca w części operacyjnej instrukcja wejścia READ wykazuje pierwszą z wymienionych faz — wprowadzenia danych wejściowych. Zgodnie z tą instrukcją następuje czytanie liczb, na których będą wykonywane żądane operacje. Wartości wczytanych liczb są przypisywane zmiennym X i Y.

dokończenie na str. 24



Rys. Michał Przybyłowski

dokończenie ze str. 23

Fazę drugą uruchamiają cztery instrukcje przypisania, w których kolejno wyznacza się sumę, różnicę, iloczyn i iloraz wczytanych liczb. Każda z tych instrukcji składa się z nazwy zmiennej, operatora przypisania ":=" i wyrażenia (w tych wypadkach arytmetycznego). Występujące w wyrażeniach operatory +, —, •, / oznaczają odpowiednio dodawanie, odejmowanie, mnożenie, dzielenie. Należy zauważyć, że wartość każdego wyrażenia ma typ zgodny z typem zmiennej, do której jest przypisywana.

Do wyprowadzenia wyników na ekran monitora (realizacja fazy trzeciej) wykorzystano instrukcję WRITELN. Instrukcja ta, podana bez parametrów, umożliwia przejście do nowej linii. Wykonanie instrukcji WRITELN z parametrami jest podobne do wykonania instrukcji WRITE, przy czym po wyprowadzeniu wartości parametrów następuje przejście do nowej linii. Parametrami instrukcji WRITELN są tutaj, oprócz tekstów, nazwy zmiennych, których wartości mają być wyprowadzane. Określona jest przy tym liczba pozycji znakowych, jakie ma zajmować wyprowadzana wartość. Dla zmiennej typu real określono ponadto liczbę pozycji znakowych przeznaczonych na część ułamkową (w tym przypadku 5). Rozplanowując rozmieszczenie wyników na ekranie monitora założono, że wprowadzane liczby będą liczbami z zakresu (1,99). Dla liczb 27 i 13 uzyskano następujący wynik działania programu:

pierwsza liczba = 27 *druga liczba* = 13
suma = 40 *różnica* = 14 *iloczyn* = 351 *iloraz* = 2.07692

Liczby te zostały wprowadzone z klawiatury (wpisanie liczb 27 13 i naciśnięcie klawisza **RETURN** lub **ENTER**).

W przykładzie 2 zamieszczono tekst objaśniający, ujęty w nawiasy klamrowe. Jest to komentarz nie mający żadnego wpływu na przebieg obliczeń (jest pomijany przez kompilator). Komentarze będą używane również w dalszych programach w celu zwiększenia czytelności programów i ułatwienia zrozumienia ich treści.

Dotychczas poznane instrukcje umożliwiają już wykonanie prostych obliczeń na wprowadzonych danych oraz prezentację wyników. Przeważnie jest to niewystarczające, gdyż wykonanie takich działań często zależy od spełnienia określonych warunków. Zagadnienie to zilustrujemy bliżej na przykładzie wyznaczania pierwiastków rzeczywistych trójmianu kwadratowego (przykład 3). Jak wiemy, liczba pierwiastków rzeczywistych zależy od wartości

Przykład 3

```
PROGRAM prog 3,
VAR a, b, c, (współczynniki trójmianu)
     x1, x2 (pierwiastki trójmianu)
     delta:real; (zmienna pomocnicza)
BEGIN
     READLN (a);
     IF a < = 0 THEN
     WRITELN (' niedopuszczalna wartość współczynnika
     a')
     ELSE
     BEGIN
     READ (b, c);
     WRITELN;
     WRITELN (' dla współczynników: a = ', a);
     WRITELN ('                     b = ', b);
     WRITELN ('                     c = ', c);
     delta: = b*b—4*a*c; (obliczenie wyróżnika trójmianu)
     IF delta < 0 THEN
        WRITELN (' brak pierwiastków rzeczywistych')
     ELSE
        IF delta = 0 THEN
     BEGIN
        x 1 : = —b/ (2*a);
        WRITELN (' trójmian ma jeden pierwiastek
                   podwójny x = ', x 1)
     END
     ELSE
     BEGIN
        x 1: =(—b—SQRT (delta)) / (2*a);
        x 2: =(—b+SQRT (delta)) / (2*a);
        WRITELN;
        WRITELN (' trójmian ma dwa pierwiastki: x 1 = ',
                   x 1);
        WRITELN ('.                             x 2 = ', x 2)
     END
   END
 END.
```

wyróżnika trójmianu. Zatem w zależności od jego wartości należy wykonać inną grupę instrukcji. Wybór umożliwia instrukcja alternatywy o postaci:

IF warunek THEN instrukcja 1 ELSE instrukcja 2

Wykonanie instrukcji alternatywy polega na zbadaniu warunku i jeśli jest on spełniony, następuje wykonanie instrukcji 1. W przeciwnym razie wykonywana jest instrukcja 2. W warunkach z przykładu 3 zastosowano operatory relacyjne: < (mniejszy), <= (mniejszy lub równy), = (równy). Pozostałe operatory relacyjne to: > (większy), >= (większy lub równy), <> (nierówny). W celu umożliwienia wykonania określonej grupy instrukcji w wypadku spełnienia (niespełnienia) warunku wykorzystano instrukcję złożoną. Instrukcja złożona jest sekwencją instrukcji zawartą między parą słów kluczowych BEGIN — END. Jest ona traktowana jako pojedyncza instrukcja. Do wyznaczania wartości pierwiastka wyróżnika trójmianu użyto standardowej funkcji SQRT. Odpowiednią kolejność wykonywania działań w wyrażeniach arytmetycznych zapewniają nawiasy (podobnie, jak w powszechnie stosowanej notacji matematycznej). Instrukcję READLN, wykorzystaną do wczytania współczynnika "a", wykonuje się jak instrukcję READ, przy czym po wczytaniu danej wejściowej następuje przeskok do następnej linii wczytania (w tym przypadku do następnego wiersza ekranu). Wartości wyznaczonych pierwiastków (jeśli istnieją) wyprowadzane są w standardowym formacie zmiennoprzecinkowym, przyjętym dla liczb rzeczywistych. Ma on postać:

□□ #.##########E * ##

dla liczb większych lub równych zero oraz postać:

□ - #.##########E * ##

dla liczb mniejszych od zera, przy czym □ reprezentuje spację, # reprezentuje cyfrę dziesiętną, *reprezentuje znak + lub —.

Wartość tak wyprowadzonej liczby należy odczytywać jako

$$c \cdot 10$$

gdzie:

c — liczba znajdująca się przed literą E (cecha),
m — liczba znajdująca się za literą E (mantysa).

Program przedstawiony w przykładzie 3 wykonano dla następujących wartości współczynników trójmianu a = 2,67, b = 13,63, c = 7,21 i uzyskano następujące wyniki:

dla współczynników: a = 2.6700000000E+00
b = 1.3630000000E+01
c = 7.2100000000E+00
trójmian ma dwa pierwiastki: x 1 = —4.5055210206E+00
x 2 = —5.9934789329E—01

Dane zostały wprowadzone z klawiatury w następujący sposób:

— wpisanie liczby 2.67 i wciśnięcie klawisza RETURN,
— wpisanie liczb 13.63 i 7.21 i wciśnięcie klawisza RETURN.

Zamiast klawisza RETURN można używać również klawisza ENTRY

Wprowadzona instrukcja alternatywy może być wykorzystywana do opisu działań o wybiorczym przebiegu. Nie pozwala natomiast opisać operacji wykonywanych powtarzalnie. Jedną z instrukcji dostępnych w języku PASCAL, umożliwiających taki opis, jest instrukcja WHILE o postaci:

WHILE *wyrlog* Do
instr

gdzie:

wyrlog — wyrażenie logiczne przyjmujące wartość TRUE(prawda) lub FALSE (fałsz),
instr — pojedyncza instrukcja języka PASCAL

Powyższy zapis oznacza, że wykonanie instrukcji „instr" ma być powtarzalne tak długo, aż wartość wyrażenia logicznego „wyrlog" zostanie doprowadzona do wartości FALSE. Zauważmy, że jeżeli już przed wykonaniem instrukcji

WHILE wartością „wyrlog" jest FALSE, to instrukcja „instr" nie jest w ogóle wykonywana. Instrukcję WHILE wykorzystano w programie wyznaczania największego wspólnego podzielnika dwóch całkowitych liczb dodatnich (przykład 4).

Przykład 4.

```
PROGRAM prog 4;
    VAR x,y: integer;
    BEGIN
      READ (x, y);
      WRITELN;
      WRITE (' największy wspólny podzielnik liczb
        ',x,' i ',y,' wynosi ');
      WHILE x<>y DO
    IF x>y THEN
    x: = x—y
    ELSE
    y: = y—x;
    WRITELN (x)
  END
```

Nietrudno zauważyć, że wpływ na wartość wyrażenia logicznego w instrukcji WHILE (x<>y) ma wykonanie instrukcji alternatywy, a więc instrukcji „instr". Należy przy tym zdawać sobie sprawę z tego, że jeżeli po żadnym z powtórzeń instrukcji „instr" wyrażenie logiczne nie osiągnie wartości FALSE, to wykonywanie instrukcji WHILE nigdy się nie zakończy. Mówi się wtedy często o „zapętleniu" programu.

Wyrażenia występujące w przykładzie 4 zawierają nazwy tych samych zmiennych po obu stronach operatora „: = ", np.

x: = x - y.

W takim wypadku przy obliczaniu wartości wyrażenia brana jest pod uwagę wartość zmiennej x przed wykonaniem instrukcji podstawienia. Wyznaczona wartość wyrażenia jest następnie przypisywana zmiennej x jako nowa wartość (a więc poprzednia wartość zostaje zniszczona).

Program wykonano dla liczb 256 i 64. Uzyskano następujące wyniki:

Największy wspólny podzielnik liczb 256 i 64 wynosi 64

Ponieważ w instrukcjach wyjścia nie określono liczby pozycji znakowych dla wyprowadzonych liczb całkowitych, zajmują one tyle pozycji, ile mają cyfr znaczących.

Kolejne wiadomości o języku PASCAL — w następnej części.

4. Zadania

Wychodząc z założenia, że programowania nie można nauczyć się bez intensywnego i samodzielnego opracowywania programów, ich kompilacji, uruchamiania i testowania, w każdej części, będziemy przedstawiali zadania do samodzielnego opracowania. Zestawy zadań postaramy się tak dobrać, aby ich opracowanie było możliwe na podstawie już przedstawionych wiadomości o języku. Zachęcamy Czytelnika do rozwiązania chociażby jednego z zadań.

Zadanie 1

Napisać program wyznaczający wartość wyrażenia:

a) $\frac{x^2 - ax^3}{2ax}$;

b) $\frac{(a \cdot b)^2 + \frac{c}{d}}{a - (c - \frac{d}{b})^2}$

Zadanie 2

Napisać program, który sprawdza czy trzy całkowite liczby dodatnie a, b, c, są długościami boków trójkąta i jakiego (równoramiennego, równobocznego, prostokątnego).

Zadanie 3

Napisać program obliczania wartości szeregu

$$1 + \frac{x^2}{2} + \frac{x^4}{2 \cdot 4} + \frac{x^6}{2 \cdot 4 \cdot 6} + \dots\dots\dots$$

---

1 Implementację języka uzyskuje się przez opracowanie translatora, tj. programu tłumaczącego teksty z danego języka (źródłowego) na inny język (wynikowy). Translatory dzielimy na kompilatory i interpretery. System TURBO PASCAL wyposażony jest w kompilator języka PASCAL.

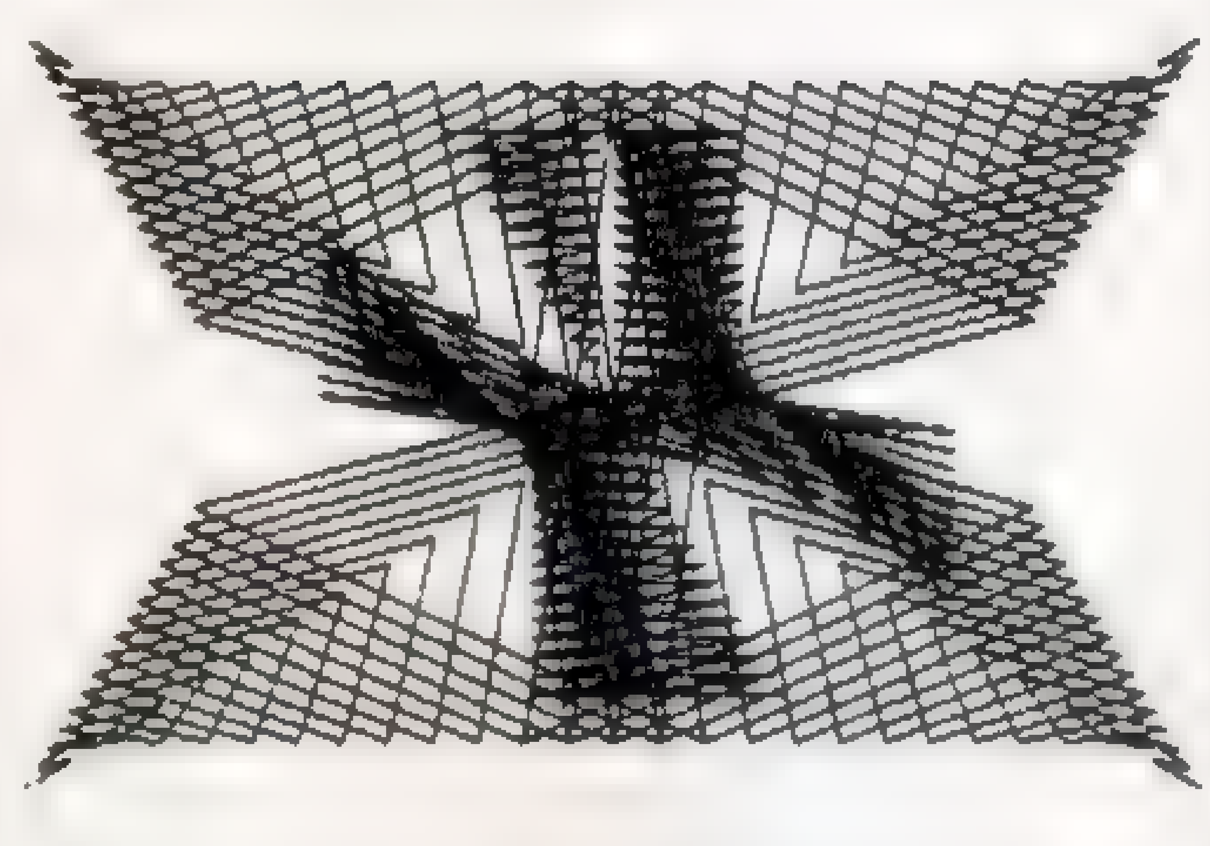
2 Edytor ekranowy pozwala na bezpośrednie działanie na tekście aktualnie wyświetlanym na ekranie monitora. Za pomocą odpowiednich komend tekst taki można rozsuwać, wstawiać nowe słowa, kasować określone słowa lub wiersze tekstu itp.

3 Słowa kluczowe języka są słowami o określonym znaczeniu i mogą być używane tylko zgodnie z regułami dopuszczanymi przez język. W przykładach programów słowa kluczowe są napisane, dla odróżnienia, dużymi literami.

4 Jest to podział logiczny. W programach, części programów zaliczane umownie do poszczególnych faz z reguły się przeplatają.

S.R.

```
10 CLS
20 ORIGIN 320,200
30 FOR x=1 TO 10
35 FOR y=1 TO 15.
40 DRAW -7*x,10*y
45 DRAW -5*y,10*y
47 NEXT y
48 ORIGIN 390,200
50 DRAW 5*y,-10*x
60 NEXT x
70 ORIGIN 320,200
80 FOR k=1 TO 10
90 FOR m=1 TO 15
100 DRAW 7*k,-10*m
110 DRAW 5*m,-10*m
120 NEXT m
125 ORIGIN 270,200
130 DRAW -5*m,10*k
135 NEXT k
137 ORIGIN 580,400
140 FOR x=1 TO 15
143 DRAW -20*x,-15*x
145 DRAW -5*x,-10*x
147 DRAW -20*x,-25
155 NEXT x
160 ORIGIN 580,0
170 FOR x=1 TO 15
180 DRAW -20*x,15*x
185 DRAW -5*x,10*x
190 DRAW -20*x,25
200 NEXT x
210 ORIGIN 60,0
220 FOR x=1 TO 15
230 DRAW 20*x,15*x
240 DRAW 5*x,10*x
250 DRAW 20*x,25
260 NEXT x
270 ORIGIN 60,400
280 FOR x=1 TO 15
290 DRAW 20*x,-15*x
300 DRAW 5*x,-10*x
310 DRAW 20*x,-25
320 NEXT x
```



Grafika komputerowa (Amstrad)

PROGRAM 37

ATARI

„Węglowodory", to prosty program edukacyjny, przeznaczony do sprawdzenia wiadomości o węglowodorach od C_1H_4 do C_8H_{16}. Program może być użyty jako sprawdzian wiedzy lub do zobrazowania wzorów strukturalnych. Przy niewielkim nakładzie pracy możliwe jest rozbudowanie programu o inne grupy związków organicznych.

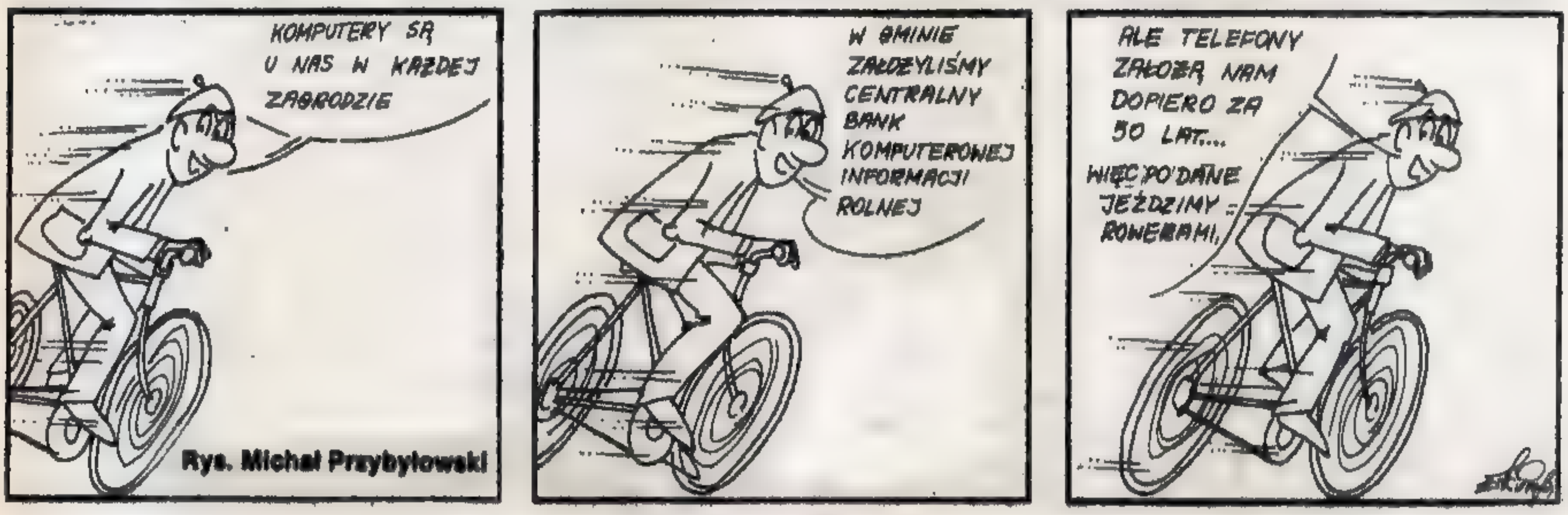
```
5 REM ** WEGLOWODORY **
7 OPEN #1,4,0,"K:"
10 DIM A$(90),B$(7),C$(75),D$(6)
12 DIM E$(6),F$(40),Q$(1)
14 F$="                                      "
16 Q$=" "
20 GOSUB 1000
23 ? "}":POKE 752,1
25 POSITION 14,2:? "[ WEGLOWODORY ]"
30 POSITION 6,8:? "Czy potrzebujesz wyjasnienia"
35 POSITION 16,10:? "[T]ak/[N]ie"
45 GET #1,X
50 IF X=84 THEN GOTO 200
55 IF X<>84 THEN GOTO 65
65 P=0:PP=0
80 I=INT(RND(0)*15):J=I*6+1:K=I*5+1
82 B$=A$(J,J+5)
83 D$=C$(K,K+4)
85 IF PP=0 THEN 92
90 FOR F=1 TO 500:NEXT F
92 GOSUB 1200
99 POSITION 14,2:? "[ WEGLOWODORY ]"
101 POSITION 1,20:? "[  Wzory strukturalne        W         ]"
102 POSITION 1,21:? "[  Prawidlowe odpowiedzi     Ocena     ]"
103 POSITION 1,22:? "[  Uruchomienie T Kasowanie N          ]"
105 POSITION 27,21:? P
107 POSITION 7,7:? B$
110 POSITION 7,9:INPUT E$
111 PP=PP+1
112 IF E$="W" THEN GOSUB 400
114 IF E$="T" THEN 23
116 IF E$="N" THEN 1300
120 L=LEN(E$)
122 IF L=4 THEN E$(LEN(E$)+1)=Q$
126 IF PP<16 THEN 133
127 IF P>=7 THEN POSITION 36,21:? "3"
128 IF P>=10 THEN POSITION 36,21:? "4"
129 IF P>=13 THEN POSITION 36,21:? "5"
130 IF P<7 THEN POSITION 36,21:? "2"
132 IF PP>=16 THEN 1100
133 IF E$=D$ THEN 300
134 IF E$="W" THEN 80
135 POSITION 5,16:? "Zle-sprobuj jeszcze raz"
140 GOTO 90
200 REM OBJASNIENIE
205 POSITION 2,4:? "Sprawdzenie wiadomosci o alkanach"
210 POSITION 2,6:? "i alkenach."
215 POSITION 2,8:? "Gdy pokaze sie nazwa weglowodoru"
220 POSITION 2,10:? "podaj jego wzor sumaryczny."
225 POSITION 2,12:? "Wciskajac klawisz W i RETURN otrzy-"
```

```
230 POSITION 2,14:? "mujesz wzor strukturalny."
235 POSITION 2,16:? "Klawisz T i RETURN uruchamiasz"
240 POSITION 2,18:? "program od poczatku."
245 POSITION 2,20:? "Klawisz T i RETURN kasuje program."
250 POSITION 2,22:? "[ Wcisnij dowolny klawisz ]"
260 GET #1,X
265 ? "}"
270 IF X<>27 THEN 65
300 POSITION 5,16:? "Wynik dobry - kontynuuj"
310 P=P+1:POSITION 27,21:? P
320 GOTO 80
400 REM WZORY STRUKTURALNE
402 L=0
405 IF I>7 THEN 450
407 I=I+1
410 POSITION 20,8:? "H":POSITION 20,9:? "|":POSITION 18,10:? "H-C-H":POSITION 20
,11:? "|":POSITION 20,12:? "H"
415 IF I=1 THEN 495
425 FOR A=2 TO I:L=L+2
430 POSITION 20+L,8:? "H":POSITION 20+L,9:? "|":POSITION 20+L,10:? "C-H":POSITIO
N 20+L,11:? "|":POSITION 20+L,12:? "H"
435 NEXT A
440 GOTO 495
450 POSITION 20,8:? "H H":POSITION 20,9:? "| |":POSITION 20,10:? "C=C":POSITION
20,11:? "| |":POSITION 20,12:? "H H"
455 IF I=8 THEN 495
460 POSITION 22,8:? " ":POSITION 22,9:? " "
465 FOR A=9 TO I:L=L+2
470 POSITION 22+L,8:? "H":POSITION 22+L,9:? "|":POSITION 21+L,10:? "-C-H":POSITI
ON 22+L,11:? "|":POSITION 22+L,12:? "H"
475 NEXT A
495 FOR F=1 TO 700:NEXT F
500 RETURN
1000 A$="METAN ETAN  PROPANBUTAN PENTANHEKSANHEPTANOKTAN ETEN  PROPENBUTEN PENTE
NHEKSENHEPTENOKTEN "
1010 C$="C1H4 C2H6 C3H8 C4H10C5H12C6H14C7H16C8H18C2H4 C3H6 C4H8 C5H10C6H12C7H14C
8H16"
1020 RETURN
1100 GOSUB 1200
1110 POSITION 5,16:? "Czy chcesz sprobowac jeszcze raz"
1120 POSITION 16,18:? "[T]ak/[N]ie"
1130 GOTO 110
1200 FOR F=4 TO 19
1210 POSITION 0,F:? F$
1220 NEXT F
1230 RETURN
1300 ? "}":POKE 752,0:NEW
```

UWAGA !

Sposob wprowadzania niektorych znakow niedostepnych w druku :

[T] - REVERSE VIDEO a potem T
} - ESC a nastepnie CONTROL i CLEAR



Rys. Michał Przybyłowski

OKNA

Po długim oczekiwaniu rynek oprogramowania mikrokomputerów wzbogacił się o systemy „interfejsu graficznego". Sprawiają one, iż komputer wydaje się bardziej „życzliwy".

Firma DIGITAL RESEARCH była pierwszą, która dostarczyła użytkową wersję systemu graficznego — GEM. Znakomicie ułatwia on wykonywanie żmudnych i kłopotliwych czynności związanych z przekazywaniem komunikatów do systemu operacyjnego mikrokomputera, a także ułatwia odczyt informacji adresowanych przez maszynę do użytkownika.

Praktycznie, w dużej części służy temu tak zwana myszka. Jest to urządzenie w postaci niewielkiego (jak paczka papierosów) pudełka z klawiszami (najczęściej od jednego do trzech klawiszy). W dolnej płaszczyźnie pudełka znajduje się kulka, która, podobnie do zainstalowanej w końcówce długopisu, przesuwana po dowolnej płaszczyźnie, obraca się. Każdy ruch (obrót) kulki, dzięki odpowiednim układom elektronicznym myszki, sygnalizowany jest komputerowi (za pośrednictwem kabla — ogonka myszki).

Przesuwanie myszki, np. po stole, odzwierciedlane jest na ekranie monitora zmianą położenia charakterystycznego punktu, tak zwanego znacznika.

Zatem urządzenie to pozwala między innymi na wskazywanie jednego z wielu wyświetlanych na ekranie komunikatów, np. składową menu — „wyczyść ekran". Gdy znacznik znajdzie się w polu wyświetlanego komunikatu, należy po prostu nacisnąć określony klawisz myszki. Jest to jednoznaczne z napisaniem wyróżnionych słów za pośrednictwem klawiatury.

OKNA (produkt firmy MICROSOFT) także zapewniają realizację tego typu zadań. Ponadto pozwalają na pseudowieloprogramowość. Oznacza to, że stwarzane jest wrażenie jednoczesnej realizacji więcej niż jednego użytkowego programu.

Podstawową ideą tej formy graficznej komunikacji z mikrokomputerem jest uproszczenie czynności manualnych związanych z kierowaniem pracą systemu, a także poprawienie jakości postaci informacji przekazywanej przez mikro.

Na przykład, jeśli przetwarzane są jednocześnie dwa programy, dwie wyróżnione części ekranu, umownie nazywane **oknami**, prezentują obrazy odzwierciedlające aktualny stan procesu przetwarzania informacji przez te programy. Monitor służy do „podglądania" pracy mikro.

Podobnie jak ekrany monitorów w supersamie służą do „podglądania" wielu miejsc sklepu jednocześnie przez jedną osobę. W przypadku sklepu źródłem informacji są kamery. W mikro rolę tę pełni np. system OKNA.

Myszka jest wykorzystywana do szeregu różnych zadań, jednym z nich jest wskazywanie wybranej czynności, którą ma wykonać mikro, na przykład: nowy rozmiar okna, otworzenie nowego okna.

Czynności związane z instalowaniem systemu OKNA w nowym komputerze nie są zbyt złożone. Wymagane jest tylko wspomniane manipulowanie myszką. Polega ono na wskazywaniu wyróżnionych elementów menu, które jest wyświetlane na ekranie w jednym z okien.

Zazwyczaj w pierwszej kolejności dokonuje się tym sposobem odpowiednie przystosowanie dysku do pracy pod kontrolą okien. Następnie należy przekazać do systemu informacje o konfiguracji sprzętowej (rodzaj drukarki, wielkość i typ pamięci itp.).

Sterowanie systemem OKNA jest tak proste, że najczęściej podręcznik okazuje się zbędny. Jednak mając na uwadze małe doświadczenie użytkowników końcówek mikro, firma MICROSOFT zamierza jednak sprzedawać odpowiednią dokumentację.

OKNA wykorzystują tę samą metodę prezentacji co konkurencyjne systemy: GEM i MACINTOSH, jednak nie dysponują możliwością polegającą na wyświetlaniu komend sterowania systemem oraz zbiorów w postaci IKON.

Jednym z podstawowych założeń współczesnych metod porozumiewania się obrazem, jest maksymalne ograniczenie sztucznych, wyuczonych symboli, jakimi są między innymi litery i cyfry. Lansowany jest powrót do naturalnego języka, jakim wydają się obrazy, w swej uproszczonej formie (przy zachowaniu meritum treści) są nimi, według umownej terminologii, ikony. Najczęściej ilustrują przedmiot lub czynność. Przykładem takiej konwencji ikon są niektóre oznaczenia w miejscach publicznych, a także część znaków drogowych.

W odniesieniu do techniki mikrokomputerowej prace w tym kierunku prowadzone są od dosyć dawna i ich wynikiem jest między innymi SNOBOl 4 — język ikon.

Ikony w sposób istotny usprawniły i ułatwiły proces porozumiewania się człowieka z komputerem i odwrotnie. Często zamiast mało czytelnych słów niewielka rycina ukazana na ekranie monitora dostarcza nam szybko i bezbłędnie określoną treść.

Obecnie ikony dostępne są we wszystkich profesjonalnych mikrokomputerach. Pokonują bariery językowe i są najbliższe naturalnej tendencji porozumiewania się. Redukują czas uczenia się, jest mniej kłopotów podczas pracy z mikro, staje się ona atrakcyjna. Doskonałą ilustracją tej metody jest większość gier komputerowych, które dzięki ikonom są zrozumiałe w czasie kilku chwil, nawet dla dzieci. Okazuje się bowiem, że język jakim posługuje się autor gry nie ma znaczenia. W zasadzie bawiący się nie muszą umieć czytać. Klasycznym przykładem takiej gry jest SHADOWFIRE (zabawa, której twórcą jest Danton Designs z firmy BEYOND).

Kiedy zamierzamy sprawdzić zawartość dysku — OKNA wyświetlają aktualną listę zbiorów w postaci tekstu. Natomiast GEM i MACINTOSH oferują tekst i ikony.

Wydaje się, że jednak OKNA są systemem lepszym od swych konkurentów. Górują między innymi szybkością odpowiedzi (na polecenia wydawane za pomocą myszki), łatwiejsze jest sterowanie pracą mikrokomputera i uzyskuje się zdecydowanie bardziej czytelną formę prezentacji menu.

Szybkie otwieranie różnych okien, „podglądanie" różnych newralgicznych punktów realizowanego programu, np.: procesu kontroli danych wejściowych, postaci danych zapisywanych na dysku, pośrednich wyników przetwarzania, następuje praktycznie bezzwłocznie, ale pod warunkiem, że przetwarzany jest tylko jeden program. Przyczyna tkwi w szybkości procesora, który jest udostępniany przez system kolejno realizowanym programom. Wyświetlane okna są „zdjęciami" zrobionymi podczas ostatniego kontaktu programu z procesorem. Stąd okres między kolejnymi zdjęciami (oknami) tego samego programu rośnie wraz z liczbą przetwarzanych programów. Gdy pracuje jeden program, okres ten jest niezauważalny.

Niestety OKNA są w pełni wykorzystane tylko wówczas, gdy wspomagają pracę oryginalnych programów MICROSOFTa. Dołączany jest wówczas tak zwany pamiętnik programu użytkowego, notatnik dla użytkownika, kalkulator i zegar. Wszystkie te zdolności są odpowiednio prezentowane na ekranie monitora w oknach, np. zegar w postaci tarczy i tradycyjnych wskazówek.

System ten posiada także prostą bazę danych, rejestrowane są one w postaci obrazów i tekstów.

Przykładem firmowego oprogramowania w pełni wykorzystującego możliwości OKIEN jest PAINT. Pozwala on tworzyć obrazy o wyszukanej formie i dowolnej treści. Stanowi konkurencję dla uznanego przez profesjonalistów

dokończenie na str. 30







Przyjechali... aż zaprzęgiem. Oprócz wielu atrakcji, przywieźli również komputery. A najwięcej radości z „Dni Berlina" w Warszawie miały dzieci.

Foto: Ryszard Rogoń



CHIŃSKIE KOMPUTERY PERSONALNE

Chińska Republika Ludowa oferuje mikrokomputery oznaczone 0520A, 0520 C-E, 0520 C-H kompatybilne z IBM PC. Model 0520 C-H ma mikroprocesor 8088, 512 KB pamięci operacyjnej RAM, dwa mechanizmy dysków elastycznych o gęstości zapisu 360 KB oraz dysk stały 20 MB.

W monitorze kolorowym ekran przy modzie graficznym pracuje z 648 x 504 punktami bezpośrednio adresowalnymi, w monitorze monochromatycznym z 972 x 700 punktami bezpośrednio adresowalnymi.

ARM

Mikroprocesor ARM, czyli Acorn RISC (Redunded Instruction Set Computer) Machine, o architekturze z przetwarzaniem potokowym (pipelining), może wykonać 3 miliony rozkazów w ciągu sekundy. Zbudowany jest z 25 000 tranzystorów, umieszczonych na powierzchni 7 mm^2, a wykonanych technologią CMOS. ARM ma 25 pełnych rejestrów 32-bitowych. Mikroprocesor dysponuje 26-bitową magistralą adresową, czyli może adresować przestrzeń 64 MB a dokładniej — adresuje 67 108 864 słowa 32-bitowe. Wyposażony jest w 32-bitową magistralę danych.

Dyski laserowe

Firma AGA Inc. wprowadza system dysków laserowych dla komputerów personalnych IBM. Dyski o średnicy 12 cali mają umożliwiać przy zapisie danych tekstowych osiągnięcie pojemności 3 GB, przy zapisie obrazów graficznych — pojemność 50 GB.

J. W.

LIGA MYŚLĄCYCH

Zadanie nr 1

Dwie beczki zawierają razem 240 litrów wody. Gdyby z pierwszej beczki przelać do drugiej tyle litrów wody, żeby zawartość drugiej beczki podwoiła się, a następnie z drugiej przelać do pierwszej tyle litrów wody, żeby zawartość pierwszej beczki podwoiła się to w obu beczkach będzie jednakowa liczba litrów wody.
Ile litrów wody było pierwotnie w każdej beczce?

Zadanie nr 2

Trzech robotników może wykonać pracę w ciągu t godzin. Pierwszy z nich, pracując sam, może wykonać pracę dwa razy szybciej od trzeciego, a o godzinę krócej od drugiego. W jakim czasie każdy z nich może wykonać tę pracę samodzielnie?

Zadanie nr 3

Ile najmniej kuponów TOTO LOTKA należy skreślić, aby uzyskać co najmniej jedno prawidłowe skreślenie sześciu liczb przy założeniu, że skreślamy sześć liczb z 49? Proszę podać sposób rozwiązania.

Rozwiązania zadań prosimy przysyłać pod adresem redakcji do końca grudnia br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

dokończenie ze str. 21

```
140 LET p(i)=i
150 NEXT i
154 FOR h=1 TO n
155 LET s=0
160 FOR i=1 TO n
170 LET t=ABS (a(i,p(i)))
180 IF t>=s THEN  LET s=t: LET j=i
190 NEXT i
200 LET g=p(j): LET p(j)=p(h)
210 LET p(h)=g
220 LET s=a(h,g)
230 LET w=h+1
240 FOR j=w TO z
250 LET q=p(j)
260 LET a(h,q)=a(h,q)/s
270 LET t=a(h,q)
280 FOR i=w TO n
290 LET a(i,q)=a(i,q)-a(i,g)*t
300 NEXT i
310 NEXT j
320 NEXT h
330 FOR h=n TO 1 STEP -1
340 LET s=a(h,z)
350 FOR j=h+1 TO n
360 LET q=p(j)
370 LET s=s-a(h,q)*x(q)
380 NEXT j
390 LET x(p(h))=s
400 NEXT h
410 REM  Wydruk   rozwiazania                ukladu rownan
420 FOR i=1 TO n
430 PRINT "x(";i;") = ";x(i)
440 NEXT i
```

dokończenie ze str. 28

pakietu o nazwie MACPAINT (zbudowany on został przez Macintosha na komputer Apple).

Dostatecznie duża pamięć komputera pozwala na jednoczesne wykorzystywanie kilku wersji tego samego programu. Jest to bardzo użyteczne np. podczas stosowania PAINTa bowiem zbiór obrazów może być wówczas wspólny dla kilku programów. Wielu użytkowników korzysta z tego samego zbioru obrazów, także obrazy są przekazywane między programami.

Jednym z najaktualniejszych zamierzeń MICROSOFTa (twórcy OKIEN) jest rozwiązanie problemu wieloprogramowości i wielodostępu. Prace w tym kierunku uzależnione są od uzyskania odpowiedniej wersji systemu operacyjnego MS DOS.

Ciekawostkę stanowi zapewne całkowicie poprawna praca OKIEN pod systemem GEM.

Odpowiedzią IBMa na prezentowane oprogramowanie Macintosha, OKIEN MICROSOFTa oraz GEMa firmy DIGITAL RESEARCH jest TOPVIEW.

Zamiarem IBMa było osiągnięcie efektu „życzliwych zachowań" personalnych komputerów. Jednak porównując TOPVIEW z OKNAMI lub z GEMem należy z całą stanowczością stwierdzić, iż jest to zdecydowanie gorsze oprogramowanie.

Powierzchownie TOPVIEW jest bardziej podobny do OKIEN niż do GEMa lub Macintosha. System ten nie reprezentuje jednak programu graficznego, nie używa ikon, ponadto sterowanie tym systemem jest bardzo kłopotliwe. Wymagana jest myszka i klawiatura. Część poleceń należy wybierać myszką, a część tradycyjnie przekazywać za pośrednictwem klawiatury. Na przykład, aby cokolwiek napisać, najpierw trzeba skorzystać z myszki, by ustalić miejsce kursora na ekranie monitora mikro, dopiero potem można napisać wybrany komunikat na klawiaturze.

W zasadzie TOPVIEW jest sterowany klawiaturą.

Pierwszą informacją wyświetlaną na ekranie przez ten system jest „TOPVIEW WITA" i pierwszą czynnością, którą należy wykonać jest naciśnięcie klawisza by program pracował dalej — czynność, która nie ma sensu. Prawdopodobnie IBM chciał w ten sposób zamanifestować komputerową życzliwość...

OKNA i TOPVIEW wymagają pewnych informacji o realizowanych programach użytkowych. Informacje te są, za pomocą specjalnych edytorów, rejestrowane w Program Information File (PIF).

TOPVIEW ma kłopoty z innymi programami niż ze stworzonymi przez IBM. Np. GEM i OKNA doskonale dają sobie radę z LOTUSem 1-2-3, natomiast TOPVIEW jest tu nieużyteczny. Nawet w LOGO interpreter IBM wprowadza bałagan na ekranie, praktycznie uniemożliwiający pracę.

Zasygnalizowane prace MICROSOFTa, MACINTOSHa, DIGITAL RESEARCH, a także IBMa, bez względu na już uzyskane efekty, są objawem stosunkowo nowego kierunku w informatyce — doskonalenie kontaktu człowiek — komputer. Często efekty tych prac nazywane są interfejsem użytkownika, interfejsem graficznym „łatwe w użyciu". Jednak najtrudniej tendencję tę oddaje określenie: ŻYCZLIWOŚĆ KOMPUTERÓW

Opracował: Włodzimierz GOGOŁEK

Pocztowa giełda

Rozwiązanie krzyżówki z „IKS-a" nr 5

Hasło brzmi: IKS UCZY, BAWI, POMAGA". Bony pieniężne (1000 zł) wylosowali: Tomasz Dąbrowski — Słupsk, Henryk Duklas — Zgierz, Jolanta Mężyk — Ustroń, Dorota Niezgoda — Sandomierz, Artur Pacholec — Przemyśl.

Nagrody książkowe otrzymują: Magdalena Swatowska — Warszawa, Przemysław Krzak — Koziegłowy, Arek Klimczak — Leśna, Krzysztof Pielarz — Janów Lubelski, Andrzej Eckersdorf — Gorzów Wlkp., Andrzej Ciszek — Gliwice, Janina Sajdak — Warszawa, Paweł Nowak — Poznań, Ewa Sokoll — Bytom, Bartosz Szpunt — Koszalin.

UWAGA — KONKURS!

Miłośnicy literatury science fiction z oryginalnymi pomysłami i fantazją! Każdy z Was ma szansę spróbować swych sił w naszym konkursie na nowelę lub opowiadanie pod hasłem „Przed nami XXI wiek".

„Najfantastyczniejsze" utwory zostaną uhonorowane przez ziemskie jury, które przyzna nagrody:

I — 15 tys. złotych
II — 10 tys. złotych
III — 5 tys. złotych

Nad ich sprawiedliwym i obiektywnym podziałem czuwać będzie komputer. Autorzy materiałów zakwalifikowanych do druku zostaną ponadto usatysfakcjonowani godziwym honorarium zgodnie z obowiązującymi stawkami.

Entuzjastów informatyki i komputerów nie musimy przekonywać, że i fantazja musi mieć swoje ścisłe parametry. A więc mile widziane będą prace o objętości nie przekraczającej 15 stron maszynopisu bądź czytelnego rękopisu. Jak będzie wyglądał nasz glob opanowany przez komputery n-tej generacji, w jaki sposób informatyka zmieni życie człowieka XXI wieku, jak homo sapiens ułoży sobie koegzystencję z rozumnymi maszynami? To tylko niektóre sugestie organizatorów dotyczące tematyki nadesłanych utworów.

Prace opatrzone godłem należy nadsyłać do końca grudnia, dołączając kopertę podpisaną godłem, zawierającą dane o autorze, pod adresem:

„IKS" — ul. Grzybowska 77
00-950 W-wa

Przychylności weny, zdyscyplinowanej fantazji — życzy **REDAKCJA**

PS Zastrzegamy sobie prawo pierwszeństwa publikacji prac nagrodzonych oraz wybranych spośród nie nagrodzonych.

W naszym komputerlandzie

W naszym komputerlandzie rozsiano ostatnio plotki, że według wszelkich reguł prawdopodobieństwa istnieją duże szanse, iż nowy rok będzie nosił numer 1987. Zaniepokojony tymi doniesieniami pracowity Spektruś zwołał kierowany przez siebie zespół robotów na roboczą naradę. Najstarsze i najzacniejsze roboty wypowiedziały się więc konstruktywnie na interesujący wszystkich temat.

Pierwszy robot swoją wypowiedź utrzymał w tonie optymistycznym, stwierdzając jednak, że byłby za ustaleniem numeru 1975, gdyż w komputerlandzie uzyskano właśnie poziom techniki komputerowej przewidzianej w planie na ten rok.

Robot drugi krytycznie ustosunkował się do tej kwestii, gdyż jego zdaniem nie ma sensu dyskutować o roku 1987, skoro nie wyjaśniona jest jeszcze sprawa roku 467 przed naszą erą.

Trzeci dyskutant proponował, by w komputerlandzie przeprowadzić badania ankietowe, które zadecydują o tym czy będzie to rok 1987 czy jakiś inny. Badania najlepiej byłoby przeprowadzić w III kwartale roku przyszłego, a wyniki po opracowaniu przez komputer byłyby znane najdalej pod koniec I półrocza roku następnego.

A czwarty mówca był za wyrównaniem wszystkich cyferek i ustaleniem na przykład numeru 2222. Jeśli bowiem nawet jakiś spekulant będzie próbował sztuczek czytania numeru od końca, to i tak wyjdzie mu to samo, zaś numer 1987 czytany od końca daje 7891, a to już wielka przesada i w ogóle brak wyobraźni.

W swoim podsumowującym słowie Spektruś zauważył, że wstępne rozeznanie pozwala założyć, że numer następnego roku będzie pochodną funkcji roku poprzedniego według wzoru:

$$N = R^2 + \sqrt{(X - Y_z)} + n$$

co zostanie obliczone, jak tylko zespół robotów połapie się, o co chodzi. W porównaniu jednak do roku 1986, rok następny powinien wzrosnąć o około 0,050352 procenta. Dla porównania — jest to wzrost znaczący, zwłaszcza w zestawieniu z latami poprzednimi...

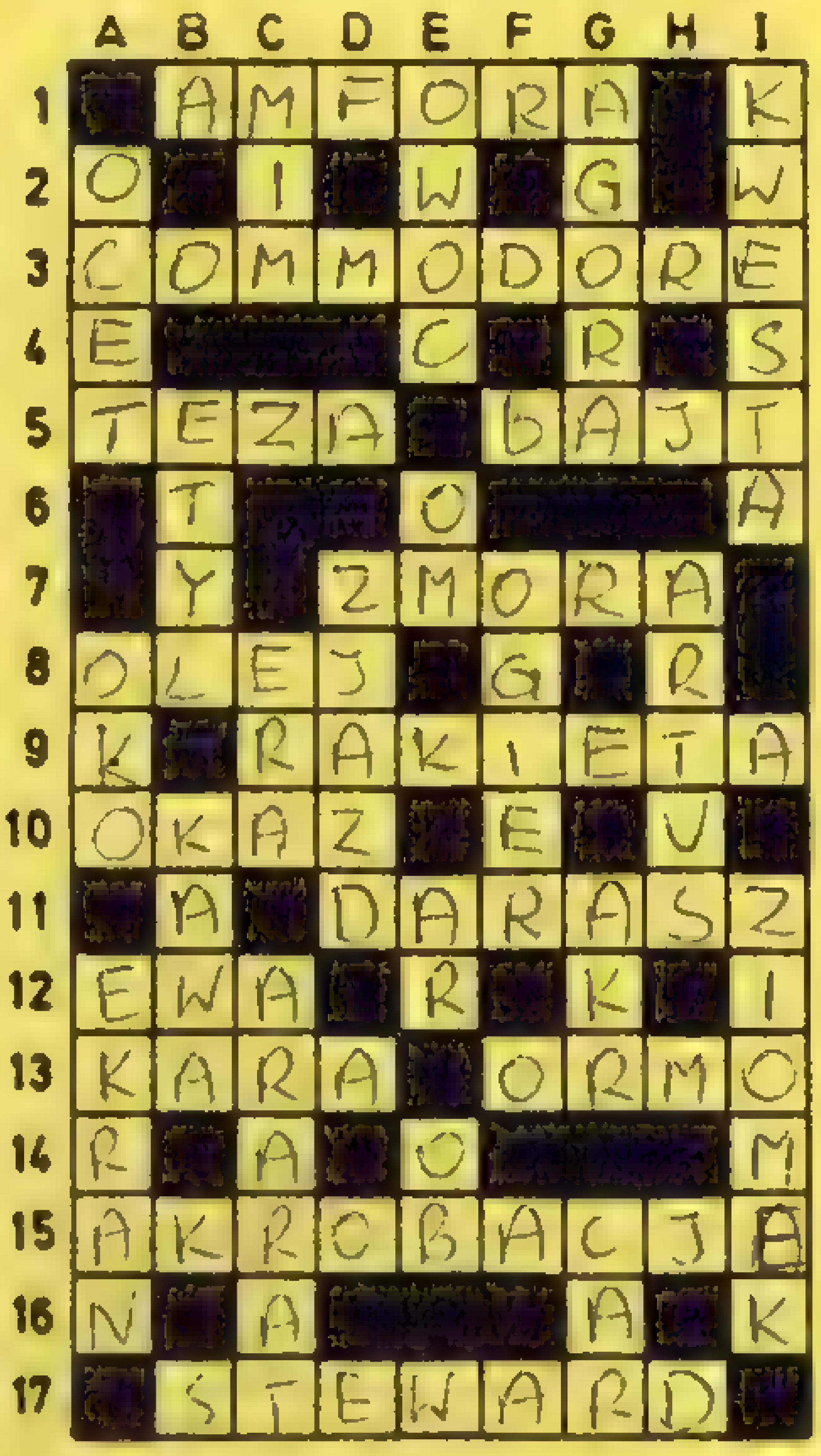
Podglądał:
Eugeniusz MLECZAK



Nie wiemy, kiedy na rynku ukaże się ELWRO 800 Junior, ale na pewno w następnym numerze „IKS-a" znajdziecie:

- *kolejny odcinek Basica*
- *„Jak to zrobić" cz. 2*
- rozwiązania zadań z „Ligi Myślących"
- oraz wiele innych ciekawych informacji

Krzyżówka nr 7



POZIOMO: 1-B naczynie lub tytoń 3-A jeden z rodziny komputerów, 5-A założenie, 5-F jednostka pamięci, 7-D może być senna, 8-A silnikowy lub roślinny, 9-C ...tenisowa 10-A coś rzadko spotykanego, 11-D działacz emigracyjny czł. Młodej Polski, 12-A rajska dziewczyna, 13-A w tytule powieści F. Dostojewskiego, 13-F rezerwa MO, 15-A popisy cyrkowe, 17-B w załodze statku.

PIONOWO: 1-C nietypowy aktor, 1-E jabłko, 1-G rynek w starożytnej Grecji, 1-I akcja na cele dobroczynne, 2-A uzyskuje się go z drzewa lub owoców, 5-B C_2H_5, 6-E jednostka oporu elektrycznego, 7-D złot, 7-F koń, 7-H jego imieniem nazwano Dwór w Gdańsku z XIV w., 8-A pańskie konia tuczy, 8-C okres w dziejach Ziemi, 10-B mała czarna, 11-E 0,01 ha, 11-G jednostka powierzchni, 11-I rodak, 12-A tytuł czasopisma, 12-C wulkaniczny masyw na Wyż. Armeńskiej, 14-E rzeka w ZSRR, 15-G król w Bułgarii.

HASŁO: 3—A, 5—C, 7—B, 9—H, 16—G, 15—H, 9—F, 15—B, 4—I, 9—D.

ROZWIĄZANIA (tylko hasło) prosimy przesyłać pod adresem redakcji do końca grudnia. Wśród czytelników, którzy nadeślą prawidłowe rozwiązania rozlos[illegible]my bony pieniężne i nagrody książkow[illegible]

Mikrociekawostki



Nowa japońska propozycja — SHARP MZ-5600 to cała seria 16-bitowych mikrokomputerów (MZ-5631, MZ-5641, MZ-5645) zbudowanych na procesorze 8086-2 (8 MHz). Pamięć RAM ma standardowo 256 KB z możliwością rozbudowy do 512 KB. Komputery tej serii mogą być wyposażone w dysk sztywny o pojemności 10 MB. Standardowym systemem operacyjnym na MZ-5600 jest MS-DOS, a językiem programowania oczywiście BASIC.



Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8241. Nr ind. 382809 P-67

INFORMATYKA KOMPUTERY SYSTEMY

CENA — 50 zł

Dodatek „Żołnierza Wolności" nr 8/1986 ISSN 0860—2794



Za parę tygodni rozpoczniemy nowy rok bardziej dojrzali. Osiem wydanych numerów „IKS-a" zaświadcza, że wyszliśmy już z wieku niemowlęcego. Podsumowania jednak nie będzie, bowiem jaki jest nasz miesięcznik Czytelnicy wiedzą doskonale. Liczymy się z ich zdaniem, a pomysły, dobre oczywiście, starannie wykorzystujemy. Łamy „IKS-a" dla wszystkich, którzy mają coś ciekawego do powiedzenia, stoją otworem. Otrzymujemy wiele listów, szczególnie serdecznie staramy się przyjmować gości w naszej redakcji. Jesteśmy zapraszani na wiele mikrokomputerowych imprez.

Zainteresowanie mikrokomputerami nie słabnie. Sprzętu jest coraz więcej. Przybywa też wiadomości o jego możliwościach. Już dziś zdarza się, że kilkunastoletni młodzieńcy zawstydzają dojrzałych profesjonalistów. Nadal jednak jesteśmy na początku drogi. Informatyka porwała wielu, niestety dla znacznej części młodzieży stała się również przykrym obowiązkiem.

Niestety bywa i tak. Kiedy w Warszawie rozpoczynała się wystawa Mikro Expo wokół Muzeum Techniki hulał mroźny, listopadowy wiatr. Nie przeszkadzało to jednak domorosłym informatykom w uformowaniu cierpliwie stojącej kolejki. Tak było w sobotę i niedzielę (8 i 9 listopada). Od wtorku na wystawę przychodziły szkoły. *„I co my tu będziemy robić przez te pół godziny"* pytali najbardziej znudzeni. Prawda, co mieli robić, kiedy tylko nieliczni wiedzieli po co tu przyszli. W ten sposób już wkrótce okazać się może, że jesteśmy na dobrej drodze do zagłaskania informatyki, aż do obrzydzenia.

Jeszcze raz trzeba sobie zdać sprawę z tego, że mikrokomputer jest tylko narzędziem. Porównać go można do samochodu. Proszę bardzo: posiadanie prawa jazdy nie jest czymś nadzwyczajnym, trzeba znać zasady ruchu, umieć wykonać podstawowe czynności techniczne, ot chociażby zmienić koło — to wystarczy. Znakomita większość kierowców nie „grzebie" w silniku, poważniejsze naprawy pozostawiając specjalistom.

Z komputerem jest tak samo, dokładnie tak samo: trzeba poznać zasady dialogu z tym urządzeniem i umieć go uruchomić. A środek pozostawmy specjalistom. Pamiętajmy, że z komputerem trzeba się oswoić, bo większość użytkowników nie będzie w przyszłości pisać programów! Będzie je jedynie eksploatować. Tak, tak. Ucząc zatem podstaw nie wymagajmy gruntownej wiedzy technicznej. Przyszły użytkownik nie musi wiedzieć jak działa mikroprocesor, nie jest mu potrzebna wiedza o układach logicznych, tak jak kierowcy nie jest potrzebna znajomość na przykład ilości zębów w przekładni głównej — bo i po co!

Nie straszmy zatem informatyką, pokażmy jedynie jej możliwości. I zapamiętajmy, że tylko nieliczni będą genialnymi konstruktorami i wspaniałymi programistami — większość pójdzie tylko ich śladem.

Pokonywaniu tych właśnie barier, niechęci, niezrozumienia, niemalże przestrachu chcemy służyć i nadal liczymy na Waszą pomoc.

Wszystkim Czytelnikom Przyjaciołom i Sympatykom życzymy udanego startu w nowy rok.

Redakcja

NOWINKI

Firma PHILIPS SUBSYSTEMS and PERIPHERALS z Nowego Jorku pracuje nad nowym typem optycznego dysku typu „pisz raz, czytaj wiele razy" (write once read many) o pojemności 500 Megabajtów. Będzie on kompatybilny z CD ROM (compact disk read only memory). Philips nazwał te nowe dyski CD-PROM (compact disk programmable read only memory)

* * *

Holenderska firma DOC-data planuje wykonać optyczny mechanizm taśmowy, który będzie rejestrował 1 Terabajt (1000 Gigabajtów danych). Urządzenie to nazwano DOCwheel, wykorzystuje ono 128 optycznych kaset (optical cassettes) o rozmiarach standardowych audio-kaset (4,4x2,2x0,6 cala) Każda kaseta ma pojemność 8 Gigabajtów. To nadzwyczajne urządzenie będzie dostępne w 1987 roku

* * *

Natomiast już dzisiaj można kupić BACKUP Tape System firmy IRWIN Jest to pamięć taśmowa o pojemności 40 Megabajtów. Informacje rejestrowane są na kasecie typowej wielkości. Można ją schować do kieszeni marynarki. Dzięki autokorekcji błędów gwarantowana jest poprawność odczytu bez konieczności weryfikacji zapisanych na kasecie informacji. Proponowane (oczywiste) zastosowanie — składowanie twardych dysków 10, 20 i 40 Megabajtowych.

* * *

KURZWELL zaprezentował foniczny terminal, który umożliwia wykorzystanie wypowiadanych słów i fraz do sterowania, wprowadzania i obierania danych z mikrokomputerów. System ten automatycznie „przekłada" słowa komend i dane na postać wymaganą przez wejście w komputerze stosowane dla typowej klawiatury. Innymi słowy można wykorzystać klawiaturę lub mikrofon, ekran lub głośnik.

* * *

Firma MITSUBISHI proponuje tani (395 dolarów) digitizer o aktywnej powierzchni tablicy 21x30 cm. Digitizer umożliwia operowanie z szybkością 60 punktów na sekundę i rozdzielczością 0,1 mm. Tablica współpracuje z AutoCAD, P-CAD, PC-Paint itp. (Digitizer jest urządzeniem wejściowym mikro, które, podobnie jak myszka, zmienia położenie znacznika na ekranie monitora. Miejsce stołu zajmuje wspomniana tablica, myszki — „pióro")

* * *

HEWLETT-PACKARD wyprodukował maszynę kompatybilną z IBM PC/AT, która może pracować w języku angielskim lub jednym z czterech azjatyckich: japońskim, koreańskim, chińskim lub uproszczonym chińskim. Maszyny te pracują w oparciu o procesor 80286. Dołączona jest bogata oferta uzupełniająca (RAM, floppy 1,2 Mb, oprogramowanie). Wybierając wersję japońską (3,1) MS-DOS staje się systemem operacyjnym w języku japońskim. W ROM-ie zapisano około 7000 japońskich znaków. Znaki Kanji są wyświetlane na ekranie jako matryce 24x24 pixele.

* * *

Pojemność typowego optycznego dysku wynosi 1 Gigabajt. Jest to ekwiwalentem około 400 000 stron A4, 500 do 1000 obrazów czarno-białych, 2000 mikrofisz, 8000 m taśmy magnetycznej FUJITSU twierdzi, że to dopiero początek, bowiem propozycja tej firmy to 12-calowy dysk 3 Gigabajty z możliwością zapisu i odczytu informacji Póki co można kupić CM100 Philipsa 0,6 Giga (115x320x267 mm — 5 kg)

W.G.

POLSKIE ZNAKI NA ZX SPECTRUM

Wszystkie programy, które powstają z myślą o polskim użytkowniku powinny być wyposażone w cały zestaw liter polskiego alfabetu. Dotyczy to szczególnie programów dydaktycznych, programów przetwarzających teksty oraz programów, w których występuje lista nazwisk lub dowolnych innych nazw. Stosowanie liter polskiego alfabetu w programach podyktowane jest nie tylko wymaganiami dydaktycznymi czy formalnymi. To właśnie język przez długie lata był ostoją naszej narodowości.

Żaden z zachodnich mikrokomputerów nie jest wyposażony we wszystkie litery polskiego alfabetu. Nie występują bowiem polskie litery diakrytyzowane, takie jak Ą, ą, Ć, ć, Ę, ę itd

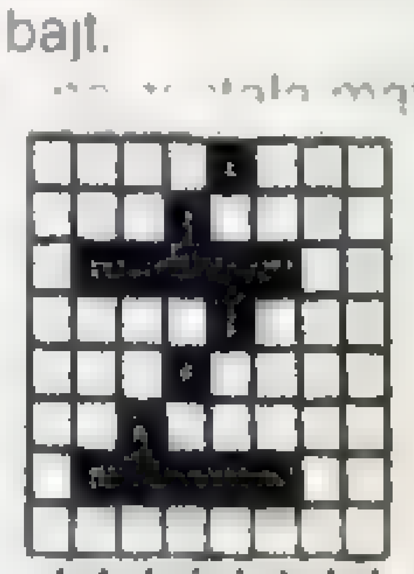
W mikrokomputerze ZX Spectrum można stosunkowo łatwo zdefiniować dowolny znak, w szczególności więc również i polskie litery diakrytyzowane. Łącznie zdefiniować można 21 znaków, które dostępne są w trybie graficznym na klawiszach A—U. Definiując znak, trzeba rozstrzygnąć dwie kwestie

— kształt znaku
— przyporządkowanie znaku do jednego z klawiszy A—U.

Wszystkie znaki definiowane są w matrycy 8X8 pól (pikseli). Kazdy rząd składający się z ośmiu pikseli może być [illegible]

oznacza pole zaczernione (przy załozeniu pisma pozytywowego, tj. trybu INVERSEØ)a Ø pole nie zaczernione. Liczby te wpisywane są następnie za pomocą instrukcji POKE w tzw. obszar UDG (user defined graphics) pod adresem wybranego klawisza A—U. Łatwo zauważyć, że jeden tak zdefiniowany znak zajmuje 8 bajtów pamięci, podczas gdy wszystkie znaki i słowa kluczowe dostępne wprost z klawiatury zajmują 1 bajt.

Ponizej przedstaw[illegible] matryca znaku „ź":



żżżżżżżż

```
10 REM definiowanie znaku
20 POKE USR "g",BIN 00001000
30 POKE USR "g"+1,BIN 00010000
40 POKE USR "g"+2,BIN 01111100
50 POKE USR "g"+3,BIN 00001000
60 POKE USR "g"+4,BIN 00010000
70 POKE USR "g"+5,BIN 00100000
80 POKE USR "g"+6,BIN 01111100
90 POKE USR "g"+7,BIN 0
```

[illegible] pętli FOR, nalezy jednak przeliczyć liczby binarne na dziesiętne. Stosowny fragment programu definiujący znak „ź" wyglądałby wówczas następująco:

```
10 REM definiowanie znaku
20 LET k$="g"
30 FOR i=0 TO 7
40 READ a
50 POKE USR k$+i,a
60 NEXT i
70 DATA 8,16,124,8,16,32,124,0
```

Tak zdefiniowany „ź" dostępny jest w trybie graficznym po naciśnięciu klawisza „g"

W związku z dużym upowszechnieniem mikrokomputera ZX Spectrum celowe byłoby ujednolicenie przyporządkowania poszczególnych liter diakrytyzowanych do klawiszy A—U. O ile dla małych liter ą, ć, ę itd. panuje zgodność — gdyż przyporządkowuje się je do klawiszy odpowiednio A, C E — o tyle dla dużych liter zgodności takiej nie ma. Ponadto pewien chaos wprowadzają znaki „ź", „Ź", „ż", „Ż", które nie mieszczą się w przedziale A—U. Najlepszym kryterium dla przyporządkowania znaków do klawiszy jest sąsiedztwo w alfabecie (porządek alfabetyczny) oraz — dla znaków „ź" i „ż" — skojarzenie z wymową francuską

Należy podkreślić, że w ASCII kody znaków zdefiniowanych przez UDG są większe od kodów duzych i małych liter alfabetu angielskiego, co utrudnia leksykograficzne porządkowanie słów. Problem mozna rozwiązać stosując umiejętne kodowanie, np. nazwisko „Ślusarczyk" powinno być zapisane w komputerze jako „S!lusarczyk". Przy wprowadzaniu danych trzeba analizować pierwszy znak i — jeśli jest to znak z zakresu UDG — przeprowadza się kodowanie. Przy wyprowadzaniu danych (na ekran lub drukarkę) trzeba analizować drugi znak nazwy i — jeśli jest to znak uzyty do kodowania (!), dwa pierwsze znaki nazwy zastępuje się odpowiednim znakiem z zakresu UDG. Kodowanie i dekodowanie powinno być realizowane przez odrębne podprogramy. Znak użyty do kodowania powinien być tak dobrany, aby jego kod był zawarty pomiędzy kodami dużych i małych liter. Jest to niezbędne dla zapewnienia prawidłowego sortowania. Warunek ten spełnia np. znak !

Alfabet polski zawiera 18 liter diakrytyzowanych (małych i duzych razem). Ponieważ obszar UDG pozwala na zdefiniowanie 21 znaków, pozostałe 3 znaki mozna wykorzystać do celów specjalnych (np. bardzo przydatna jest kreska do podkreśleń tekstu)

Znaki raz zdefiniowane zgodnie z przedstawionym opisem należy zapisać na taśmie za pomocą instrukcji

```
SAVE "Znaki-pol" CODE 65368,168
```

i uzywać w kazdym programie, ładując je instrukcją

```
LOAD "Znaki-pol" CODE
```

lub

```
LOAD "" CODE
```

dokończenie na str. 6











MIKRO EXPO'86



Niebywałym powodzeniem na wystawie MIKRO EXPO'86 cieszył się „IKS" (a szczególnie wcześniejsze jego numery). Było również na co popatrzeć...

Foto: Jan Zelman

PROGRAM

„ZGADYWANKA" (ATARI) jest komputerową wersją popularnej gry „Master mind". Jest trudniejsza od pierwowzoru, polega bowiem na odgadnięciu liczby z przedziału od 0000 do 9999. Może w niej brać udział do 10 graczy, można zatem urządzić turniej logicznego myślenia.

Istnieją dwa poziomy gry; łatwy, dla początkujących, w którym każda cyfra liczby jest inna oraz trudny, dla zaawansowanych, w którym mogą wystąpić powtarzające się cyfry.

UWAGA: W linii 110 należy wpisać w cudzysłowach po B $ i C $ następujące znaki graficzne, które nie zostały wydrukowane, odpowiednio CTRL P oraz CTRL T. Te same znaki należy wpisać w nawiasy w linii 1450, w takiej samej kolejności.

```
90 REM << ZGADYWANKA >>
100 DIM T(10),C(10),Z(2,4),K(4)
105 DIM O$(1),B$(2),C$(2),S(4)
110 B$=" ":C$=" "
115 OPEN #1,4,0,"K:"
117 FOR I=1 TO 10:T(I)=0:NEXT I
120 GOSUB 1000:REM STRONA TYTUL
OWA
400 FOR P=1 TO Q
405 POKE 752,0
410 GRAPHICS 0:? "   UWAGA GRA
CZ ";P;", WYBIERAM LICZBE.":? :
? :?
420 FOR I=1 TO 6:C(I)=0:NEXT I
430 N=0:FOR I=1 TO 4
440 N1=INT(10*RND(0)):IF A9=1 T
HEN 470
450 IF C(N1+1)=1 THEN 440
460 C(N1+1)=1
470 N=N+(N1*(10^(I-1))):NEXT I
480 N1=N:T5=0
490 T1=INT(N/1000):N=N-(T1*1000
)
500 H1=INT(N/100):N=N-(H1*100)
510 E1=INT(N/10):N=N-(E1*10)
520 Z(1,1)=T1:Z(1,2)=H1:Z(1,3)=
E1:Z(1,4)=N
530 IF T5<>20 THEN 550
540 ? :? "ZBYT WIELE PROB, SZUK
ANA LICZBA: ";N1:GOTO 720
550 POSITION 1,22:? "PODAJ SWOJ
A LICZBE: "
560 FOR I=1 TO 4:S(I)=0:K(I)=0:
NEXT I:X1=0:T5=T5+1
570 FOR KL=5 TO 11 STEP 2
580 GOSUB 950
590 IF KL=5 THEN T=C:POSITION K
L,T5:? T:Z(2,1)=T
594 IF KL=7 THEN H=C:POSITION K
L,T5:? H:Z(2,2)=H
598 IF KL=9 THEN E=C:POSITION K
L,T5:? E:Z(2,3)=E
600 IF KL=11 THEN O=C:POSITION
KL,T5:? O:Z(2,4)=O
605 NEXT KL
610 G=T*1000+H*100+E*10+O
615 POSITION 1,22:? "
620 POSITION 18,T5:POKE 752,1
622 FOR I=4 TO 1 STEP -1
624 IF Z(2,I)=Z(1,I) THEN ? B$;
:K(I)=2
626 NEXT I
628 IF G=N1 THEN 710
630 X=INT(4*RND(0))+1:IF S(X)=1
 THEN 630
640 S(X)=1:FOR L=1 TO 4:IF Z(2,
X)<>Z(1,L) THEN 700
645 IF K(L)=2 THEN 700
650 IF K(L)=1 THEN 700
660 K(L)=1:IF X<>L THEN 680
670 POKE 752,1:? B$;:GOTO 690
680 POKE 752,1:? C$;
690 L=5
700 NEXT L:X1=X1+1:IF X1=4 THEN
 530
705 GOTO 630
710 ? :? :? :? :? "      BARDZO
 DOBRZE, ODGADLES LICZBE W    ";
T5;" RUCHACH."
715 FOR I=1 TO 600:NEXT I
720 ? :? :T(P)=T(P)+T5:NEXT P:R
=R+1
730 ? :? "       PO ZAKONCZENIU R
UNDY ";R:?
740 ? "GRACZ","RAZEM","SREDNIA"
,"STOPIEN"
750 ? :FOR I=1 TO Q
760 ? " ";I," ";T(I),:T9=INT(
(T(I)/R)*100)/100:? " ";T9,
770 GOSUB 830
780 NEXT I
790 ? :? :?
800 ? "NASTEPNA RUNDA? (T/N) ";
:INPUT O$
810 IF O$="T" THEN 400
815 IF O$<>"T" AND O$<>"N" THEN
 800
820 STOP
830 T9=T9+((A9-1)*4)
840 IF T9<=4 THEN ? "EKSPERT":R
ETURN
850 IF T9<=8 THEN ? "WYJATKOWO"
:RETURN
860 IF T9<=12 THEN ? "DOBRZE":R
ETURN
870 IF T9<=15 THEN ? "SREDNIO":
RETURN
880 IF T9<=18 THEN ? "SLABO":RE
TURN
890 IF T9<=20 THEN ? "BARDZO SL
ABO":RETURN
900 ? "WYMAGA":? "","","","TREN
INGU":RETURN
950 REM PPROGRAM GET
960 GET #1,C:C=C-48
965 IF C=78 THEN KL=KL-2:POSITI
ON KL,T5:? " "
970 IF C<0 OR C>9 THEN 960
980 RETURN
1000 REM STRONA TYTULOWA
1010 DL=PEEK(560)+256*PEEK(561)
1020 GRAPHICS 0:SETCOLOR 2,0,0:
POKE 580,1:POKE 752,2
1030 SETCOLOR 1,0,15:SETCOLOR 0
,2,8:SETCOLOR 3,11,7:POKE 82,0
1040 POKE DL+9,7:POKE DL+13,7
1050 POSITION 5,4:? "zgadywanka
"
1060 POSITION 3,15:? "OPTION -
POZIOM GRY:"
1070 POSITION 3,17:? "SELECT -
ILOSC GRACZY:"
1080 POSITION 3,19:? "START  -
POCZATEK GRY"
1090 POSITION 3,21:? "HELP   -
```

dokonczenie na str. 6

dokończenie ze str. 5

```
INSTRUKCJA"
1100 A9=1:GOSUB 1200
1110 Q=0:GOSUB 1300
1120 W=PEEK(53279)
1130 IF W=3 THEN GOSUB 1200
1140 IF W=5 THEN GOSUB 1300
1150 IF W=6 THEN RETURN
1160 IF PEEK(732)=17 THEN GOSUB
 1400
1170 GOTO 1120
1200 REM POZIOM GRY
1210 A9= NOT A9
1215 FOR I=1 TO 50:NEXT I
1220 IF A9=0 THEN 1240
1230 POSITION 25,15:? "LATWY /
¢öÑã½₽":RETURN
1240 POSITION 25,15:? "ʮà¢ö₽ /
TRUDNY":RETURN
1300 REM ILOSC GRACZY
1305 FOR I=1 TO 50:NEXT I
1310 IF Q=10 THEN Q=0
1320 Q=Q+1:POSITION 30,17:? "
 "
1330 POSITION 30,17:? STR$(Q)
1340 RETURN
1400 REM INSTRUKCJA
1410 POKE 732,0:GRAPHICS 0:POKE
 82,2:POKE 83,39
1420 POSITION 15,1:? "ZASADY GR
Y":? " "
1430 ? "   WYBIERAM LICZBE MIED
ZY 0000-9999.  SPROBUJ ODGADNAC
 TE LICZBE W MINIMAL- NEJ ILOSC
I PROB. ZA KAZDYM RAZ"
1440 ? "NEJ ILOSCI PROB. ZA KAZ
DYM RAZEM DAM  CI WSKAZOWKI JAK
 BLISKI JESTES PO-"
1450 ? "PRAWNEJ ODPOWIEDZI. WSK
AZOWKAMI BEDA  <CZARNE>() I/LUB
 <BIALE>(). JESLI"
1460 ? "POWIEM <BIALE> TO ZNACZ
Y, ZE JEDNA Z  CYFR JEST POPRAW
NA, LECZ NIE JEST NA"
1470 ? "SWOIM MIEJSCU. JESLI PO
WIEM <CZARNE>, WTEDY MASZ POPRA
WNA CYFRE NA SWOIM"
1480 ? "MIEJSCU. POZWALA CI TO
WNIOSKOWAC     JAKIE CYFRY SA P
OPRAWNE."
1490 ? " ":? " ":? "Jesli skonc
zyles czytac, nacisnij    dowo
lny klawisz.":? " ":POKE 764,25
5
1495 IF PEEK(764)=255 THEN 1495
1500 ? " ":? " ":? " ":? " ":?
" ":? " "
1510 ? "  ISTNIEJA DWA POZIOMY
TRUDNOSCI GRY. WERSJA LATWA - D
LA POCZATKUJACYCH."
1520 ? "NIE BEDZIE MIEC TYCH SA
MYCH CYFR UZY- TYCH DWA RAZY W
LICZBIE. WERSJA DLA"
1530 ? "ZAAWANSOWANYCH MOZE MIE
C NAWET CZTERY TAKIE SAME CYFRY
."
1540 ? "   MOZE GRAC OD 1 DO 10
 GRACZY."
1550 ? " ":? " ":? " ":? "*****
*****  POWODZENIA!  **********"
1560 ? " ":? " ":? " ":? "   Je
sli skonczyles czytac, nacisnij
  dowolny klawisz."
1570 POKE 764,255
1580 IF PEEK(764)=255 THEN 1580
1590 POP :GOTO 1000
```


dokończenie ze str. 3

Jeżeli powyższe instrukcje ładowania będą umieszczone w programie jako pierwsze instrukcje wykonywalne, a znaki zapisane zostaną na taśmie bezpośrednio z programem, możliwe jest automatyczne ładowanie polskiego alfabetu.

Co zrobić gdy w programie potrzebne są dwa lub więcej alfabetów (np. słowniki)? Można zwiększyć obszar UDG korzystając ze zmiennej systemowej o tej samej nazwie (adresy 23675 i 23676) lub wykorzystać części wspólne znaków. Jeżeli np. zdefiniowaliśmy znak „ą", to chcąc uzyskać znak niemieckiego alfabetu „ä", wystarczy zlikwidować ogonek u dołu znaku i dopisać kropki u góry. Częścią wspólną, którą można wykorzystać dla obu liter jest tu znak „a". Zamiana znaku „ą" na „ä" odbywa się dzięki

```
POKE USR "a"  ,BIN 00101000
POKE USR "a"+7,BIN 0
```

```
POKE USR "a"  ,40
POKE USR "a"+7,0
```

Przejście odwrotne od „ä" do „ą" polegające na usunięciu kropek u góry znaku i dopisaniu ogonka u dołu wykonują instrukcje

```
POKE USR "a"  ,BIN 0
POKE USR "a"+7,BIN 00000010
```

lub

```
POKE USR "a"  ,0
POKE USR "a"+7,2
```

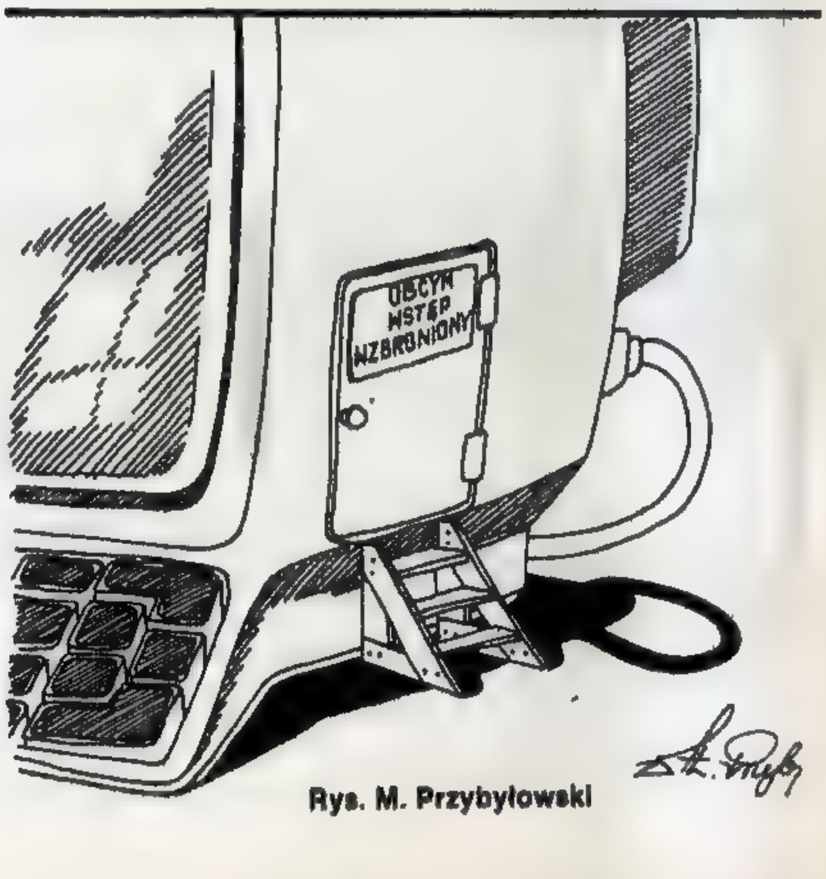
Oczywiście obydwa znaki „ą" i „a" dostępne są w trybie graficznym na klawiszu „a", przy czym w danej chwili dostępny jest tylko jeden z nich. Powyższy sposób „przeładowania" alfabetów jest bardzo szybki i prosty. Przeładowanie całego alfabetu wymaga zaledwie kilku instrukcji POKE i może być zrealizowane przez podprogram.

Kończąc pragnę zachęcić Czytelników do korzystania w swoich programach ze wszystkich liter polskiego alfabetu. Pomni nawoływania Mikołaja Reja

„niechaj narodowie wżdy postronni znają
iż Polacy nie gęsi, iż swój język mają"

dbajmy o poprawną polszczyznę we własnych programach, które wykonywane przez komputer — pisane są przecież — dla ludzi.

Janusz MORBITZER



Rys. M. Przybyłowski

PROGRAM 39

SPECTRUM

CYKLOIDA

Krąg toczy się po prostej. Związany z nim punkt leżący na okręgu pozostawia „ślad" znacząc swą drogę. Efektem tej „pracy" punktu będzie krzywa okresowa o długości łuku równej dla jednego cyklu ośmiu promieniom okręgu. Wystartujmy program. Linie 190... 240 wczytują 107 bajtów tworzących trzy podprogramy w języku wewnętrznym Z80. Nie można bez nich marzyć o poprawnej animacji.

Menu pozwala wybrać 9 wykresów cykloidy w trzech wariantach. „Pracujący" punkt może leżeć na okręgu (cykloida zwyczajna), na zewnątrz (cykloida wydłużona) lub wewnątrz niego (cykloida skrócona). W każdym wariancie możemy obejrzeć trzy charakterystyczne przebiegi podając liczbę pełnych obrotów okręgu.

Ekran podzielony jest na trzy części identycznie, jak to zrobiono firmowo. W dolnej części przesuwany jest okrąg, aby go tam umieścić (zahacza o linie edycyjne) trzeba go narysować w części środkowej (linia 420) i przerzucić w dół (linia 450). Operację wykonuje część programu maszynowego nazwana żartobliwie „desantem okręgu".

Od linii 460 rozpoczyna się główna pętla realizująca wykres. Położenie okręgu w dolnym ekranie jest zsynchronizowane z obliczonym położeniem promienia wodzącego rysowanym na górnym ekranie (linie 480... 510). Teraz obie części ekranu są logicznie sumowane przed kod „nakładanie ekranu" i wyświetlane w części środkowej dając efekt animacji. Okrąg przesuwa się (linie 550... 610) i cały cykl powtarza się. Przemieszczanie okręgu realizuje procedura „scroll prawy". Proces zachodzi co jeden punkt w każdym przebiegu głównej pętli.

Kreślenie krzywej można zatrzymać w każdym momencie dowolnym klawiszem i podobnym wznowić (stopklatka).

Rezygnacja z opisów i wzorów jest świadoma — można je znaleźć w książkach.

K.M.

```
10 REM ** cykloida 86
20 OVER 0: INVERSE 0: FLASH 0: BRIGHT 0
30 CLEAR 59999
40 REM grafiki
50 LET c=16224
60 FOR a=0 TO 23
70 POKE USR "a"+a, PEEK (c+a)
80 IF a=7 THEN LET c=16240
90 IF a=15 THEN LET c=16320
100 NEXT a
110 POKE USR "a"+3,24
120 POKE USR "a"+8,16
130 POKE USR "a"+16,16
140 LET c$=" CYKLOIDA"
150 LET d$="zwyczajna"
160 LET e$="wyd"+ CHR$ 144+"u"+ CHR$ 146
+"ona"
170 LET f$="skr"+ CHR$ 145+"cona"
180 REM kod maszynowy
190 LET a=60000
200 READ r
210 IF r<0 THEN GOTO 250
220 POKE a,r
230 LET a=a+1
240 GOTO 200
245 REM menu
250 BORDER 5: PAPER 5: INK 0: CLS
260 PRINT '''c$''" 1 "+d$''" 2 "+e$''" 3
 "+f$
270 IF PEEK 23560=49 THEN LET r1=0: GOTO
 310
280 IF PEEK 23560=50 THEN LET r1=5: GOTO
 310
290 IF PEEK 23560=51 THEN LET r1=-5: GOT
O 310
300 GOTO 270
310 INPUT AT 22,0;AT 15,4;"liczba cykli
(1..3)"'i
320 LET i=ABS (INT i)
330 IF i<1 OR i>3 THEN GOTO 310
340 REM cykloida
350 BORDER 1
360 PAPER 1: INK 1: CLS
370 FOR a=22784 TO 23039
380 POKE a,14
390 NEXT a
400 POKE 23624,9: INPUT ()
410 LET r=15
420 CIRCLE INK 8;PAPER 8;r,74,r
430 PLOT r,123: DRAW 225,0: DRAW -5,3
440 PLOT r,123: DRAW 0,50: DRAW 3,-5
450 RANDOMIZE USR 60067
460 FOR a=1 TO 225 STEP i
470 IF LEN INKEY$ THEN GOSUB 680
480 LET alfa=a*PI/95*i
490 LET x=-(r+r1)* SIN alfa
500 LET y=-(r+r1)* COS alfa
510 GOSUB 650
520 RANDOMIZE USR 60079
530 GOSUB 650
540 PLOT PEEK 23677,PEEK 23678
550 FOR b=1 TO i
560 POKE 23602,0
570 POKE 23603,80
580 POKE 23604,64
590 POKE 23605,32
600 RANDOMIZE USR 6E4
610 NEXT b
620 NEXT a
630 PRINT INK 9;AT 18,3;c$;"  ";
631 IF NOT r1 THEN PRINT INK 9;d$
632 IF r1>0 THEN PRINT INK 9;e$
```

dokończenie na str. 8

dokończenie ze str. 7

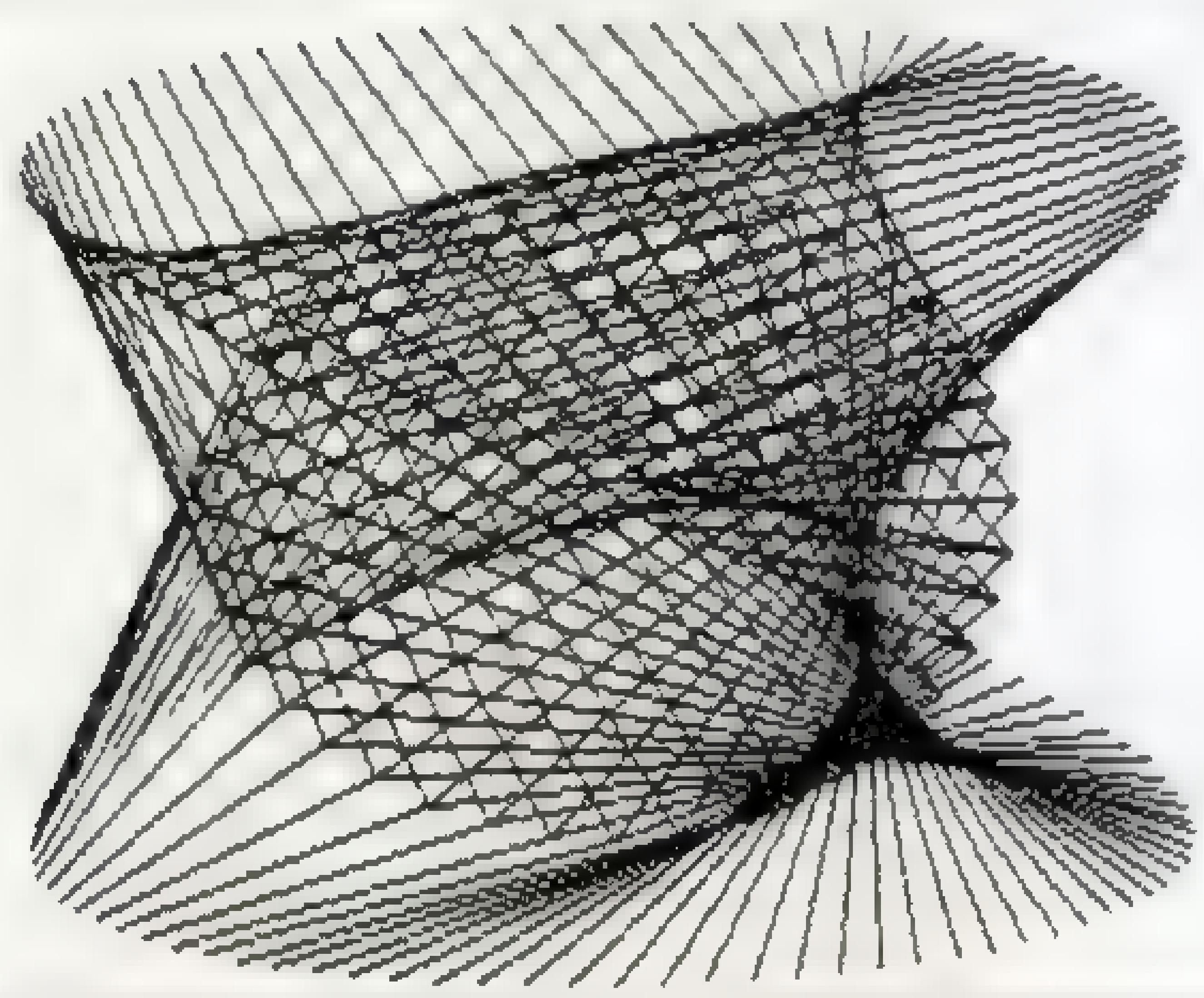
```
633 IF r1<0 THEN PRINT INK 9;f$
634 IF NOT LEN INKEY$ THEN GOTO 634
635 GOTO 350
650 POKE 23677,r+a: POKE 23678,138
660 DRAW OVER 1;x,y
670 RETURN
680 IF LEN INKEY$ THEN GOTO 680
690 IF NOT LEN INKEY$ THEN GOTO 690
700 RETURN
710 DATA 042,050,092,237,075,052,092,167
,203,030,035,005,032,250,042,050,092,205
,131,234,034,050,092,237,075,052,092,013
,200,237,067,052,092,024,228,124,230,007
,254,007,040,002,036,201,125,230,224,254
,224,040,007,017,224,006,167,237,082,201
720 DATA 124,254,087,200,017,032,000,025
,201,033,000,072,017,000,080,001,000,008
,237,176,201,033,000,064,017,000,080,001
,000,008,221,033,000,072,026,182,221,119
,000,035,019,221,035,011,120,177,032,242
,201

 730 DATA -1
```

```
  5 ;scroll prawy
 10                 LD     HL,(23602)
 20                 LD     BC,(23604)
 30 L1              AND    A
 40 L2              RR     (HL)
 50                 INC    HL
 60                 DEC    B
 70                 JR     NZ,L2
 80                 LD     HL,(23602)
 90                 CALL   CH
100                 LD     (23602),HL
110                 LD     BC,(23604)
120                 DEC    C
130                 RET    Z
140                 LD     (23604),BC
150                 JR     L1
160 CH              LD     A,H
170                 AND    #07
180                 CP     #07
190                 JR     Z,L3
200                 INC    H
210                 RET
220 L3              LD     A,L
230                 AND    #E0
240                 CP     #E0
250                 JR     Z,L4
260                 LD     DE,01760
270                 AND    A
280                 SBC    HL,DE
290                 RET
300 L4              LD     A,H
310                 CP     #57
320                 RET    Z
330                 LD     DE,00032
340                 ADD    HL,DE
350                 RET
370 ;desant okregu
390                 LD     HL,18432
400                 LD     DE,20480
410                 LD     BC,2048
420                 LDIR
430                 RET
440 ;nakladanie ekranu
460                 LD     HL,16384
470                 LD     DE,20480
480                 LD     BC,2048
490                 LD     IX,18432
500 LOOP            LD     A,(DE)
510                 OR     (HL)
520                 LD     (IX),A
530                 INC    HL
540                 INC    DE
550                 INC    IX
560                 DEC    BC
570                 LD     A,B
580                 OR     C
590                 JR     NZ,LOOP
600                 RET
```

Grafika (AMSTRAD)

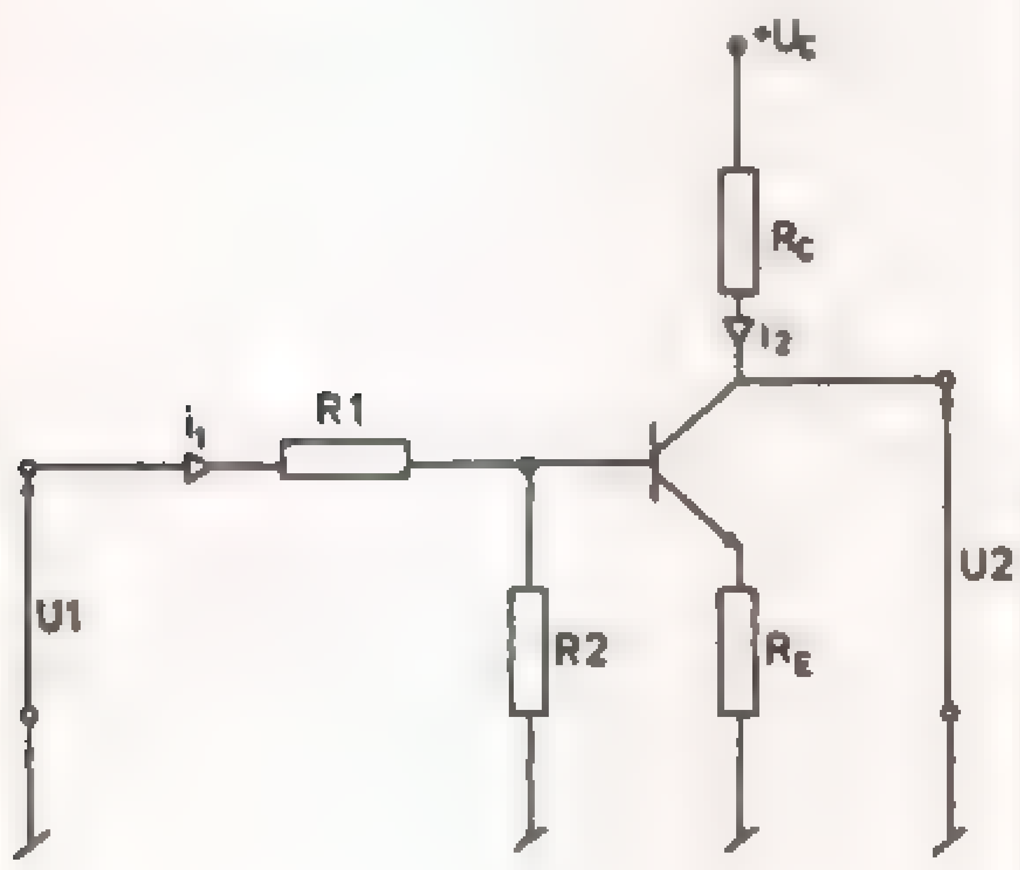
```
10 CLS
20 LET z=0
30 b%=RND*5+1
40 c%=RND*5+1
50 ORIGIN 320,200
60 FOR a=0 TO 100 STEP PI/20
70 x%=140*COS(a)
80 y%=140*SIN(a)
90 MOVE x%,y%
100 DRAW 200*SIN(a/b%),200*COS(a/c%)
110 LET z=z+1
120 IF z=600 GOTO 140
130 NEXT a
140 PRINT "koniec"
```



Logika i arytmetyka w technice cyfrowej

Poprzedni odcinek tego cyklu zapoznał Was z architekturą mikrokomputera i mikroprocesora. Czas przejść do podstaw matematycznych, terminologii i pojęć, których znajomość jest niezbędna każdemu, kto pragnie poznać działanie elementów mikrokomputera.

Podstawowym elementem, używanym do budowy mikroprocesorów i pozostałych elementów systemów mikrokomputerowych jest tranzystor, elektroniczny element półprzewodnikowy o trzech wyprowadzeniach: bazie, emitarze i kolektorze. Tranzystor, przedstawiony na rys. 3., może stabilnie i pewnie pracować w dwóch stanach.



Rys. 3. Tranzystor jako element dwustanowy

1° Gdy na wejście tranzystora, podane zostanie napięcie U_1 bliskie 0 V, wówczas prąd wyjściowy jest prawie równy zeru i napięcie U_2 na wyjściu tranzystora bliskie jest napięciu U_c.
Tranzystor nie przewodzi, jest rozwarty.
2° Gdy na wejście podane zostanie napięcie bliskie U_c, wówczas prąd wyjściowy jest bardzo duży, a napięcie U_2 bliskie jest 0 V.
Tranzystor jest nasycony, zwarty.

Dwa powyższe stany, oznaczone umownie 1 **(jedynka logiczna)** i 0 **(zero logiczne)** oraz to, że ludzie mają tendencję do używania zwrotów **prawda** (1) lub **fałsz** (0) powoduje, iż naturalnym systemem liczenia w komputerach jest **dwójkowy system liczbowy.**

Cała informacja dla mikroprocesora oraz innych elementów systemu mikrokomputerowego przedstawiona jest jako informacja dwójkowa, za pomocą wielkości fizycznych, zwanych **sygnałami dwuwartościowymi (dwustanowymi).** W danym systemie można używać sygnału 3 V (z tolerancją, 1 V) jako logicznej jedynki (1) i sygnału 0,5 V (z tolerancją, 0,5 V) jako logicznego zera (0).

Systemy elektroniczne, zbudowane z elementów (układów) dwustanowych, nazywane są **systemami cyfrowymi.** Układy dwustanowe, inaczej cyfrowe, na swoich wejściach akceptują sygnały dwuwartościowe o określonej tolerancji. Na wyjściach tych układów pojawiają się również sygnały dwuwartościowe.

Obróbkę **informacji dwójkowych** można prowadzić za pomocą układów cyfrowych zwanych **bramkami logicznymi.** W tabeli 1 przedstawiono osiem bramek wykonujących operacje logiczne. Operacje negacji, sumy logicznej i iloczynu logicznego, trzy podstawowe operacje logiczne, łatwo można przeprowadzić za pomocą układów tranzystorowych. Do budowy układów scalonych, realizujących powyższe operacje, układów popularnie zwanych bramkami, wykorzystano właśnie tranzystory.

Operacjami logicznymi i zmiennymi dwójkowymi zajmuje się **algebra Boole'a.** Ze znanych wam już symboli operacji logicznych oraz ze zmiennych dwójkowych, nawiasów i znaku równości można utworzyć wyrażenie algebraiczne, funkcję boolowską, która dla danej wartości zmiennych przyjmuje wartość 1 lub 0.

Daną funkcję boolowską, inaczej **funkcję logiczną,** można przedstawić za pomocą układu złożonego z bramek logicznych.

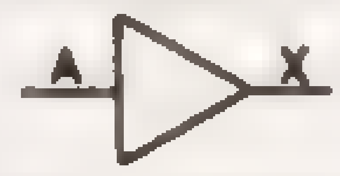
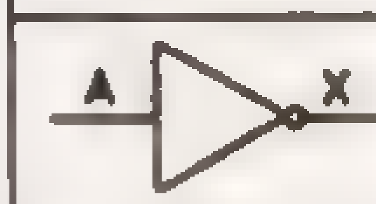
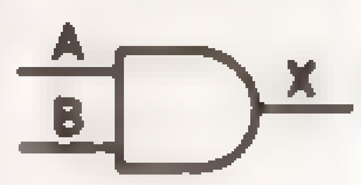
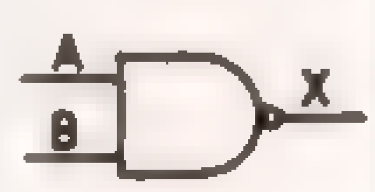
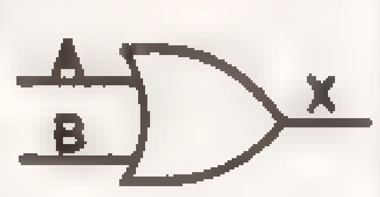
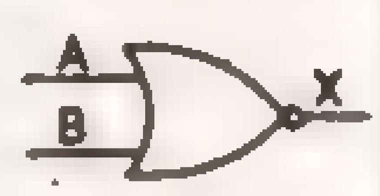
symbol graficzny	nazwa bramki	funkcja algebraiczna nazwa operacji	tablica funkcji
	bufor	$X = A$	A X 0 0 1 1
	inwerter NOT	$X = \bar{A}$ negacja (dopełnienie)	A X 0 1 1 0
	AND	$X = A \cdot B$ iloczyn logiczny	A B X 0 0 0 0 1 0 1 0 0 1 1 1
	NAND	$X = \overline{A \cdot B}$ zanegowany iloczyn logiczny	A B X 0 0 1 0 1 1 1 0 1 1 1 0
	OR	$X = A + B$ suma logiczna	A B X 0 0 0 0 1 1 1 0 1 1 1 1
	NOR	$X = \overline{A + B}$ zanegowana suma logiczna	A B X 0 0 1 0 1 1 1 1 0 1 1 0
	EXCLUSIVE-OR	$X = A \oplus B$	A B X 0 0 0 0 1 1 1 0 1 1 1 0
	EXCLUSIVE-NOR	$X \quad A \odot B$ równoważność	A B X 0 0 1 0 1 0 1 0 0 1 1 1

TABELA 1 BRAMKI LOGICZNE

dokończenie na str. 10

dokończenie ze str. 9

Dobra znajomość poniżej przedstawionych podstawowych zależności algebry Boole'a, pozwala na otrzymywanie prostszych funkcji logicznych, realizowanych za pomocą mniejszej ilości bramek

TABLICA 1

PODSTAWOWE ZALEŻNOŚCI ALGEBRY BOOLE'A

(1) $a \cdot 0 = 0$

(2) $a \cdot 1 = a$

(3) $a \cdot a = a$

(4) $a \cdot \bar{a} = 0$ */

(5) $a \cdot b = b \cdot a$

(6) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

(7) $a \cdot (b + c) = a \cdot b + a \cdot c$

(8) $a + 0 = a$

(9) $a + 1 = 1$

(10) $a + a = a$

(11) $a + a = 1$

(12) $a + b = b + a$

(13) $a + (b + c) = (a + b) + c$

(14) $a + b \cdot c = (a + b) \cdot (a + c)$

(15) $\bar{\bar{a}} = a$

(16) $\overline{a \cdot b} = \bar{a} + \bar{b}$

(17) $\overline{a + b} = \bar{a} \cdot \bar{b}$

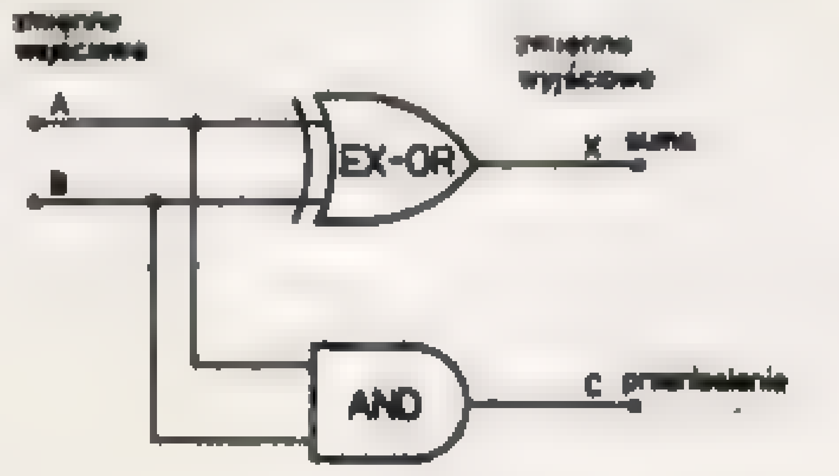
*/ $\bar{a}$ — „a" zanegowane

Dwie ostatnie zależności, oznaczone (16, (17), nazywane **prawami de Morgana,** są bardzo ważne przy działaniach na funkcjach NAND i NOR. Z praw tych wynika, że dowolną funkcję boolowską można zrealizować za pomocą bramek jednego z dwóch typów: NAND lub NOR. Nie jest to bez znaczenia, w związku m. in. z tym, podstawowymi, powszechnie wykorzystywanymi bramkami są NAND i NOR, umieszczone (po cztery danego typu) w układach 7400 i 7402

Jedną z metod upraszczania postaci funkcji logicznych jest metoda **tablic Karnaugha,** znana też pod nazwą metody **wykresów Veitcha.** Metoda ta polega na minimalizacji funkcji przez łączenie w grupy jedynek i jest pomocna przy przechodzeniu od opisu słownego problemu do postaci funkcji boolowskiej

Połączenie bramek logicznych tworzy **układ kombinacyjny,** w którym sygnały dwuwartościowe na wyjściach są jednoznacznie określonymi funkcjami kombinacji zer i jedynek, podanych w danej chwili na wejścia. Wśród przykładów układów kombinacyjnych można podać półsumator i sumator pełny.

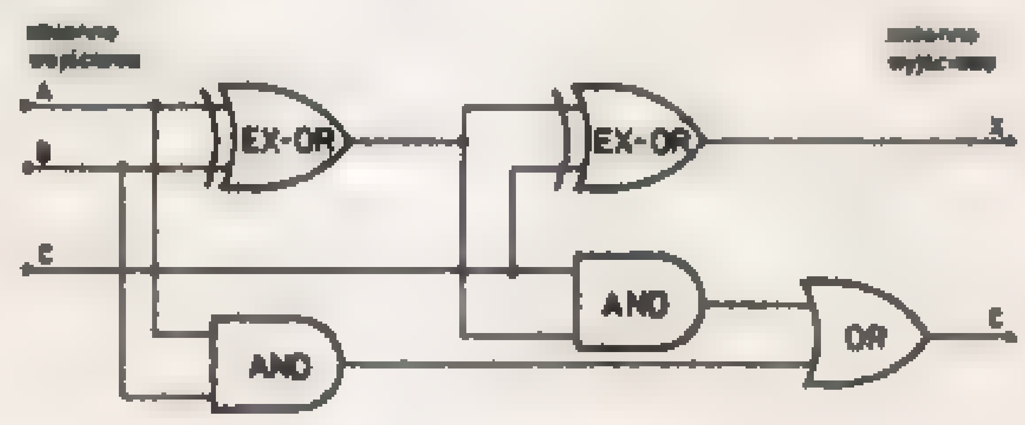
Półsumator, przedstawiony na rys. 4,



A	B	X	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Rys. 4. Układ półsumatora

umożliwia arytmetyczne dodawanie dwóch bitów. Natomiast sumator pełny, przedstawiony na rys. 5, umożliwia



A	B	C	X	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Rys. 5. Układ sumatora pełnego

arytmetyczne dodawanie trzech bitów wejściowych. Na wejście C podawane jest przeniesienie z poprzedniej, mniej znaczącej pozycji.

Układ, w którym, oprócz bramek logicznych, znajdują się układy pamiętające, zwane przerzutnikami jest **układem sekwencyjnym.** W takim układzie sygnały dwuwartościowe na wyjściach są kombinacją zer i jedynek na wejściach, jak również zależą od stanów przerzutników, czyli od ostatnio zapamiętanych wartości sygnałów

Układy sekwencyjne można podzielić na: **synchroniczne,** w których momenty działania poszczególnych elementów elektronicznych taktowane są **impulsami zegarowymi** i **asynchroniczne,** bez wyróżnionych impulsów zegarowych.

Do przechowywania informacji dwójkowych wykorzystywane są cyfrowe układy scalone zwane **rejestrami.** Rejestr zawiera w sobie **przerzutniki** — komórki logiczne do przechowywania jednego bitu informacji oraz może posiadać bramki kombinacyjne, ułatwiające przetwarzanie informacji.

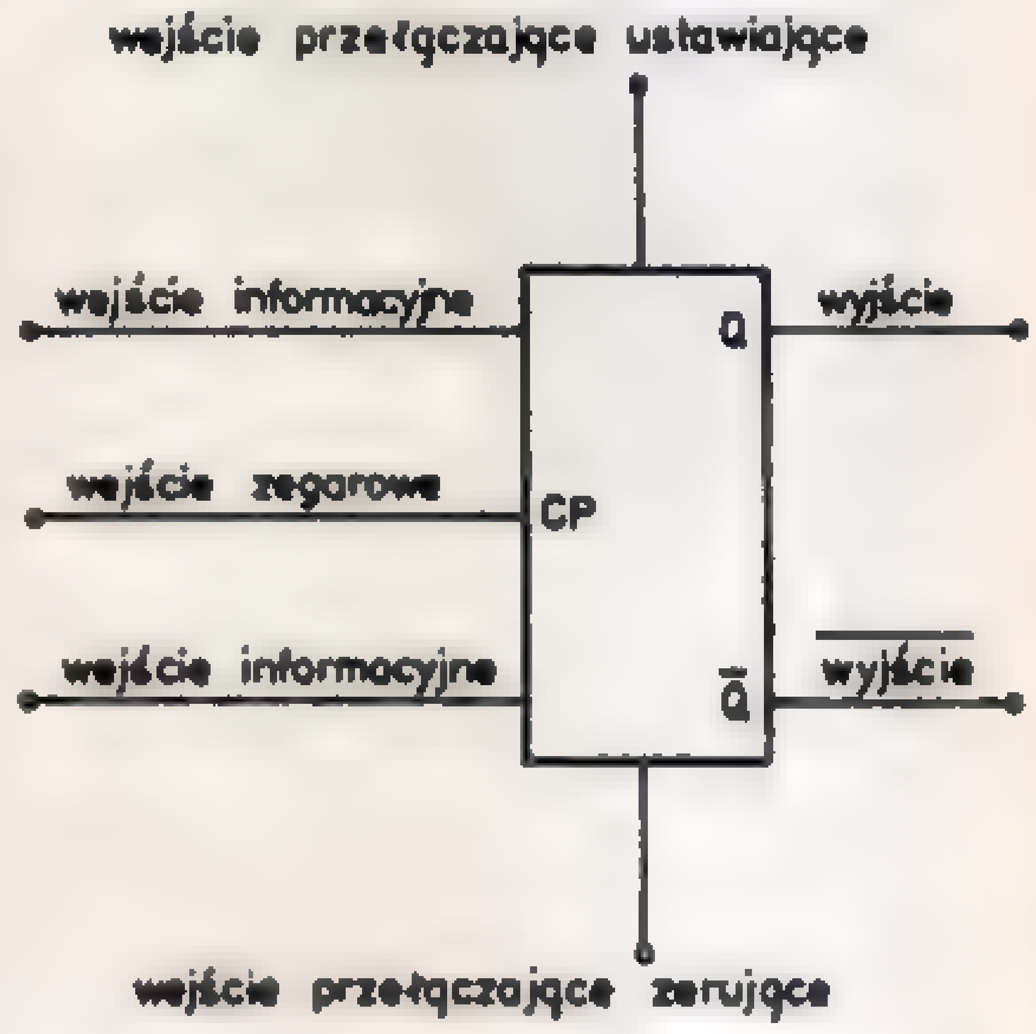
Przerzutniki można podzielić na **asynchroniczne R-S** oraz **synchroniczne,** wykorzystywane w systemach komputerowych. Przerzutniki synchroniczne można podzielić na:

— **wyzwalane zboczem,** zmieniające swój stan w chwili zmiany wartości sygnału zegarowego z 1 na 0 lub z 0 na 1,

— **wyzwalane poziomem** inaczej zatrzaskowe typu „latch", przenoszące zmiany z wejścia na wyjście przy jedynce logicznej na wejściu zegarowym;

— **dwuzboczowe typu Master-Slave,** składające się z przerzutnika głównego i przerzutnika pomocniczego

Symbol graficzny przerzutnika synchronicznego przedstawia rys. 6



Rys. 6. Przerzutnik synchroniczny

Upraszczając można stwierdzić, iż w zależności od posiadanych wejść przerzutniki można sklasyfikować jako **przerzutniki JK, przerzutniki D, przerzutniki T.**

Rejestry można sklasyfikować w następujący sposób

— **szeregowe,** do szeregowego (kolejnego) wprowadzania i wyprowadzania informacji, bit po bicie,

— **równoległe,** do równoległego (jednoczesnego) wprowadzania i wyprowadzania informacji,

— **szeregowo-równoległe,** do szeregowego wprowadzenia i równoległego wyprowadzenia informacji;

— **równoległo-szeregowe,** do równoległego wprowadzenia i szeregowego wyprowadzenia informacji

Rejestry szeregowe mogą być **rejestrami jednokierunkowymi,** przesuwającymi informację w lewo **lub** w prawo albo **rejestrami dwukierunkowymi,** przesuwającymi informację w lewo i w prawo. Rejestry równoległe wykorzystywane są do budowania **pamięci buforowych,** przechowujących informacje podawane w sposób równoległy.

Rejestr zmieniający stan w określonej kolejności nazywany jest **licznikiem.** Licznik służy do zliczenia i zapamiętania liczby impulsów podanych na jego wejścia zliczające. Licznik można podzielić m.in. na **jednokierunkowe** i **dwukierunkowe (rewersyjne)**

oraz na **asynchroniczne, synchroniczne, asynchroniczno-synchroniczne.** Liczniki zbudowane są, podobnie jak rejestry, z przerzutników i bramek kombinacyjnych.

Innymi układami, które spotkacie w systemie cyfrowym są układy do przetwarzania kodów. Układy służące do zmiany kodu, w czasie komunikacji człowiek ⇆ system, nazywane są **przetwornikami (konwerterami)** kodów. Do zamiany informacji dwójkowej na kod pierścieniowy „1 z n" służą **dekodery,** do odwrotnej zamiany zaś **enkodery.** Pozostałe przetworniki nazywa się **transkoderami.**

Do realizacji **transmisji (przesyłania) informacji** wykorzystywane są **multipleksery** i **demultipleksery** czyli przetworniki formatu słów z równoległego na szeregowy oraz do transmisji linią przesyłową z szeregowego na równoległy.

W części sterującej każdego systemu cyfrowego spotykane są **układy uzależnień czasowych.** Są wśród nich **przerzutniki monostabilne, generatory fali prostokątnej, rozdzielacze, selektory impulsów** itp.

Do wykonywania działań arytmetycznych w systemach cyfrowych wykorzystywane są **sumatory; subtraktory** (do odejmowania); sumatory uzupełnione o układy przesuwające, umożliwiające wykonywanie mnożenia i dzielenia; **komparatory,** porównujące wielkość kilku liczb; **multiplikatory,** do realizacji operacji mnożenia; **jednostki arytmetyczno-logiczne ALU,** wykonujące szereg operacji arytmetycznych; **układy kontroli parzystości** itp.

Kolejne grupy układów systemów cyfrowych to: **układy sprzęgające,** sterujące m.in. wskaźnikami cyfrowymi, współpracujące z klawiaturą oraz **układy transmisji,** do przekazywania informacji na duże odległości.

Tak w skrócie wygląda charakterystyka układów systemów cyfrowych.

Wróćmy do arytmetyki systemów cyfrowych. System liczbowy **o podstawie** R jest systemem, w którym każda z r cyfr oznaczona jest różnym **symbolem.** W **dziesiętnym systemie liczbowym,** systemie o podstawie r = 10, używa się dziesięciu symboli: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. W **dwójkowym systemie liczbowym** r = 2 i używane są dwa symbole: 0,1. W **ósemkowym (oktalnym) systemie liczbowym** r = 8 i używanych jest osiem symboli: 0, 1, 2, 3, 4, 5, 6, 7. W **szesnastkowym (heksadecymalnym) systemie liczbowym** r = 16 i używa się szesnastu symboli: 0,1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

W systemach cyfrowych spotkacie się właśnie z tymi systemami liczbowymi. Zapiszmy przykładowo w każdym z tych systemów liczbę 1024.

Liczba w systemie dziesiętnym zapisywana jest następująco.

(18)

$$L_{10} = \sum_{i=-m}^{n-1} a_i \cdot 10^i = a_{n-1} \cdot 10^{n-1} + a_{n-2} \cdot 10^{n-2} + \ldots + a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + \ldots + a_{-m} \cdot 10^{-m}$$

1024 to

$$1 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0$$

Liczba w systemie dwójkowym zapisywana jest następująco:

(19)

$$L_2 = \sum_{i=-m}^{n-1} a_i \cdot 2^i = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \ldots + a_{-m} \cdot 2^{-m}$$

1024_{10} to

$$1 \cdot 2^{10} + 0 \cdot 2^9 + 0 \cdot 2^8 + \ldots + 0 \cdot 2^1 + 0 \cdot 2^0$$

$$1024_{10} = 100\ 0000\ 0000_2$$

Liczba w systemie ósemkowym zapisywana jest następująco:

(20)

$$L_8 = \sum_{i=-m}^{n-1} a_i \cdot 8^i = a_{n-1} \cdot 8^{n-1} + a_{n-2} \cdot 8^{n-2} + \ldots + a_3 \cdot 8^3 + a_2 \cdot 8^2 + a_1 \cdot 8^1 + a_0 \cdot 8^0 + a_{-1} \cdot 8^{-1} + \ldots + a_{-m} \cdot 8^{-m}$$

1024_{10} to

$$2 \cdot 8^3 + 0 \cdot 8^2 + 0 \cdot 8^1 + 0 \cdot 8^0$$

$$1024_{10} = 2000_8$$

Liczba w systemie szesnastkowym zapisywana jest następująco:

(21)

$$L_{16} = \sum_{i=-m}^{n-1} a_i \cdot 16^i = a_{n-1} \cdot 16^{n-1} + a_{n-2} \cdot 16^{n-2} + \ldots + a_3 \cdot 16^3 + a_2 \cdot 16^2 + a_1 \cdot 16^1 + a_0 \cdot 16^0 + a_{-1} \cdot 16^{-1} + \ldots + a_{-m} \cdot 16^{-m}$$

1024_{10} to

$$4 \cdot 16^2 + 0 \cdot 16^1 + 0 \cdot 16^0$$

$$1024_{10} = 400_{16}$$

zestawiając.

$$1024_{10} = 100\ 0000\ 0000_2 = 2000_8 = 400_{16}$$

Przykłady liczb całkowitych, zapisanych w różnych systemach przedstawiono w tablicy 2.

Tablica 2

LICZBY W RÓŻNYCH SYSTEMACH LICZBOWYCH

System dziesiętny	System szesnastkowy
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	[illegible]
11	[illegible]
12	C
13	D
14	E
15	F
16	10
20	14
40	28
50	32
100	64

System ósemkowy	System dwójkowy
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
10	1000
11	1001
12	1010
13	1011
14	1100
15	1101
16	1110
17	1111
20	10000
24	10100
48	101000
62	110010
144	1100100

Od arytmetyki rozpoczniemy nasze następne spotkanie.

Jacek WOJTALA

KOMPUTERY W LOTNICTWIE



Fot. 1 Komputery pomagają w kontroli ruchu powietrznego

Komputery zadomowiły się na dobre we współczesnym lotnictwie, przejmując wiele różnorodnych zadań obciążających do tej pory człowieka bądź wykonywanych przez rozbudowane i zawodne urządzenia elektromechaniczne. Pełnią one wiele funkcji usługowych w urządzeniach naziemnych, rewolucjonizując szkolenie pilotów i obsług technicznych dzięki realistycznej symulacji warunków lotu i złożonych systemów samolotu, usprawniając naziemną obsługę techniczną i diagnostykę, pracując w systemach ochrony lotnisk, pomagając w kontroli ruchu powietrznego czy też w rezerwacji miejsc na rejsy pasażerskie.

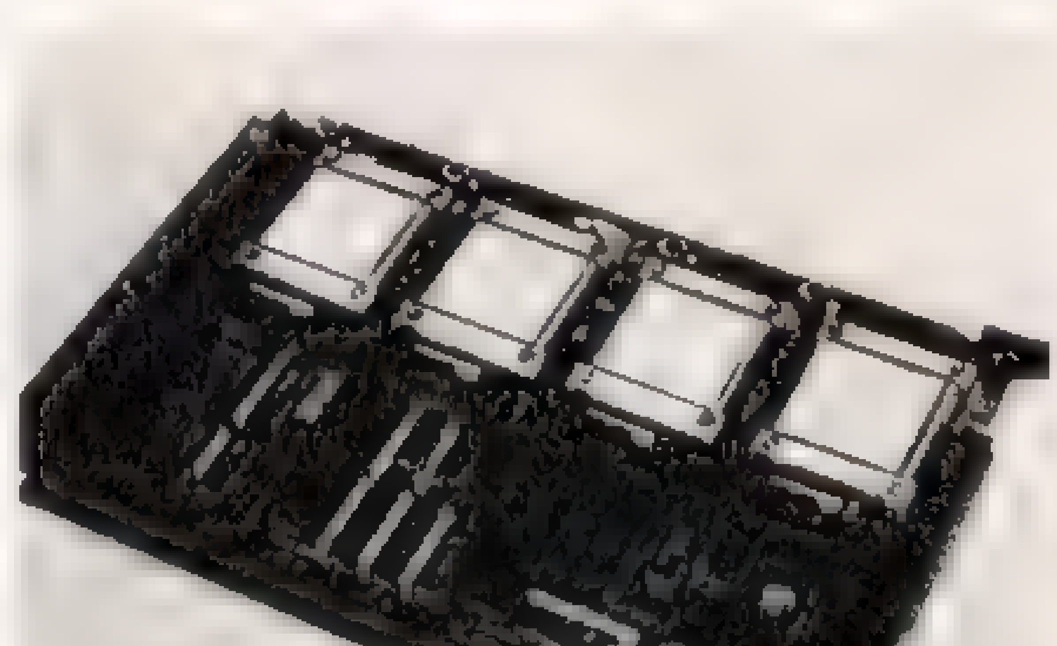
W dalszej części artykułu zajmiemy się jednak komputerami znajdującymi się na pokładzie samolotów. Mają one wiele zadań, z których najważniejsze, to:

— obliczenia nawigacyjne,
— pilotowanie samolotu,
— kontrola funkcjonowania elementów wyposażenia lotniczego,
— sterowanie pracą silników,
— sterowanie systemów dystrybucji paliwa
— przetwarzanie informacji pilotażowo-informacyjnej na elektroniczne układy obrazowania (wskaźniki),
— obliczenia do kierowania ogniem i bombardowania (w samolotach wojskowych).

Zadania te komputery pokładowe wykonywać muszą niejednokrotnie w bardzo trudnych warunkach, wymaga się więc od nich wyjątkowej odporności na wpływy otaczającego środowiska, jak np. możliwości pracy w temperaturach od —55°C do +80°C, odporności na zmiany ciśnienia, tj. poprawnej pracy zarówno na powierzchni Ziemi, jak też na wysokościach rzędu 15—25 km, odporności na przeciążenia w granicach 7—9 (przeciążenie — stosunek przyspieszeń występujących w locie po torze krzywoliniowym do przyspieszenia ziemskiego), wibracje w zakresie częstotliwości 10—1000 Hz, kurz, aerozole soli itp... a wszystko to w wilgotności względnej aż do 95%. Pomimo to komputery pokładowe pracują bezawaryjnie przez kilka tysięcy godzin ku zadowoleniu użytkowników.

Różnorodność zadań doprowadziła do pewnej specjalizacji i w chwili obecnej wyróżnić można komputery nawigacyjne, komputery danych powietrznych, komputery-przeliczniki kierowania bronią, komputery zarządzające urządzeniami awionicznymi i inne wąskospecjalizowane maszyny znajdujące się głównie na pokładzie samolotów bojowych, współpracujące z radarem, systemami kontroli podwieszanego uzbrojenia, radiokomunikacji, dystrybucji paliwa itp... Stało się tak jednak nie od razu...

Pierwsze próby wprowadzenia komputerów w lotnictwie sięgają lat pięćdziesiątych i dotyczą głównie maszyn analogowych. Wyjątkiem jest tutaj maszyna cyfrowa opracowana we wczesnych latach pięćdziesiątych, służąca jako przelicznik do kierowania ogniem rakiet odpalanych z myśliwca F-102. Była to konstrukcja oparta na lampach miniaturowych, funkcjonująca z szybkością setek operacji na sekundę. Jej program operacyjny złożony z kilkuset słów, zapisany był obwodowo na małym bębnie magnetycznym, w języku maszynowym.



Fot. 4 Magnetyczna pamięć pęcherzykowa komputera systemu kierowania lotem

Burzliwy rozwój techniki komputerowej w latach sześćdziesiątych sprawił, że jednemu centralnemu komputerowi na pokładzie samolotu usiłowano podporządkować całość rozbudowanych urządzeń pokładowych, niejako wbrew usługowym funkcjom komputera, który jest tylko instrumentem pomocniczym, a nie celem samym w sobie. Obecnie komputery cyfrowe (procesory) rozmieszczone są w sposób zdecentralizowany, zwykle w pobliżu czujników bądź wskaźników, z którymi współpracują, rozproszone w różnych miejscach samolotu, komunikując się wzajemnie poprzez magistrale danych integrujące wszystkie urządzenia awioniczne w jeden zwarty system. Dzięki nim możliwości pilotażowe rozszerzone zostały poza granice dotychczas niewyobrażalne. Przykładami mogą być tutaj systemy ślepego lądowania, pilot automatyczny, systemy nawigacji według profilu terenu czy też elektroniczne systemy sterowania (ang. „fly by wire")

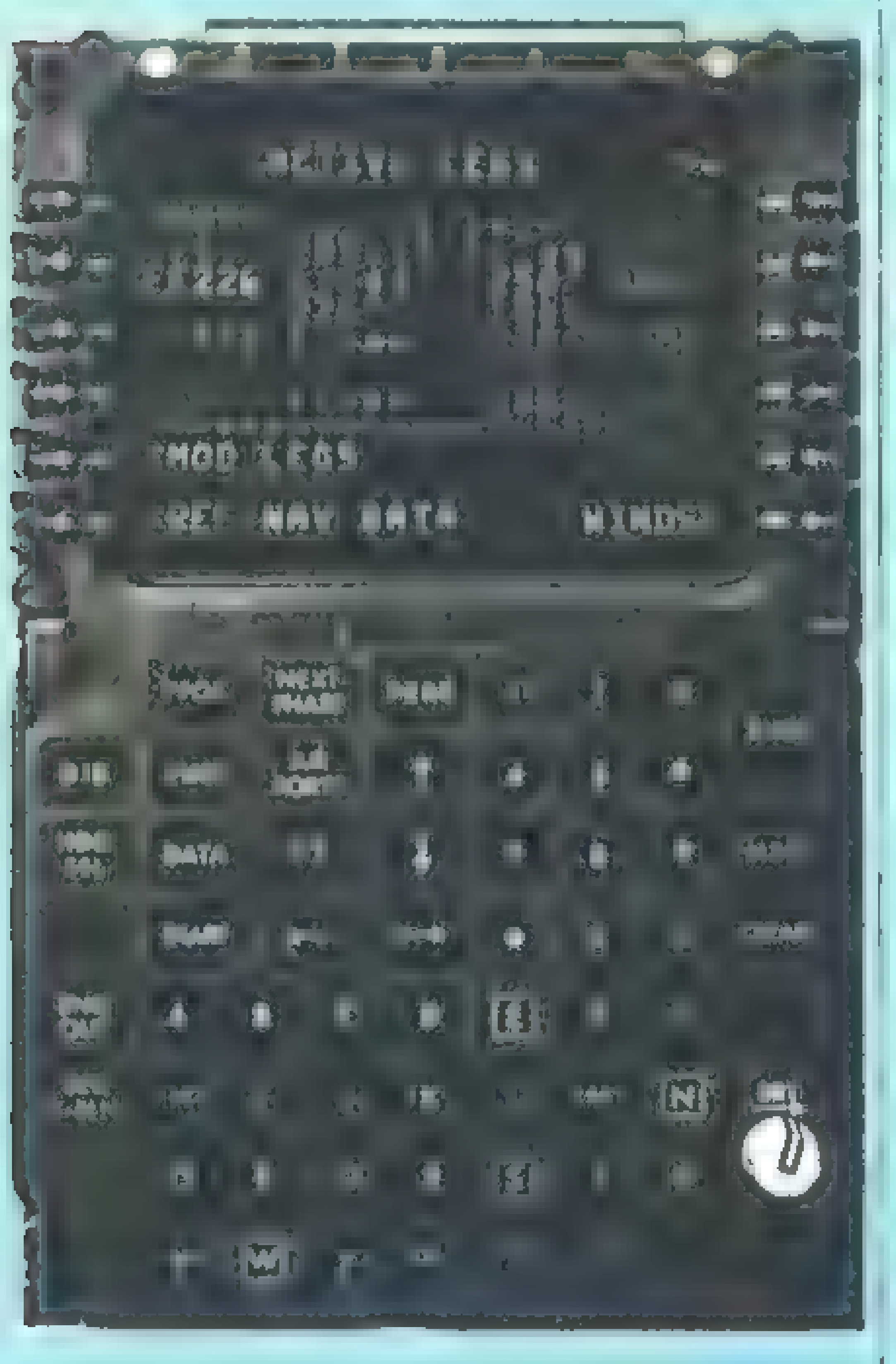
Właśnie opracowanie elektronicznego systemu sterowania samolotem było przełomem w wykorzystywaniu komputerów pokładowych. We wcześniejszych hydro-mechanicznych systemach sterowania projektanci zdani byli na łaskę i niełaskę charakterystyk aerodynamicznych samolotu. System „fly by wire" pozwala na nowatorskie rozwiązywanie zagadnień sterowania samolotem poprzez wprowadzanie sił sterujących dokładnie tam, gdzie trzeba i kiedy trzeba. Efektem jest samolot, który zachowuje się tak, jakby jego konfiguracja była idealnie dopasowana do warunków zastanych w danym momencie, zwany krótko CCV (ang „control configured vehicle"), tj. samolot o konfigurowanej stateczności. Złożone problemy aerodynamiki lotu rozwiązywane są przez komputery za pomocą metod optymalizacji. Zaletami elektronicznego systemu sterowania są

— duża zwrotność samolotu
— mniejsze rozmiary powierzchni sterowych,
— niedopuszczanie do przeciągnięcia samolotu poprzez umieszczenie ograniczników w pętlach sterowania

Istota systemu „fly by wire" polega na usunięciu tradycyjnych mechanicznych połączeń pomiędzy pilotem a powierzchniami sterowymi i zastąpieniu ich sygnałem elektrycznym analogowym bądź cyfrowym, który na swej drodze do urządzeń wykonawczych przetwarzany jest przez specjalizowane komputery pokładowe sprzężone z

różnorodnymi czujnikami informującymi o parametrach lotu. Ponad połowa oprogramowania systemu przeznaczona jest do kontroli parametrów lotu i utrzymania odpowiednich wartości granicznych, tj. prędkości, współczynników obciążenia itp.

W celu zwiększenia niezawodności stosuje się potrójnie lub poczwórnie rezerwowane (rezerwowanie — stosowanie dodatkowych (rezerwowych) systemów lub elementów strukturalnych) systemy sterowania zapewniające zmniejszenie ryzyka przypadku utraty samolotu z powodu całkowitego uszkodzenia układu sterowania do 1 na 10 milionów tj. 10^{-7} milionów. Wyliczanie samolotów wyposażonych w elektroniczny system sterowania zająłoby dużo miejsca, wspomnę więc tylko, ze samolot pasażerski „Concorde" posiada poczwórnie rezerwowany analogowy system sterowania, samolot wielozadaniowy „Tornado" potrójnie rezerwowany system analogowy, a samolot myśliwski F-18 poczwórnie rezerwowany cyfrowy. Prom kosmiczny „Space Shuttle" zaopatrzony jest również w poczwórnie rezerwowany cyfrowy system sterowania.



Fot. 3 Moduł wprowadzania i zobrazowania informacji komputera systemu kierowania lotem

Wprowadzenie obwodów o dużej i bardzo duzej skali integracji doprowadziło do rewolucji w dziedzinie wykorzystywania systemów sterowania i informowania na pokładzie samolotu. Komputery (procesory) pozwoliły na realizację techniczną wielu nowych urządzeń i zmiany jakościowe urządzeń dotychczas istniejących, a mianowicie: wskaźników refleksyjnych (obraz wyświetlany na przedniej szybie kabiny), wielofunkcyjnych wskaźników barwnych, wielofunkcyjnych radarów dopplerowskich, obserwacyjnych urządzeń termalnych (termowizyjnych), laserowych dalmierzy, oświetlaczy i termolokatorów, sterowanych mikroprocesorowo urządzeń walki radioelektronicznej, systemów nawigacji według profilu terenu i wspomnianych juz elektronicznych układów sterowania samolotem. Te nowe rozwiązania zwiększyły znacznie bezpieczeństwo latania poprzez dokładne i szybkie informowanie pilota, poprawienie manewrowości i umozliwienie lotów w praktycznie każdych warunkach atmosferycznych. W samolotach wojskowych poprawiły istotnie bojową efektywność poprzez zwiększenie celności i skuteczności ognia oraz wprowadzenie skuteczniejszych urządzeń walki elektronicznej. Ten jakościowy skok technologiczny stał się mozliwy dzięki przejęciu przez komputery co najmniej czterech zadań niemożliwych dotychczas do wykonania przez urządzenia. Należą do nich:

— łączenie w sposób komplementarny wskazań wielu czujników pokładowych pozwalających na uzyskanie jakościowo wyższych parametrów (wskazań), jak ma to miejsce w przypadku komputerów nawigacyjnych przetwarzających dane z układu nawigacji bezwładnościowej, radaru dopplerowskiego, radiokompasu, itp...,
— zastąpienie złożonych urządzeń elektromechanicznych, jak ma to miejsce w systemach nawigacji bezwładnościowej, radarach dopplerowskich, systemach stabilizacji częstotliwości prądów i napięć zasilających, przelicznikach kierowania ogniem itp...,
— przetwarzanie danych z czujników jak np. procesory radarowe, procesory danych powietrznych, procesory obrazowe wskaźników barwnych na lampach kineskopowych itp.
— wykorzystanie nowych zjawisk fizycznych w konstrukcji urządzeń pokładowych, dotychczas niemożliwych do realizacji, jak np. żyroskop laserowy w systemie nawigacji bezwładnościowej.

Jednym z ciekawszych zastosowań komputerów w lotnictwie komunikacyjnym jest komputerowy system kierowania lotem. Integruje on funkcje systemów nawigacji, automatycznego sterowania silnikami i autopilota. Umożliwia on bezpieczne prowadzenie samolotu według zadanej trasy przelotu, oblicza optymalną prędkość lotu i wysokość, zużycie paliwa i czasy przylotu do wyznaczonych punktów trasy i lotnisk. Dokładny opis techniczny systemu jest trudny i niezrozumiały dla średnio przygotowanego czytelnika. Jego struktura oparta jest na kilku procesorach, co pozwala uzyskać większe szybkości działania niż w przypadku jednego procesora centralnego. Są to dostępne komercyjnie procesory 16-bitowe typu Intel 8086, 8087 i 8088. Pamięć swobodnego dostępu RAM może być trojakiego rodzaju: magnetyczna rdzeniowa, magnetyczna na dyskach lub magnetyczna pęcherzykowa. Ta ostatnia stosowana jest w najnowocześniejszych rozwiązaniach ze względu na dużą gęstość upakowania danych i występowanie części ruchomych, co zwiększa jej niezawodność. Objętość programów komputera systemu kierowania lotem zawiera się w granicach 256 do 512 kilosłów.

Najbardziej nasycone techniką mikrokomputerową są samoloty bojowe. Pilot współczesnego samolotu wielozadaniowego musi wykryć cele, zidentyfikować, śledzić i zniszczyć, w czym pomocne mu są najnowocześniejsze systemy pokładowe i bronie. W toku wykonywania zadania komputery pokładowe wykonują miliony obliczeń w



Fot. 2 „Skomputeryzowana kabina samolotu pasażerskiego najbliższej przyszłości A-320. Widoczne wielofunkcyjne wskaźniki barwne na lampach kineskopowych i w dolnej części dwa moduły wprowadzania i zobrazowania danych systemu kierowania lotem

ułamku sekundy i podejmują różnorodne decyzje. Pilot prowadząc samolot musi ciągle kontrolować przyrządy i interpretować ich wskazania aby wykonać zadanie bojowe. W celu odciążenia pilota wszelkie decyzje i funkcje, które można zautomatyzować zostały włączone w zintegrowane podsystemy komputerowe. Oczywiście operacje wykonywane przez nie są pod kontrolą pilota i w każdej chwili może wykonać je sam lub zdając się na podsystemy komputerowe realizować swoje podstawowe zadania, tj. obmyślać taktykę walki i obserwować wskazania przyrządów. Rozwiązania systemowe współczesnych samolotów bojowych pozwalają bowiem na w pełni zautomatyzowane prowadzenie samolotu, bez ingerencji w pracę poszczególnych układów. Samolot potrzebuje ręki pilota jedynie w momencie startu i lądowania.

dokończenie na str. 14

dokończenie ze str. 13

jak również podczas manewrów w rejonie działań bojowych (celu). Przelot według zadanej trasy do wybranych lotnisk i rejonów, jak również podejście do lądowania, do wysokości kilkudziesięciu metrów, odbywają się automatycznie.

System komputerowy samolotu bojowego składa się z 1—2 komputerów zadaniowych i pewnej liczby komputerów specjalizowanych współpracujących z czujnikami i wskaźnikami. Praktycznie wszystkie nowoczesne samoloty bojowe wyposażone są w tego typu systemy komputerowe lub przynajmniej w komputerowy system nawigacyjno-celowniczy. Wszelkie istotne dane pomocne do wykonania zadania wprowadzane są do pamięci komputera przed startem samolotu. Zwykle są to współrzędne geograficzne lotnisk, punktów pośrednich trasy przelotu, współrzędne rejonów celów oraz cała gama dodatkowych warunków związanych ze specyfiką wykonywanego zadania. Dane te wprowadzać można z kart perforowanych, taśm magnetycznych bądź za pomocą klawiatury alfanumerycznej.

Komputery pokładowe są znacznie zróżnicowane zależnie od typu samolotu i zadań przed nim stawianych. Obecnie na pokładzie samolotów występują komputery z procesorami 8-, 16-, 24- i 32-bitowymi adresującymi pamięci o pojemności od 32 do 512 kilosłów i szybkości działania w granicach 200 tys. do 800 tys. operacji na sekundę. Standardem w przypadku samolotów wielozadaniowych, myśliwsko-bombowych stał się już praktycznie komputer z procesorem 16-bitowym, jakkolwiek starsze modele tych samolotów mają procesor 8-bitowy. Procesory 16-bitowe wchodzą również w skład większości opracowywanych obecnie systemów minikomputerowych cywilnych samolotów pasażerskich. Natomiast samoloty specjalnego przeznaczenia służące np. do wczesnego ostrzegania powietrznego, wykrywania i zwalczania okrętów podwodnych oraz bombowce strategiczne wyposażone są w komputery o standardzie 32-bitowym, co wiąże się z ogromem informacji zbieranej z dużego obszaru przestrzeni i przetwarzanej z czasie rzeczywistym przez pokładowe systemy komputerowe.

Podobnie zróżnicowana jest wielkość oprogramowania komputerów pokładowych. Program samolotu F-16 składa się np. ze 128 kilosłów, z czego 32 kilosłowa to program komputera-przelicznika kierowania bronią, o szybkości działania 500 tys. operacji na sekundę. Natomiast program samolotu F-18 złożony jest z 400 kilosłów i realizowany przez 13 komputerów pokładowych. Większe samoloty, jak np. samolot wczesnego ostrzegania marynarki wojennej USA, typu P-3, którego oprogramowanie ma objętość 700 kilosłów, z czego 128 kilosłów w magnetycznej pamięci rdzeniowej, bądź samolot wczesnego ostrzegania E-3A „AWACS" mający oprogramowanie o objętości 512 kilosłów, to już niemal latające ośrodki obliczeniowe. Żeby utwierdzić się w tym przekonaniu dodam, że program wykorzystywany do celów diagnostyki i obsługi technicznej tego typu samolotów ma objętość nieco ponad 1 milion słów. Od 30% do 70% programów komputerów pokładowych napisanych jest w językach wyższego rzędu, takich jak: Coral, Jovial, Fortran lub Macro. Reszta zaś w językach maszynowych typu Assembler.

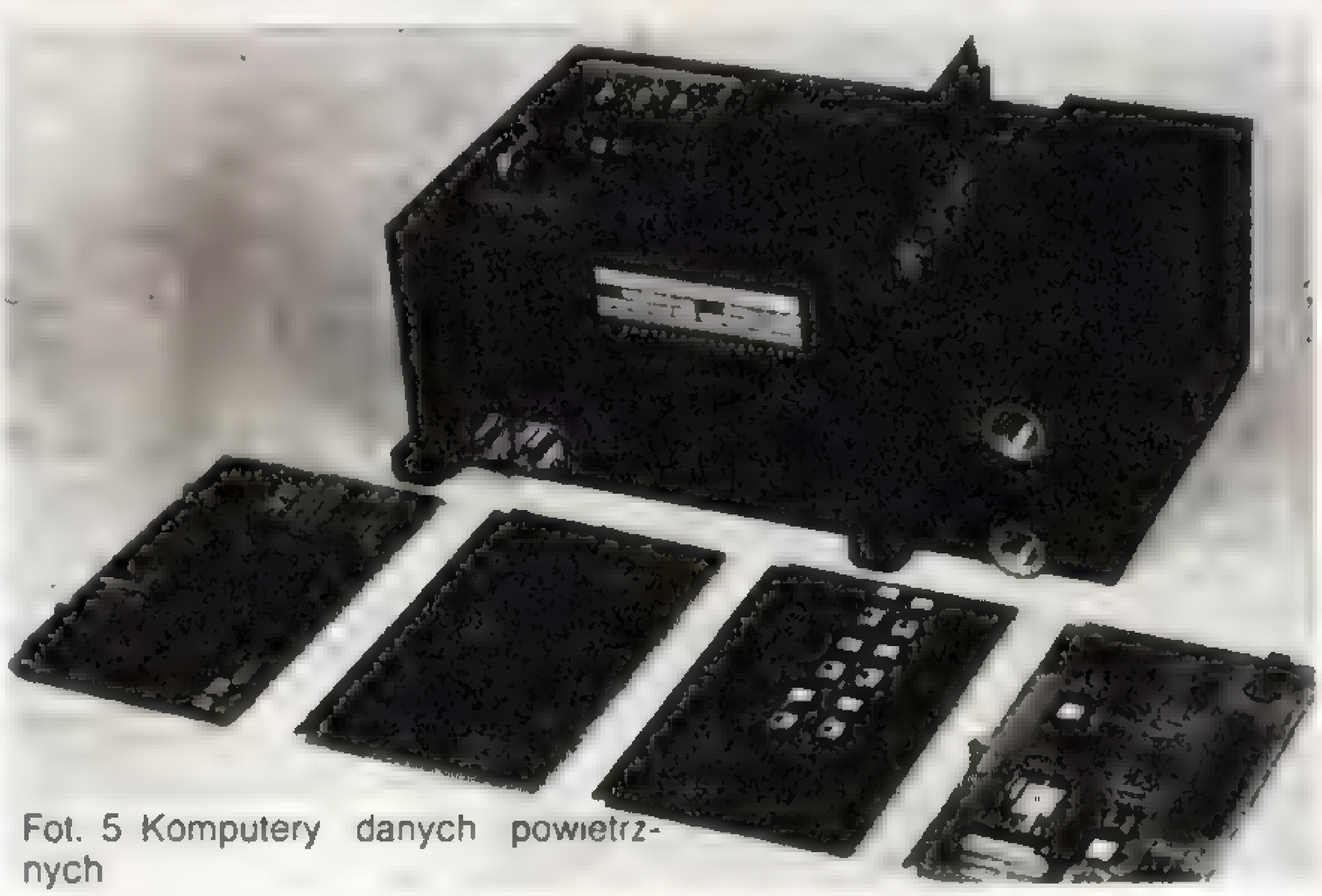
W dziedzinie technologii komputerów pokładowych o standardzie 32-bitowym dokonał się w ciągu ostatnich kilku lat ogromny postęp. Dowodem są dane porównawcze tego typu komputerów z lat 1980 i 1985, opublikowane przez amerykańską firmę IBM.

Do światowej czołówki w produkcji pokładowych komputerów lotniczych należą firmy amerykańskie, angielskie i francuskie. Komputery o wysokim standardzie produkowane są również w Szwecji, RFN, Izraelu i we Włoszech. W krajach socjalistycznych komputery lotnicze produkowane są głównie w ZSRR.

Dzień dzisiejszy przynosi bowiem ciągle nowe rozwiązania techniczne. Dopiero w chwili obecnej konstruktorzy systemów awionicznych zaczynają pojmować, jak różnorodne zadania spełniać mogą napływające w znacznej ilości komputery i w jaki sposób można je wykorzystać najbardziej efektywnie. Jednymi z nowszych pomysłów, w toku prób, są np. cyfrowe systemy sterowania głosem i fotel wyrzucany, którego katapultowaniem steruje układ mikroprocesorowy.

ZBIGNIEW KAROLCZYK

	1980 komputer syst. „AWACS" typu CC-2	**1985 komputer AN/UYK-41**
szybkość działania	2 mln oper./s	3 mln oper./s
pojemność pamięci	2,5 Mbitów	2 Mbity
pobór mocy	6500 W	100 W
objętość	1036 dm³	17 dm³
	komputer nawig.-celowniczy bomb. B-52	**komputer pojedynczo-modułowy**
szybkość działania	400 tys. oper./s	500 tys. oper./s
pojemność pamięci	128 Kbitów	128 Kbitów
pobór mocy	450 W	5 W
objętość	25,2 dm³	0,56 dm³ (pojed. karta)



Fot. 5 Komputery danych powietrznych

Trudne początki, ale...

Rozmowa z Tomaszem Okólskim — szefem klubu komputerowego przy ZW ZSMP we Wrocławiu.

● **Trudno się dodzwonić do klubu, ale za to chyba drzwi do niego są otwarte.**

— Telefonować trzeba przez centralę ZW ZSMP i na pewno uzyska się połączenie. Do niedawna rzeczywiście były trudności, gdyż organizowaliśmy zajęcia w terenie. Nikogo tu nie było. Mamy już jednak swój kąt. Drzwi, jak widać są otwarte.

● **Ale dopiero się tutaj urządzacie. Nie widać sprzętu.**

— To ostatnie przygotowania do otwarcia klubu komputerowego z prawdziwego zdarzenia. Jak widać pomieszczenie jest duże. Chcemy ruszyć ostro w listopadzie.

● **Co tu kiedyś było?**

— Bufet Zarządu Wojewódzkiego ZSMP.

● **Nikt nie miał do was pretensji, że zabraliście lokal na klub?**

— Przydzielono go nam i nie słyszałem żadnych głosów krytycznych pod naszym adresem. Wręcz przeciwnie. Inicjatywa z powołaniem klubu została przyjęta życzliwie. Działacze młodzieżowi pomagają nam w wielu sprawach.

● **Od kiedy faktycznie istnieje klub?**

— Od czerwca

● **I odtąd nie ruszyliście jeszcze z miejsca?**

— Od tamtej pory działaliśmy, jak wspomniałem, w terenie, prowadząc akcję pod nazwą „latający komputer". Po prostu jeździliśmy po gminach województwa wrocławskiego ze sprzętem, propagując problematykę mikrokomputerową wśród tamtejszej młodzieży i władz terenowych. W czasie wakacji udało się nam zainteresować szerokie rzesze młodych ludzi tą dziedziną wiedzy, zachęcić do powoływania klubów.

● **I udało się?**

— Owszem. Zainteresowanie było bardzo duże. W gminach Sobótka i Kąty Wrocławskie już zakupiono sprzęt i powstały kluby. Ponadto uczestniczyliśmy w obozie młodzieżowym. Sprzęt otrzymaliśmy na ten czas z zakładów ELWRO. Również z tego zakładu oddelegowano specjalistę programowania, który prowadził zajęcia na wysokim poziomie metodycznym. Uczył programowania, obchodzenia się ze sprzętem, przekazywał wiele fachowych informacji. Cenimy sobie taką pomoc. Zresztą we Wrocławiu mamy wielu zwolenników. Utrzymujemy też ścisły kontakt z Elektronicznymi Zakładami Naukowymi. Najcenniejsze jest to, że będziemy mogli korzystać z fachowego wsparcia specjalistów wysokiej klasy. Może także i sprzętu.

● **A jaki macie własny sprzęt?**

— Taki chyba sam jak pozostałe kluby wyposażone przez CB TMT. A więc trzy mikrokomputery UNIPOLBRIT. Mamy też jeden SPECTRUM ufundowany przez ZW ZSMP. Ale obecnie jest uszkodzony i mamy kłopoty z naprawą.

● **Nie jest to wiele, tym bardziej że zainteresowanie młodzieży waszym klubem jest duże. Czym więc się będziecie zajmować na początek?**

— Przede wszystkim wymianą doświadczeń, ułatwieniem dostępu do literatury, oprogramowania dla wszystkich chętnych z całego województwa wrocławskiego. Zgłosiła się już duża grupa ze Strzelina. Mamy sygnały o zainteresowaniu grup młodzieży z wielu szkół średnich. Są także zgłoszenia indywidualne. Każdy może do nas przyjść. Jak to się mówi: prosto z ulicy.

● **Czy ci, którzy przyszli, są zadowoleni?**

— Różnie z tym bywa. Niektórzy odchodzą dość szybko, gdy dowiadują się, że nie prowadzimy gier i innych atrakcji. Po prostu traktujemy klub komputerowy nie jako salon gier, lecz placówkę kształcącą młodych ludzi. Dlatego zostają ci, którzy traktują klub jako źródło zdobywania nowych wiadomości, poznawania tajników maszyny, programowania. Z nimi też wiążemy największe nadzieje.

● **Sądzę, że na początek macie już zarysowany jakiś program działania lub tematy, którymi zajmiecie się szczegółowo.**

— Jeszcze nie. Najpierw chcemy ludzi, a potem niech wszyscy ustalą czym powinniśmy się zajmować. Formuła jest otwarta. Na owoce naszej działalności będzie trzeba trochę poczekać.

● **Co z nazwą klubu?**

— Będzie również zależała od jego członków.

● **Jak zostałeś szefem?**

— Studiuję w Akademii Ekonomicznej i tam należę do koła naukowego informatyki. Jest to zresztą moja życiowa pasja. Właściwie przypadkowo trafiłem do tego klubu. Zaproponowano mi tę pracę, a wiadomo, że student nie pogardzi żadnym groszem. Ponadto będzie dla mnie dużą satysfakcją nauczanie innych obcowania z komputerem.

● **Czego oczekujesz od „IKS-a"?**

— Programy, które drukujecie, mnie osobiście już niewiele dają, choć są przydatne innym. Natomiast oczekuję na jeszcze więcej świeżych informacji o sprzęcie, na prezentację oprogramowania.

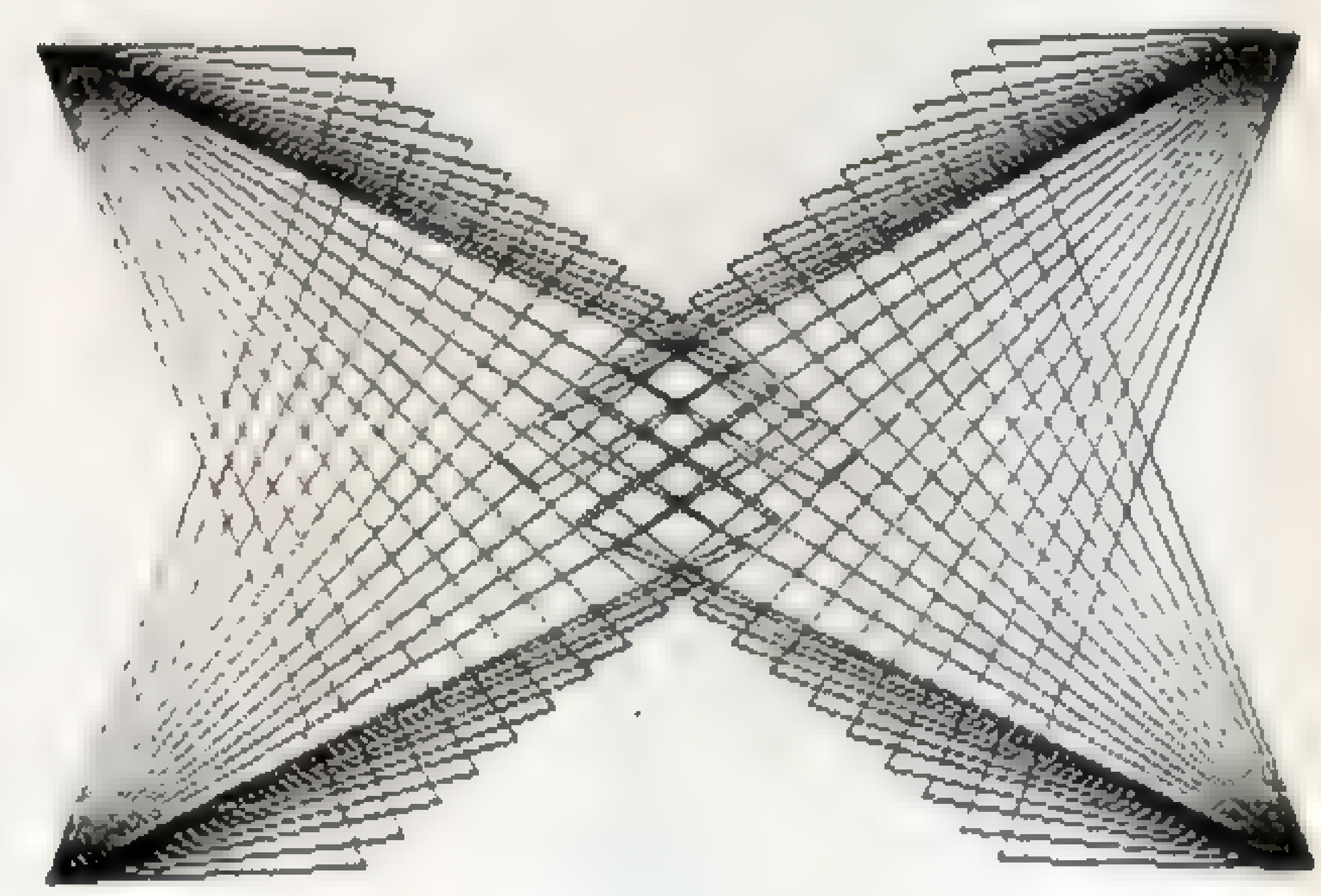
● **Mam nadzieję, że programy, które powstaną w waszym klubie trafią na łamy „IKS-a".**

— Chętnie byśmy je tam widzieli. I skorzystamy z oferty.

● **Życzę więc udanego startu.**

rozmawiał: MAREK SIENIAWSKI

```
10 CLS
30 ORIGIN 250,200
40 FOR x=1 TO 13
60 DRAW 15*x,-15*x
65 DRAW 320,200
70 DRAW 15*x,15*x
80 DRAW 320,-200
90 NEXT x
100 ORIGIN 390,200
110 FOR y=1 TO 13
120 DRAW -15*y,-15*y
130 DRAW -320,200
150 DRAW -15*y,15*y
160 DRAW -320,-200
170 NEXT y
```



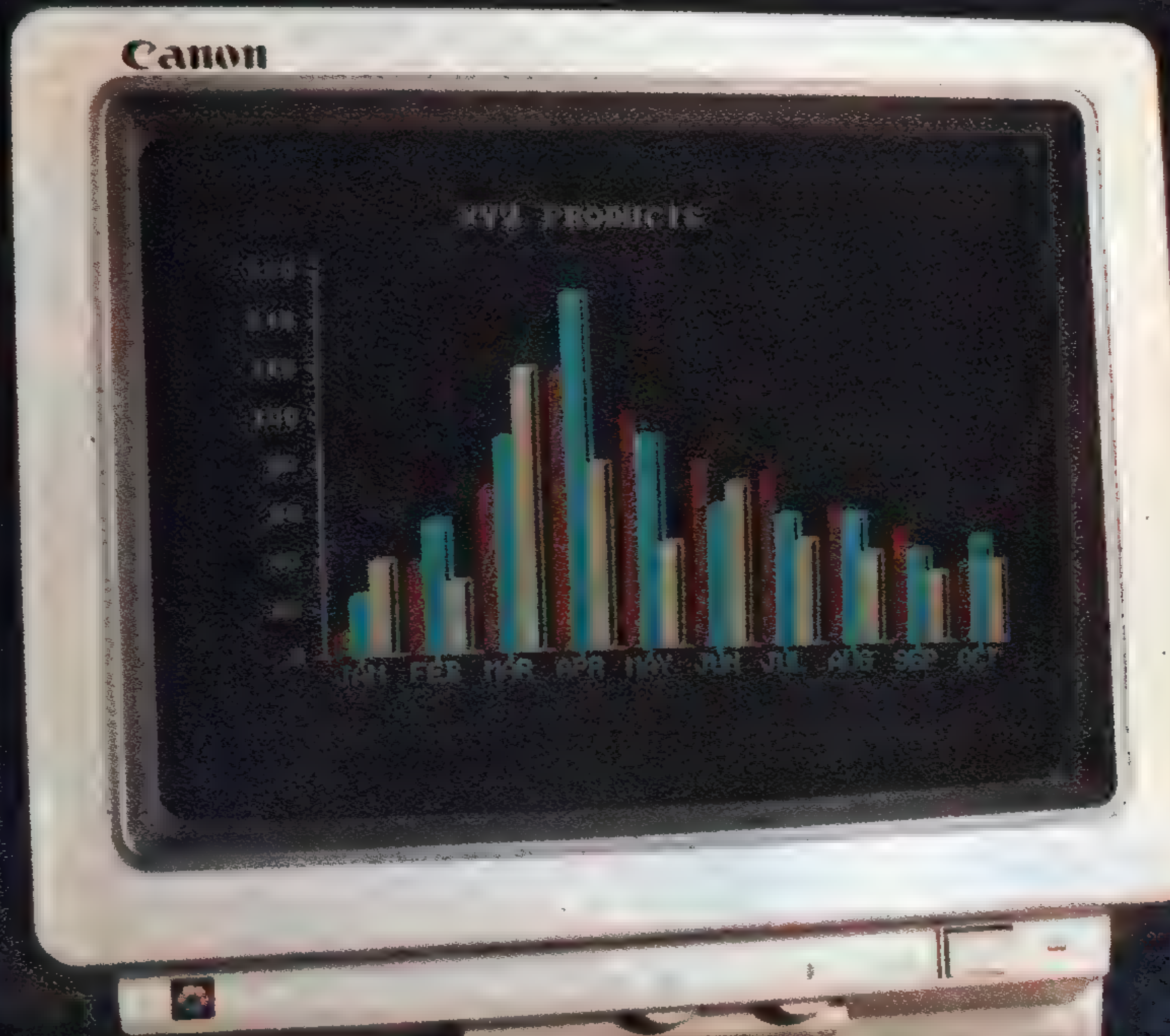
GRAFIKA (AMSTRAD)

IKS 1987

	I STYCZEŃ					II LUTY				
PN		5	12	19	26		2	9	16	23
WT		6	13	20	27		3	10	17	24
ŚR		7	14	21	28		4	11	18	25
CZ	1	8	15	22	29		5	12	19	26
PT	2	9	16	23	30		6	13	20	27
SO	3	10	17	24	31		7	14	21	28
N	4	11	18	25		1	8	15	22	

	III MARZEC						IV KWIECIEŃ				
PN		2	9	16	23	30		6	13	20	27
WT		3	10	17	24	31		7	14	21	28
ŚR		4	11	18	25		1	8	15	22	29
CZ		5	12	19	26		2	9	16	23	30
PT		6	13	20	27		3	10	17	24	
SO		7	14	21	28		4	11	18	25	
N	1	8	15	22	29		5	12	19	26	

	V MAJ					VI CZERWIEC				
PN		4	11	18	25	1	8	15	22	29
WT		5	12	19	26	2	9	16	23	30
ŚR		6	13	20	27	3	10	17	24	
CZ		7	14	21	28	4	11	18	25	
PT	1	8	15	22	29	5	12	19	26	



SO	2	9	16	23	30		6	13	20	27	
N	3	10	17	24	31		7	14	21	28	

VII LIPIEC						VIII SIERPIEŃ					
PN		6	13	20	27		3	10	17	24	31
WT		7	14	21	28		4	11	18	25	
ŚR	1	8	15	22	29		5	12	19	26	
CZ	2	9	16	23	30		6	13	20	27	
PT	3	10	17	24	31		7	14	21	28	
SO	4	11	18	25		1	8	15	22	29	
N	5	12	19	26		2	9	16	23	30	

IX WRZESIEŃ						X PAŹDZIERNIK				
PN		7	14	21	28		5	12	19	26
WT	1	8	15	22	29		6	13	20	27
ŚR	2	9	16	23	30		7	14	21	28
CZ	3	10	17	24		1	8	15	22	29
PT	4	11	18	25		2	9	16	23	30
SO	5	12	19	26		3	10	17	24	31
N	6	13	20	27		4	11	18	25	

XI LISTOPAD						XII GRUDZIEŃ					
PN		2	9	16	23	30		7	14	21	28
WT		3	10	17	24		1	8	15	22	29
ŚR		4	11	18	25		2	9	16	23	30
CZ		5	12	19	26		3	10	17	24	31
PT		6	13	20	27		4	11	18	25	
SO		7	14	21	28		5	12	19	26	
N	1	8	15	22	29		6	13	20	27	



Dean Wspaniały

ANDRZEJ MONASTYRSKI

— Dean, kochanie, wstawaj. Dochodzi szósta trzydzieści... Melodyjny kobiecy głos napełnił pokój. Wydawało się, że płynął zewsząd. Z góry, z dołu, od okien, ścian i drzwi. Jego natężenie stopniowo rosło. Od szeptu przechodził do unisono, by zakończyć się w ciepłym mezzoforte.

Dean leniwie przeciągnął się na wygodnym pneumatycznym tapczanie, przyjmującym przez cały czas kształt jego ciała. Okna powoli rozjaśniały się miłym, łagodnym blaskiem, tak różnym od natrętnych promieni słonecznych. Sprawiał to specjalny system filtrów optycznych. Leżący mężczyzna powoli uchylił powieki. Nie był ostatnio w najlepszym nastroju. Chyba ze dwa razy w zeszłym miesiącu wydało mu się, że nie może porozumieć się bez kłopotu z najnowszym modelem omalizomatu X 10TW95, jego podstawowym narzędziem pracy. A gdyby to zauważył przypadkiem Warner — lub któryś z jego donosicieli, ot, choćby James — Dean nie miałby czego szukać w firmie. Więc dla poprawy samopoczucia zamówił sobie ten poranny program.

Na dużym ekranie ukazała się postać pięknej dziewczyny.

— Dean, przypominam ci, że już powinieneś wstać. Dziś jest czwartek, piętnastego czerwca. Temperatura wynosi 18 stopni, będzie słonecznie, w południe może być do 25 stopni.

Jedna z najatrakcyjniejszych modelek znanej firmy Simson — Cloth uśmiechała się teraz tylko dla niego, ukazując nieskazitelną biel zębów i urocze dołeczki na buzi.

— Wanna już czeka. Dziś masz kąpiel jaśminową. No, rusz się wreszcie — pogroziła figlarnie paluszkiem — bo ci się grzanki przypalą i kawa wystygnie! A do tego, jak będziesz marudził, czeka cię w windzie spotkanie z Jamesem..

Dean zmarszczył brwi. Nie cierpiał tego ulizanego cwaniaczka. Takiego, który zawsze i wszędzie musi być pierwszy, musi ze swym przypochlebnym uśmieszkiem wejść od rana w oczy bossowi, przymilić się sekretarce, opowiedzieć najnowszy dowcip współpasażerom w windzie.

— Stary karierowicz — mruknął pod nosem.

— To stary karierowicz, obrzydliwy typ — powtórzyła jak echo złotoloka Miriam z ekranu. — Nie daj mu się dziś prześcignąć. Musisz być lepszy. Będziesz lepszy...

Dean Robinson podniósł się wreszcie z posłania. Podszedł do niewielkiej klawiatury i przebiegł palcami po przyciskach. Zaprogramował sobie cały poranek. Miriam ponowiła zaproszenie do łazienki z wyjątkowo kuszącym uśmiechem.

— Dean, skarbie, pośpiesz się. Czekam...

Wszedł do sporego seledynowo-błękitnego pomieszczenia. Na ekranie zajmującym pół ściany widniała już Miriam. Stała pod prysznicem, a deszcz srebrzystych kropelek rozpryskiwał się dokoła.

— No, nie gap się tak — posłyszał jej lekko strofujący głos — jakbyś mnie w życiu nie widział. Nie marudź, czas ucieka

Odwrócił oczy, nieco skonfudowany. Wydawało mu się, że to realna Miriam, a nie jej ekranowa postać. Tymczasem ze ścian, sufitu i podłogi co chwila wysuwały się kolejne przyrządy gimnastyczne. Drążki, poręcze, kółka, sprężyny, worki treningowe. Robinson każdemu z nich poświęcał chwilę uwagi, by przejść do następnego. Kazał je zainstalować, gdy stwierdził, że mięśnie mu wiotczeją, brzuch zaczyna się zaokrąglać, a sylwetka traci swój poprawny wygląd.

Wreszcie zmęczony wszedł do pachnącej kąpieli, poddając się łagodnemu podwodnemu masażowi. Odświeżony zabrał się z apetytem do śniadania. W tym czasie Miriam zabawiała go z ekranu.

— W dzisiejszych gazetach nie ma nic ciekawego. Notowania giełdy bez specjalnych zmian. W Europie spokój. W Meksyku wybuchł jakiś wulkan. Uważaj, nie poplam spodni kawą! Boss nie cierpi najmniejszej plamki na ubraniach pracowników, pamiętasz?

— Stary pedant — skrzywił się Dean.

— Stary, głupawy pedant — zaszczebiotała Miriam — ale od niego wszystko zależy. I dzisiejszy dzień i twoja pozycja w pracy i nadzieja na awans. Więc nie marudź i uważaj. Zostaw już te kanapki! Nie zauważyłeś, że ci przybyło na wadze? A boss nie znosi otyłych. Sam chudy jak szczapa nie cierpi grubasów.

Dean cofnął rękę. Szkoda mu było świeżutkich grzanek zaserwowanych przez automat, ale musiał pohamować apetyt. To prawda, Ted Warner bez skrupułów pozbywał się każdego, kto przekroczył 70 kilogramów. I diabli wiedzieli, skąd brał te informacje? Podobno Ted kazał umieścić wagę w wycieraczce, ale nikt tego nie potrafił sprawdzić. Za to w firmie pracowali wyłącznie szczupli, wysportowani panowie, jednakowo ubrani i jednakowo pachnący lawendą. Wyłącznie lawendą

Robinson poczuł narastającą złość i niechęć do pracy, szefa i świata. Codziennie ten sam rytuał, ten sam strój, ten sam zapach. Ale boss był twardy Wymagał, płacił i nie tolerował najmniejszych odstępstw od tego swoistego regulaminu.

— Gorzej, jak w wojsku — warknął. — Tam mogłem odpiąć guzik, podwinąć rękawy, rozluźnić pas, a nawet wylać sobie na łeb garniec jakichkolwiek perfum i nikt nie miał pretensji...

— Nie marudź, Dean — posłyszał znów strofujący głos. — Warner jest pedantyczny, ale dobrze płaci. Inni bossowie bywają gorsi.

— Racja — pomyślał.

— Kończ śniadanie. Pośpiesz się. Masz dziś szansę. Ted wkrótce będzie konferował z facetami z Chemical Industry. Będzie chyba chciał mieć tam swego człowieka. Może właśnie ciebie tam wsadzi? Zrób dziś na nim dobre wrażenie. Bądź miły, nie daj się wyprzedzić Jamesowi. Zabłyśnij czymś, postaraj się. Zrób coś, co go do ciebie przekona. No, pomyśl o czymś takim, zadziałaj dziś mądrze...

— Głupia, wydekoltowana papugo — warknął Dean. — Sam dobrze to wszystko wiem. Powtarzasz mi ciągle tylko to, co ci kazałem nagrać. A bądź taka mądra i powiedz wreszcie jak i czym tu błysnąć?

— Postaraj się Dean. Kochany, zrób coś. Przecież jesteś inteligentny, mądry, pracowity. Może być trudny dzień, ale łatwo dasz sobie radę. Tylko nie zapomnij zagłuszyć ten jaśmin lawendą!

— Jesteś głupia gęś! Stale gęgasz to samo! Mam już dosyć twoich wspaniałych rad! Chyba będę musiał ciebie wymienić. O, już wiem! Zamówię sobie Kate. Ona jest lepsza. Ma zgrabniejsze nogi, wiesz?

dokończenie na str. 23



Rys. M. Przybyłowski

PASCAL (2)

Materiał zaprezentowany w części pierwszej został tak dobrany, aby po jego opanowaniu możliwe było pisanie i uruchamianie prostych programów. W kolejnych częściach, na niezbyt skomplikowanych przykładach, będziemy omawiali wybrane elementy języka. W tej części zajmiemy się w głównej mierze instrukcjami umożliwiającymi zapisywanie w języku PASCAL czynności wykonywanych powtarzalnie. Zgodnie z zapowiedzią zamieszczamy również opis komend edytora, niezbędnych do poprawiania programów źródłowych oraz zadania do samodzielnego rozwiązania.

1. Instrukcje iteracyjne (powtarzania).

Rozpatrzmy następujący problem:

„Wyznaczyć maksymalną liczbę spośród zadanych N liczb rzeczywistych".

Najprostszym rozwiązaniem jest uznanie pierwszej liczby za maksymalną i porównanie jej z liczbą następną. Liczbę większą, z porównywanych, traktuje się jako maksymalną i porównuje ją z kolejną liczbą. Zatem po N-1 porównaniach uzyskujemy liczbę największą spośród zadanych. Program wykonujący przedstawione zadanie przedstawiono w przykładzie 5.

```
PROGRAM prog5;
  VAR n,              {ilosc liczb rzeczywistych}
      i:integer;      {licznik powtorzen}
      x,              {kolejna liczba rzeczywista}
      max:real;       {liczba maksymalna}
  BEGIN
    READLN(n);
    i:=1;
    READLN(x);
    max:=x;
    WHILE i <= n-1 DO
      BEGIN
        READLN(x);
        i:=i+1;
        IF max < x THEN max:=x
      END;
    WRITELN;
    WRITELN('maksymalna liczba sposrod wprowadzonych to ',max)
  END.
```

Do opisu powtarzających się czynności wykorzystano znaną już instrukcję WHILE. Nietrudno zauważyć, że w tym wypadku stosowanie tej instrukcji nie jest zbyt wygodne. Konieczne jest bowiem ustalenie liczby powtórzeń operacji wykonanych zgodnie z instrukcją WHILE oraz określenie warunku kończącego jej wykonanie. Warunek powinien być spełniony wtedy, gdy osiągnie się wymaganą liczbę powtórzeń. Nieprecyzyjne określenie warunku prowadzi do niepoprawnej liczby powtórzeń (najczęściej o 1 mniejsza lub o 1 większa od wymaganej).

Bardziej naturalną konstrukcją do opisu czynności powtarzanych określoną ilość razy jest instrukcja o postaci:

FOR i: = e_1 TO e_2 DO
instr

gdzie:

i — zmienna kontrolna (licznik powtórzeń),
e_1 — początkowa wartość zmiennej kontrolnej,
e_2 — końcowa wartość zmiennej kontrolnej,
instr — dowolna instrukcja.

Zapis ten oznacza, że instrukcja „instr" wykonywana jest dla kolejnych wartości zmiennej kontrolnej „i" od wartości początkowej „e_1" z krokiem 1 do wartości końcowej „e_2". Wartość zmiennej kontrolnej jest zmieniana automatycznie, zatem zmiana taka nie jest dopuszczalna w ramach instrukcji „instr". Zmienna kontrolna może być natomiast wykorzystywana jako argument w wyrażeniach, jeżeli takie występują w instrukcji „instr". Przyjmuje się, że po wykonaniu instrukcji FOR wartość zmiennej kontrolnej jest określona.

Program wyznaczania maksymalnej liczby spośród zadanych N liczb rzeczywistych z wykorzystaniem omówionej instrukcji FOR zamieszczono w przykładzie 6.

```
PROGRAM prog6;
  VAR n,              {ilosc liczb rzeczywistych}
      i:integer;      {zmienna kontrolna}
      x,              {kolejna liczba rzeczywista}
      max:real;       {liczba maksymalna}
  BEGIN
    READLN(n);
    READLN(x);
    max:=x;
    FOR i:=1 TO n-1 DO
      BEGIN
        READLN(x);
        IF max < x THEN max:=x
      END;
    WRITELN;
    WRITELN('maksymalna liczba sposrod wprowadzonych to ',max)
  END.
```

Program „prog. 5" i „prog. 6" uruchomiono dla N = 7 oraz liczb rzeczywistych 2,57; — 12; 26,48; — 1,31; 5,126 i 7,33 uzyskując identyczny wynik o postaci:

```
maksymalna liczba sposrod
wprowadzonych to 2.6480000000E+01
```

W obu programach wykorzystano instrukcję warunkową

IF warunek THEN instr

gdzie:

warunek — wyrażenie logiczne,
instr — dowolna instrukcja.

Jeżeli warunek jest spełniony, to zostaje wykonana instrukcja „instr", a jej zakończenie jest uważane za zrealizowanie całej instrukcji warunkowej. W przypadku przeciwnym instrukcja „instr" nie jest wykonywana i uważa się, że cała instrukcja warunkowa została zrealizowana.

Po tym koniecznym wyjaśnieniu powróćmy ponownie do własności języka PASCAL w zakresie opisu czynności wykonywanych powtarzalnie.

Rozważmy problem obliczania wartości szeregu

$$1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \dots$$

z zadaną dokładnością. Można go rozwiązać przez dodawanie do siebie kolejno obliczanych wartości wyrazów szeregu. Należy przy tym zauważyć, że:

— wartości kolejnych wyrazów szeregu można wyznaczyć z zależności

$$a_n + 1 = a_n \cdot \frac{X}{n+1}$$

przyjmując, że do $a_0 = 1$ (a_n — n-ty wyraz szeregu),

— o dokładności obliczeń decyduje wartość ostatniego obliczonego wyrazu szeregu.

Czytelnicy, którzy rozwiązali zadanie nr 3, bez trudu napiszą program obliczania wartości podanego szeregu, wykorzystując znaną już instrukcję WHILE.

dokończenie na str. 21

AMSTRAD

W Londynie, we wrześniu tego roku, firma AMSTRAD zaprezentowała swój najnowszy model, komputer osobisty PC 1512, kompatybilny z IBM PC. Porównując PC 1512 z niewiele starszymi, dobrze znanymi w naszym kraju, modelami CPC 6128 i JOYCE PCW 8256, należy stwierdzić, że to duży krok do przodu. Można też sądzić, iż model ten będzie cieszyć się olbrzymią popularnością nie tylko w W. Brytanii.



W AMSTRADZIE PC 1512 zastosowano 16-bitowy mikroprocesor 8086, pracujący z częstotliwością zegara 8 MHz. Mikroprocesor adresuje 512 KB pamięci RAM, którą można zwiększyć o 128 KB. Szybkość działania mikroprocesora może być zwiększona po dołączeniu do niego koprocesora arytmetycznego 8087, dla którego wykonano miejsce w płycie systemu. Nie wykonano natomiast złączek do rozszerzenia pamięci operacyjnej ponad pojemność 640 KB. W systemie zastosowano kwarcowy zegar czasu rzeczywistego z baterią, działającą po wyłączeniu zasilania.

PC 1512 przełamuje lansowany dotąd przez firmę AMSTRAD standard dyskietek 3-calowych. W modelu tym zastosowano dwustronne mechanizmy dyskowe, pozwalające na dyskach elastycznych 5 1/4 cala osiągnąć gęstość zapisu 360 KB, dokładniej 368 640 bajtów. Jako pamięć masową zastosowano również dysk stały o pojemności 10 MB lub 20 MB, co jest nowością w modelach firmy.

AMSTRAD PC jest wyposażony w interfejs szeregowy RS 232 C i równoległy typu Centronics. Ten drugi służyć może do podłączenia bardzo dobrych drukarek matrycowych DMP 2000 lub DMP 3000. Trzy zastosowane złączki umożliwiają dołączenie do konfiguracji podstawowej wspomnianego dysku stałego, modemu, przetwornika a/c lub c/a, dodatkowego portu dla drukarki. Istnieje również złączka dla pióra świetlnego.

PC 1512 ma klawiaturę typu QWERTY z 85 klawiszami, która ma wbudowaną złączkę dla drążka sterowego. W skład wyposażenia podstawowego wchodzi również „myszka".

Z najnowszym AMSTRADEM mogą pracować monitor kolorowy PC CM lub monochromatyczny PC MM. Pozwalają one na organizację ekranu w standardzie dużej rozdzielczości (high resolution) w modzie graficznym: w 16 lub



2 kolorach z 640 x 200 punktami bezpośrednio adresowalnymi.

W standardzie średniej rozdzielczości (medium resolution) mogą być wykorzystywane 3 palety czterokolorowe z 320 x 200 punktami bezpośrednio adresowalnymi. W modzie alfanumerycznym organizacja ekranu może być następująca: 16 kolorów w 25 liniach po 80 znaków lub 25 liniach po 40 znaków.

Waga poszczególnych elementów systemu AMSTRAD PC 1512 jest następująca: jednostka centralna z dwiema stacjami dysków 7,75 kg; klawiatura 1,17 kg; PC CM 11,6 kg; PC MM 7,4 kg; DMP 3000 4,2 kg.

Przedstawione parametry AMSTRAD-a PC nie są rewelacyjne, niemniej czynią go atrakcyjnym na rynku europejskim, tym bardziej że upodobniają go do IBM PC/XT.

Teraz krótka charakterystyka oprogramowania. W PC 1512 zastosowano dyskowy system operacyjny wersję MS DOS3.20 oraz DOS Plus firmy DIGITAL RESEARCH. Firma ta opracowała również GEM, ułatwiający pracę w wyżej wymienionych modach graficznych. Firma LOCOMOTIVE SOFTWARE opracowała nową wersję języka Basic — BASIC 2.

W pamięci stałej ROM AMSTRADA PC umieszczono ROM BIOS (Basic Input/ Output System), pracujący bezpośrednio ze wszystkimi układami elektronicznymi, w tym peryferyjnymi, stanowiącymi system PC 1512.

Wśród programów użytkowych, oferowanych przez firmę a znanych na pewno dobrze użytkownikom IBM-ów, są: program do redagowania tekstów WordStar 1512; program dla dyrektorów i ekonomistów SuperCalc 3, ze swoimi pie chartami i bar chartami, oraz inne, w tym: LOTUS 1-2-3; SIDEKICK; PFS: FILE; DBMS; REFLEX.

Nie zapomniano o miłośnikach gier. Przygotowano dla nich szachy, golf, wyścigi samochodowe, zawody zimowe.

Na koniec ceny

— 1 x360 KB,PC MM £ 399,
— 2 x360 KB,PC MM £ 499,
— 1 x360 KB,PC CM £ 599,
— 2 x360 KB,PC CM £ 699,
— 1 x360 KB,10 MB,PC MM£ 699,
— 1 x360 KB,10 MB,PC CM£ 849,
— 1 x360 KB,20 MB,PC MM£ 799,
— 1 x360 KB,20 MB,PC CM£ 949

Rozszerzenie RAM o 128 KB zwiększa cenę o £44

Jacek WOJTALA

dokończenie ze str. 19

Program, dający w wyniku wykonania taki sam efekt, zamieszczono w przykładzie 7. Zamiast instrukcji WHILE, do opisu czynności wykonywanych powtarzalnie wykorzystano instrukcję REPEAT o postaci

REPEAT
I_1
I_2

I_n
UNTIL wyrlog

gdzie: $I_1, I_2, ..., I_n$ — dowolne instrukcje,
wyrlog — wyrażenie logiczne

Zapis ten oznacza, że ciąg instrukcji zawarty pomiędzy słowami kluczowymi REPEAT—UNTIL wykonywany jest tak długo (przynajmniej raz — porównaj z instrukcją WHILE), aż wartość wyrażenia logicznego „wyrlog" będzie równa TRUE. Należy przy tym zdawać sobie sprawę z tego, że jeżeli w żadnym z powtórzeń wykonania ciągu instrukcji wyrazenie logiczne „wyrlog" nie osiągnie wartości TRUE, to wykonanie instrukcji REPEAT nigdy się nie zakończy.

```
PROGRAM prog7;
  VAR n:integer;      {numer wyrazu szeregu}
      s,              {wartosc szeregu}
      p,              {wartosc kolejnego wyrazu}
      eps,            {dokladnosc obliczen}
      x:real;
  BEGIN
    s:=0;
    p:=1;
    n:=0;
    READ(x,eps);
    REPEAT
      s:=s+p;
      n:=n+1;
      p:=p*x/(n+1)
    UNTIL ABS(p) < eps;
    WRITELN;
    WRITELN('wartosc szeregu dla x =',x);
    WRITELN('obliczona z dokladnoscia eps =',eps);
    WRITELN('wynosi s =',s)
  END.
```

Program „Prog 7" wykonano dla X = 1 i eps = 10^{-10} (eps można wprowadzić w postaci 1e—10 lub 0.0000000001) uzyskując wynik:

```
wartosc szeregu dla x =  1.0000000000E+00
obliczona z dokladnoscia eps =  1.0000000000E-10
wynosi s =  1.7182818284E+00
```

W programie wykorzystano standardową funkcję arytmetyczną ABS, obliczającą wartość absolutną argumentu „P" („P" zależy od „X", a więc może przyjmować wartości mniejsze od zera). W języku PASCAL dostępne są jeszcze inne standardowe funkcje, np. obliczające wartość sinusa kąta czy logarytmu naturalnego z danej liczby. Funkcje standardowe zostaną omówione w jednej z kolejnych części.

Aby zamknąć listę instrukcji iteracyjnych języka PASCAL, opiszemy drugą postać instrukcji FOR (FOR—DOWNTO)

Jest ona następująca

FOR i: = e_1 DOWNTO e_2DO

instr

gdzie

i, e_1 e_2, instr — mają identyczne znaczenie, jak w już opisanej instrukcji FOR (FOR—TO).

Zapis ten oznacza, że instrukcja „instr" wykonywana jest dla kolejnych wartości zmiennej kontrolnej „i" od wartości początkowej „e_1", przy czym wartość zmiennej kontrolnej jest zmniejszona o 1 (a nie zwiększona, jak w instrukcji FOR—TO) po każdym wykonaniu instrukcji „instr. Przykłady wykorzystania instrukcji FOR—DOWNTO podamy w odpowiednim czasie. Biorąc pod uwagę dotychczas poznane własności języka PASCAL, trudno jest znaleźć problem, w rozwiązaniu którego naturalne byłoby zastosowanie instrukcji FOR—DOWNTO.

2. Wybrane komendy edytora

Pierwsze próby uruchomienia programu z reguły kończą się niepowodzeniem. Przyczyny są następujące.

— w programie źródłowym wystąpił zapis niezgodny z regułami języka PASCAL (sygnalizowany jest błąd komplikacji),

— podczas wykonania programu wystąpił błąd, np. próba wczytania danej o postaci niedopuszczalnej dla typu zmiennej, której ma być przypisana (sygnalizowany jest błąd wykonania),

— w programie wystąpiły błędy logiczne (program wykonał się, lecz wyniki nie są zgodne z oczekiwanymi). W każdym z tych wypadków należy wywołać edytor przez wciśnięcie klawisza:

— ESC — po sygnalizacji błędu komplikacji lub wykonania; na ekranie monitora zostanie wyświetlony program źródłowy, a kursor wskaże miejsce, w którym wystąpił błąd,

— E — gdy stwierdziliśmy błędy logiczne; na ekranie monitora zostanie wyświetlony program źródłowy a kursor będzie wskazywał pierwszy znak w pierwszej linii programu

Wykorzystując komendy edytora programu można poprawić i uruchomić go ponownie. Edytor dostarcza 45 komend, z których opiszemy na razie kilka, naszym zdaniem, niezbędnych

Komendy edytora składają się z jednego lub dwóch znaków. Będziemy je zapisywać symbolicznie w następujący sposób:

— <CTRL> — znak 1 (w przypadku komend jednoznakowych),

— <CTRL> — znak 1 — znak 2 (w przypadku komend dwuznakowych).

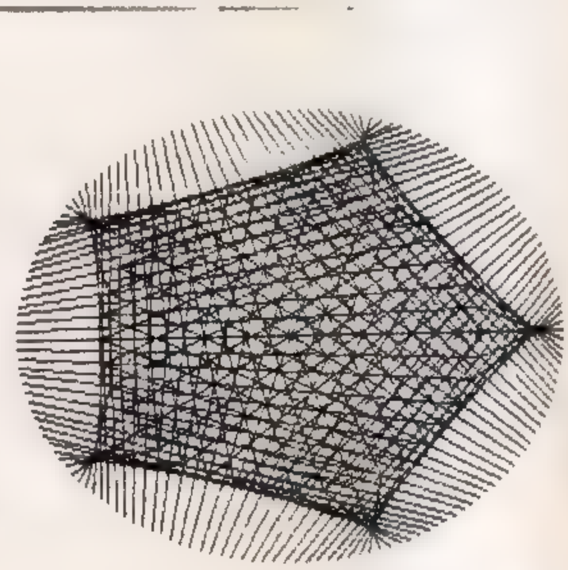
Zapisy te oznaczają, że przy wciśniętym klawiszu CONTROL należy nacisnąć klawisze ze znakiem „znak 1", a dla komend dwuznakowych, należy nacisnąć następnie klawisz ze znakiem „znak 2".

Poprawienie programu może polegać m. in. na:

— wstawieniu nowych znaków w w...

dokończenie na str. 22

GRAFIKA (AMSTRAD)



dokończenie ze str. 21

— skasowaniu określonych znaków w wierszu,
— wstawieniu nowego wiersza do programu,
— skasowaniu określonego wiersza programu.

Edytor ma dwa tryby wstawiania znaków w wierszu. INSERT oraz OVERWRITE. W trybie INSERT nowy znak zostaje wprowadzony na pozycję kursora, a pozostała część wiersza (od kursora w prawo do końca wiersza) jest przesuwana w prawo. W trybie OVERWRITE nowy znak zostaje wprowadzony na miejsce znaku, wskazywanego przez kursor. Zmianę trybu wstawienia znaków powoduje komenda <CTRL> — O—V. Do skasowania znaku znajdującego się przed kursorem wykorzystywany jest klawisz DEL. Do przesunięcia kursora można wykorzystać jedną z komend:

<CTRL> — S — przesunięcie kursora w prawo o jeden znak.

<CTRL> — D — przesunięcie kursora w lewo o jeden znak.

<CTRL> — E — przesunięcie kursora o jeden wiersz w górę.

<CTRL> — X — przesunięcie kursora o jeden wiersz w dół.

<CTRL> — W — tzw. przewinięcie w górę (przesunięcie tekstu wyświetlonego na ekranie o jeden wiesz w dół).

<CTRL> — Z — tzw. przewinięcie w dół (przesunięcie tekstu wyświetlonego na ekranie o jeden wiersz w górę).

Wstawienie nowego wiersza umożliwia komenda <CTRL> — N. Powoduje ona wstawienie nowego wiersza za wierszem, w którym znajduje się kursor. Znaki, od pozycji kursora w prawo do końca wiersza, zostają przeniesione do nowego wiersza. Pozycja kursora się nie zmienia. Do kasowania wiersza, w którym znajduje się kursor, służy komenda <CTRL> — Y. Po skasowaniu wiersza tekst znajdujący się poniżej kasowanego wiersza jest przesuwany o jeden wiersz w górę. Kursor przesuwa się na pozycję pierwszego znaku. Przy stosowaniu tej komendy należy zachować ostrożność, gdyż nie ma możliwości odzyskania skasowanego wiersza.

3. Zadania

Zadanie 4

Napisać program obliczający wartość funkcji

$$F(k) = K!$$

dla zadanej liczby naturalnej K.

Zadanie 5

Napisać program, który wczytuje dwie liczby całkowite i sprawdza, czy są one względnie pierwsze. Dwie liczby są względnie pierwsze, jeżeli nie mają wspólnych podzielników różnych od 1 i —1.

Zadanie 6

Napisać program obliczający cosinus kąta między dwoma wektorami $\{a_i\}$ i $\{b_i\}$
o n współrzędnych z zależności:

$$\cos\alpha = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{1}^{n} a_i^2 \sum_{1}^{n} b_i^2}}$$

Uwaga: współrzędne wektorów wczytywać parami

a_i, b_i (i = 1,2..., n).

Stefan ROZMUS

Z ostatniej chwili

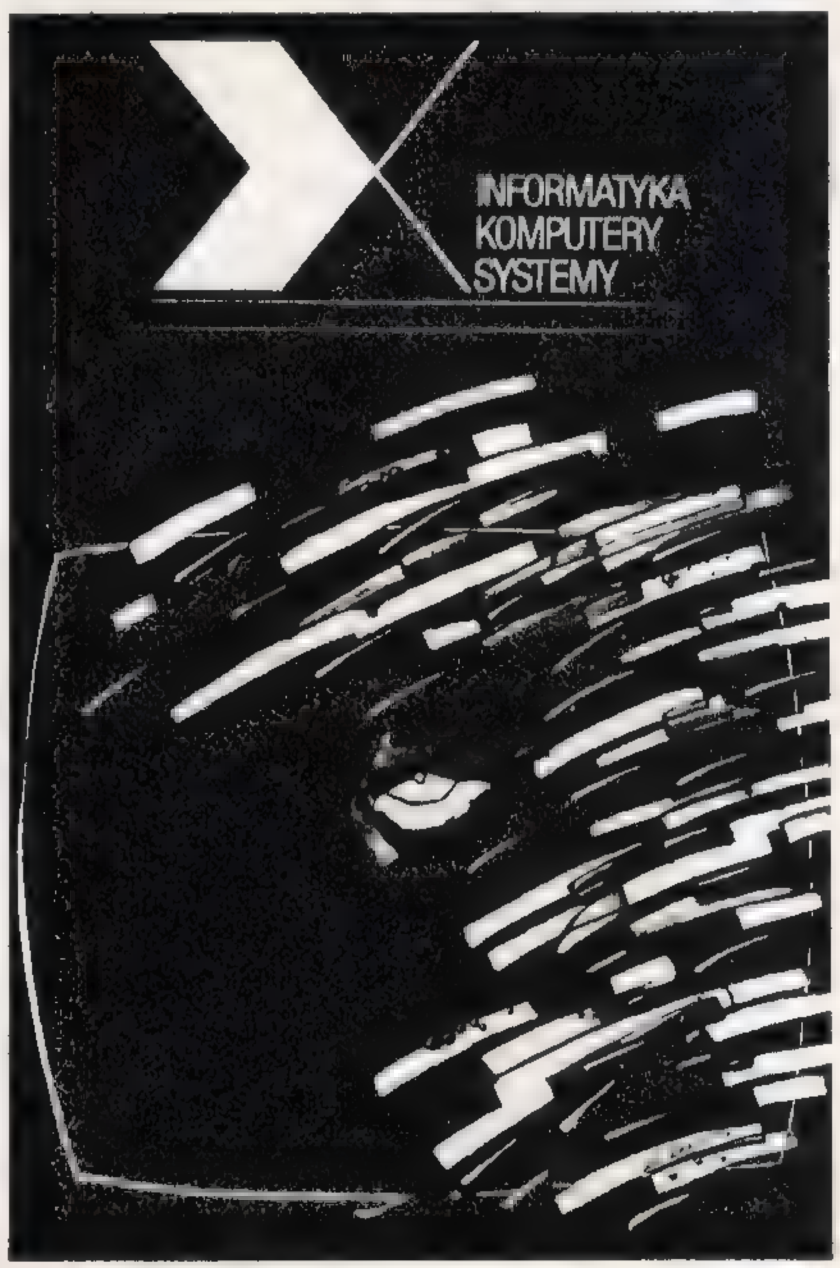
NATIONAL SEMICONDUCTOR 32332

Jest to procesor pracujący z częstotliwością zegara 10,12 lub 15 MHz. Magistrala danych może być zaprojektowana jako 8-, 16- lub 32-bitowa. Mikroprocesor 32332 adresuje 16-Megabajtową przestrzeń adresową, która po dodaniu 32-bitowego rejestru może być zwiększona do 4 Gigabajtów.

CP/M — 8000

Firmy DIGITAL RESEARCH i ZILOG opracowały system operacyjny CP/M dla 16-bitowego mikroprocesora Z-8000 (i jego kolejnych wersji Z-8001, Z-8002, Z-8003, Z-8004).

System wymaga pamięci operacyjnej o pojemności minimum 176 KB, zalecana pojemność 256 KB. CP/M — 8000 działa z kompilatorami języków: BASIC, C, FORTRAN, PASCAL z ASSEMBLEREM i różnymi programami użytkowymi. System może być umieszczony na 8-calowych dyskach elastycznych SS/SD lub 5 1/4 calowych dyskach DS/DD.



„IKS" od stycznia w nowej szacie...

„Zegar" na MERITUM

„Zegar" to podprogram obsługi przerwania NMI mikroprocesora Z-80. Umożliwia stały odczyt czasu podczas pracy z mikrokomputerem.

Wadą programu jest konieczność zatrzymywania „Zegara" przed zapisem i odczytem programów z taśmy. „Zegar" zatrzymuje się instrukcją OUT 250,0 i ponownie uruchamia instrukcję OUT 250,0. Czas uaktualnia się poprzez wpisywanie do odpowiednich komórek pamięci wartości: godzin, minut i sekund. Przykład zapisu nowego czasu do zegara POKE 16500,12' sekundy
POKE 16501,59' minuty
POKE 16502,10' godziny

Wartości adresów są stałe. Operacje związane z zatrzymaniem, uruchomieniem i zmianą czasu w zegarze wykonuje się w trybie kalkulatorowym. W przypadku wyzerowania systemu należy kolejno wykonać instrukcje:
POKE 165526,17: POKE 16527,126: PRINT USR (0)

Spowodują one wykonanie programu maszynowego (od adresu 32273), który wznowi pracę „Zegara". Po wpisaniu i uruchomieniu programu można wykonać rozkaz NEW.

PIOTR BIEDRZYCKI
LVI LO — Warszawa

```
8000      FOR A=32144 TO 32297
8005 READ A$
8010 N=ASC(LEFT$(A$,1))-48
8015 N=N-INT(N/12)*7
8020 M=ASC(RIGHT$(A$,1))-48
8025 M=M-INT(M/12)*7
8030 POKE A,N*16+M
8035      NEXT A
8040  ' KOD MASZYNOWY ZEGARA
8045 DATA F5,3E,B0,D3,FB,3E,C2,D3
8050 DATA FA,3E,D8,D3,FA,E5,C5,CD
8055 DATA AA,7D,CD,DC,7D,C1,E1,F1
8060 DATA ED,45,21,73,40,34,3E,1E
8065 DATA BE,C0,36,00,3C,D3,FE,3D
8070 DATA D3,FE,23,34,3E,3C,BE,C0
8075 DATA 36,00,23,34,BE,C0,36,00
8080 DATA 3E,FF,06,35,D3,FE,10,FE
8085 DATA 3D,20,F7,3E,18,23,34,BE
8090 DATA C0,36,00,C9,D5,11,74,40
8095 DATA 21,3F,3C,06,03,CD,F4,7D
8100 DATA 2B,36,3A,2B,13,10,F6,23
8105 DATA 36,20,D1,C9,1A,CD,05,7E
8110 DATA E6,0F,F6,30,77,79,E6,0F
8115 DATA F6,30,2B,77,C9,0E,FF,0C
8120 DATA D6,0A,F2,07,7E,A7,C6,0A
8125 DATA C9,00,21,90,7D,22,10,40
8130 DATA 3E,0F,D3,F7,3E,B0,D3,FB
8135 DATA 3E,00,D3,FA,D3,FA,00,C3
8140 DATA CC,06
8145 CLS:PRINT@ 520,"";
8150 INPUT"PODAJ GODZ ,MIN ,SEK "
     ;G,M,S
8155 POKE 16500,S:POKE 16501,M :
     POKE 16502,G
8160 POKE 32294,201:
     POKE 16526,17:POKE 16527,126
     :U=USR(0)    :POKE 32294,0
8165 GOTO 8165
```

dokończenie ze str. 18

Ale Miriam nie reagowała i kontynuowała swój monolog.

— Dean, jesteś znakomity! Doskonale dziś wyglądasz! I nie przejmuj się, że masz trudny dzień. Jesteś w znakomitej formie, widziałam to podczas ćwiczeń. Wykonałeś je lepiej, niż wcześniej, daję słowo. Zasłużyłeś na dużą buźkę! Ale już zbieraj się, bo najwyższy czas

W odpowiedzi wzruszył ramionami. Widział w łazienkowym lustrze swe odbicie, a i ćwiczenia wcale nie poszły mu najlepiej

— Pamiętaj — ciągnęła Miriam — wkrótce stary Bob Andersson zrezygnuje z roboty. Może właśnie dziś Tedowi przyjdzie myśl, żeby ciebie dać na jego miejsce? Zarządzałbyś całym pierwszym zespołem analiz magazynowych! To by było wspaniałe! Dean, mógłbyś nareszcie zmienić mieszkanie, kupić nowy samochód, ożenić się... No, chłopie, nie zmarnuj szansy! Dean, jesteś jednym z najinteligentniejszych facetów spośród wszystkich, jakich znam...

— Kretynko! — wrzasnął dziko Dean. — Zamknij ten swój uśmiechnięty dziób! Przestań powtarzać mi do znudzenia to wszystko!

Cholera wie, czy stary Bob tak szybko odejdzie na emeryturę! Cholera wie, czy Ted zaproponuje mi jego stanowisko, czy nie wybierze tego gnojka Jamesa! Cholera wie, czy zechce mnie ulokować w Chemical Industry! Cholera wie, czy to wszystko prawda, co sobie wymyśliłem! A teraz ty mi to w kółko powtarzasz!

Uśmiechnięta twarz Miriam ani na moment nie zmieniła swego wyrazu. Kwadrofonia nadal przekazywała jej ciepły, łagodny głos.

— Dean, skarbie, więcej pewności siebie. Dziś bez wątpienia dokonasz czegoś, czym zapewnisz sobie przychylność Warnera. Nie przejmuj się niczym. Wal do roboty i wracaj szybko do domu. Będę czekała na ciebie. Śmiało Dean, odwagi, chłopie. Pamiętaj, czekam! Pa, kochany. A teraz do roboty. No, wal...

Dean z ponurą miną chwycił ze stołu ciężki szklany klosz od sera i z całej siły z rozmachem walnął nim w uroczo uśmiechającą się z ekranu buzię. Zanim jednak krucha powłoka ekranu i szkło klosza rozmieniły się na tysiące okruchów, posłyszał jeszcze zachwycony głos dziewczyny:

— Dean, jesteś wspa...

ANDRZEJ MONASTYRSKI

5

BASIC

(Podstawy programowania)

PAMIĘĆ KASETOWA: INSTRUKCJE SYSTEMOWE LOAD, SAVE, VERIFY, MERGE (wiadomości podstawowe)

ZX Spectrum współpracuje z magnetofonem kasetowym. Taśmy magnetyczne (kasety) pełnią rolę pamięci zewnętrznej (pamięci pomocniczej). Na kasety nagrywamy programy przeznaczone do wielokrotnej realizacji.

Do nagrywania programów posługujemy się instrukcją postaci

SAVE „nazwa"

— nazwa — jest to nazwa identyfikująca dany program na taśmie (jest to innymi słowy etykieta programu) zaczyna się od litery i jest ciągiem liter i cyfr, liczba znaków w nazwie ≤ 10

Załóżmy, że wprowadziliśmy program do komputera i chcemy go zapamiętać na taśmie magnetycznej z etykietą (nazwą) „rysunek".

W tym celu musimy wykonać następujące czynności:

1. podłączyć magnetofon do komputera (wejście **„MIC"**) i ustawić taśmę na miejscu pustym (do zapisu)
2. napisać instrukcję:
SAVE „rysuneK" (i **ENTER**)
3. komputer wypisze na ekranie
Start tape then press any key
4. włączyć magnetofon (do zapisu)
5. nacisnąć dowolny klawisz w ZX Spectrum
6. na ekranie pokazują się różnej wielkości paski poprzeczne, poprawny zapis jest sygnalizowany przez komputer napisem w lewym dolnym rogu postaci: Ø OK

Zapisany program na taśmie należałoby sprawdzić, czy dobrze się nagrał. Instrukcja VERIFY porównuje program z taśmy z zawartością pamięci komputera i ma postać:

VERIFY „nazwa"

— nazwa — jest to nazwa identyfikująca dany program na taśmie (etykieta programu)

Sprawdźmy nagrany przez nas program. W tym celu musimy wykonać następujące czynności:

1. przewinąć taśmę tak, aby ustawić ją (mniej więcej) przed początkiem sprawdzanego programu
2. włączyć przewód łączący wejścia: słuchawki w magnetofonie a w komputerze „EAR"
3. napisać instrukcję:
VERIFY „rysunek" (i **ENTER**)
4. włączyć magnetofon (do odczytu)
5. następuje szukanie na taśmie programu o etykiecie (nazwie) rysunek. W trakcie szukania programu komputer wypisuje na ekranie nazwy każdej informacji na taśmie na jaką po drodze natrafi. W momencie znalezienia programu o etykiecie rysunek pojawi się na ekranie napis
Program: rysunek

Następnie na ekranie pokazywać się będą różnej szerokości poprzeczne paski. Poprawne zweryfikowanie (sprawdzenie) programu sygnalizowane jest przez komputer w lewym dolnym rogu ekranu napisem postaci:

Ø **OK,** Ø:1

Do wczytywania programów zapisanych na taśmie służy instrukcja postaci

LOAD „nazwa"

— nazwa — jest to nazwa identyfikująca dany program na taśmie (etykieta programu)

Chcemy wczytać uprzednio nagrany program o etykiecie (nazwie) rysunek.

W celu wczytania programu musimy wykonać następujące czynności:

1. włożyć do magnetofonu odpowiednią kasetę tzn. taką, na której zapisany jest program, który chcemy wczytać
2. ustawić taśmę w miejscu (przybliżonym) naszego programu
3. włączyć przewód łączący wejścia: słuchawki w magnetofonie, a w komputerze „EAR"
4. napisać instrukcję
LOAD „rysunek" (i **ENTER**)
5. włączyć magnetofon (do odczytu)
6. następuje szukanie na taśmie programu o etykiecie (nazwie) rysunek. W trakcie szukania programu komputer wypisuje na ekranie nazwę każdej informacji na taśmie na jaką po drodze natrafi. W momencie znalezienia programu o etykiecie rysunek pojawi się na ekranie napis.
Program: rysunek

Następnie na ekranie pokazywać się będą różnej szerokości poprzeczne paski. Poprawne wczytanie programu sygnalizowane jest przez komputer w lewym dolnym rogu ekranu napisem postaci

Ø OK, Ø:1

Możemy wtedy program wykonywać (**RUN** i **ENTER** itd.) Instrukcja **LOAD** usuwa stary program i zmienne z pamięci komputera przed wczytaniem nowych z taśmy. Istnieje inna instrukcja — **MERGE** — która tego nie robi. MERGE usuwa tylko te linie i zmienne, które zastępuje nowymi, o tym samym numerze lub nazwie

Instrukcja **MERGE** ma postać

MERGE „nazwa

— nazwa — jest to nazwa identyfikująca dany program na taśmie (etykieta programu)

Czynności, które wykonujemy są takie same jak w LOAD.

Instrukcję MERGE stosujemy wtedy, gdy „składamy" dwa programy (lub kilka programów) w jeden.

Przypuśćmy, że mamy dane dwa programy

```
10 REM program 1
20
|
100
110
.
300
```

```
10 REM program 2
20
|
100
310
|
500
```

Chcemy z tych programów złożyć jeden program postaci.

```
10 REM program 2
20                                linie z programu 2
|
100
110                               linie z programu 1
|
300
```

310 · · linie z programu 2

500

Programy nagrane są na taśmę z etykietami (nazwami) program 1 i program 2. Wykonujemy następujące czynności

LOAD „program 1" (i ENTER)

po wczytaniu tego programu (Ø OK) piszemy

MERGE „program 2" (i ENTER)

po wczytaniu tego programu (Ø) naciśnijmy **LIST** i sprawdźmy czy otrzymaliśmy w programie te linie, które nas interesują.

Linie **programu 1** o numerach 10÷100 będą zastąpione liniami programu 2 o tych samych numerach. Linie programu 1 o nr 110÷300 pozostaną bez zmian, natomiast linie o numerach 310÷500 są liniami programu 2.

W instrukcjach **VERIFY, LOAD** i **MERGE** nazwa programu może być tekstem pustym tzn. możemy napisać:

VERIFY" "LOAD" "MERGE"

wówczas komputer nie dba o nazwę programu i bierze pierwszy program z taśmy, na który natrafi.

Uwaga: w nazwach programów (w tych instrukcjach) rozróżniane są duże i małe litery, np. RYSUNEK i rysunek nie oznaczają tej samej nazwy

Jeżeli podczas czytania z taśmy lub sprawdzania (weryfikowania) programu komputer wypisze:

R Tape loading error

oznacza to, że wystąpił błąd we wczytywaniu (program na taśmie został odnaleziony, ale z jakiegoś powodu nie mógł zostać wczytany lub zweryfikowany).

W przypadku weryfikowania, najczęściej należy dany program nagrać od nowa (**SAVE**) i ponownie weryfikować (**VERIFY**)

W przypadku odczytywania (**LOAD** lub **MERGE**) należy zmieniać ustawienie głośności i tonów w magnetofonie. Jeżeli po kilku próbach nie uda się wczytać programu, należy zaniechać wczytywania (szczególnie wtedy, gdy taśma była nagrywana na innym magnetofonie).

Z.W.

Jak to zrobić (2)

Wychodząc z założenia, że nawet najprostszy przykład wyjaśni więcej, niż wielogodzinny wykład chcę podbudować poprzednie rozważania o player'ach programem wykorzystującym tę technikę.

Jest to bardzo prosty program, pomagający w przyswojeniu czterech podstawowych działań arytmetycznych. Są w nim kolejno losowo wybierane działania i generowane ich składniki (czynniki). Pojawiają się one na monitorze w postaci równania z jedną niewiadomą, np:

9 : x =3

a osoba „ucząca" się powinna wprowadzić prawidłową, jej zdaniem, wartość zmiennej x

Cały proces przebiega w kilku częściach o zmiennym stopniu trudności, osiąganym przez zmianę zakresu zgadywanych liczb.

W programie efekt ten otrzymujemy przez zwiększenie liczby wartości możliwych do generowania za pomocą następującej instrukcji:

I=INT (RMD (Ø) * Poziom)

oraz przesunięcie wartości bazowej wykorzystywanej do tworzenia zgadywanych liczb

x = BAZA + I

Pięknie, ale gdzie są nasze *player'y*. Otóż spójrzmy na ekran.

Prezentują one kolejne zmiany przebiegające po udzieleniu prawidłowej odpowiedzi.

W programie technika *player-missile* wykorzystana została do zobrazowania ruchu „ludzika". Jego przesuwanie się jest uzależnione od poprawności odpowiedzi i ilustruje moment zbliżenia się do oceny bardzo dobrej lub sygnalizuje niebezpieczeństwo otrzymania dwójki. Dodatkowo, w czasie gdy grający myśli, jakiej udzielić odpowiedzi, ludzik ze zniecierpliwieniem tupie nogą.

W tekście programu oznaczone zostały odpowiednie fragmenty realizujące „odpowiedzialne" za opisane zachowanie ludzika. By nie odebrać przyjemności rozszyfrowania, jak one działają, nie chcę ich dogłębnie analizować.

Życzę dobrej zabawy. **GRZEŚ**

```
10 OPEN #2,4,0,"K:"
15 DIM A$(20),OP$(1),K$(1)
20 A$="                    "
25 PL=0:GOSUB 2205:REM CZYNNOSCI       POCZATKOWE  I   DEFINIOWANIE
   PLAYERA 0
30 FOR PL=1 TO 3:REM TWORZENIE         POZOSTALYCH PLAYEROW
35 GOSUB 2235:NEXT PL
40 MS=90 REM MIEJSCE WYSWIETLANIA      PLAYERA NA POCZATKU GRY
45 L=5:REM USTAWIENIE LUDZIKA
50 P=4:REM (P-3) OKRESLA POZIOM        TRUDNOSCI
55 D=0:REM PRZESUNIECIE ZAKRESU        LICZB
60 REM S T A R T
65 IF P>8 THEN GOTO 900:REM GRA       MA PIECI POZIOMOW
70 REM ORGANIZACJA EKRANU
75 POSITION 1,8
80 PRINT #6;"2________________5"
85 POKE 53248,MS:REM WYSWIETLENIE      PLAYERA 0 - STOJACY LUDZIK
90 POSITION 11,0
95 PRINT #6,"czesc ";P-3
100 REM GENEROWANIE WYRAZENIA
105 J=D+INT(RND(0)*P)+1
110 I=D+INT(RND(0)*P)+1
115 X=D+INT(RND(0)*P)+1
120 C=INT(RND(0)*8)+1
125 ON C GOTO 130,130,135,140,145,145,150,155
130 I=J+X:OP$="+":GOTO 200+50*(C-1)
135 I=J-X:OP$="-":GOTO 200
140 I=X-J:OP$="-":GOTO 250
145 I=J*X:OP$="*":GOTO 200+50*(C-5)
150 J=I*X:OP$=":":GOTO 200
155 X=J*I:OP$=":":GOTO 250
195 REM WYPISANIE DZIALANIE
200 POSITION 0,2:PRINT #6;A$:POSITION 1,2
205 PRINT #6;J;" ";OP$;" X = ";I
210 C=1:REM WSKAZNIK TYPU DZIALANIA
215 GOTO 300
250 POSITION 0,2:PRINT #6;A$:POSITION 1,2
255 PRINT #6;"X ";OP$;" ";J;" = ";I
260 C=2
295 REM POBRANIE WYNIKU
300 POSITION 0,10
305 PRINT #6;"podaj wartosc  x    ";
310 LK=0:IC=0:POSITION 17,10
315 IF PEEK(764)<>255 THEN 330
320 REM OCZEKIWANIE NA WCISNIECIE         KLAWISZA
325 GOSUB 2320:GOTO 315
330 GET #2,K:IF K=155 THEN 380
335 REM KONTROLA POPRAWNOSCI
340 IC=IC+1:IF IC>3 THEN 300
345 IF K<48 OR K>57 THEN 300
350 PRINT #6,CHR$(K);
355 REM OBLICZANIE WARTOSCI LICZBY
360 LK=10*LK+K-48
370 GOTO 315
375 REM ANALIZA ODPOWIEDZI
380 POSITION 0,10:PRINT #6;A$
```

dokończenie na str. 26

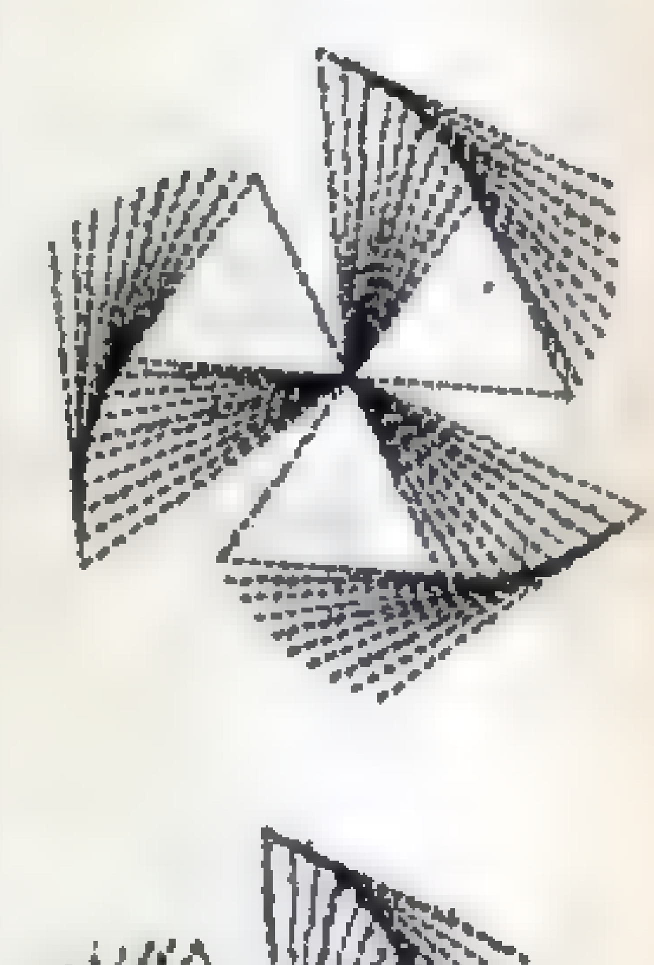
dokonczenie ze str. 25

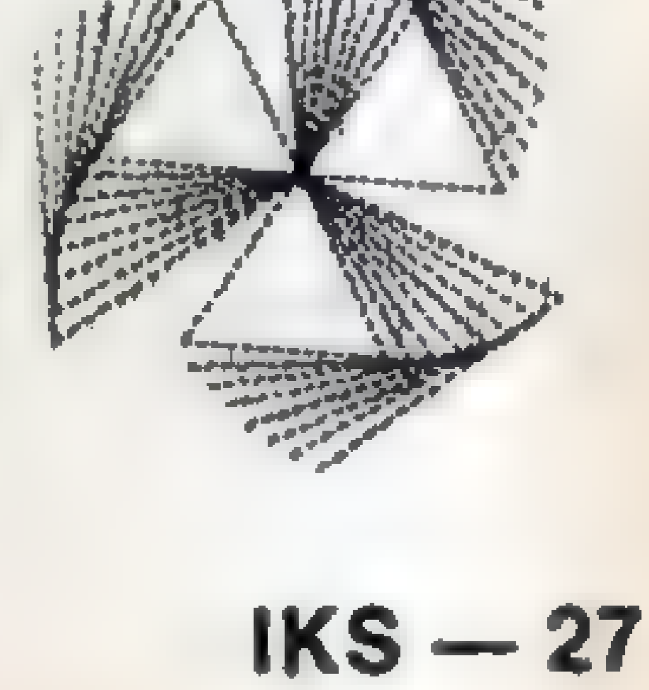
```
385 POSITION 1,2:PRINT #6;A$
390 POSITION 1,2:IF C=2 THEN 405
395 PRINT #6;J;" ";OP$;" ";X;" = ";I
400 GOTO 415
405 PRINT #6;X;" ";OP$;" ";J;" = ";I
410 REM SPRAWDZENIE POPRAWNOSCI
415 IF LK=X THEN 500
420 REM ODPOWIEDZ NIEPRAWIDLOWA
425 L=L-1:IF L<1 THEN 460
430 POKE 712,53:POKE 708,250
435 GOSUB 2500:REM RUCH W LEWO NA        ZMIENIONYM TLE
440 FOR N=1 TO 5:GOSUB 2300:NEXT N
445 POKE 712,250:POKE 708,50
450 GOTO 105
455 REM D W O J A  KONIEC GRY
460 POSITION 1,8:PRINT #6;"2"
465 GOSUB 2300:PRINT #6;CHR$(125)
470 POSITION 1,3
475 PRINT #6;"MUSISZ  JESZCZE"
480 POSITION 3,5
485 PRINT #6;"POPRACOWAC"
490 GOTO 1000
495 REM ODPOWIEDZ POPRAWNA
500 L=L+1:IF L>19 THEN 520
505 GOSUB 2400:REM RUCH W PRAWO
510 GOTO 105:REM NASTEPNE DZIALANIE
515 REM OSIAGNIETA PIATKA
520 POSITION 18,8:PRINT #6;"5"
525 GOSUB 2300:GOSUB 1150
530 PRINT #6;CHR$(125)
535 POSITION 2,3:PRINT #6;"B R A W O"
540 FOR N=1 TO 3:GOSUB 2300:NEXT N
545 PRINT #6;CHR$(125)
550 REM USTAWIENIE PARAMETROW            NASTEPNEJ CZESCI
555 P=P+1:D=D+2:L=5:MS=90
560 GOTO 65
895 REM P I A T K A   KONIEC GRY
900 PRINT #6;CHR$(125)
905 POSITION 5,3
910 PRINT #6;"WSPANIALE ZNASZ"
915 POSITION 2,5
920 PRINT #6;" M A T E M A T Y K E"
1000 GOTO 1000
1145 REM SYGNAL DZWIEKOWY
1150 FOR X=1 TO 3
1155 SOUND 0,82,10,8:GOSUB 2300
1160 SOUND 0,0,0,0:GOSUB 2301
1165 NEXT X
1170 SOUND 0,103,10,8
1175 GOSUB 2300:GOSUB 2300
1180 SOUND 0,0,0,0
1185 FOR N=1 TO 5:GOSUB 2300:NEXT N
1190 RETURN
2200 REM POCZATEK PROGRAMU
2205 M=PEEK(106)-16:GRAPHICS 18
2210 POKE 712,250:POKE 708,50
2215 POKE 709,176
2220 POKE 559,62:POKE 54279,M
```

```
2225 POKE 106,M:POKE 53277,3
2230 REM DEFINIOWANIE PLAYERA  PL
2235 POKE 704+PL,160:POKE 53256+PL,2
2240 S=M*256+1024+PL*256
2245 REM TWORZENIE OBRAZU PLAYERA
2250 RESTORE 2275+PL
2255 FOR L=150 TO 170
2260 READ B:POKE S+L,B
2265 NEXT L
2270 RETURN
2275 DATA 16,56,56,56,16,124,254,186,186,186,186,16,40,40,40,40,40,108,0,0,0
2276 DATA 8,28,28,28,8,62,93,93,93,89,158,8,20,36,68,36,100,12,0,0,0
2277 DATA 16,56,56,56,16,56,124,186,170,186,121,16,40,36,34,36,38,48,0,0,0
2278 DATA 8,28,28,28,8,28,28,28,28,28,28,28,28,8,8,8,8,8,0,0,0
2300 FOR T=0 TO 100:NEXT T:RETURN
2301 FOR T=0 TO 15:NEXT T:RETURN
2315 REM TUPANIE NOGA                    ZMIANA WARTOSCI  W OBRAZIE PLAYERA
2320 S=M*256+1024+166
2325 POKE S,PEEK(S)+128
2330 GOSUB 2301:GOSUB 2301
2335 POKE S,PEEK(S)-128
2340 GOSUB 2301
2345 RETURN
2390 REM R U C H                          POWSTALY PRZEZ WYMIANE PLAYEROW
2395 REM RUCH W PRAWO
2400 POKE 53248,1:GOSUB 2301:ILK=0
2405 ILK=ILK+1:IF ILK=8 THEN 2435
2410 MS=MS+1
2415 POKE 53251,1:POKE 53250,MS
2420 GOSUB 2301
2425 POKE 53251,MS:POKE 53250,1
2430 GOSUB 2301:GOTO 2405
2435 POKE 53251,1:POKE 53248,MS
2440 RETURN
2495 REM RUCH W LEWO
2500 POKE 53248,1:GOSUB 2301:ILK=0
2505 ILK=ILK+1:IF ILK=8 THEN 2535
2510 MS=MS-1
2515 POKE 53251,1:POKE 53249,MS
2520 GOSUB 2301
2525 POKE 53251,MS:POKE 53249,1
2530 GOSUB 2301:GOTO 2505
2535 POKE 53251,1:POKE 53248,MS
2540 RETURN
```

JANUSZ MILLER
SHARP PC-1500
WIATRAKI

```
10:BEEP 10,100,10
   0
20:"A":CLEAR :
   USING :RADIAN
30:INPUT "ILE WIA
   TRAKOW ? N=";N
   :N=ABS N:IF N<
   3GOTO 10
40:INPUT "KOLOR ?
    U=";U:U=ABS U
   :U=INT U:IF U>
   3GOTO 10
50:INPUT "MNOZNIK
    ? M=";M:M=ABS
   M:IF M=0GOTO 1
   0
60:GRAPH :LINE -(
   0,0)-(100,0),9
   :SORGN
70:R=M:IF R>60LET
   M=M-1:GOTO 70
80:S=π/(4*N):D=R/
   (2*N):DIM W(6,
   2)
90:U=U-1:IF U<1
   LET U=1
100:FOR L=1TO N
110:T=0:U=U+1:IF U
   >3LET U=U-3
120:FOR K=1TO 6
130:W(K,1)=R*COS (
   T+F)
140:W(K,2)=R*SIN (
   T+F):T=T+π/3
150:NEXT K:COLOR U
160:FOR K=1TO 6
170:IF K=2OR K=4OR
   K=6GOTO 230
180:X=W(K,1):Y=W(K
   ,2)
185:IF L>1THEN
   LINE -(0,0)-(X
   ,Y),9:GOTO 200
190:LINE -(0,0)-(X
   ,Y),1
200:P=K+1:IF P>6
   LET P=P-6
210:A=W(P,1):B=W(P
   ,2):LINE -(X,Y
   )-(A,B),1
220:LINE -(A,B)-(0
   ,0),1
230:NEXT K:F=F+S:R
   =R+D
240:NEXT L:TEXT :
   END
```





EKRAN GRAFICZNY

COORD 23677/8

Obszar ekranu zajmuje w Spectrum 6912 bajtów (adresy 16384.. 23295), przy czym ostatnie 768 bajtów jest przeznaczone na atrybuty. W trybie graficznym ekran tworzy sieć punktów o architekturze 192 linie na 256 kolumn, lecz dla BASIC'a dostępnych jest 176 linii, gdyż pominięto część edycyjną ekranu. Pomimo dobrych algorytmów procedury graficzne nie są pozbawione drobnych, lecz utrudniających życie elementów. Szczególną krytyką darzą wszyscy sposób przekazywania parametrów do instrukcji DRAW.

Z chwilą włączenia komputera oraz po rozkazach RUN, CLEAR, NEW i CLS punkt kursora graficznego ma współrzędne (0, 0). Spróbujmy stworzyć funkcję ustawiania pozycji tego kursora podobnie, jak to ma miejsce w Amstrad — BASIC (rozkaz MOVE). Wykorzystamy niestety jedyne, graficzne zmienne systemowe COORD. Adres 23677 trzyma kolumnę ostatnio postawionego punktu (współrzędna x), zaś 23678 linię (współrzędna y)

```
200 POKE 23677,x: POKE 23678,y
: RETURN
```

Aby przesunąć kursor do pozycji np. (80, 100) wczytujemy
LET x = 80: LET y = 100 : GOSUB 200

Oczywiście punkt ten nie będzie „zapalony" w przeciwieństwie do instrukcji PLOT 80, 100

Rysowanie linii szczególnie początkującym nastręcza wiele kłopotów. Lepiej przecież podać współrzędne punktu, w którym linia będzie zakończona

```
300 DRAW x- PEEK 23677,y- PEEK
 23678: RETURN
```

Przykładowo, aby narysować linię od punktu (10, 15) do punktu (200, 75) wczytujemy

```
10 LET x=10: LET y=15: GOSUB 2
00
20 LET x=200: LET y=75: GOSUB
300
```

Innym sposobem rysowania linii jest wykorzystanie współrzędnych biegunowych, jak to ma miejsce w LOGO. Należy ustalić azymut w stopniach i podać długość linii w punktach. Oto podprogram

```
400 IF ABS azymut>360 THEN RET
URN
410 PLOT PEEK 23677, PEEK 2367
8
420 DRAW np* SIN (azymut* PI /
180),np* COS (azymut* PI /180)
430 RETURN
```

Parametry podajemy następująco
LET azymut = 45: LET np. = 100: GOSUB 400

Symulacja komend LOGO: prawo i lewo, nie powinna teraz nastręczać trudności

Powróćmy jeszcze do problemu braku możliwości stawiania punktów w dolnych liniach ekranu i nauczmy komputer. Zakładamy, że numery dodatkowych linii graficznych będą od 0 do 15, przy czym punkt (0, 0) będzie w lewym, górnym rogu prostokąta edycji. Stawianie punktów umożliwi nam procedura w języku wewnętrznym 280 o identyfikatorze EDIT—PLOT zapisana pod assemblerem GENS3M21:

```
  5 ;edit-plot   '86
 10              ORG   61000
 20              LD    HL,(23677)
 30              LD    A,H
 40              CP    16
 50              JR    C,NOERR
 60              RST   8
 70              DEFB  4
 80 NOERR        AND   7
 90              ADD   A,80
100              LD    D,A
110              LD    A,H
120              LD    E,192
130              CP    8
140              JR    C,LINIA
150              SET   5,E
160 LINIA        LD    A,L
170              AND   248
180              RRCA
190              RRCA
200              RRCA
210              ADD   A,E
220              LD    E,A
230              LD    A,L
240              AND   7
250              LD    B,A
260              INC   B
270              XOR   A
280              SCF
290 NRBIT        RRA
300              DJNZ  NRBIT
310              EX    DE,HL
320              BIT   1,(IY+87)
330              JR    NZ,OVER1
360              OR    (HL)
365              LD    (HL),A
370              RET
380 OVER1        XOR   (HL)
385              LD    (HL),A
390              RET
```

linia 20 — wydobycie współrzędnych punktów z COORD.
linie 30—40 — współrzędna y nie może przekroczyć 15;
linie 50—70 — błąd, jeśli przekroczyła,
linie 80—220 — wyszukanie bajtu na ekranie, w którym punkt będzie postawiony;
linie 230—310 — wyszukanie numeru ustawionego bitu;
linie 320—330 — czy tryb OVER 1 — skok, jeśli tak;
linie 360—370 — wydruk punktu w trybie normalnym;
linie 380—390 — wydruk punktu w trybie OVER 1.

Aby skorzystać z procedury zastosujmy podprogram

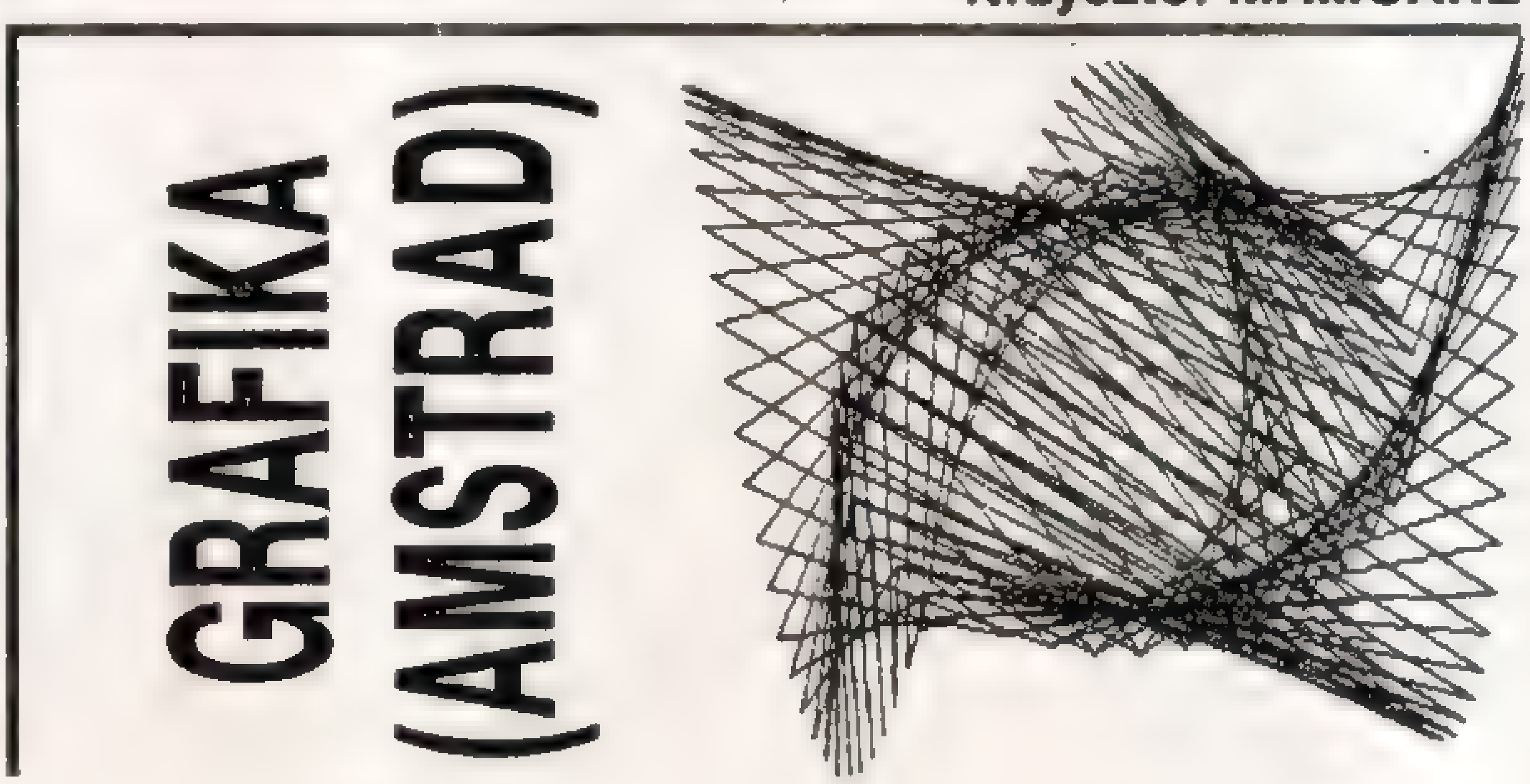
```
500 POKE 23677,x: POKE 23678,y
510 IF USR 61000 THEN
520 RETURN
```

Narysowanie linii prostej wygląda następująco:

```
10 FOR a=0 TO 255
20 LET x=a: LET y=15
30 GOSUB 500
40 NEXT a
50 GOTO 50: REM obrona przed r
aportem
```

Procedura EDIT—PLOT uwzględnia tryb OVER, jeśli go włączymy. W następnym numerze jeszcze o ekranie...

Krzysztof MAMCARZ



Dysponując chwilowo wolnym czasem oraz komputerem, zespół robotów, kierowany przez pracowitego Spektrusia, postanowił założyć komputerowy system matrymonialny, żeby ułatwić ludziom wybór partnera lub partnerki. Roboty zachwycone były tym pomysłem do tego stopnia, że nie brały pod uwagę ewentualności, iż kiedykolwiek ktoś może mieć z tego powodu do nich pretensje.

Przezorny Spektruś od początku miał jednak wątpliwości. Koncepcja, że maszyna dostarczać będzie żonę lub męża jego zdaniem zniechęcała do małżeństwa w ogóle, skoro pretensje będzie można mieć tylko do mikroprocesorów i programu. A jak się coś zatnie? A jak zabraknie części zamiennych? A jak wyłączą prąd? A jak coś nawali? Trzeba będzie przepraszać, a co gorsza przez lata wysłuchiwać wyzwisk: ty tranzystorze skubany!

Mimo Spektrusiowych wątpliwości system ostatecznie uruchomiono, a pierwszym efektem jego pracy było skojarzenie 72-letniego kawalera, który chciał udowodnić, że stary człowiek może z 80-letnią panienką, która zapomniała już, o co mu chodzi, a ponadto była głucha.

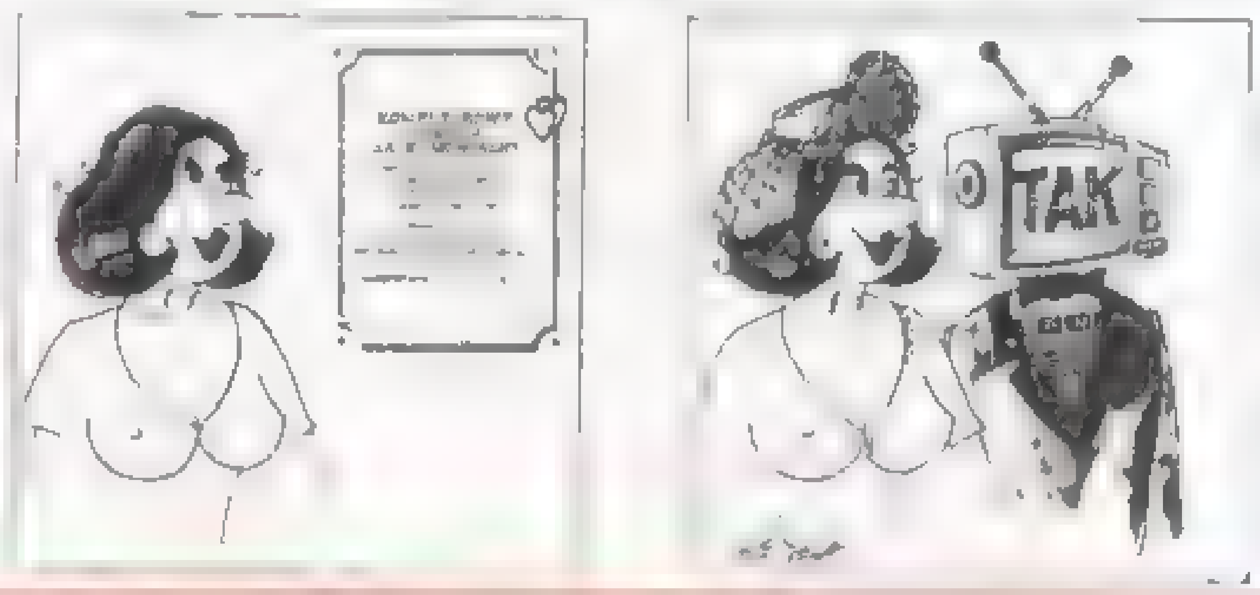
Odpowiednia reklama zapewniała przedsięwzięciu popularność i powodzenie, jednak do komputerowej firmy matrymonialnej zaczęły napływać spostrzeżenia, propozycje i sugestie, wśród których przewijały się koncepcje całowania w różne miejsca i jeszcze bardziej wyrafinowanych pieszczot.

W tej sytuacji Spektruś, widzący jasno swą powinność polecił powołać bezkomputerową, wieloliczydłową firmę rozwodową, „żeby była jakaś całość".

W komputerlandzie znowu wszyscy mają pełne ręce roboty...

Podglądał: **EUGENIUSZ MLECZAK**

W NASZYM KOMPUTERLANDZIE



REDAGUJ RAZEM Z NAMI!

Z pewnością każdy z Was przeszedł dziesiątki labiryntów, stoczył sporo walk z kosmitami, czy też pędził po niejednym torze wyścigowym. Prędko też zauważyliście, że wiele gier jest do siebie podobnych i szybko nudzą. Zatem proponujemy byście razem z nami redagowali namiastkę listy przebojów, a właściwie tych programów, które najchętniej ładujemy do naszego „Atari". Ponieważ niewielu z nas ma stację dysków — umawiamy się tylko na programy kasetowe. Uwzględniamy tylko gry komputerowe i programy użytkowe

Pozycja	Nazwa i rodzaj programu (producent i data)	Grafika	Dźwięk	Atrakcyjność	Szczególne wartości zabawowe, edukacyjne i inne	Ogólna ocena
1	„Spy vs Spy" — cz. II „Walka szpiegów" — cz. II gra przygodowa	10	8	10	Gra na 2 joysticki. Wymagane logiczne myślenie i zdolność podzielnej uwagi oraz szybkiego kojarzenia szczegółów. Dodatkową atrakcją piękną grafika trójwymiarowa. Dużo emocji.	9
2	„The Last V-8" „Ostatnie V-8" gra zręcznościowa Mastertronic 1985	9	10	7	Dojechanie samochodem do bazy przed upływem czasu, wymaga mistrzowskiego opanowania joysticka. Doskonały podkład muzyczny. Wyrabia refleks. — 6 pkt	9
3	„Sega's Spy Hunter" gra przygodowa U.S. Gold	8	8	8	Kto zna scenerię i dynamikę przygód Jamesa Bonda — ten wie już o co chodzi. Szybka akcja z dreszczykiem, samochodami, motocyklami, motorówką etc. Wyświetlane są komunikaty. — 6 pkt.	8
4	„Montezuma's Revenge" (Zemsta Montezumy) gra labiryntowo-przygodowa Utopia Software 83	9	6	10	Pasjonująca gra, w której stawką są zaklęte skarby wodza Azteków. Sceneria koresponduje z wyobrażeniem o azteckich labiryntach czyhających w nich niespodziankach. — 8 pkt.	8
5	„Hacker" (Komputerowy gangster) Activision 1985 gra strategiczno-przygodowa	6	4	10	Sensacyjna przygoda ze skompletowaniem planu „Magma". Aktywny udział w grze, wymagana podstawowa znajomość angielskiego i geografii stref czasowych. Uczy logicznego myślenia	7

Odpadają w przedbiegach te, które zamiast bawić i uczyć, drażnią swą agresywnością lub noszą znamiona hazardu

Proponujemy skalę ocen od 1—10, przy czym ocena końcowa nie jest średnią arytmetyczną. Decydują też walory szczególne programów. A więc, rozpoczynamy zabawę od typów Pawła listopad'86. Zapewne macie lepsze. Czekamy!

O ile gier na „Atari" jest już około 500 — gorzej z programami użytkowymi. Posiadacze DOS-ów mają większe szanse. Ja polecam na razie „Statystykę" i „Elektrownię atomową" w wydaniu polskim, choć szkoda, że za wygórowaną cenę — studio komputerowe nie zdobyło się na oryginalność i programy te przypominają do złudzenia pewne starocie, jak chociażby „SCRAM" – Symulator Reaktora Jądrowego. Ale, lepsze to niż nic

Nadal dobrą zabawą jest „Magic Painter" z bogatym menu — pozwalającym na komputerowe malarstwo — prawie w całym tego słowa znaczeniu

Życzę przyjemnej zabawy

* * *

NA 32 KOLUMNACH

Poniższy program pozwala na wprowadzenie istotnych modyfikacji w obszarze ekranu i Display List — zarządzających generacją obrazu na monitorze. Jason Grant ułożył „pięciolinię", która daje Ci nowe możliwości w porównaniu z edytorem tekstu. Trochę „żonglerki" POKE-ami, a efekt widoczny będzie w postaci 32-kolumnowego edytora ekranu w zerowym trybie grafiki. Przekonaj się sam:

```
10 POKE 82,0:POKE 83,31:GRAPHICS 0:DS=
PEEK(88)+PEEK(89)*256:DL=1539
20 FOR I=1536 TO 1538:POKE I,112:NEXT
I:FOR I=1 TO 24:POKE DL,66
30 POKE DL+2,INT(DS/256):POKE DL+1,DS-
256*PEEK(DL+2):DS=DS+40
40 DL=DL+3:NEXT I:POKE 559,33:POKE DL+
2,6
50 POKE DLL,65:POKE DL+1,0:POKE 560,0:
POKE 561,6
```

NOWE URZĄDZENIE

Jeżeli posiadasz już „Atari" oraz współpracującą z nim drukarkę — pomyśl o elektronicznej maszynie do pisania Nowy wydatek? — ależ nie. Wystarczy pięcioliniowy program by Twój komputerowy keyboard spełniał funkcję

Pocztowa giełda

Rozwiązanie krzyżówki z „IKS-a" nr 6

Hasło brzmi: „PRZYPOMINAMY O PRENUMERACIE IKS-a". Bony pieniężne (1000 zł) wylosowali: Paweł Dąbrowski — Łódź, Łukasz Foltyn — Radom, Danuta Jagiełła — Wrocław, Maciej Śliwiński — Konin, Bogusław Trzeciak — Milicz.

Nagrody książkowe otrzymują: Ewa Cofur — Będzin, Janusz Hełpa — Wolsztyn, Krzysztof Grabacki — Kraków, Wojciech Pilski — Gdańsk, Wiesława Kurczak — Waliszew, Antoni Rudź — Sokółka, Barbara Bańska — Ostrzeszów, Waldemar Jaróg — Nowy Targ, Paweł Hamerlik — Dąbrowa Górn., Adam Darda — Świętochłowice.

Prosimy o podanie dokładnego adresu Piotra Burka z Kamiennej Góry, który wylosował nagrodę książkową za prawidłowe rozwiązanie krzyżówki z nr. 2 „IKS-a".

PILNE! Zakupię następujące materiały firmy SHARP:
1. pisaki czarne nr EA-850 B;
2. pisaki barwne nr EA-850 C;
3. papier (rolki) nr EA-1500 P.

Wiadomość: pisemnie pod adresem redakcji. Telefonicznie: 22-65-93 (Warszawa) Janusz Miller

Kupię — klawiaturę lub folię pod klawiaturę do ZX 81. Michał Pracel, ul. Bosmańska 28/12. 81-116 Gdynia

Kupię schemat ideowy mikrokomputera ZX Spectrum 48 K. Jacek Zieliński, ul. Bieruta 8/7 11-100 Lidzbark Warmiński woj. olsztyńskie

Sprzedam ZX Spectrum 48 K. Joanna Piotrowska, ul. Kniewskiego 25/5, Poznań

Kupię Commodore VIC 20. Adam Rusin, ul. Reja 11/36, 43-300 Bielsko-Biała

Odpowiedzi na zadania „Ligi Myślących" z 5 numeru „IKS-a"

Zadanie 1:

Niech pierwszy robotnik kopie rów w czasie t_1, a drugi w czasie t_2. Z treści zadania wynika, że cały rów robotnicy wykopali w ciągu 12 godzin, stąd w ciągu godziny wykopali następującą część rowu

$$\frac{1}{t_1} + \frac{1}{t_2} = \frac{1}{12}$$

Gdyby pierwszy robotnik sam wykonał połowę pracy, a drugi resztę to wykopaliby rów w czasie 25 godzin, stąd

$$\frac{t_1}{2} + \frac{t_2}{2} = 25$$

A więc otrzymujemy układ równań

$$\begin{cases} \frac{1}{t_1} + \frac{1}{t_2} = \frac{1}{12} \\ t_1 + t_2 = 50 \end{cases}$$

Rozwiązując powyższy układ równań otrzymujemy dwa rozwiązania

$t_1 = 30$ lub $t_1 = 20$

a stąd $t_2 = 20$ lub $t_2 = 30$

Odpowiedź: Wykonując pracę samodzielnie, jeden z robotników kopałby rów przez 20 godzin, a drugi przez 30 godzin

Zadanie 2

W zadaniu tym należało obliczyć ilość lat matematyka greckiego, nazwanego w średniowieczu „ojcem algebry" Diofantosa. Oznaczmy przez x — długość życia Diofantosa

Z treści zadania możemy ułożyć następujące równanie

$$\frac{1}{6}x + \frac{1}{12}x + \frac{1}{7}x + 5 + \frac{1}{2}x + 4 = x$$

stąd $x = 84$

Odpowiedź: Diofantos zmarł mając 84 lata

dokończenie ze str. 30

klawiatury nowoczesnej maszyny do pisania. Spróbuj!

Gdyby w pisaniu tekstów przeszkadzał Ci kursor, możesz „schować" go; wpisując instrukcję: POKE 752, I.

Teraz już możesz napisać do nas na swojej nowoczesnej maszynie. Powodzenia!

```
10 PRINT CHR$(125):OPEN #1,4,0,"K:"
20 OPEN #2,8,0,"P:"
30 OPEN #3,8,0,"S:"
40 FOR GO=0 TO 1 STEP 0:GET #1,K
50 PUT #2,K:PUT #3,K:GOTO 40
```

I jeszcze ciekawostka. Jak podaje „Atari User" — który program opublikował — autorami tej praktycznej „pięciolinii" są uczniowie 5 klasy szkoły podstawowej. A może okażesz się lepszy od nich...„Pogłówkuj"!

Kolumnę opracował:
Paweł Kowalewski uczeń XIV LO im. K. Gottwalda w Warszawie. Wykorzystano teksty z „Atari User" nr 4/86



Rys. M. Przybyłowski

KRZYŻÓWKA NOWOROCZNA



POZIOMO: 1-A komputer jej nie ma, 1-L krąży wokół jądra, 4-A synonim nawałnicy, 4-I część przedstawienia, 4-N rodzaj nadwozia samochodu osobowego. 6-A tygodnik zmotoryzowanych, 6-N namiestnik, 8-A architekt japoński — realizował „Expo-70", 8-N skrzydlaty koń 10-A rodzina chrząszczy, 10-N miasto woj. nad. rz. Mleczna, 12-B. lód w kawałkach, 12-0 jeden z trzech w trójkącie, 14-A gra zręcznościowa, 14-N miasto we Włoszech, 16-A aparatura do sondowania jonosfery, 16-K często kończy się bólem brzucha, 19-A Nazwa pierwszego parowozu; 19-I struś..., 19-Ł polonijne przedsiębiorstwo komputerowe.

PIONOWO: 1-A nauka o przetwarzaniu informacji, 1-D zjednoczenie, 1-F podobno się zdarza, 1-H imię żeńskie, 1-L miasto w woj. suwalskim, 1-M miasto w Holandii, 1-O świątynia buddyjska w Japonii, 1-S, gwiazdy o największych rozmiarach i jasności, 2-J 1987, 4-E państwo w Europie, 4-N park rozrywki, 8-C chorwacki pisarz (1876-1949), 8-P na tarczy herbowej, 13-A srebrna moneta 13-E, produkt węgla, 13-N stragan, 13-S imię męskie, 16-D szpilkowate kości, 16-F Międzynarodowa Organizacjia Radia i Telewizji, 16-H zanieczyszcza atmosferę, 16-L tytuł oficerski w dawnej Turcji, 16-M Stan w USA, 16-O lubi imponować, 17-J pamięć w komputerze.

Hasło: 3-F, 4-S, 16-E, 17-F, 14-P, 17-L, 19-M, 14-N, 4-A, 12-R, 1-D

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca stycznia 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8325. Nr ind. 382809 P-82

1 1987 (9)

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" ISSN 0860—2794 Cena 50 zł



Tajemnice szyfru – nagroda komputer

W szponach ATARI

Noworoczne życzenia, które Wam składaliśmy, są oczywiście życzeniami i dla nas — mamy nadzieję, że nadal pozostaniecie naszymi Czytelnikami.

Zmieniliśmy winietę. Czy będzie się podobać? Chociaż oczywiście nie ona jest w „IKS-ie" najważniejsza. A co wewnątrz? Przede wszystkim więcej programów, krótkich, interesujących i coraz lepszych. Już dziś zapowiedzieć możemy specjalne wydanie „IKS-a", tylko z programami — w kioskach powinno się znaleźć jeszcze w pierwszym kwartale. Będą to programy oryginalne, nigdzie dotychczas nie publikowane, przeznaczone na najpopularniejsze komputery, a napisane w Basicu i Pascalu, których naukę rozpoczęliśmy w ubiegłym roku.

Wśród opublikowanych programów znajdą się również samodzielne prace naszych Czytelników, programy sprawdzone, dobre. Nadal czekamy na Wasze propozycje.

Na łamach „IKS-a" nadal publikować będziemy wykłady (choć to raczej odstraszające słowo), Pascala, Basica — w zmienionej nieco formule. Więcej też będzie informacji o tym, jak lepiej wykorzystywać posiadany sprzęt i to zarówno pod względem programowym (software) jak i sprzętowym (hardware). To wszystko już od pierwszego numeru.

I jeszcze jedno: w ubiegłym roku do rąk Czytelników trafiło osiem numerów „IKS-a", razem 1 600 000 egzemplarzy. W tym roku wydamy 12 numerów, które, jak obiecuje drukarnia, trafiać będą do kiosków regularnie co miesiąc.

Życzymy przyjemnej lektury i pożytecznej zabawy.

REDAKCJA

To była okazja

W radiowym studiu 20 stycznia spotkali się przedstawiciele redakcji wszystkich pism komputerowych. Był **„Komputer"**, był **„Bajtek"** i **„IKS"**. Gospodarzem byli dziennikarze „Radiokomputera".

Spotkanie trwało godzinę. Słuchacze, ale tylko ci, których późna pora emisji (22.50 — 23.50) nie odstraszyła, byli świadkami wymiany jedynie kurtuazyjnych zwrotów, z których nic konkretnego nie wynika. Tymczasem wiemy z pewnością: **wszystkie pisma ukazywać się będą nadal**, a o ich kształcie w dużym stopniu zadecydują Czytelnicy.

Wiesław CETERA

Komputer czeka na Ciebie!

Chcesz być jego posiadaczem, to odczytaj treść ukrytą pod postacią ciąg [illegible]

```
@Ys 2*)"zNr)x¤5s
dUiK1r!#zN"#o!7N
dP^J¤¤Yu4,"/ PsN
d0pU+¤zT*"!-yqcN
```

Z[illegible] drukarce znak $

Ale uwaga! Jest to zbiór znaków kodu ASCII zastosowanego w komputerze Spectrum. Kod ten choć określony jest międzynarodową normą różni się w kilku miejscach w zależności od tego, w jakim sprzęcie został zastosowany. Na przykład ta sama treść zakodowana przez mikrokomputer Sharp ma postać:

```
@Ys 2*)"zNr)x$5s
dUiK1r!#zN"#o!7N
dP^J$$Yu4,"/ PsN
d0pU+$zT*"!-yqcN
```

Nietrudno dostrzec drobne różnice.

Teraz do dzieła. Szukamy klucza. Dla ułatwienia podajemy: **nasz klucz ma szesnaście znaków i jest nim data zapisana w międzynarodowym systemie:**

DATA: RRRR.MM.DD

(np. DATA:1410.07.15 — bitwa pod Grunwaldem, który to zapis nie jest oczywiście poszukiwanym kluczem)!

Na czym polega szyfrowanie i odczytywanie tajnopisów przeczytacie poniżej. Aby odczytać nasz tajnopis nie jest potrzebny komputer — wystarczy kartka papieru i ołówek.

W następnym numerze dla tych, którym nie udało się znaleźć klucza i odczytać zakodowanej informacji podamy bliższe informacje o kluczu. Rozwiązanie konkursu już w maju.

Na zwycięzcę czeka komputer. Zatem do dzieła.

TYLKO DLA WTAJEMNICZONYCH

Każda publikacja (a więc i ta) jest ogólnodostępna. Ten numer „IKS-a" każdy może kupić i przeczytać. Są jednak informacje przeznaczone tylko dla upoważnionych. Jak je przesyłać? W zwykłej papierowej kopercie? Przez kuriera? Jak to chronić? Problem ten (w dużym stopniu) rozwiązuje znana od 50 lat technika szyfracji.

SZYFRACJA polega na takiej zmianie formy, aby treść była możliwa do zrozumienia tylko przez osobę upoważnioną. Zmiana formy zapisu (zwykle) polega na tym, iż dany jest zapis tekstu zastrzeżonego. Klucz szyfrowy oraz metoda zmiany formy zapisu (czyli METODA SZYFRACJI I DESZYFRACJI) znane są nadawcy i upoważnionemu odbiorcy.

Nasza metoda szyfracji jest następująca:

1. pobieramy KOLEJNY znak TEKSTU,
2. pobieramy KOLEJNY znak KLUCZA,
3. wykonujemy „DODAWANIE ZNAKÓW",
4. „SUMA" jest KOLEJNYM znakiem KRYPTOGRAMU.

KRYPTOGRAM ma tę samą treść co tekst, lecz FORMA jego jest taka, iż treść JEST NIEZROZUMIAŁA dla osoby nieupoważnionej.

Ten „nieczytelny" tekst można przesłać zwykłą pocztą. Treść kryptogramu może zrozumieć tylko upoważniony, a więc taki odbiorca, który ma klucz szyfrowy.

A oto metoda deszyfracji

1. pobieramy kolejny znak kryptogramu,
2. pobieramy kolejny znak klucza,
3. wykonujemy „ODEJMOWANIE ZNAKÓW",
4. „Różnica" jest kolejnym znakiem odtworzonego tekstu.

A cóż to jest „dodawanie" czy „odejmowanie znaków"? Czyli jak szyfruje komputer?

Do szyfracji nadaje się każdy (duży) komputer, zaś mikro-

Z	N	Z	N	Z	N	Z	N	Z	N	Z	N
	32	0	48	@	64	P	80		96	p	112
!	33	1	49	A	65	Q	81	a	97	q	113
"	34	2	50	B	66	R	82	b	98	r	114
#	35	3	51	C	67	S	83	c	99	s	115
$	36	4	52	D	68	T	84	d	100	t	116
%	37	5	53	E	69	U	85	e	101	u	117
&	38	6	54	F	70	V	86	f	102	v	118
'	39	7	55	G	71	W	87	g	103	w	119
(	40	8	56	H	72	X	88	h	104	x	120
)	41	9	57	I	73	Y	89	i	105	y	121
*	42	:	58	J	74	Z	90	j	106	z	122
+	43	;	59	K	75	[	91	k	107	{	123
,	44	<	60	L	76	¥	92*	l	108	¦	124
-	45	=	61	M	77	]	93	m	109	}	125
.	46	>	62	N	78	^	94	n	110	~	126
/	47	?	63	O	79	_	95	o	111	●	127**

Z – znak (litera, cyfra)
N – numer ASCII
*) Symbol yena (waluty japońskiej)
**) Ciemne tło

Rys. 1a

Z	N	Z	N	Z	N	Z	N	Z	N	Z	N
spacja	32	1	48	@	64	P	80	£	96	p	112
!	33	2	49	A	65	Q	81	a	97	q	113
"	34	3	50	B	66	R	82	b	98	r	114
#	35	4	51	C	67	S	[illegible]	c	99	s	115
$	36	5	52	D	68	T	84	d	100	t	116
%	37	6	53	E	69	U	85	e	101	u	117
&	38	7	54	F	70	V	86	f	102	v	118
'	39	8	55	G	71	W	87	g	103	w	119
(	40	9	56	H	72	X	88	h	104	x	120
)	41	0	57	I	73	Y	89	i	105	y	121
*	42	:	58	J	74	Z	[illegible]	j	106	z	122
+	43	;	59	K	75	[	91	k	107	{	123
,	44	<	60	L	76	\	92	l	108	|	124
−	45	=	[illegible]	M	77	]	93	m	109	}	125
.	[illegible]	>	[illegible]	N	78	↑	94	n	110	~	126
/	47	?	63	O	79	_	[illegible]	o	111	©	127

Rys. 1b

komputer tylko taki, który ma dwie cechy:
— posługuje się kodem ASCII
— wykonuje operacje znakowe.

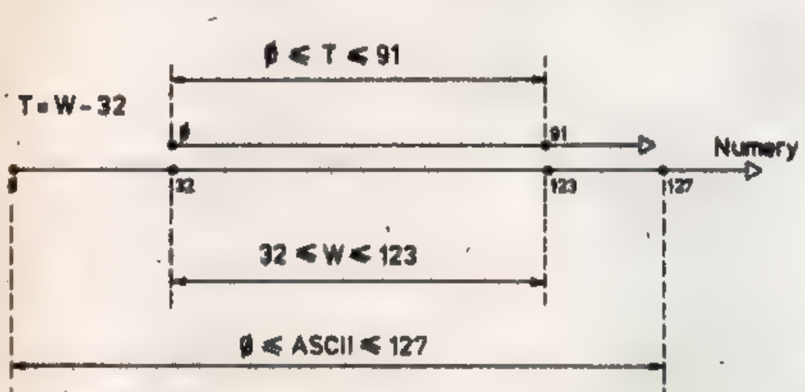
Rysunki 1a i 1b podają tablice międzynarodowego kodu ASCII. 1a to wersja wykorzystana przez firmę SHARP, zaś 1b to kod ASCII zastosowany w „poczciwym" Spectrum. Kod ten polega na tym, iż każdemu znakowi (każdej literze, każdej cyfrze, każdemu symbolowi) odpowiada jeden numer kodu (ten jest zwykle podawany w postaci zero-jedynkowej, my przyjęliśmy dziesiętną).

1. ASC "A" (ENTER) 65
2. CHR$ 65 (ENTER) A
3. A$ = "123ABC" (ENTER) 123ABC
   LEFT$ (A$,4) (ENTER) 123A
4. RIGHT$ (A$,4) (ENTER) 3ABC
5. LEN A$ (ENTER) 6
6. B$ = "WAC" : C$ = "1234"
   B$ = B$ + "−" + C$ (ENTER) WAC−1234
7. O$ = "_ _ _ _ _" (ENTER)
   LEN O$ (ENTER) 5

Uwaga: _ oznacza odstęp

Rys. 2

Rysunek 2 podaje wykorzystane operacje znakowe:
1) przetworzenie znaku na numer kodu ASCII,
2) przetworzenie numeru na znak,
3) wyodrębnienie 4 kolejnych znaków licząc od LEWEJ,
4) wyodrębnienie 4 kolejnych znaków licząc od PRAWEJ,
5) wyznaczenie liczby znaków,
6) rozbudowa pamięci znaku,
7) wypełnienie odstępami pamięci znaku.



Rys. 3

Rysunek 3 podaje zakresy numerów. Kod ASCII operuje numerami od 0 do 127. Wykorzystywana w programie zmienna W rejestruje numery od 32 do 123. Druga zmienna:

$$T = W - 32$$

rejestruje numery od 0 do 91

1. TEKST "P" ⟶ ASC "P" (ENTER) 80
2. KLUCZ "W" ⟶ ASC "W" (ENTER) 87
3. W = numer TEKSTU + numer KLUCZA
   W = 80 + 87 = 167
4. T = W − 32 = 167 − 32 = 135
5. T = 135 > 91 ⟶ T = T − 91 = 135 − 91 = 44
6. W = T + 32 = 44 + 32 = 76
7. CHR$ 76 (ENTER) L
8. Znak SZYFRU "L"

SZYFRACJA (P + W ⟶ L)

Rys. 4

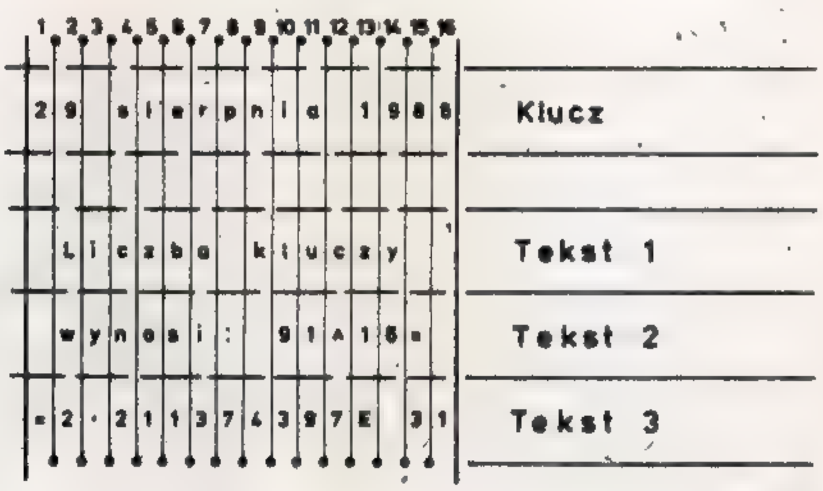
Rysunek 4 podaje przykład szyfracji jednego znaku.
Oto opis OPERACJI SZYFROWANIA:
1) znak TEKSTU przetwarzamy na NUMER kodu ASCII,
2) znak KLUCZA przetwarzamy na NUMER,
3) DODAJEMY NUMERY. Sumą jest zmienna W,
4) wyliczamy zmienną T = W — 32
5) ponieważ zakres zmiennej T zawiera w przedziale od 0 do 91, zatem jej wartość SPROWADZAMY do właściwego zakresu,
6) wyliczamy numer szyfru: W = T + 32,
7) przetwarzamy numer szyfru na znak szyfru,
8) ten znak szyfru dopisujemy do kryptogramu.

Jak widać, SZYFRACJA polega na „dodawaniu znaków":

NUMER KRYPTOGRAMU = NUMER TEKSTU + NUMER KLUCZA

DESZYFRACJA zaś na odwrót — polega na „odejmowaniu znaków":

NUMER ODTWORZONEGO TEKSTU = NUMER KRYPTOGRAMU — NUMER KLUCZA



Rys. 5

Rysunek 5 podaje czynności wstępne SZYFRACJI TEKSTU. Klucz jest naiwny (data ukończenia tego

dokończenie na str. 4

dokończenie ze str. 3

programu). Treścią jest **LICZBA KLUCZY WYNOSI: 91 ↑ 16 = 2.211374397E.31**

Ponieważ w pamięci znakowej SHARP PC-1500 można zmieścić tylko 16 znaków, zaś różnych znaków jest 91 zatem:

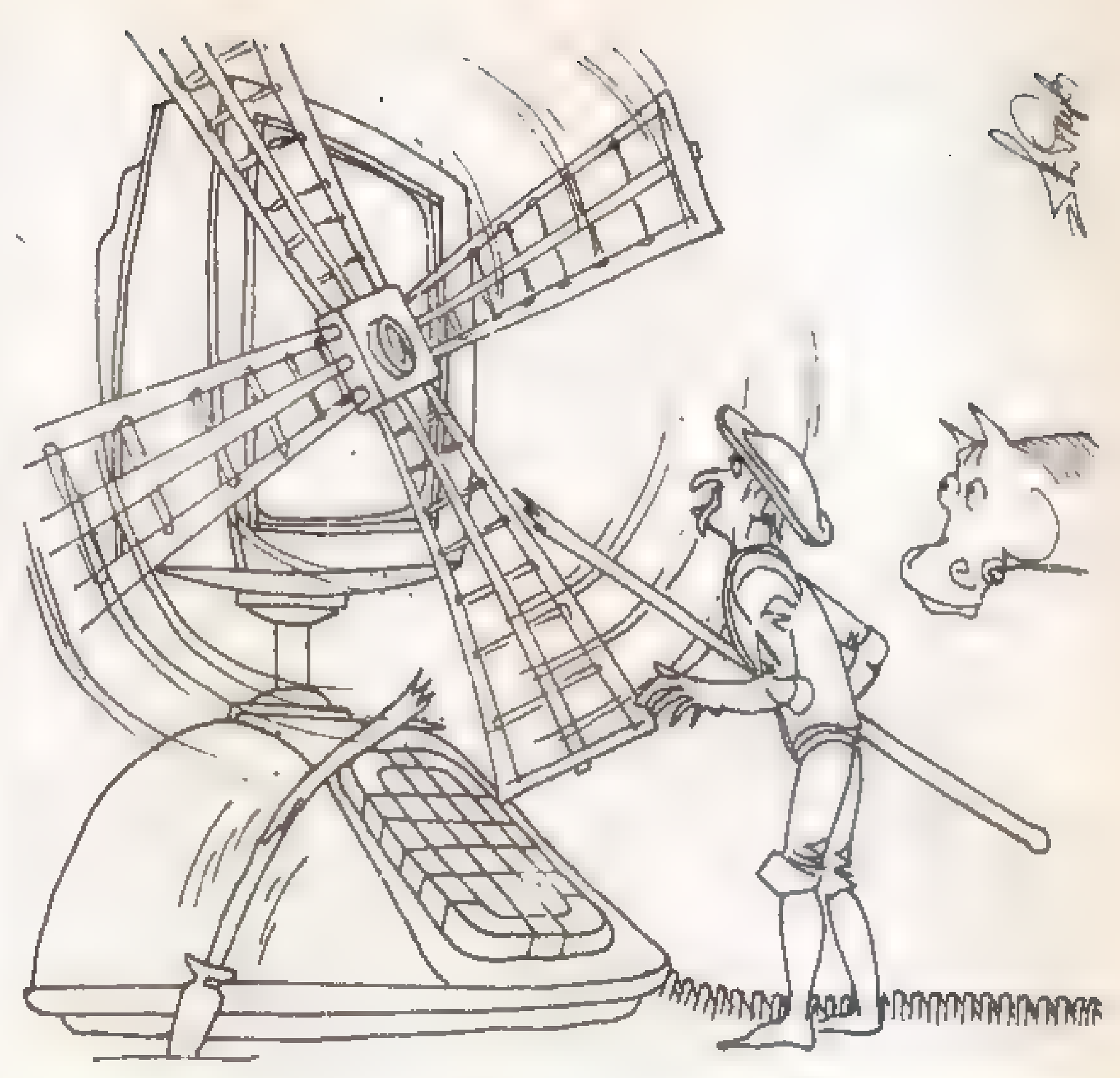
Liczba kluczy
91 ↑ 16

wynosi (ponad) dwadzieścia dwa i... trzydzieści zer, czyli 22 kwintyliony!!!

Janusz MILLER

— program dla SHARPA str. 28

— program dla SPECTRUM str. 30



Rys. Michał Przybyłowski

ABAKUS

Pierwszy w Polsce klub minikomputerowy — ABAKUS powstał w 1983 r. Od listopada 1984 r. ABAKUS jest Stowarzyszeniem Mikrokomputerowym posiadającym osobowość prawną.

Celem jego działania jest popularyzacja mikroinformatyki i jej zastosowań, a w szerszym ujęciu — kształtowanie kultury mikroinformatycznej w PRL.

Stowarzyszenie prowadzi swoją działalność na terenie Warszawy przez kluby komputerowe, istniejące przy Warszawskim Ośrodku Kultury i w Domu Kultury na Bródnie przy ul. Wysockiego. Jeszcze w tym roku zostaną uruchomione kolejne kluby, m.in. w DK w Rembertowie, przy SM Jary na Ursynowie i zostanie wznowiona praca klubu na Woli przy ul. Brożka.

Ale kluby mikrokomputerowe to tylko jedna z form działalności stowarzyszenia.

Cele ABAKUS-a są szersze. Chcemy organizować imprezy popularyzujące mikroinformatykę o ogólnopolskim zasięgu. Będziemy tworzyć płaszczyznę spotkań i wymiany doświadczeń osobom interesującym się informatyką, śledzić i rozpowszechniać światowe osiągnięcia w zakresie zastosowań mikroinformatyki. ABAKUS zorganizował już kilkadziesiąt pokazów, organizował giełdy komputerowe, konkursy. Szerokim echem odbiły się w Polsce organizowane przez Stowarzyszenie — DNI KULTURY MIKROINFORMATYCZNEJ (maj—czerwiec 1985) — DNI KULTURY VIDEOINFORMATYCZNEJ—INFOVIDEO'86 (maj—czerwiec 1986).

ABAKUS nawiązał współpracę w Hobby Computer Club (HCC) — największą w świecie organizacją społeczną zrzeszającą miłośników mikrokomputerów.

Dzisiaj podajemy do publicznej wiadomości wyniki konkursu na grę edukacyjną. Jego uczestnicy zbudowali wspaniałe programy, które jednocześnie uczą i bawią.

W klubach mikrokomputerowych stowarzyszenia prowadzona jest działalność szkoleniowa z podstaw budowy, obsługi i programowania mikrokomputerów dla osób w różnym wieku i z różnych środowisk. Rozpoczynają działalność sekcje zainteresowań, do których zapraszamy dzieci i młodzież. Komputery dla członków stowarzyszenia udostępniamy także na zamówienie — telefonicznie można uzgodnić z instruktorem dzień, godzinę i czas korzystania ze sprzętu.

Zapraszamy wszystkich chętnych do współpracy, do włączenia się w tworzenie kultury mikroinformatycznej w Polsce. Stowarzyszenie opiera swą działalność na pracy społecznej członków, którymi mogą być osoby prawne lub fizyczne. Swoją działalność stowarzyszenie prowadzi na bazie funduszów gromadzonych ze składek członkowskich, darowizn i dotacji.

Stowarzyszenie Mikrokomputerowe ABAKUS działa w tymczasowej siedzibie przy Warszawskim Ośrodku Kultury w Warszawie przy ul. Elektoralnej 12 (tel. 20-48-38 wewn. 205). Kieruje nim prezes Leszek Wilk i dwaj wiceprezesi Krzysztof Świętoń i Dorota Winkler.

Oto wynik konkursu na grę edukacyjną:

I nagroda — mikrokomputer ATARI 130 za program „ABC" L. Kuciński

II nagroda — mikrokomputer TEXAS za program „ORTOGRAFIA" J. Połępa

III nagroda — radiomagnetofon za program „Nauka Jazdy" K. Wysocki

IV, V, VI nagroda — kasety z programami

— J. Kańkowski za program „Mnożenie"

— A. Zimny za program „Naukowiec"

— M. Dąbrowski za program „Sok jabłkowy"





Foto: Jan Zelman

PĘTLA

Konstrukcją programową często używaną w programach jest pętla. Jest ona najprostszym sposobem wielokrotnego wykonywania grupy instrukcji (w szczególnym przypadku może to być pojedyncza instrukcja lub instrukcja pusta), aż do momentu zajścia określonego warunku powodującego zakończenie jej działania. Pętle wymagają odejścia od normalnego, sekwencyjnego interpretowania instrukcji programowych. W celu zdefiniowania pętli niezbędnym staje się określenie następujących elementów jej struktury:

1. Specyfikacja początku i końca pętli.
2. Specyfikacja warunku kończącego pętlę.

Pętle możemy programować wykorzystując podstawowe instrukcje sterujące takie jak IF i GOTO. Instrukcja GOTO definiuje pierwszy z wymienionych elementów struktury pętli, natomiast instrukcja IF definiuje drugi element. Do tworzenia pętli można również wykorzystywać instrukcje przeznaczone specjalnie do realizacji pętli np.: FOR, REPEAT, WHILE i inne w zależności od stosowanego języka programowania. Poniżej przedstawione są trzy przykłady pętli rozwiązujących ten sam problem: obliczenie sumy liczb naturalnych od 1 do N.

A)

```
0 INPUTN
10 FOR I=1 TO N
20 SUM=SUM+I
30 NEXT
40 END
```

B)

```
0 INPUTN
5 I=1
10 IF I>N THEN GOTO 40
20 SUM=SUM+I
30 I=I+1: GOTO 10
40 END
```

C)

```
0 INPUTN
5 I=0
10 I=I+1
20 SUM=SUM+I
30 IF I<N THEN GOTO 10
40 END
```

Wszystkie pętle wymagają wykonania testu pozwalającego określić, czy pętlę należy zakończyć, czy kontynuować dalej? Realizacja tego testu może być na początku pętli (pętla 1B) lub na jej końcu (pętla 1C). Należy zauważyć, że wykonanie testu na końcu pętli powoduje co najmniej jednokrotną realizację grupy instrukcji zawartych w pętli. Umieszczenie testu na początku pętli może wykluczyć w ogóle realizację instrukcji obejmowanych przez pętlę.

Nieumiejętne korzystanie z pętli prowadzi do powstawania programów zbyt wolnych lub zajmujących wiele pamięci. Istnieje kilka praktycznych reguł pozwalających programować sprawne pętle.

W pierwszym rzędzie należy zastanowić się, czy do rozwiązania zadania niezbędne jest wykorzystanie pętli? Otóż okazuje się, że w wielu przypadkach pętle nie są konieczne. Przedstawiony wcześniej problem można rozwiązać bez korzystania z pętli. Kolejne liczby od 1 do N mogą być interpretowane jako wyrazy ciągu arytmetycznego. Suma S_n n początkowych kolejnych wyrazów ciągu arytmetycznego wyraża się wzorem:

$$S_n = \frac{a_1 + a_n}{2} n$$

gdzie: a_1 — pierwszy wyraz ciągu,
a_n — n-ty wyraz ciągu

Korzystając z tej własności program upraszcza się do postaci:

```
10 SUM = N * (1 + N)/2
```

Zyskaliśmy na zajętości pamięci i szybkości działania programu.

W następnych przykładach możemy zatrudnić nasze komputery do wykonywania różnych pętli. Pierwszy z nich obrazuje możliwość łączenia pętli:

```
5 INPUT N
10 FOR I=1 TO N
20 X=X+I
30 NEXT
40 FOR I=J TO N
50 Y=Y-I
60 NEXT
```

```
5 INPUT N
10 FOR I=1 TO N
20 X=X+I
30 Y=Y-I
40 NEXT
```

Zamiast dwóch pętli obejmujących wiersze 10—30 i 40—60 wykonywana jest jedna pętla obejmująca wiersze 10—40. Połączenie pętli pozwoliło na zmniejszenie zajętości pamięci i wzrost szybkości działania programu.

Kolejny przykład ilustruje organizację zagnieżdżeń pętli:

```
10 FOR K=1 TO 100
20 FOR J=1 TO 50
30 FOR I=1 TO 10
40 NEXT:NEXT:NEXT
```

```
10 FOR I=1 TO 10
20 FOR J=1 TO 50
30 FOR K=1 TO 100
40 NEXT:NEXT:NEXT
```

Zakładając, że kolejność występowania pętli nie wpływa na uzyskiwany efekt końcowy możemy organizować pętle w różnej kolejności. Dwa sposoby demonstruje przykład. Realizacja wersji pierwszej na mikrokomputerze C-64 wynosi około 2 minut, zaś wersji drugiej 1 minutę. Co powoduje taką różnicę szybkości działania programów? Otóż zwróćmy uwagę na liczbę otwarć i zamknięć pętli w obu wersjach (wiąże się z tym inicjowanie zmiennej sterującej pętlą oraz testowanie warunku zakończenia pętli)

	wersja A	wersja B
liczba otwarć	1 + 100 + + 5000 = 5101	1 + 50 + + 500 = 551
liczba zamknięć	50000 + 5000 + + 100 = 55100	50000 + 500 + + 10 = 50510
Razem	60201	51061

Jasne staje się teraz, dlaczego wersja B jest szybsza. Należy pamiętać, że pętle należy tak organizować, aby pętle zewnętrzne były powtarzane najmniej razy. Ograniczymy w ten sposób liczbę otwarć i zamknięć pętli.

Następna seria przykładów posłuży omówieniu zasad optymalizacji pętli.

```
5 N=50:DIM T1(N,N),T2(N),T3(N):
Y=1:INPUT W
10 FOR I=1 TO N
20 FOR J=1 TO N
30 IF W=0 THEN S=S+SIN(X)+T1(I,J)
:GOTO 50
40 S=S+T2(I)+T3(J)+X/Y
50 NEXT:NEXT
60 END
```

```
5 N=50:DIM T1(N,N),T2(N),T3(N):
Y=1:INPUT W
10 IF W=0 THEN GOTO 60
20 FOR I=1 TO N
30 FOR J=1 TO N
40 S=S+T2(I)+T3(J)+X/Y
50 NEXT:NEXT:GOTO 100
60 FOR I=1 TO N
70 FOR J=1 TO N
80 S=S+SIN(X)+T1(I,J)
90 NEXT:NEXT
100 END
```

dokończenie na str. 6

dokończenie ze str. 5

```
5 N=50:DIM T1(N,N),T2(N),T3(N):
Y=1:INPUT W
10 IF W=0 THEN GOTO 80
20 XY=X/Y
30 FOR I=1 TO N: TZ=T2(I)+XY
40 FOR J=1 TO N
50 S=S+T3(J)+TZ
60 NEXT:NEXT:GOTO 120
70 SI=SIN(X)
80 FOR I=1 TO N
90 FOR J=1 TO N
100 S=S+SI+T1(I,J)
110 NEXT:NEXT
120 END
```

Ulepszanie pętli należy zawsze zaczynać od pętli wewnętrznych. Zminimalizowanie operacji wewnątrz tych pętli zostaje zwielokrotnione przez działanie pętli. Do podstawowych sposobów pozwalających na uzyskanie wymiernych efektów należą:

- usuwanie niezmiennych wyrażeń z pętli;
- zmniejszanie liczby odwołań do obiektów ze wskaźnikami (np. tablice);
- właściwe ustawianie kolejności wyrażeń logicznych.

Wykorzystywanie dwóch pierwszych sposobów z reguły powoduje wzrost wielkości pamięci zajmowanej przez program, jak również wzrost szybkości działania programu.

Wersja pierwsza w prezentowanych przykładach zajmuje najmniej pamięci, lecz jest najwolniejsza. Drugą wersję uzyskujemy wyciągając badanie warunku W = 0 przed pętle z jednoczesnym rozłączeniem pętli, gdyż jest on niezmiennym wyrażeniem pętli. Wersja trzecia uzyskana jest po usunięciu wszystkich pozostałych niezmiennych wyrażeń z odpowiednich pętli. Czasy realizacji poszczególnych wersji na mikrokomputerze C-64 są następujące.

	wersja 1	wersja 2	wersja 3
W = 1	40 s	32 s	22 s
W = 0	45 s	36 s	26 s

Uzyskiwane w przykładach oszczędności czasu działania poszczególnych wersji programów są rzędu kilku lub kilkudziesięciu sekund. Wykorzystanie jednak wersji wewnątrz kolejnej pętli powoduje zwielokrotnienie oszczędności.

Proponuję przetestowanie ich w omawianym zakresie. Pamiętajmy: pętla pętli nie równa!

S. WASILEWSKI

NA SKRÓTY

Proponujemy wspólną zabawę — pracę. Polegać ona będzie na tłumaczeniu oraz uzupełnianiu najczęściej stosowanych skrótów związanych z mikro. Dzisiaj prezentujemy skróty wybrane (zauważone) przez nas, łącznie z ich oryginalną interpretacją.

Oczekujemy na inne, Waszym zdaniem ważne, skróty oraz ich tłumaczenia (interpretację). W kolejnym numerze „IKS-a" będziemy prezentować aktualny stan naszego słownika.

ASCII — American Standard Code for Information Interchange
AT — for Advanced Technology
BASIC — Beginner's All-purpose Symbolic Instruction Code
BIOS — Basic Input-Output System
bps — bits per second
CAD — Computer-Aided Design
CAM — Computer Aided Manufacturing
CAMAC — Computer Application of Measurments
CD ROM — Compact Disc ROM
CIA — Complex Interface Adapter
CMOS — Complementary Metal-Oxide Semiconductor
CPI — Characters Per Inch
CPS — Characters Per Second
CPU — Central Processing Unit
DCE — Data Communication Equipment
DIP — Dual In-line Package
DMA — Direct Memory Access
DOS — Disc Operating System
DRAW — Digital Read And Write
DTE — Data Terminal Equipment
EARS — Electronic Audio Recognition System
EDRAW — Erasable Optical Disc Technology
EGA — Enhanced Graphics Adapter
ELD — Electroluminescent Display
EPROM — Erasable PROM
FF — Form Feed
FPU — Floating Point Processing Unit
GEM — Graphics Environment Manager
GSX — Graphics System Extension
IBM — International Business Machines
IC — Integrated Circuit
ICL — International Computers Ltd
I/O — Input/Output
LAN — Local Area Network
LCD — Liquid-Crystal Display
LF — Line Feed
MDA — Monochrome Display Adapter
MNU — Memory Menagement Unit
NLQ — Near Letter Quality
OCR — Optical Character Reader
PC — Personal Computer
PGS — Professional Graphics System
PIF — Program Information File
PIPS — Pan Information Processing System
POST — Power-On Self Test
PROM — Programmable ROM
PSG — Programmable Sound Generator
RAM — Randim Access (Read/Write) Memory
ROM — Read Only Memory
RGB — Red-Green-Blue
RISC — Reduced Instruction Set Computer
SCCS — Source Code Control System
SID — Sound Interface Device
VIC — Video Interface Chip
WIMPS — Windows, Icons, Mice and Pull-Down Menus
WORM — Write Once, Read Many
YHP — Yokogawa Hewlett-Packard

WŁODZIMIERZ GOGOŁEK



Rys. Michał Przybyłowski

PROGRAM 1

SPECTRUM

Prosta ewidencja literatury

Wydaje się, że ciekawą kontrpropozycją wobec programu nr 16 BIBLIOTEKA z „IKS-a" nr 4/1986 jest niżej prezentowana „Prosta ewidencja literatury" (Ewidencja). Zasadnicze cechy proponowanej wersji różnią ją od programu Biblioteka.

1. Program Ewidencja jest stosunkowo krótki (ok. 6 razy krótszy od Biblioteki).
2. Pozwala wybierać pozycje, które dotyczą wybranego tematu lub dwóch wybranych tematów jednocześnie. (Łatwo można zwiększyć liczbę tematów, które jednocześnie muszą występować przy określonej pozycji by została ona wybrana).
2. Wprowadzamy do komputera nie nazwy, lecz kody. (Unikamy czasochłonnego przepisywania tekstów na klawiaturze i co ważniejsze — zbiór pamięci RAM mieści swobodnie informacje o ponad 2000 pozycji. Ponadto wprowadzanie informacji o każdej pozycji trwa 2 — 3 sekundy, bowiem każda składa się z kilku lub kilkunastu znaków).
4. Niezbędny jest zeszyt (skorowidz), do którego wprowadzamy informacje głównie tekstowe, nie podlegające komputerowej analizie.

W zasadzie Ewidencja powstała dwa lata temu na konkretne (własne) zamówienie. W chwili gdy stwierdziłem, że aby znaleźć informacje na przykład dotyczące „kolorowych ikon w mikrokomputerach", muszę przejrzeć kilkanaście lub więcej pozycji z mojej półki, bez pewności, że nie pominąłem istotnego źródła danych na ten temat. Oczywiście „propozycja" to nie tylko książka, czasopismo, ale głównie kserokopia lub po prostu własne notatki, streszczenia.

Pierwszym zabiegiem wdrażania Ewidencji było założenie (w zeszycie) mojego prywatnego języka ewidencji, którego słowami są dwuznakowe kody:

a) tematów, np.: DL — drukarki laserowe, UN — unix, CE — ceny, DO — dyski optyczne itp. Praktycznie wyróżniłem 60 tematów,
b) wydawnictw, tytułów, np.: BA — Bajtek, BY — BYTE, IX — IKS,
c) autorów (znaczących) np.: MC — Mc Cormick, EZ — Ezra Shapiro.

Kody b) i c) praktycznie okazały się zbyteczne, zatem można je pominąć. Radzę zwrócić uwagę na a)!

Pozostało jeszcze nadanie numerów każdej pozycji, przejrzenie ich i wyróżnienie tematów, których dotyczą. Informacje te notuję skrupulatnie w zeszycie, na przykład: BI, PR, SP, 495 X zapis ten oznacza (zgodnie z moim słownikiem): Biblioteka, program, Spectrum, pozycja na półce i w komputerze 495, autor nie ma kodu, wydawnictwo IKS. Każdej pozycji można przyporządkować do 15 tematów, jeśli mniej, należy nacisnąć spację.

Po wprowadzeniu naszych pozycji do komputera i na kasety (na wszelki wypadek koniecznie na dwie) chcąc wybrać znaną mi literaturę (o której istnieniu mogłem zapomnieć) na temat PR i SP (czyli programy na Spectrum) wystarczy uruchomić Ewidencję i napisać PR,SP — w odpowiedzi na ekranie ukażą się numery pozycji dotyczące tematów PR i SP np.:

TEMAT: pr, sp — Pozycja 351 X
Pozycja 380 BA
itp.

Pierwsze uruchomienie
LET n = 1
GO TO 10

Po komunikacie: Będzie DIM! Należy wcisnąć dowolny klawisz. Pojawi się informacja: ZAPISAŁEM 0 ELEMENTÓW oraz pytania:
KONTROLA (k)?
DOGRYWANIE (d)?
WYBIERANIE (w)?
odpowiedzią powinien być znak d (bo nie ma jeszcze co kontrolować i wybierać). Na pytanie: podaj numer kolejny, należy nacisnąć klawisz 1.
Na ekranie pojawi się instrukcja: Podaj kod literatury w formacie:
999AABB
999 — 3-cyfrowy numer literatury (jest on sprawdzany), w naszym przypadku numerem tym jest: 001
AA — dwuliterowy kod autora, (opcjonalnie).
BB — dwuliterowy kod wydawnictwa, (opcjonalnie).
Kolejną instrukcją jest: NAPISZ KOD CECHY:
w odpowiedzi piszemy dwuliterowy kod tematu np.: aa. Można go podać do 15 tematów, o każdy zapyta komputer. Jeśli tematów jest mniej niż 15, wystarczy nacisnąć znak spacji i wracamy do kodu literatury. Spacja zamiast kodu literatury powoduje powrót do menu. Teraz już jest co kontrolować i wybierać.
k — kontrola. Należy odpowiedzieć na pytanie: Podaj n: co oznacza, że możemy sprawdzać nasz zbiór od dowolnego miejsca = n.

Gdy zdecydujemy się na wybieranie w, na ekranie pojawia się pytanie: CZEGO SZUKAĆ? PYTAJ, NP.: ab,ac LUB TYLKO ab.

Gdy chcemy szukać literatury dotyczącej tematu programy, piszemy pr, gdy chodzi nam o pozycje dotyczące programów, ale tylko tych, które zamieniają komputer w zabawkę, piszemy pr, gr.

W odpowiedzi komputer przedstawi nam pozycje związane z oboma tematami pr (programy) i gr (gry), w formie wcześniej prezentowanego przykładu. Naciśnięcie dowolnego klawisza powoduje powrót do: CZEGO SZUKAĆ...

Kolejne uruchomienia

Po ściągnięciu programu z kasety uruchamiamy go GO TO 100 nigdy RUN

Składujemy program przez SAVE, oczywiście wcześniej go zawieszając. Proponuję nadawać nazwę z aktualną datą np. ewid231186.

Teksty podkreślone ukazują się na ekranie w wyniku pracy komputera lub naciśnięcia klawisza na klawiaturze komputera.

dokończenie na str. 8

dokończenie ze str. 7

```
 45 REM PROSTA EWIDENCJA LITERATURY
 50 CLS : PRINT "Bedzie DIM!"
 55 PAUSE 0
 60 DIM z$(200,38)
105 POKE 23609,50.
110 CLS
115 PRINT "Zapisalem ";(n-1);" elementow"
120 INPUT "Kontrola    (k)?"'"Dogrywanie (d)?"'"Wybieranie (w)?";k$
122 IF k$="k" THEN  GO TO 600
124 IF k$="w" THEN  GO TO 300
129 INPUT "Podaj numer kolejny";n
130 FOR n=n TO 200
135 INPUT "Podaj kod literatury w formacie:"'"999AABB     ";k$
136 PRINT "
137 PRINT n;" ";k$;" ": IF k$=" " THEN  LET k$="**"
140 LET z$(n)=k$: IF k$="**" THEN  GO TO 120
141 FOR a=1 TO 3
142 IF CODE z$(n,a)<48 OR CODE z$(n,a)>57 THEN  GO TO 148
143 NEXT a
144 GO TO 150
148 PRINT "BLAD kodu literatury": PAUSE 0: GO TO 135
150 IF k$="**" THEN  GO TO 120
155 IF VAL z$(n,1 TO 3)=n THEN  GO TO 160
157 PRINT "BLAD numeru": PAUSE 0: GO TO 135
160 FOR w=8 TO 37 STEP 2
170 INPUT "Napisz kod cechy:";k$
173 PRINT "   ";k$;", "
175 IF k$=" " THEN  LET k$="**"
180 LET z$(n,w TO w+2)=k$
190 IF k$="**" THEN  GO TO 205
200 NEXT w
210 NEXT n
220 PRINT "Przepelnienie": STOP
310 DIM l$(5)
320 CLS : INPUT "Czego szukac?. Pytaj,np.:ab,ac  lub tylko ab";l$(1 TO 5)
340 FOR n=1 TO 200
350 FOR w=8 TO 37 STEP 2
360 IF z$(n,1 TO 2)="**" THEN  GO TO 400
370 IF l$(1 TO 2)=z$(n,w TO w+1) THEN  GO TO 420
380 IF z$(n,w TO w+1)="**" THEN  GO TO 395
390 NEXT w
395 NEXT n
400 PRINT  INVERSE 1;"Brak tematu w zbiorze": PAUSE 0: GO TO 300
420 REM Drugi klucz
430 IF l$(3)<>"," THEN  GO TO 520
440 FOR w=8 TO 37 STEP 2
450 IF l$(4 TO 5)=z$(n,w TO w+1) THEN  GO TO 520
460 GO TO 380
520 REM Druk
530 PRINT "Temat: ";l$(1 TO 2);",";l$(4 TO 5);". Pozycja: ";z$(n,1 TO 7)
540 GO TO 380
610 INPUT "Podaj n: ";n
620 FOR n=n TO 200
650 PRINT " ";n;" ";z$(n,1 TO 7);" ";z$(n,8 TO 38)
660 IF z$(n,1 TO 2)="**" THEN  GO TO 700
670 NEXT n
700 PAUSE 0: GO TO 100
```

W.G.

PROGRAM 2

AMSTRAD

```
10 ' ** AMSTRAD BASIC
20 ' ** kolo '86
30 ' ** !KOLO,P1,P2,P3,P4
40 ' ** P1,P2 - wsp. srodka
50 ' ** P3    - promien
60 ' ** P4    - INK
70 DATA "180C6B004600320031000000000003EF
6DDBDC0CDCCBBED530AA0220CA0DD7E00CDDEBBD
D6603DD6E022204A0DD5607DD5E06DD6605DD6E0
4CDC9BB010000ED4306A02A04A02208A0CB25CB1
4E5D1210300AFED522202A02A06A0ED5B08A0E5D
5CDC8A0D1E1AFED"
80 DATA "52FA76A0ED5B0AA02A0CA0CDC9BBC92
A02A0010000ED42F29BA0ED5B06A0CB23CB12CB2
3CB1221060019ED5B02A01918242A06A0ED5B08A
0AFED52CB25CB14CB25CB14110A0019ED5B02A01
9ED5B08A01BED5308A02202A02A06A0232206A01
88FED5B06A02A08"
90 DATA "A0CDEABBED5B08A02A06A0CDEABB210
000ED4B06A0AFED42E5E5ED5B08A0CDEABBD12A0
8A0CDEABB210000ED4B08A0AFED42E5E5ED5B06A
0CDEABBD12A06A0CDEABBE1D1E5D5CDEABBE1D1C
DEABBC9012BA12127A1CDD1BCC9FCA62BA130A1C
30EA04B4F4CCF00"
100 RESTORE
110 MEMORY &9FFF
120 bajt=&A000
130 FOR a=1 TO 3
140 READ a$
150 k=1
160 GOSUB 200
170 NEXT
180 CALL &A11D
190 DELETE -250
200 WHILE k<207
210 POKE bajt,VAL ("&"+MID$(a$,k,2))
```

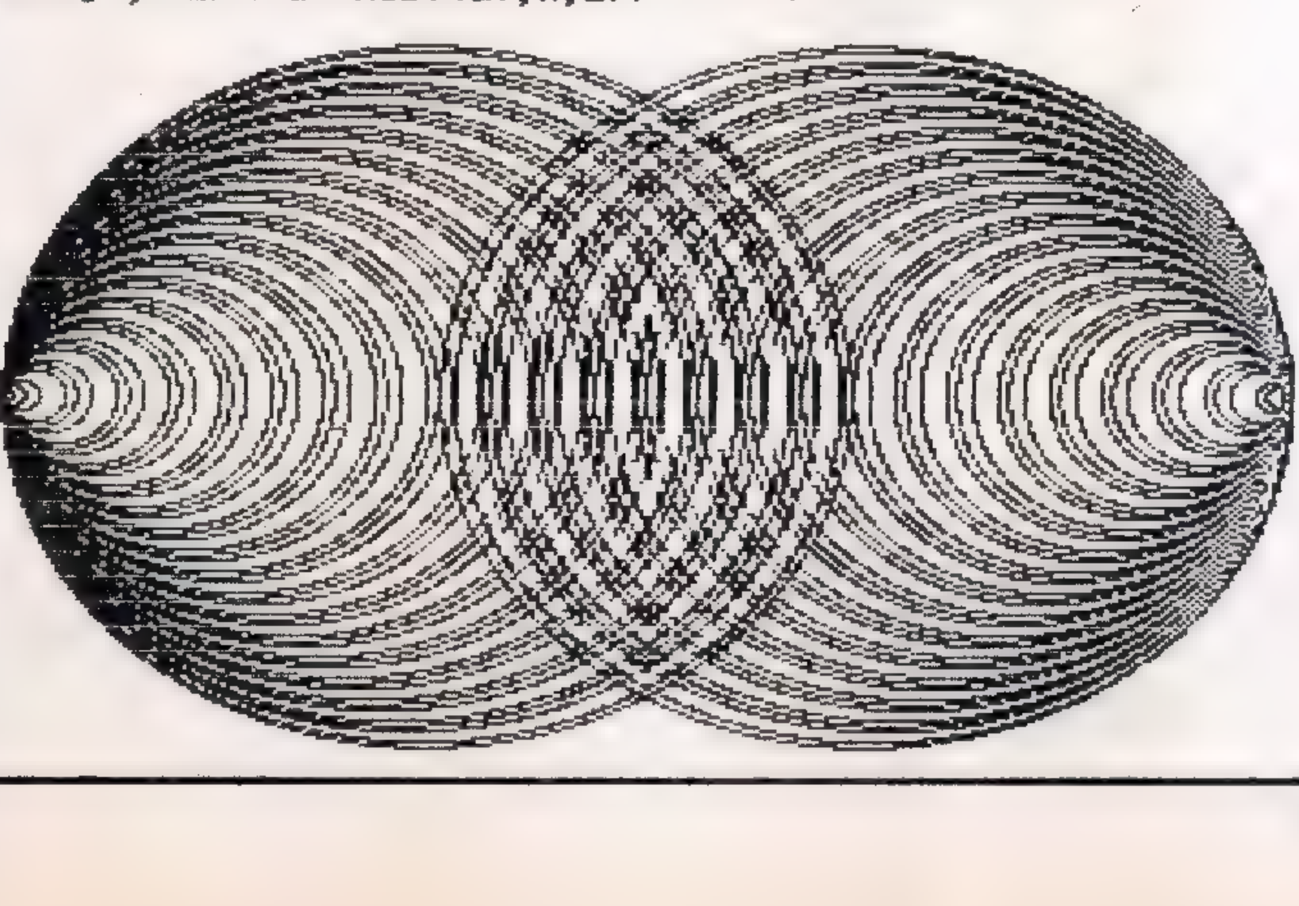
Dość dziwne wydaje się, że przy tak dużych możliwościach Amstrada zabrakło w nim procedury, która rysowałaby koło. Prezentowany program powoduje zainstalowanie komendy, która tę lukę wypełnia. Po jego uruchomieniu nastąpi wczytanie kodu maszynowego od adresu & A000, na który składają się procedury: instalacyjna i obsługująca.

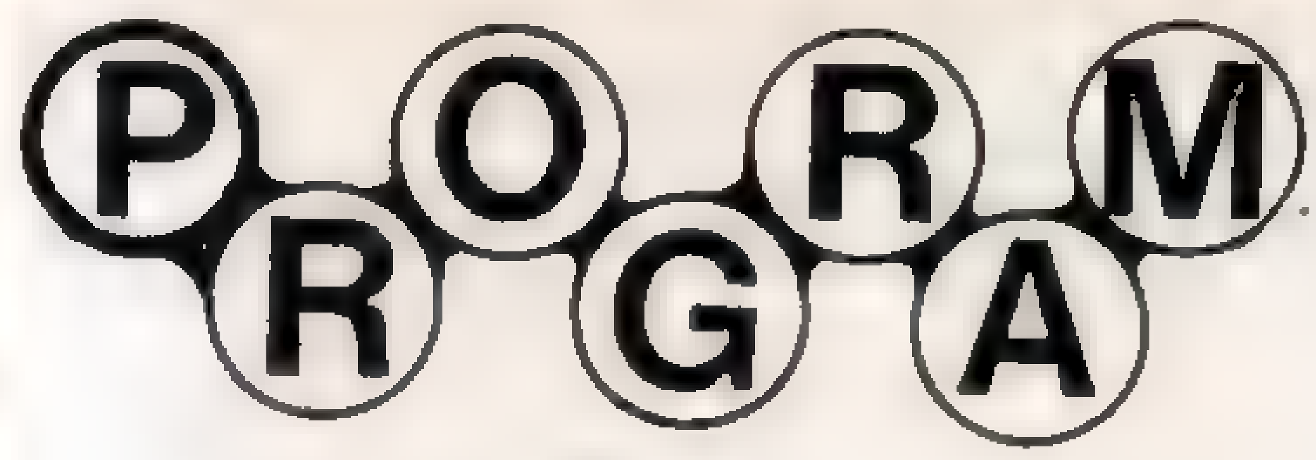
Nowa komenda: KOŁO wymaga czterech parametrów: współrzędnych środka, długości promienia oraz numeru kałamarza (INK), z którego ma być wzięty kolor do narysowania figury. Np.

: KOŁO, 320, 200, 50, 3

Rysowanie odbywa się dość szybko, a wnętrze koła może być wypełnione przez komendę FILL (co jest niemożliwe przy metodzie podanej w instrukcji firmowej). Jako zachętę do wykorzystywania procedury podajemy krótki program demonstracyjny, wyglądający w kolorach bardzo efektownie.

```
220 k=k+2
230 bajt=bajt+1
240 WEND
250 RETURN
260 ' DEMONSTRACJA
270 ' FANTAZJA
280 CLS
290 FOR a=0 TO 200 STEP 3
300 !KOLO,a,200,a,a
310 !KOLO,600-a,200,a,a
320 NEXT
330 END
```





3

SPECTRUM

Można inaczej — Głos mają Czytelnicy

Pamięć programu

Opublikowany w „IKS" nr 2/86 program „Pamięć operacyjna" posiada dwie istotne niedogodności.

1. listowanie rozpoczyna się zawsze od pierwszego wiersza programu — co przy długich programach jest bardzo niewygodne,
2. program „nie wie", gdzie kończy się program w BASICu i listuje pamięć obszaru zmiennych co dezorientuje użytkownika — (pojawiają się numery nieistniejących w programie wierszy).

Pozwalam sobie zatem zaproponować zmiany w wierszach 9010 i 9020 oraz dopisanie wierszy 9015, 9016 i 9017.

Znak ⁘ zastępuje nieczytelny na drukarce znak $

```
9000 REM  86.1  LIST MEM
9010 CLS: LET s=PEEK 23635+PEEK 23636*256-1: LET  k=PEEK  23627+
     PEEK  23628*256-1:  LET  p=PEEK   (s+1)*256+PEEK   (s+2):
     INPUT "LIST od wiersza:";(p);" t/n "; LINE a$:  IF  a$<>"n"
     THEN GO TO 9210
9015 INPUT "wiersz:";w: PRINT FLASH 1; "CZEKAJ": FOR i=s  TO  k:
     IF PEEK i=13 THEN LET p=PEEK (i+1)*256+PEEK (i+2): IF  p>=w
     THEN LET s=i+1: CLS: GO TO 9220
9016 NEXT i
9017 PRINT w; "NIE MA TAKIEGO WIERSZA!": BEEP  1,1:  PAUSE  170:
     GO TO 9010

9020 LET p=PEEK s
9030 IF p=13 THEN GO TO 9200
9040 IF p=14 THEN GO TO 9300
9050 IF p<10 THEN PRINT s; "    "; " .  ";
9060 IF p>9 AND p<100 THEN PRINT s; "    ";p;"     ";
9070 IF p>99 THEN PRINT s; " :  ";p; " ·  ";
9080 IF p>31 THEN PRINT PAPER 6; CHR$ p
9090 IF p<32 THEN PRINT
9100 LET s=s+1: GO TO 9020
9200 PRINT s;  "    ";p;  "    ";  PAPER 4;   "KONIEC  WIERSZA":
        IF s=k THEN INPUT FLASH 1; "KONIEC PROGRAMU (ENTER)";
        LINE a$: STOP
9205 PRINT
9210 LET s=s+1: LET p=PEEK s*256+PEEK (s+1)
9220 PRINT s; "    "; PAPER 4; "WIERSZ NR: ";p; "
9230 LET s=s+2: LET p=PEEK s+PEEK (s+1)*256
9240 PRINT s; "    "; PAPER 6; "Ilosc bajtow= ";p
9250 LET s=s+1: GO TO 9100
9300 PRINT s; "    ";p; "    "; PAPER 4; "Stala liczbowa:"
9310 PRINT s+1; ".   ";
9320 FOR f=s+1 TO s+5: PRINT PEEK f; " , ";: NEXT f
9350 PRINT: LET s=s+5: GO TO 9100
```

W wierszu 9010: zmienna s = start programu w BASICu (zmienna systemowa PROG), zmienna k = koniec programu w BASICu (zmienna systemowa VARS), zmienna p = numer pierwszego wiersza programu, zmienna a = pytanie czy listować od wiersza (p).

W wierszu 9015 wprowadzenie poszukiwanego numeru wiersza — zmienna (w), w pętli szukanie adresu zadanego wiersza.

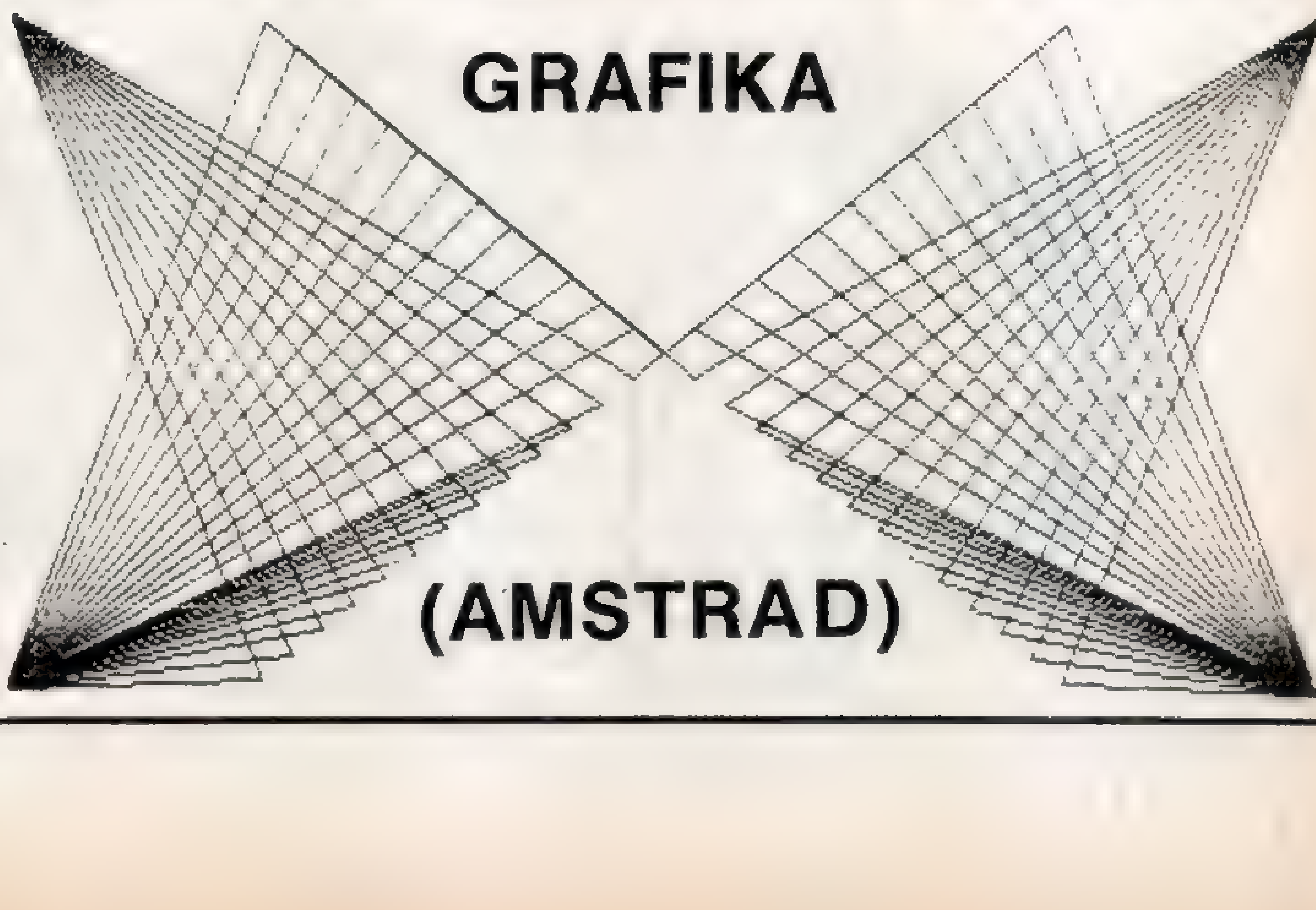
W wierszu 9017 wydrukowanie komunikatu — jeśli zadany numer wiersza nie istnieje.

W wierszu 9020 po wydrukowaniu komunikatu: „KONIEC WIERSZA" sprawdzenie czy jest to koniec programu.

ANDRZEJ HOROCH

GRAFIKA (AMSTRAD)

```
10 CLS
15 ORIGIN 320,200
20 FOR x=1 TO 13
25 DRAW 15*x,-15*x
50 DRAW 320,200
60 ORIGIN 320,200
75 DRAW 15*x,15*x
85 DRAW 320,-200
86 NEXT x
90 ORIGIN 320,200
93 FOR y=1 TO 13
95 DRAW -15*y,-15*y
100 DRAW -320,200
105 ORIGIN 320,200
110 DRAW -15*y,15*y
115 DRAW -320,-200
120 NEXT y
```



PROGRAM 4

ATARI

KOMPUTER ZEGAREM

Jednym z częstszych pytań zadawanych przez posiadaczy ATARI 800XL jest pytanie o sposób pomiaru czasu przez komputer. Dlatego w bieżącym numerze przedstawimy wykorzystanie jednego z zegarów programowych. Jest to tak zwany zegar czasu bieżącego (RTCLOCK), który liczy od zera i przechowuje wartość trzybajtową. Znajduje się on w pamięci RAM o adresach 18, 19, 20 i działa od włączenia komputera do chwili jego wyłączenia. Zliczą on przerwania VBLANK, które występują w regularnych odstępach czasu, zgodnie ze standardem sygnału telewizyjnego (co 1/60 sekundy dla NTSC i co 1/50 sekundy dla PAL). Każde z tych miejsc pamięci działa jak część licznika. Procedura obsługi przerwania systemu operacyjnego dodaje jedynkę do bajtu o adresie 20. Gdy jego zawartość osiągnie 255, następne przerwanie powoduje dodanie jedynki do bajtu 19 i wyzerowanie bajtu 20. Sytuacja taka występuje co 5,1 sekundy. Gdy wartości bajtów 19 i 20 osiągną liczbę 255, następne przerwanie powoduje dodanie jedynki do bajtu o adresie 18 i wyzerowanie bajtów 19 i 20. Zdarza się to co 21,845 minuty. Z chwilą, gdy wszystkie trzy bajty osiągną wartość 255, następne przerwanie powoduje ich wyzerowanie. Zjawisko takie wystąpi po 16 777 215 przerwaniach, czyli po około 93,2 godzinach.

Dla sprawdzenia powyższych wywodów wprowadźmy do pamięci komputera następujące instrukcje.

```
10 CZAS=PEEK(18)*65536+PEEK(19)
*256+PEEK(20)
20 PRINT "CZAS= ";CZAS
```

Po napisaniu RUN (i RETURN) otrzymamy na ekranie liczbę. Gdy po chwili ponownie napiszemy RUN, otrzymamy inną, większą liczbę. Jest to liczba przerwań VBLANK od momentu włączenia komputera. Aby otrzymać czas włączenia komputera, musimy tę liczbę podzielić przez 50. W tym celu zmodyfikujemy powyższe instrukcje następująco:

```
10 CZAS=PEEK(18)*65536+PEEK(19)
*256+PEEK(20)
20 CZAS=INT(CZAS/50)
30 PRINT "CZAS= ";CZAS;" s"
```

W celu rozpoczęcia pomiaru czasu musimy oznaczyć moment rozpoczęcia pomiaru. Możemy to zrobić jednym z następujących sposobów:

1) zapamiętując aktualną wartość zegara czasu bieżącego,
2) wstawiając zera do komórek o adresach 18, 19 i 20 pamięci.

Poniżej podajemy przykład wykorzystujący pierwszy sposób. Po oznaczeniu chwili początkowej możemy w każdym momencie określić czas, jaki minął od tej chwili. W poniższym przykładzie wykorzystujemy tę samą procedurę odczytu czasu (linie programu 6000—6020) zarówno do oznaczenia początku pomiaru (linie 100, 110), jak i do odczytu bieżącej wartości zegara (linia 140). Po odjęciu obu wartości (linia 150) otrzymujemy czas oczekiwania komputera na odpowiedź, oczywiście w sekundach.

```
100 GOSUB 6000:REM ODCZYT CZASU
 POCZATKOWEGO
110 CZASP=CZAS
120 PRINT "ILE TO JEST 4*7 ";
130 INPUT X
140 GOSUB 6000:REM ODCZYT CZASU
 KONCOWEGO
150 CZASU=CZAS-CZASP
160 PRINT "POTRZEBOWALES NA ODP
OWIEDZ AZ ";CZASU;" S."
170 STOP
6000 REM PROCEDURA ODCZYTU CZAS
U
6010 CZAS=INT((PEEK(18)*65536+P
EEK(19)*256+PEEK(20))/50)
6020 RETURN
```

Na zakończenie proponujemy program wykorzystujący zegar czasu bieżącego i używający drugiego z wymienionych sposobów oznaczania momentu zainicjowania zegara, to jest wyzerowania zawartości komórek o adresach 18, 19 i 20. Poniższy przykład demonstruje sposób przerobienia komputera na budzik, możliwe jest ustawienie aktualnego czasu oraz wybranie chwili wszczęcia alarmu.

```
10 REM ZEGAR Z BUDZIKIEM
20 GRAPHICS 2:POKE 752,1:? "}"
25 ? "PODAJ AKTUALNY CZAS"
30 GOSUB 300
35 REM INICJOWANIE ZEGARA
40 POKE 20,0:POKE 19,0:POKE 18,
0
45 GODZ=G:MIN=M:SEK=S
50 ? "PODAJ CZAS BUDZENIA"
55 GOSUB 300
60 GA=G:MA=M:SA=S
65 IGODZ=GODZ:IMIN=MIN:ISEK=SEK
70 GOSUB 200:REM ODCZYT CZASU
75 IF (GA=GODZ) AND (MA=MIN) AN
D (SA<SEK) THEN GOSUB 100
78 IF SEK=0 THEN ? #6;"}"
80 POSITION 5,5:? #6;GODZ;":";M
IN;":";SEK
85 GOTO 70
90 REM SYGNAL BUDZIKA
100 FOR I=100 TO 10 STEP -10
110 SOUND 0,I,10,8
120 NEXT I
130 SOUND 0,0,0,0
140 RETURN
190 REM ODCZYT CZASU
200 C=(((PEEK(18)*256)+PEEK(19)
)*256+PEEK(20))/50
210 GODZ=INT(C/3600):C=C-(GODZ*
3600)
220 MIN=INT(C/60):SEK=INT(C-(MI
N*60))
230 SEK=SEK+ISEK:IF SEK>59 THEN
 SEK=SEK-60:MIN=MIN+1
240 MIN=MIN+IMIN:IF MIN>59 THEN
 MIN=MIN-60:GODZ=GODZ+1
250 GODZ=GODZ+IGODZ
260 GODZ=GODZ-INT(GODZ/24)*24
270 RETURN
290 REM WPROWADZANIE CZASU
300 ? "GODZINA ";:INPUT G:? "MI
NUTA  ";:INPUT M:? "SEKUNDA ";:
INPUT S
310 RETURN
```

LP

Drukarka ośmiobitowa

Cztery centymetry przewodu, lutownica i nóż — tylko tyle potrzeba, aby wbudowane w twoim komputerze Amstrad (Schneider) CPC 464 łącze Centronics rozszerzyć do 8 bitów. Przyłączenie dowolnej drukarki do Amstrada CPC 464 nie jest żadną trudnością dzięki seryjnemu łączeniu Centronics. Odpowiedni wtyk platynowy można nabyć w każdym elektronicznym sklepie, tak samo jak łącze Centronics i kabel. Szybko jednak dostrzeżono istotny mankament Amstrada; zamiast przenosić 8 bitów danych równoległych w CPC 464, ósmy bit danych stanowi nieużyteczny sygnał strobujący, co oznacza iż, ten ósmy bit nie odpowiada bitowi przewodzonemu do drukarki, gdyż funkcję tę spełnia podłączony sygnał strobujący, informujący o przekazywaniu danych do drukarki

Dla samego tekstu jest to w pełni wystarczające, gdyż w Amstradach stosowany jest kod ASCII (ASCII — American Standard Code for Information Interchange), który jest kodem 7-bitowym. Wszystkie litery, cyfry i znaki mogą po wykonaniu kroku „Print 8" zostać przesłane do drukarki. Inaczej wygląda to w przypadku grafiki. Większość drukarek może w tym rezimie pracy odwzorować dowolne symbole graficzne, szczególnie w tzw. grafice blokowej, wszystkie znaki mają kod większy od 127. Wymagają one jednak przenoszenia danych graficznych pełnych 8-bitowych wartości. Chcąc skorzystać z tych możliwości drukarki, należy wprowadzić ósmy bit danych. Rozwiązanie tego problemu jest bardzo proste i wymaga łatwego majsterkowania najprostszymi narzędziami. Potrzebne są do tego tylko następujące przybory: mała lutownica (15 Wat), 5 do 6 cm izolowanego przewodu, śrubokręt krzyżakowy, ostry, spiczasty nóż i filcowa podkładka

Zanim zabierzesz się do pracy, musisz pamiętać, że cała ta operacja spowoduje wygaśnięcie gwarancji twojego komputera. Jeśli mimo to zdecydowałeś się — do dzieła!

Najpierw należy otworzyć obudowę CPC 464 (sześć śrub na dolnej płycie) i odłączyć górną część obudowy wraz z klawiaturą i magnetofonem kasetowym, przy czym należy rozłączyć dwa połączenia wtykowe. Nie ma obawy pomylenia ich przy składaniu, wtyczki pasują tylko do swoich właściwych gniazd. Błąd jest więc wykluczony. Złącza komputera odkrywają się na dolnej płycie obudowy. Płytę ustawiamy przed sobą tak, by rozszerzanie się połączeń przebiegało na dół. W lewym, górnym rogu znajdują się teraz połączenia z joystickiem, obok na prawo łącze Centronics.

Licząc od lewej strony, znajdź 9 ścieżkę prowadzącą do bramki Centronics i zaznacz filcową podkładką Jest to właśnie połączenie 8 bitu i w Amstradzie zblokowane jest ono na masę. Musi więc być przerwane jako pierwsze. Używamy do tego celu ostro zakończonego noża. Uważnie zeskrobujemy kawałeczek ścieżki przewodzącej. Uwaga! Robimy to ostrożnie, bez zbytniej siły, by nie uszkodzić innych ścieżek. Gdy żądana ścieżka zostanie przerwana, musimy wykonać nowe połączenie z 12 nóżką kostki 8255 Szukamy więc dużej, 40-nóżkowej kostki IC poniżej bramek Centronics, na której m.in. jest napis „8255". W dolnym rzędzie nóżek, licząc od lewej strony znajdujemy 12 nóżkę i zaznaczamy ją filcową podkładką. Na odchodzącej od niej ścieżce, kilka milimetrów poniżej, znajduje się „oczko" lutownicze, do którego przylutowujemy odizolowany koniec naszego przewodu. Drugi koniec przewodu należy przylutować do zaznaczonej uprzednio dziewiątej ścieżki, bramki Centronics, tuż nad wykonaną przerwą w ścieżce, ale tak by nie zblokować złączki na całej długości. Jeszcze ostatnie sprawdzenie miejsc lutowania i ponownie można podłączyć gniazda klawiatury i magnetofonu kasetowego. W ten sposób można wykonać 8-bitowy hardware, ale rzecz jeszcze w tym, by wysterować software

Dlatego też nieodzowne są częściowo nowe procedury, niezbędne do wysterowania drukarki. Wysterowanie układa się w postaci normalnego tekstu tak jak poprzednio, zaś w specjalnych wypadkach, gdy chcemy wykorzystać ten ósmy bit, musimy ułożyć specjalną komunikację. Ósmy bit bramki Centronics jest przez małe hardwarowe sprzężenie połączony z bitem 5 na porcie C kostki 8255. W Basic będzie więc ósmy bit wprowadzony przez „OUT&F600", X0010000" lub niezależnie przez „OUT&F600,32" ustawić zaś przez „OUT&F600,0" ponownie skasować. By móc dowolny 8-bitowy znak przesłać do drukarki, należy postępować następująco:

1: Sprawdź, czy kod znaku jest większy niż 127. Jeżeli tak, to ustaw ósmy bit przez „OUT & F600,32"

2. Przekazać znak do drukarki normalnie przez „PRINT ≠ 8"

3. Poprzedni stan skasować przez „OUT & F600,0"

Listing „Printer' utilities fur 8 Bit" zawiera dwa programy ułatwiające tę procedurę. Oba programy zapisane w Basicu mogą być włączone do dowolnych programów w tym języku za pomocą instrukcji MERGE. Instrukcja "GOSUB 6000" będzie przekazywać do drukarki dowolny znak. Zmienny znak Z musi przy tym otrzymać znak kodowy. Gdy chodzi np. o sterowanie grafiką i przesyłanie znaków ośmiobitowych — wprowadzona została w tym celu linia 61000. Od tego momentu wszystkie znaki Z $ przesyłane będą równocześnie do drukarki jako ośmiobitowe

Po tak eleganckim rozwiązaniu problemu łącza należałoby zająć się kolejnym aspektem drukarki tzw. Auto-Line -Feed-Syndrom, który jest szczególnie ważny przy tworzeniu listingów. W większości drukarek do Amstrada po każdym zadrukowanym wierszu znajduje się wiersz pusty, co w związku ze zużyciem papieru nie jest zbyt pożądane. Przyczyna tego znajduje się w języku maszynowym CPC 464, który na końcu każdego wiersza wysyła nie tylko znak CR (Carriage Return) ale także dodatkowo znak LF (Line Feed). Większość drukarek przesuwa wiersz nie tylko przy sygnale CR, ale także LF. Tak więc drukarka odbierze także sygnał LF i wykona drugie przesunięcie Najlepiej w tym wypadku byłoby przeszkodzić wysłaniu sygnałów LF. Jest to rzeczywiście możliwe po małej ingerencji w język maszyny. Oczywiście, nie możemy nic zmienić w ROM-ie, ale można w tym celu dokonać pewnego przełączenia. Tym przełączeniem jest rozgałęzienie z ROM na RAM. Na odpowiadającej RAM pozycji znajduje się przeważnie rozkaz JP dla programu ROM. Stawia on przed użytkownikiem możliwość, by zamiast JP wprowadzić do języka maszyny odpowiedni program. Indirection — Jump — Block składa się z adresu & BDCD do & BDF3 w RAM i zawiera jako ostatni zapis odgałęzienia „MC WAIT PRINTER". To odgałęzienie będzie przekazane każdej drukarce z drukowanymi znakami w Akku. Poza tym odgałęzieniem znajdują się bity wolne pamięci, tak że można mały program maszynowy wpisać wprost pod adresem „MC WAIT PRINTER"

```
BDF1 FEOA CP10; LF?
BDF3 37 SCF; all o k.
BDF4 C8 RETZ; tue no
BDF5 C5 07F8 JP 07F8H; normal next
```

Program ten porównuje znaki w Akku z LF, przykłada dla ostrożności Carry Flag (symbolizuje on język maszynowy, który prowadzi kolejność programu) i powraca bez dalszej akcji, zamiast przesyłania znaku LF. Innymi słowy przeskakuje porządkowy rozkaz programu druku. W listingu znajduje się rozkaz w Basicu (wywołanie GOSUB 62000, który ten program maszynowy zapisze w pamięci. Po jednorazowym

```
5000 '---------------------------
5010 '   printer-utilities
5020 '        8 - bit
5030 '---------------------------
5040 '
5050 REM print Z (8-bit)
5060 IF z>127 THEN OUT &F600,32
5070 PRINT #8,CHR$(z);
5075 OUT &F600,0
5080 RETURN
5090 '
5100 REM print Z$ (8 bit)
5110 FOR zz=1 TO LEN(z$)
5120 z=ASC(MID$(z$,zz,1))
5130 GOSUB 5060 : NEXT
5140 RETURN
5150 '
5160 REM printer line feed off
5170 FOR z=&BDF1 TO &BDF7
5180 READ byte:POKE z,byte
5190 NEXT
5200 DATA &FE,&0A,&37,&C8,&C3,
        &F8,&07
5210 RETURN
5220 '
5230 REM Czy drukarka wylaczona ?
5240 FOR zz= 200 TO 1 STEP -1
5250 IF (INP(&F500) AND 64)=0
     THEN zz=-1
5260 NEXT
5270 IF zz=0 THEN
    PRINT "** Wlacz drukarke **"
5280 RETURN
```

wywołaniu zmiana pozostanie, aż do wyłączenia lub przeprogramowania komputera.

Instrukcję sprawdzającą dla drukarki „GOSUB 63000" można wykorzystać w wielu wypadkach. Amstrad odmówi dalszej pracy według programu w Basicu, jeśli po instrukcji PRINT ≠ 8 drukarka nie była włączona. Według programu od wiersza 63000 można sprowadzić bieżący program stanu drukarki.

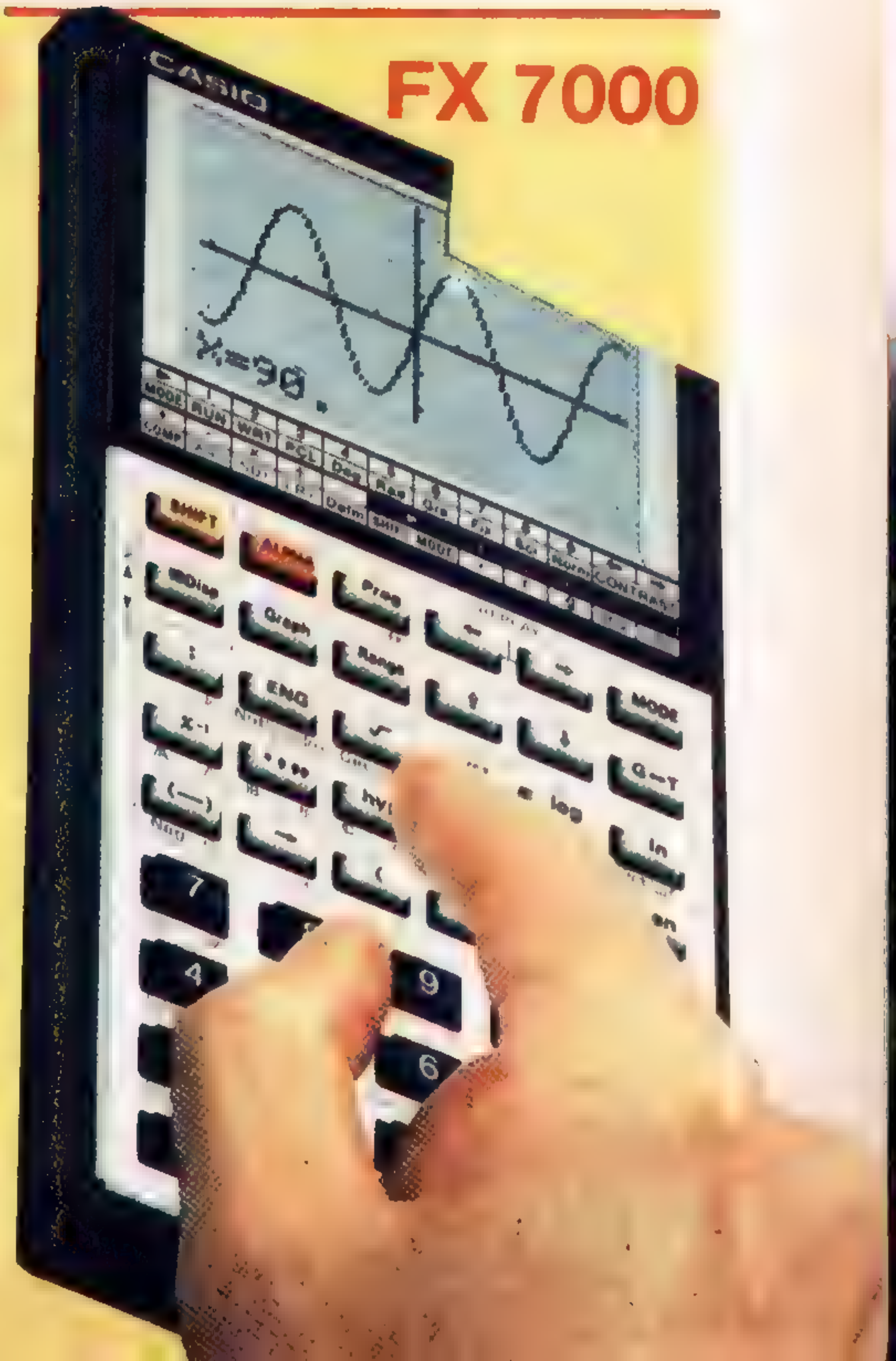
Zawarte w listingu cztery sposoby wykorzystania drukarki nie były przewidziane dla specjalnych drukarek, szczególnie pracujących razem ze wszystkimi urządzeniami, które wyposażone są w łącze Centronics. Pierwsze dwa programy są tylko do rozszerzenia hardware w opisany sposób, podczas gdy oba ostatnie podprogramy są uniwersalne

tłum. i oprac. JEM
na podst. „Happy Computer"

CASIO

Zabawka czy ... odkrycie

FX 7000



Mikrokalkulator kieszonkowy FX 7000 japońskiej firmy CASIO ma „klasyczne" rozwiązania:
— Programowanie symboliczne (jeden rozkaz = jeden klawisz),
— pamięć 422 „kroków" (czyli wciśnięć klawisza),
— tylko 26 pamięci wartości,
krótko mówiąc, przypomina on pierwsze programowane kalkulatory (na przykład model TI-58 firmy TEXAS INSTRUMENTS od lat produkowany w milionowych seriach).

Obok tych „archaicznych" rozwiązań ma on coś „supernowoczesnego": jest to ciekłokrystaliczny ekran (LCD) mieszczący 128 znaków, który swoje 5995 punktów wykorzystuje do grafiki komputerowej. I tak „na rozkaz" ukazuje się wykres zaprogramowanej funkcji.

Mało tego: można uruchomić „punkt wędrujący" po wykresie („myszkę") zaś w interesującym miejscu uzyskać współrzędne punktu. Można wykresy funkcji nakładać jeden na drugi i analizować punkty przecięcia. Prócz tego rysuje on histogram lub linie regresji z wypunktowanymi zmiennymi.

Krótko mówiąc: to „maleństwo" umożliwia jakościową analizę funkcji, dając ponadto jej wykres.

Ponieważ nie zawsze interesują nas superdokładne wyliczenia, zaś niemal zawsze ciekawy jest kształt funkcji, zatem nasuwa się pytanie postawione na wstępie: czy FX 7000 „GRAFIC" jest jeszcze jedną zabawką czy też jest on odkryciem firmy CASIO?

W szponach Atari

Firma Atari znana była na rynku amerykańskim i światowym jako producent gier elektronicznych przeznaczonych dla lokali rozrywkowych. Założyciel firmy i jej ówczesny właściciel, Nolan Buschnell, był namiętnym graczem w „Go". Jego marzeniem było wprowadzenie automatów do gier do domów. Udało mu się je zrealizować w 1975 roku. Urządzenia te do dzisiaj cieszą się ogromną popularnością. Według ocen producenta ponad 25 mln automatów model 2600 znajduje się w rękach graczy (albo na ich półkach). W 1985 roku sprzedano ich ponad milion i popyt nie słabnie. Umożliwiło to obniżenie kosztów, zmniejszenie wymiarów i wprowadzenie na rynek, zimą 1986 roku, ulepszonej wersji modelu 2600.

Tymczasem pojawiły się komputery domowe. Firma Atari zareagowała bardzo szybko. W 1979 roku powstały pierwsze, rewolucyjne modele 400 i 800, zbudowane w oparciu o 8-bitowy mikroprocesor 6502. Różniły się głównie klawiaturą. ATARI 400 miał klawiaturę foliową, zaś ATARI 800 — typową klawiaturę maszyny do pisania. Wewnętrzna budowa obu była identyczna. Charakterystyczne dla obu modeli było to, że o spełnianej przez komputer funkcji decydował dołączany z zewnątrz specjalny moduł ROM. Jednocześnie mogły być podłączone dwa takie moduły. O jakości i popularności tych komputerów najlepiej świadczy to, że sprzedano ich 26 milionów sztuk.

Silna konkurencja Commodore 64, wypierającego dzięki swej niskiej cenie i uniwersalności, Atari z rynku, zmusiła firmę do szybkiej modernizacji wyrobów. Produkcję wersji 400 i 800 wstrzymano, wprowadzając na ich miejsce, jesienią 1983 roku, nowe modele ATARI 600XL i 800XL. Różniły się one jedynie pojemnością pamięci RAM (16KB i 64KB odpowiednio). Były to udoskonalone wersje wcześniejszych modeli tej firmy. Jednak nie uzyskały oczekiwanego powodzenia. Firma znalazła się w obliczu bankructwa. W 1984 roku kupił ją dotychczasowy właściciel Commodore — Jack Tramiel. Od początku 1985 roku rozpoczął on „wojnę cenową" ze swoimi konkurentami, obniżając ceny swoich mikrokomputerów prawie o połowę.

Dzięki walce konkurencyjnej dwóch firm stałem się posiadaczem komputera ATARI 800XL. Od jesieni 1984 roku pilnie śledziłem ceny komputerów domowych, obserwowałem ich stały spadek. Nie mogłem się zdecydować. Najpopularniejszy wtedy u nas Spectrum 48KB, firmy Sinclair Research, raził niską jakością klawiatury i „niepoważnym" wyglądem. Commodore 64 był w sam raz, lecz zbyt drogi jak na moją kieszeń. ATARI 800XL zwrócił moją uwagę, gdy jego cena zrównała się z ceną Spectrum. Wykonany jest znacznie solidniej, ma normalną klawiaturę o bardzo dobrej jakości, cztery generatory dźwięku, w obudowie umieszczone są gniazda do podłączenia manipulatorów drążkowych. Nie uwzględniłem tylko jednego, o czym nie można było wyczytać w krótkich notatkach prasowych, nie może on bezpośrednio współpracować ze zwykłym magnetofonem kasetowym. Aby efektywnie wykorzystać możliwości komputera, niezbędny jest zakup przynajmniej najtańszej pamięci zewnętrznej — magnetofonu firmowego (Program Recorder). Jego cena stanowi połowę ceny komputera, ale podczas pracy jest znacznie wygodniejszy od zwykłego magnetofonu.

Radość z nabycia komputera trwała krótko. Okazało się, że zarówno programy, jak i literatura, dotycząca tego sprzętu, były nieosiągalne. Próbowałem sam pisać różne programy i gry dla dzieci, ale brakowało mi dokładnych opisów różnorodnych możliwości sprzętu i oprogramowania podstawowego. Obecnie sytuacja się nieco zmieniła. Można już zdobyć sporo programów dla ATARI, lecz nadal są trudności z uzyskaniem odpowiedniej literatury. Pewne pozycje można kupić, po bardzo wysokich cenach, na różnorodnych giełdach. Są to zwykle kiepskie odbitki kserograficzne książek w języku angielskim, albo nie najlepsze ich tłumaczenia na język polski, również mało czytelne.

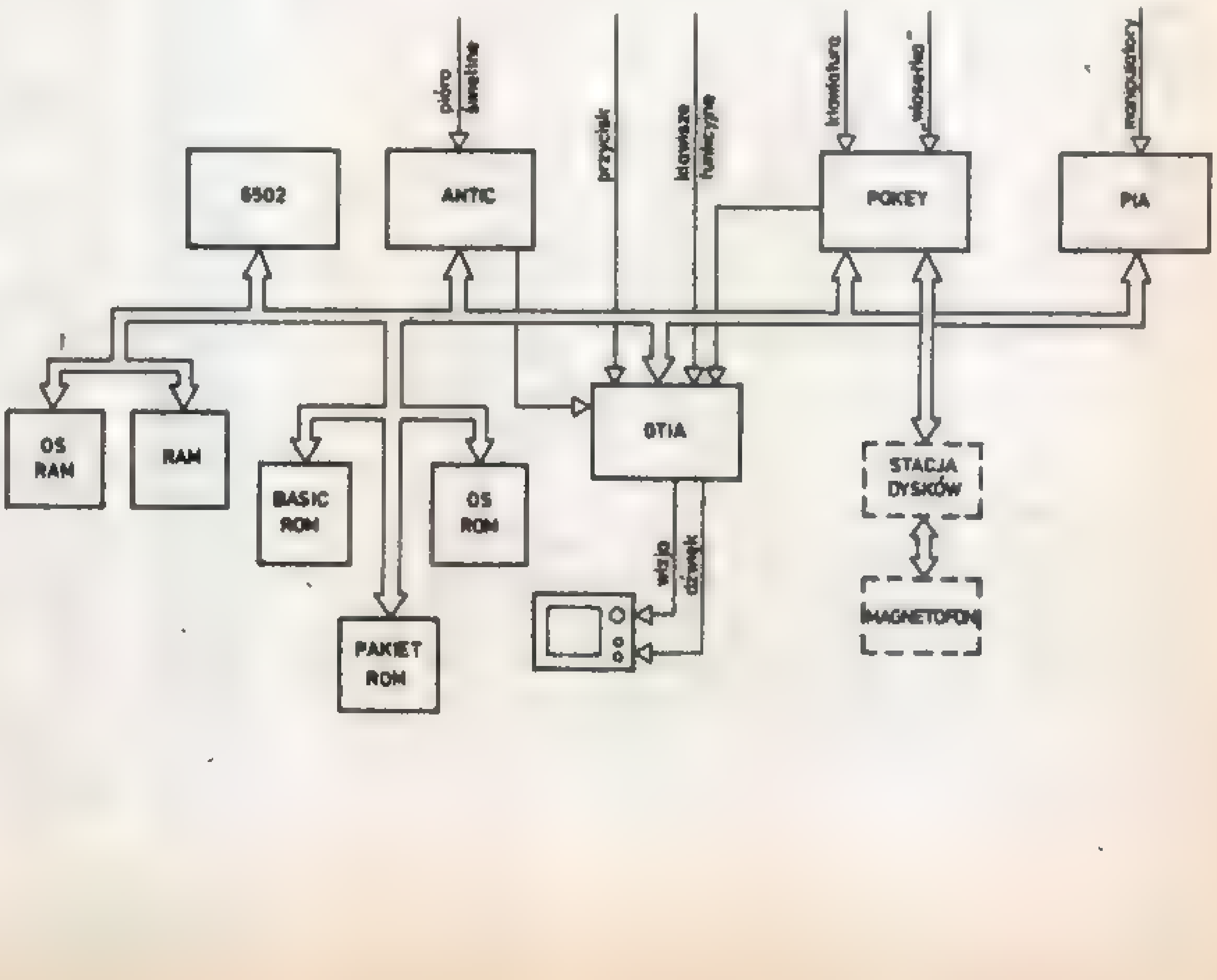
Wraz z kolegą, Tomaszem, postanowiliśmy pomóc wszystkim posiadaczom komputerów ATARI, zarówno 800XL jak i 130XE. Zgromadziliśmy trochę literatury, poznaliśmy możliwości 800XL i pragniemy podzielić się zdobytym doświadczeniem.

Pragniemy pomóc czytelnikom „IKS-a" w poznaniu swojego komputera i w jak najlepszym jego wykorzystaniu. Będziemy pisać dla tych, którym znudziły się już gry oraz tych, którzy chcą sami, w sposób twórczy, wykorzystać swój komputer nie tylko do zabawy. Przedstawimy przykłady rozwiązań pewnych problemów związanych z dźwiękiem oraz grafiką, zapoznamy z dostępnymi językami programowania, oraz gotowymi programami użytkowymi, które mogą pomóc w pracy, nauce i zabawie. W redagowaniu działu liczymy na pomoc czytelników. Mamy nadzieję, że pomożecie nam w wyborze tematów, które Was najbardziej interesują.

RZUT OKA DO ŚRODKA

Wewnętrzna konstrukcja ATARI 800XL różni się od innych, równorzędnych mu komputerów. Ma on oczywiście mikroprocesor (6502), pamięci RAM i ROM oraz układy wejścia-wyjścia (PIA — Peripheral Interface Adapter). Dodatkowo ma trzy specjalizowane układy scalone (LSI) znane jako ANTIC, GTIA i POKEY. Zostały one zaprojektowane przez inżynierów Atari głównie po to, aby odciążyć mikroprocesor od czynności pomocniczych i umożliwić mu skoncentrowanie się na obliczeniach. Podstawowe zalety ATARI 800XL wynikają głównie z możliwości tych trzech układów.

ANTIC jest układem przeznaczonym do tworzenia obrazu telewizyjnego. Jest on prawdziwym mikroprocesorem; zawiera zestaw rozkazów, program



zwany programem wyświetlania (display list) oraz dane. Program i dane do wyświetlania zapisywane są do RAM przez 6502. ANTIC pobiera te informacje, korzystając z bezpośredniego dostępu do pamięci (DMA). Rozkazy wyższego poziomu, zawarte w programie wyświetlania, tłumaczy na ciąg prostych rozkazów dla GTIA.

GTIA jest układem wyjścia telewizyjnego. ANTIC bezpośrednio steruje większością jego operacji, lecz 6502 może być zaprogramowany do ustawiania i sterowania pewnych lub wszystkich jego funkcji. GTIA tłumaczy rozkazy cyfrowe z ANTIC-u (lub 6502) na sygnały, które są wysyłane do telewizora. GTIA dodaje także pewne własne składniki, takie jak wartości koloru, grafikę gracz/pocisk (P)M — player/(missile) oraz wykrywanie kolizji. POKEY jest układem cyfrowego wejścia/wyjścia. Wypełnia on takie różnorodne zadania jak sterowanie szeregową magistralą wejścia/wyjścia, generowanie dźwięku, testowanie klawiatury i generowanie liczb losowych. Zamienia on także na postać cyfrową sygnały analogowe „wiosełek" (paddle) i steruje żądaniami przerwań maskowalnych (IRQ) z urządzeń zewnętrznych.

Wszystkie cztery układy scalone działają jednocześnie. Właściwe rozdzielenie ich funkcji w czasie projektowania zminimalizowało konflikty między nimi. Jedyny konflikt na poziomie sprzętu, między dowolnymi dwoma układami w systemie, występuje wtedy, kiedy ANTIC musi wykorzystać magistrale, adresową i danych, w celu pobrania informacji do wyświetlania. Aby to wykonać, zatrzymuje on 6502 i przejmuje kontrolę nad magistralami.

W każdym komputerze domowym ATARI wbudowany jest układ pamięci ROM, o pojemności 10KB, zawierający system operacyjny. Znaczenie tego układu jest często niedoceniane. Bez niego dysponujemy tylko ogromem możliwości, lecz nic poza tym. Ta sytuacja dotyczy wszystkich komputerów, nie tylko ATARI. Komputer jest zestawem różnorodnych układów i urządzeń. Aby wykonać jakiekolwiek zadanie, użytkownik musi sterować tymi urządzeniami (zasobami układowymi). Gdyby wszyscy programiści rozpoczynali tworzenie programów od takiego poziomu, byłaby teraz znacznie mniejsza liczba dostępnych programów. Rozwiązaniem, które było stosowane przez lata, jest wbudowanie do komputera programu, który steruje zasobami dostępnymi dla systemu i zmniejsza wysiłek programistów, niezbędny do sterowania nimi. Program ten znany jest pod różnymi nazwami: system operacyjny, główny program sterujący, monitor systemu, system wykonawczy, itp. W komputerach ATARI używana jest nazwa system operacyjny lub OS.

Używając zasobów sprzętowych komputera, system operacyjny może współpracować z różnorodnymi urządzeniami i sterować ich pracą. Do urządzeń zaliczamy: odbiornik telewizyjny lub monitor, klawiaturę, klawisze funkcyjne, manipulatory drążkowe, wiosełka, magnetofon kasetowy, stację dysków, drukarki, układ wejścia/wyjścia RS-232 i modemy.

W systemie operacyjnym można wyróżnić następujące podstawowe elementy:

Monitor. Jest to procedura systemowa, która jest uruchamiana po włączeniu komputera lub naciśnięciu klawisza RESET. Za pośrednictwem tej procedury OS przejmuje kontrolę nad systemem. Monitor odnawia system zarządzania pamięcią, inicjuje podsystem wejścia/wyjścia, ustawia wektory systemu i wybiera środowisko działania po zakończeniu inicjowania.

Obsługa przerwań. Komputer wykorzystuje standardową strukturę przerwań mikroprocesora 6502, z pewnym zewnętrznym rozszerzeniem w celu zwiększenia elastyczności działania. Przerwania generowane są przez pewne zdarzenia, do których należą: naciśnięcie jednego z klawiszy, naciśnięcie klawisza BREAK, pewne zdarzenia magistrali szeregowej, przekroczenie czasu systemowego i pionowe (puste) przejście strumienia świetlnego w telewizorze.

Wektory systemu. Dostarczają one mechanizmu, który umożliwia użytkownikowi dostęp do procedur systemu lub dostosowanie OS do specjalnych wymagań. Wektory systemu najczęściej używane są do wywoływania procedur wejścia/wyjścia, ustawiania zegarów i przekazywania sterowania do innych środowisk działania. Rozróżnia się wektory ROM, które są miejscami zawierającymi instrukcje skoku (IMP) do procedur systemu i nie mogą być zmieniane oraz wektory RAM, które zawierają dające się zmieniać adresy procedur systemu.

Podsystem wejścia/wyjścia. System operacyjny umożliwia programiście dostęp do wszystkich urządzeń zewnętrznych komputera. Podsystem wejścia/wyjścia jest zestawem procedur, które łączą wysokopoziomowe operacje wejścia/wyjścia z programami obsługi (handler'ami) urządzeń, które fizycznie sterują sprzętem wejścia/wyjścia.

Programowanie bieżące. Komputer domowy ATARI jest dobrze przygotowany do rozwiązywania problemów na bieżąco. W tym celu OS zaopatrzony jest dwa typy zegarów: układowe i programowe systemu. Zegary układowe odliczają czas w dół i mogą być używane do pomiaru czasu trwania zdarzeń w zakresie od pół mikrosekundy do kilku sekund. Zegary systemowe są programowymi, zmieniają się z częstotliwością 50 Hz i mogą być używane do zastosowań tak różnych jak pomiar czasu na magistrali szeregowej lub generowanie efektów dźwiękowych.

Zestaw znaków. Komputer wyposażony jest w tak zwany „miękki" zestaw znaków, to znaczy, że może być zmieniany. Zestaw znaków ROM używany jest dla dostarczenia standardowego zestawu znaków po włączeniu zasilania.

Pakiet zmiennoprzecinkowy. Jest to zestaw procedur matematycznych, które zwiększają możliwości systemu. Używają one arytmetyki BCD (binary codet decimal) w celu dostarczenia standardowych funkcji matematycznych (+, -, *, /), funkcji wykładniczych i logarytmicznych oraz konwersji z kodu ATASCII na BCD i z BCD na ATASCII.

OKO KOMPUTERA

COMPUTER EYES jest nowoczesnym urządzeniem do zamiany obrazu na postać cyfrową. Wspierane odpowiednim programem, po włączeniu między dowolne, standardowe źródło video i komputer ATARI, umożliwia wprowadzenie do pamięci, w ciągu kilku sekund, obrazu o wysokiej rozdzielczości. Obraz może pochodzić z kamery video, wyjścia TV, dysku optycznego, innego komputera, itp. Podłączany jest do wejścia manipulatora drążkowego. Wprowadzony obraz może być przechowany na dysku. Używanymi trybami graficznymi są: 8 — wysoka rozdzielczość; 9 — szesnaście odcieni szarości uzyskanych kosztem rozdzielczości; 7,5 — zgodny z tablicą graficzną (touch tablet). Wprowadzone obrazy mogą być modyfikowane za pomocą jednego z następujących programów: Koala Pad, Atari Touch Tablet, Micro Ilustrator lub Magniprint.

Korzystając z programu Magniprint II + można uzyskać:

— wydruk obrazów w 19 różnych powiększeniach;

— wydruk obrazów z pełnym cieniowaniem, uzyskuje się lepsze efekty niż na ekranie telewizora.

Dla użytkowników, którzy nie mają dostępu do sprzętu video, oferowane są zestawy zawierające: COMPUTER EYES + programy do przechowywania obrazu i wyświetlania, Magniprint II +, kamerę video wysokiej jakości i 3 m kabla koncentrycznego z odpowiednimi końcówkami.

Umożliwia przechowywanie zdjęć w pamięci dyskowej i wyświetlanie ich na ekranie telewizora.

LP

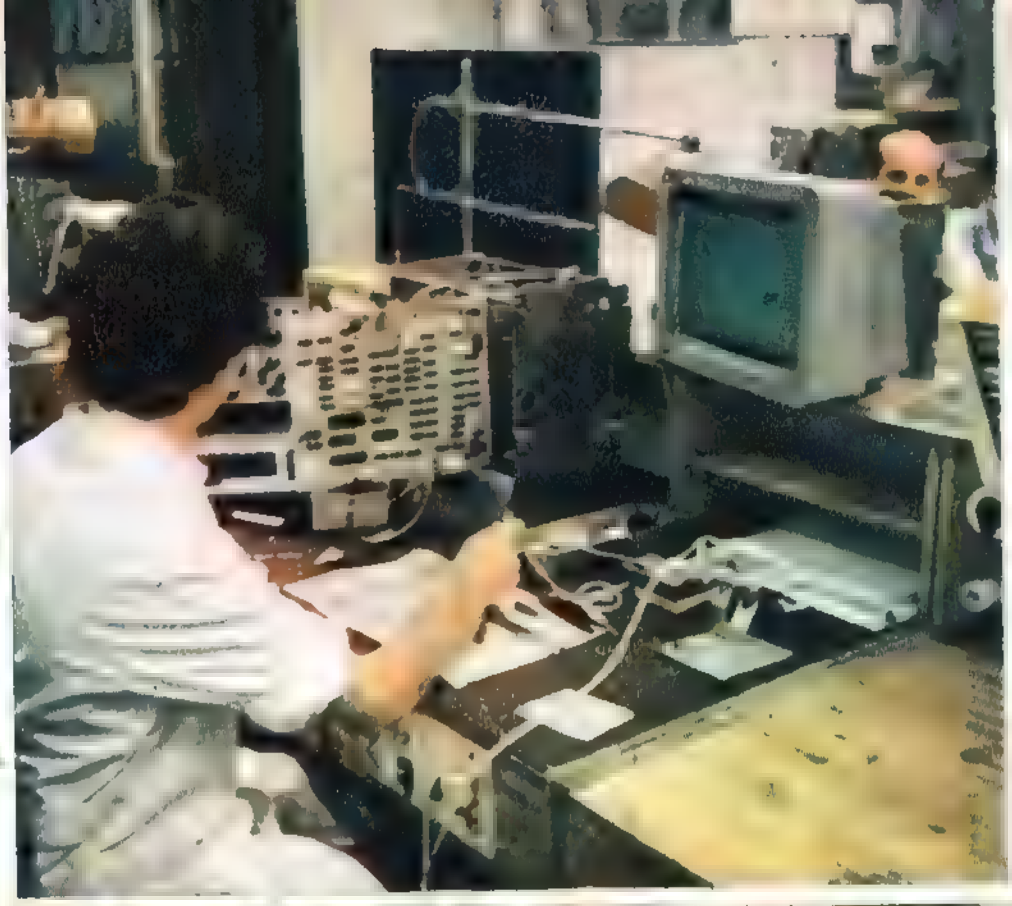






Komputero







owa taśma

Foto: Ryszard Rogoń

Jak wykorzystać niedostępną pamięć RAM

Przedstawiony poniżej program użytkowy umożliwia wykorzystanie w mikrokomputerach ATARI 800XL i 130XE pamięci RAM niedostępnej w czasie pracy z systemem operacyjnym wbudowanym w pamięć ROM komputera (OS ROM).

Mapę pamięci mikrokomputera można przedstawić w ogólnym zarysie w sposób poniższy:

nr obsz	adres dziesiętnie	adres hexadecym.	wykorzystanie
1	0—40959	$0000—$9FFF	40KB RAM/bez kilku pierwszych stron
2	40960—49151	$A000—$BFFF	8KB BASIC lub RAM
3	49152—53247	$C000—$CFFF	OS ROM—4 KB
4	53248—55295	$D000—$D7FF	2KB obszar rejestrów we/wy (ang hardware register)
5	55296—65535	$D800—$FFFF	10KB OS ROM

W czasie pracy z normalnym systemem operacyjnym OS ROM dostępne są tylko obszary 1 i 2 (ok. 48 KB), a przy włączeniu interpretera BASIC tylko obszar 1. Jednak za wyjątkiem obszaru 4, związanego z obsługą urządzeń we/wy, pozostałe obszary mogą być z powodzeniem wykorzystane. Zezwala na to konstrukcja komputera i wbudowanego weń systemu operacyjnego. System operacyjny mikrokomputera ATARI zawiera standardowe procedury (podprogramy) obsługujące urządzenia zewnętrzne. Na stałe rezydują w systemie operacyjnym podprogramy obsługujące takie urządzenia jak: klawiatura „K:", drukarka „P:", magnetofon kasetowy „C:", monitor „S:" oraz tzw. edytor ekranowy „E:". Procedury do obsługi innych urządzeń np. dysków nie rezydują na stałe w OS ROM (wyjątek stanowią procedury inicjujące ładowanie do pamięci procedur DOS), lecz są do niego „dokładane" z dyskietki. Ponadto użytkownik komputera ma możliwość wprowadzenia do OS ROM, procedur obsługujących swoje własne urządzenia, w postaci napisanych przez siebie programów. Tę właśnie możliwość wykorzystuje prezentowany program. Wprowadza on do OS ROM dwie dodatkowe procedury obsługujące urządzenia oznaczone „M:" i „N:". Urządzenie „M:" jest symulowanym, w normalnie niedostępnym obszarze RAM, dyskiem magnetycznym — stąd jego nazwa RAMDYSK. RAMDYSK można wykorzystywać pod kontrolą DOS w sposób analogiczny jak normalną dyskietkę (zamiast „D:" należy używać „M:"). Maksymalna pojemność RAMDYSKU wynosi co prawda tylko ok. 22KB (przy włączonym interpreterze BASIC jeszcze mniej, bo ok. 14KB) i jest po wyłączeniu komputera tracona, ale dostęp do zapisanych w RAMDYSKU informacji jest znacznie szybszy niż w przypadku normalnej stacji dysków.

Natomiast urządzenie „N:" (ang. null handler) symuluje wprowadzanie danych przez program na urządzenie zewnętrzne, chociaż fizycznie dane te nie są nigdzie przesyłane. Nullhandler jest szczególnie przydatny przy testowaniu pisanych programów. Wyobraźmy sobie sytuację, że testujemy program, który zapisuje 10 000 znaków na taśmę magnetyczną, tworząc jednocześnie pewne tablice pamięci. Po zapisaniu danych na kasecie program przystępuje do przetwarzania powstałych tablic. Testując taki program, ogrom czasu pochłania oczekiwanie na zakończenie przesyłania danych na taśmę magnetyczną. Natomiast jeżeli zastąpimy w testowanym programie (tylko na czas testowania) we wszystkich rozkazach związanych z pamięcią kasetową urządzenie „C:" urządzeniem „N:", to sytuacja radykalnie się poprawi. Czas testowania skróci się niewspółmiernie. Kto nie wierzy, łatwo może to sprawdzić. Ostatnia uwaga na temat tego urządzenia. Nie ma ono zaimplementowanej funkcji czytania (nie można czytać z niego informacji), toteż każda instrukcja INPUT lub GET z nim związana powoduje wystąpienie błędu w numerze 146.

Program napisany jest w języku BASIC i kodzie maszynowym, zainstalowany jest na 6 stronie pamięci od adresu $0600 do adresu $06EB. Po napisaniu programu można go zapisać na taśmę magnetyczną lub dyskietkę. Program uruchamia się instrukcją RUN. Po pojawieniu się komunikatu READY urządzenia „M:" i „N:" są gotowe do użycia. Dla użytkowników komputera ze stacją dysków warto polecić zapisanie binarnego programu do zbioru-AUTORUN. SYS, co zapewni jego automatyczne uruchomienie po każdorazowym załadowaniu DOS-a. Natomiast instalacja komputera bez stacji dysków wymaga, niestety, przy każdorazowym włączeniu komputera ładowania programu do pamięci.

Wykorzystanie RAMDYSKU może być wielorakie. Należy zaznaczyć, że wszystkie jego elementy uwidaczniają się podczas pracy z dyskowym systemem operacyjnym. Ale również pracując bez DOS-a, użytkownik zyskuje wiele cennych możliwości. Na przykład: można do RAMDYSKU wprowadzić instrukcję SAVE „M:" program w BASIC-u i uruchamiać go, gdy istnieje taka potrzeba (RUN „M:"). Zaznaczyć należy, że oprócz programu przechowywanego w RAMDYSKU można załadować program do „normalnego" RAM-u. Opcja ta jest szczególnie przydatna, gdy zachodzi potrzeba częstego uruchamiania dwóch lub więcej programów — pozwala ona oszczędzać czas potrzebny na ładowanie programów. Można w obszarze RAMDYSKU utworzyć minibazę danych. Trzeba jednakże pamiętać, że bez DOS-a dostęp do zapisanych w niej informacji będzie dostępem sekwencyjnym. Zawartość takiej bazy danych może być skopiowana na kasetę i wprowadzana do RAMDYSKU za pomocą prostego programu w BASIC-u. Można w obszarze RAMDYSKU przechowywać grafikę do tworzonych gier, można go wykorzystać jako bufor do przekazywania danych pomiędzy programami, może on z powodzeniem zastąpić wyjście programowe na drukarkę, a zapisany do niego wydruk można skopiować na kasetę lub dyskietkę. Inicjatywa co do sposobu jego wykorzystania należy do programisty. Proponuję samemu spróbować.

Opracował: H. KRASUSKI

```
10000 REM RAMDYSK
10010 S=0:RESTORE 10100
10020 FOR A=1536 TO 1771:READ D:POKE A,D
:S=S+D:NEXT A
10030 IF S<>25924 THEN ? "BLAD DANYCH":S
TOP
10090 A=USR(1594)
10100 DATA 160,0,185,26,3,201,0,240,9,20
0,200,200,192,34,208,242,56
10110 DATA 96,162,0,189,62,6,153,26,3,23
2,200,224,6,208,244,174,0
10120 DATA 160,232,138,142,0,160,77,0,16
0,208,4,206,0,160,96,169,160
10130 DATA 141,99,6,141,235,6,96,104,76,
0,6,78,68,6,77,83,6,105,6
10140 DATA 105,6,107,6,105,6,105,6,107,6
,76,106,6,97,6,105,6,160,6
10150 DATA 108,6,105,6,105,6,76,106,6,16
9,192,133,75,169,0,133,74
10160 DATA 160,1,96,162,208,228,75,208,4
,162,216,134,75,160,0,196
10170 DATA 75,240,28,32,209,6,145,74,32,
224,6,230,74,196,74,208,2
10180 DATA 230,75,208,165,74,141,234,6,1
65,75,141,235,6,96,32,98,6
10190 DATA 160,146,76,142,6,160,208,196,
75,208,4,160,216,132,75,172
10200 DATA 234,6,174,235,6,228,75,208,7,
196,74,208,3,160,136,96,160
10210 DATA 0,32,209,6,177,74,32,224,6,23
0,74,196,74,208,2,230,75,160
10220 DATA 1,96,120,140,14,212,174,1,211
,160,254,140,1,211,160,0,96
10230 DATA 142,1,211,162,255,142,14,212,
88,96,0,192
```

Czy komputer może być życzliwy?



W idealnym świecie mikrokomputerów powinien zachowywać się jak doskonały asystent. Należałoby go wówczas włączyć, wprowadzić kilka poleceń (instrukcji) i zostawić samego by po powrocie stwierdzić, że zadana praca została wykonana zgodnie z poleceniem.

Niestety, otaczający nas świat nie jest idealny. Poza barierami technologicznymi, przyczyn takiego stanu należy upatrywać w tym, że część ludzi nie zrobi (nie potrafi) nic za włączeniem komputera — znacznie mniej, zrobi coś ponadto.

Wysiłki nad pokonaniem tych barier, uczynieniem komputera bardziej „przyjaznym", „życzliwym" dają niestety ograniczone efekty. Ale projektanci są nadal przy pracy w swoich laboratoriach. Należy więc mieć nadzieję, że w niedalekiej przyszłości, spotkamy się z życzliwością komputerów.

Firmy komputerowe zapewniają, że już wkrótce użytkownicy będą mówili do mikrokomputerów i mikro będą odpowiadały (miejmy nadzieję, że grzecznie).

Technologia została opracowana, problem tkwi w kosztach tego rozwiązania. Cena procesora o odpowiedniej mocy i pamięciach masowych dla syntezy i rozpoznawania mowy musi znacznie zmaleć przed wkroczeniem tej technologii na rynek mikro.

Foniczne („mówione") wejście i wyjście informacji do i z komputera jest jak dotychczas ostatnim stopniem w światowym rozwoju oprogramowania.

Oczywiście oprogramowanie nie musi być życzliwe, by było użyteczne.

Ważnym zabiegiem w staraniach o pozyskanie była zmiana zasad konstrukcji oprogramowania z prezentacji menu skonstruowanego ze znaków na WIMP (okna, ikony, myszka i zwijane menu — windows, icons, mice and pull-down menus).

Pionierską pracę w tym zakresie wykonano dla Rank Xerox w Palo Alto Research Centre. Był to bodziec do rozpoczęcia produkcji Xerox Star i pośrednio Mac, Gem, Taxi i innych produktów typu WIMP.

W Xerox Parc zwrócono szczególną uwagę na oprogramowanie takich własności systemu, które znacznie upraszczają pracę programistów.

Ikony, menu i dialogowa skrzynka, nazywana także tabliczkami do rysowania (zasada jej działania jest dalej opisana) zminimalizowały wszystko to, czego użytkownik mikro musi się uczyć i co musi pamiętać. Jest to skrupulatnie wykorzystywane w wielu programach.

Przystępność graficznej informacji ma szczególną wartość, gdy zostaje wykorzystana jako sposób w przekazywaniu komend do systemu operacyjnego mikro. W rezultacie oprogramowanie tak rozumianej grafiki — życzliwości — znacznie upraszcza pracę operatorską.

Firmy produkujące i sprzedające oprogramowanie i sprzęt zgodziły się z tym, że ulepszaniu systemu operacyjnego maszyny powinien towarzyszyć jej „życzliwy wygląd". Wytwórcy traktują tak rozumianą życzliwość jako sposób na sprzedaż mikro użytkownikom, którzy wcześniej byli przeciwni kupieniu komputera.

Ostatnio wiele firm produkujących mikrokomputery podpisało z Microsoftem i z Digital Research umowy, by produkowane przez nich systemy operacyjne, coraz częściej nazywane graficznymi interfejsami, typu OKNA i GEM były produkowane bez ograniczeń zawężających zdolność ich pracy tylko do określonego typu komputera. Innymi słowy, dotychczas omawiane systemy operacyjne mogły praktycznie pracować tylko na sprzęcie, do którego były dołączone. Jednak entuzjazm użytkowników wobec trudnych do przecenienia możliwości „życzliwych systemów" według Digital Research okazał się nadzwyczaj wymierny, zaledwie bowiem 50% kopii GEM pracuje na oryginalnym sprzęcie, pozostałe 50% jest w rękach wspomnianych użytkowników, którzy są głęboko przekonani, iż GEM zapewnia im łatwy sposób komunikowania się z mikro.

Digital Research porównał dwa różne interfejsy—programy. Jeden znany jako VIP przekazywał informacje znakami, częściowo w postaci ikon. Jego ekran podzielony był na 3 części: w lewym górnym rogu powierzchnia pracy, menu po prawej stronie i rozwinięcie komend wzdłuż dołu ekranu. Drugi, preferowany przez użytkowników (co zapewne potwierdziło założenia badań) program to wspominany GEM — opierał się na ikonach, miał możliwości rekonfiguracji podziału ekranu. Menu ukazuje się w szczytowym pasku ekranu. Pełne menu ukazuje się w postaci „zwijanego" w dół tekstu.

W tym miejscu warto zwrócić uwagę na fakt, że dwie konkurencyjne firmy produkujące oprogramowanie Digital Research i Macintosh miały bardzo podobne rezultaty prac nad interfejsem użytkownika. Dowodzi to, że kwestia życzliwości mikro nie jest epizodem w rozwoju systemów operacyjnych mikrokomputerów. Jego dużą wagę podkreśla niekwestionowane zainteresowanie użytkowników — profesjonalistów i amatorów — także liczących się w świecie producentów systemów operacyjnych.

Rozwijając system Taxi, jego producent, Epson, rozważnie kierował się zasadą oryginalności, unikania kopii nawet takich firm jak Macintosh, miał jednak cały czas na uwadze podstawy WIMP.

Jednym z twórców życzliwości mikro za pośrednictwem systemu Taxi był Alistair Macintosh. Wykorzystał on swoje bardzo bogate doświadczenie z pracy nad wizualną percepcją w projektowaniu tego typu programów. *„Mózg jest bardzo dobry do oceny wzorów — przestrzennej percepcji —* mówi Macintosh *— gdy wejdziesz do pokoju z zawiązanymi oczami i zdejmiesz przepaskę na pół sekundy, będziesz w stanie opisać bardzo wiele — usytuowanie okien, drzwi, stołu itp. Natomiast czytanie jest wyuczonym atrybutem. Jeśli dam ci opis pokoju i odsłonię oczy, to będziesz potrzebował dużo czasu, by opisać pokój".*

W Taxi, sprawdzając podział zbiorów, widać dużą liczbę ikon, które są łatwo rozpoznawalne dzięki ich kształtowi. Porównując to z systemami Macintosha i GEM, gdzie ikony są tak małe, że niezbędny okazał się tekst opisujący typ zbioru reprezentowany przez ikony. W rezultacie znowu pośrednikiem stają się znaki alfanumeryczne.

Według oceny Macintosha rozwój grafiki jest limitowany przez użyty w IBM mikroprocesor, będący standardem w profesjonalnych mikro, a także przez pojemność dostępnej pamięci.

dokończenie na str. 20

dokończenie ze str. 19

Macintosh oczekuje w niedalekiej przyszłości systemu, który będzie reagował na każde dotknięcie klawiatury W ten sposób, by użytkownik odczuwał życzliwość komputera. Hasło „pomoc" spowoduje natychmiastowe udzielenie rad i wskazówek, co należy robić.

Użytkownicy zatrudnieni w przemyśle oczekują innych niż klawiatura pośredników, także myszka nie cieszy się zbyt wielkim powodzeniem. Chodzi o to, by posługiwać się zbliżonymi do naturalnych metodami przekazywania informacji. Ekran dotykowy wymaga po prostu dotykania palcem wyróżnionych swoich części. Znajdująca się tam informacja jest odbierana przez komputer. Zatem nie klawiatura, która ma często ponad setkę klawiszy, lecz po prostu dotknięcie palcem wybranego obrazka na ekranie — komunikatu — pośredniczy w przekazywaniu informacji do mikro

Podobnie działa tabliczka do rysowania, która pośredniczy w bezbłędnym przekazywaniu do maszyny współrzędnych miejsca dotknięcia tworzącego ją prostokąta, na przykład formatu dwóch kartek papieru. Całość ułożona jest na stole. Dotykane na tabliczce miejsca są odpowiednio odzwierciedlane na ekranie monitora. Zatem rysując na tabliczce (palcem, także igłą, ołówkiem) kwadrat, pojawia się on jednocześnie na ekranie lub dotykając odpowiednie miejsce tabliczki, wyróżniamy fragment ekranu, na którym jest na przykład symbol dysku (ikona dysku), w odpowiedzi komputer drukuje operacje możliwe do wykonania na dysku. Kopiowanie, reorganizacja, kasowanie, listowanie, czytanie itp. Nasz wybór „czytanie" (przez dotknięcie fragmentu obrazu) lub tabliczki, (w którym wyświetlone jest to słowo) powoduje kolejne pytania mikro: czytanie nazw?, wczytanie zbioru?, programu? Nasz wybór „programy?" inicjuje kolejne propozycje komputera. Tym razem są nimi nazwy programów zapisanych w dostępnych pamięciach dyskowych. Ostatecznie wybrany przez nas program (dotknięciem obrazka jego nazwy) jest wczytany do pamięci RAM. Podobnie przebiega jego uruchomienie i eksploatacja.

Jednakże doświadczeni użytkownicy wydają się niechętni wobec innych niż klawiatura pośredników. Profesjonaliści uważają, ze używanie myszki po prostu nie jest „męskie". Krytycyzmu tego rodzaju należy prawdopodobnie upatrywać także w doświadczeniach uzyskanych podczas eksploatacji systemów przetwarzania tekstów. Niezbędne okazuje się wówczas wykorzystywanie klawiatury.

Nowa Amiga Commodora ma wyjątkowo duże możliwości, często przekraczające standardy światowe. Pozwala bowiem na bardzo duży zakres korzystania z życzliwego interfejsu użytkownika z doskonale oprogramowaną myszką i typowym dostępem do klawiatury

Jednak przebojem Amigi jest pierwszy komputer, który dysponuje tekstem do *„wypowiadania, mówienia"* informacji wyjściowych. Mikro może mówić do ciebie lub po prostu czytać ci wybrane dokumenty.

Zasadniczą barierą sterowania komputera głosem są koszty. *„Mówione wejście"* lub inaczej rozpoznawanie mowy, jest dzisiaj możliwe. Na cenę tego rozwiązania wpływa technologia elementów elektronicznych — procesory, pamięci

Wiele z „fonicznych" planów osiągnął w ubiegłym roku APRICOT. W celu rozpoznania rynku, uzyskania oceny reakcji użytkowników, Apricot wprowadził na rynek mikro, który może rozpoznawać 4000 słów. Niestety, technologia ta nie została doceniona i mikro nie cieszył się oczekiwaną popularnością. Właściciele tego mikro stwierdzili, ze potrzeba zbyt dużo czasu, by komputer nauczył się rozpoznawać komendy jego właściciela. Na początku użytkownicy muszą określić, które komendy mają być rozpoznawane przez program i potem wypowiadają kolejne komendy po pięć razy, aż komputer nauczy się je rozpoznawać

Dystrybutorzy zauważyli, że trudno jest znaleźć profesjonalne zastosowania tego rodzaju rozpoznawania mowy.

Rynek oczekuje mikro ze słownikiem 20 000 słów, których wymowę rozumie i wypowiada komputer. Taki mikro wymaga pamięci gigabajtowej. Przewiduje się, że rolę tę będzie pełnił CD ROM (compact disk read-only memory) Obecnie dostępne CD ROM mają około pół gigabajta pojemności, ale czas dostępu 0,5 sekundy. Zatem CD ROM mogą być tylko częścią rozwiązania Ponadto CD ROM kosztuje dzisiaj aż 1200 funtów

Synteza dźwięków mowy ludzkiej wymaga trudnej do wyobrażenia pracy procesora. Niezbędne są szybkie procesory, takie jak INTEL 80286 i 80386

Apricot nie zmienił swoich „fonicznych" planów i dąży do zmniejszenia kosztów ich realizacji. Osiąga to przez wprowadzenie sieci komputerowych, co znacznie zmniejsza koszt ponoszony przez jednego użytkownika Ponadto wdrożono także system, który wykorzystuje interfejs opierający się na ikonach łącznie z wejściem głosem. W rozwiązaniu tym rozpoznanie mowy jest wykonywane w większości programowo

A następny krok?

Staniemy przed komputerami, z którymi można porozmawiać. Przed niot tej konwersacji, jej zakres zależny jest od oprogramowania. Przewiduję się że najczęściej będzie to system typu EKSPERT

System ekspert ma zaprogramowane zasady i informacje, które są dostateczne, by zadawać pytania użytkownikowi, oceniać odpowiedzi i odpowiadać samemu. Jeśli konwersacja odbywa się za pomocą głosu można autentycznie odczuć życzliwość komputera i sympatię... do tego urządzenia

Taka „rozmowa" wymaga nadzwyczajnych prac ekspertów, programowania, rejestracji faktów i zasad obowiązujących w systemie ekspert. Ponadto komputer powinien być „wyćwiczony" w rozumieniu tysięcy słów w swoim słowniku, bez względu na akcent mówiącego

Zapewne większość czytelników „IKS-a" z niecierpliwością oczekuje upowszechnienia sygnalizowanych objawów życzliwości komputerów. Należy jednak pamiętać, że nasze biura, redakcje, staną się wówczas niezwykle gwarnymi miejscami.

W.G.



...życzliwe czy nieżyczliwe, zabawa z nimi wspaniała!

BASIC 6

(Podstawy programowania)

Włączamy komputer i zaczynamy.

Wykład szósty: zagnieżdżone instrukcje cyklu (pętle), zasady tworzenia cykli obliczeniowych (pętli); scroll („przewijanie" ekranu), komendy: BREAK, CONTINUE; instrukcje: CLS, PAUSE;

W wykładzie czwartym zapoznaliśmy się z cyklami obliczeniowymi (pętlami), oraz z instrukcjami cyklu FOR i NEXT za pomocą, których zapisujemy pętle w BASICU.

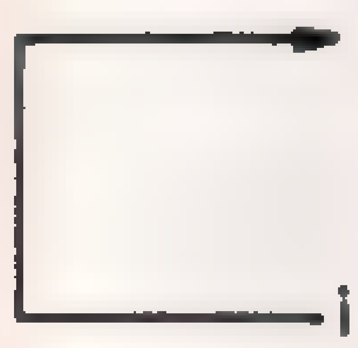
Dotychczas we wszystkich przykładach występowały pojedyncze cykle obliczeniowe postaci:

```
40 FOR I = 1 TO n STEP 1
60 REM wnętrze pętli
70 NEXT i
```

Natomiast w tym wykładzie omówimy zagnieżdżone instrukcje cyklu (pętle) oraz zasady tworzenia pętli.

W ramach jednego cyklu obliczeniowego (jednej pętli) mogą być wykonywane dalsze cykle obliczeniowe. Tak więc pętle mogą być (zagnieżdżone) jedna w drugiej, ale nigdy nie mogą zachodzić jedna na drugą. W tym wykładzie, jak i w następnych będziemy posługiwali się tylko nazwą pętle obliczeniowe. Zasady tworzenia pętli najłatwiej jest zobrazować graficznie:

1) pętla pojedyncza

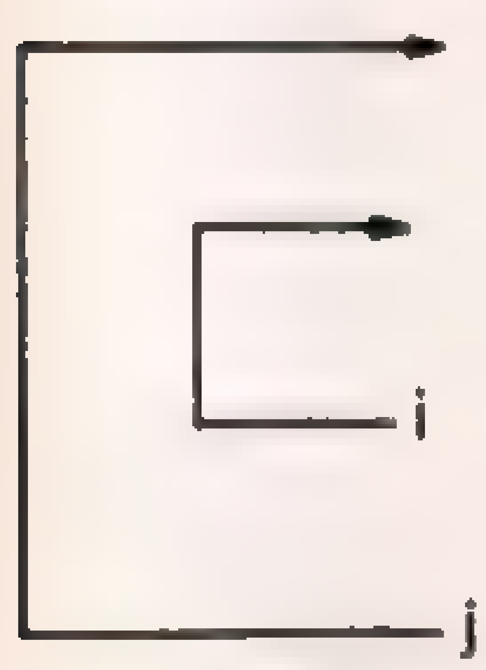


2) pętle rozłączne

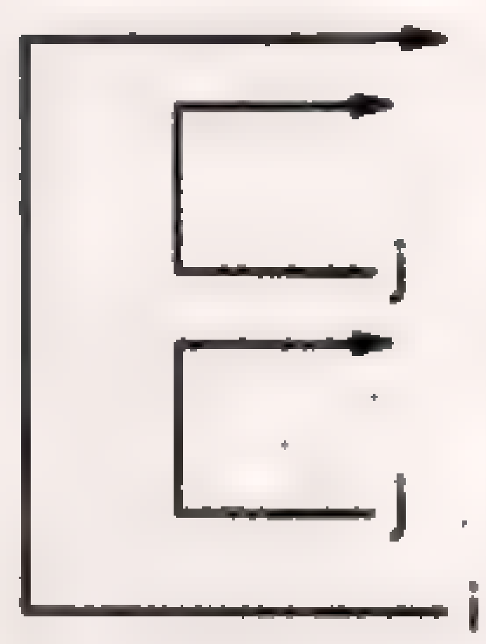


3) pętla podwójna

a)



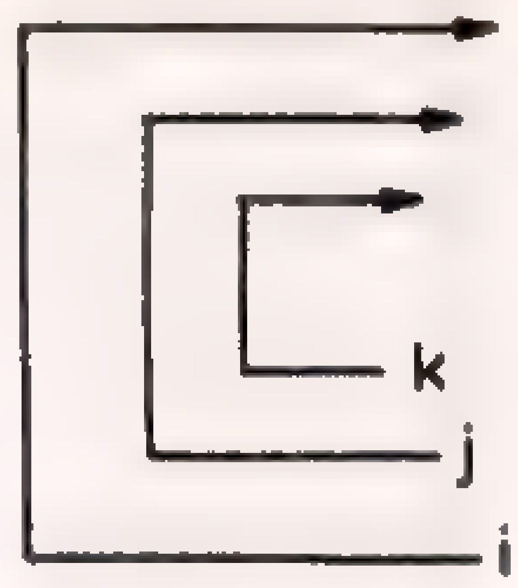
b)



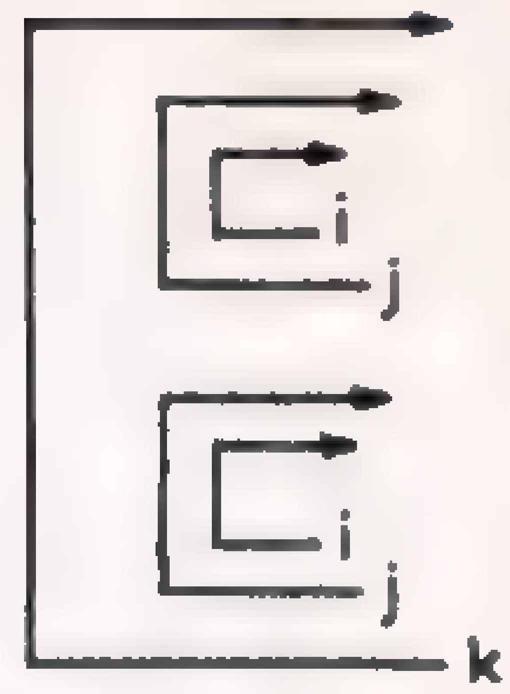
dwukrotne zagniezdzenie pętli

4) pętla trzykrotna

a)



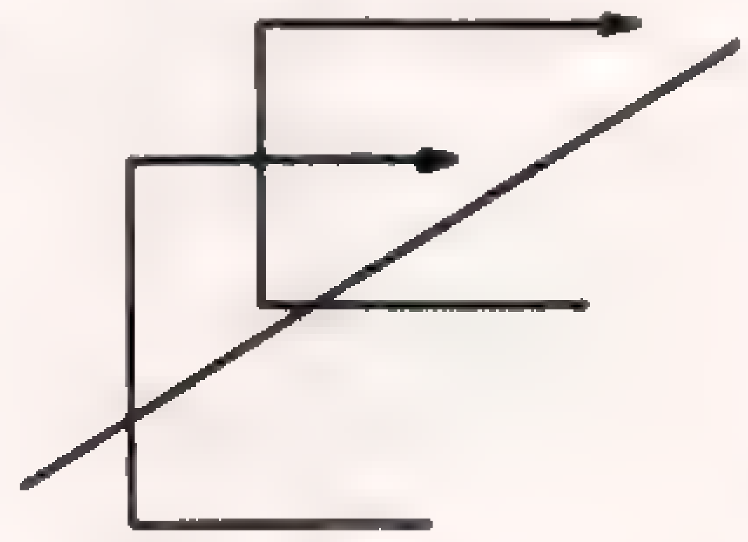
b)



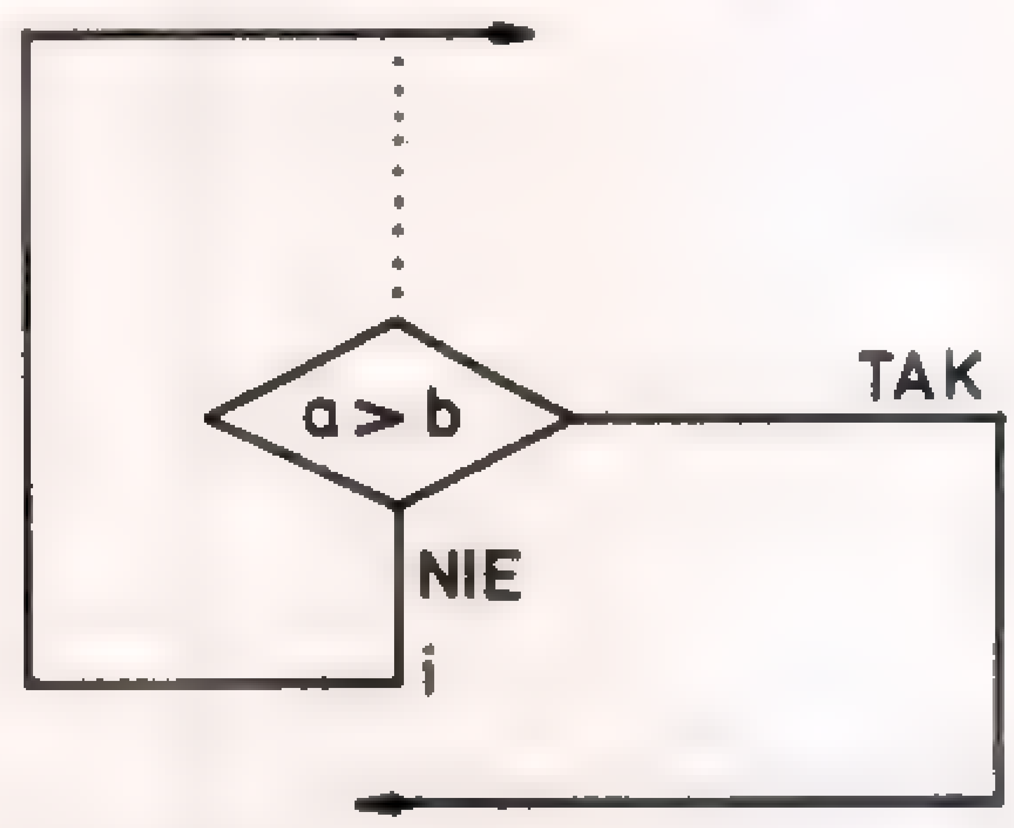
trzykrotne zagnieżdżenie pętli

5) pętle nie mogą się przecinać (zachodzić jedna na drugą):

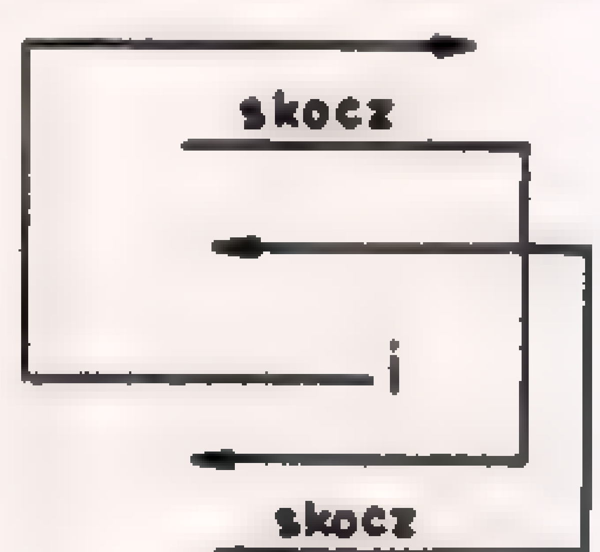
źle



6) z pętli można „wyskoczyć" za pomocą

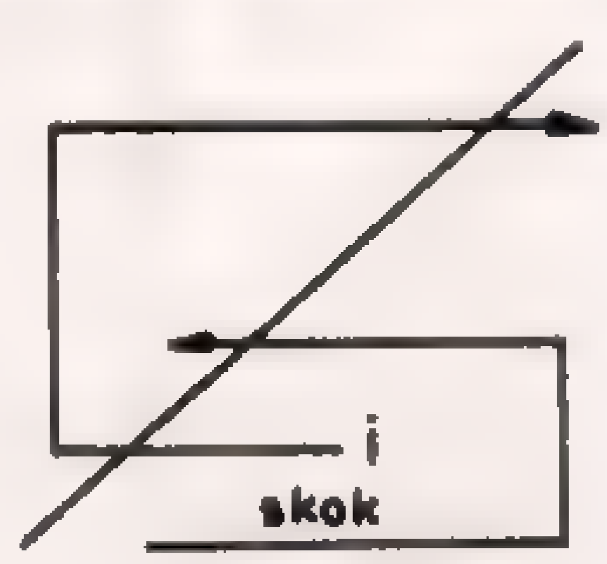


7) z pętli można „wyskoczyć" i powrócić do pętli z zachowaniem wartości zmiennej sterującej:



8) nie wolno wykonywać skoku do pętli z zewnątrz:

źle



Pętle typu 1) i 2) występowały w programach wykładu czwartego. Zajmiemy się więc omówieniem pętli typu 3) ÷ 8)

dokończenie na str. 22

dokończenie ze str. 21

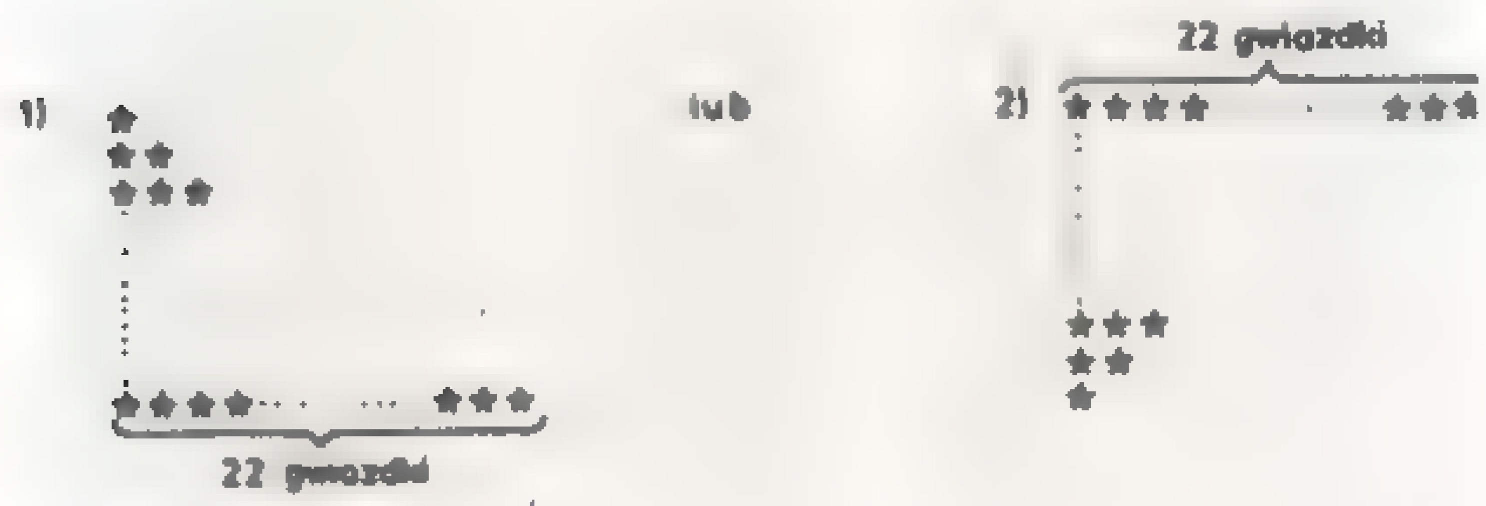
Przykład 5.1

Napisać program, który w każdym znaku ekranu drukuje (gwiazdkę)

```
 10 REM Przyklad 5.1
 15 REM ekran-22 linie po 32
        znaki
 20 REM petla typu 3a) -
dwukrotne zagniezdzenie petli
 30 FOR i=1 TO 22
 40 FOR j=1 TO 32
 50 PRINT "*";
 60 NEXT j
 70 PRINT
 80 NEXT i
```

Przykład 5.2

Napisać programy, które drukują figury postaci



a) programy:

```
 10 REM Przyklad 5.2.1
 20 REM petla typu 3a) -
dwukrotne zagniezdzenie petli
 30 FOR i=1 TO 22
 40 FOR j=1 TO i
 50 PRINT "*";
 60 NEXT j
 70 PRINT
 80 NEXT i
```

```
 10 REM Przyklad 5.2.2
 20 REM petla typu 3a) -
dwukrotne zagniezdzenie petli
 30 FOR i=22 TO 1 STEP -1
 40 FOR j=1 TO i
 50 PRINT "*";
 60 NEXT j
 70 PRINT
 80 NEXT i
```

W programach tych zmienna sterująca i jest wartością końcową zmiennej sterującej (wartość zmiennej sterującej można wykorzystywać w pętli, ale nie wolno jej zmieniać)

Przykład 5.3

Napisać program, który wypełnia ekran „szlaczkiem" postaci:

1 linia: • ★ • ★ .... • ★

2 linia: + – + – .... + –

3 linia: • ★ • ★ .... • ★

4 linia: + – + – .... + –

itd.

Program:

```
 10 REM Przyklad 5.3
 15 REM ekran - 22 linie po 32
        znaki
 20 REM petla typu 3b)
 30 FOR i=1 TO 11
 35 REM do wypelnienia jest 11
        podwojnych linii
 40 FOR j=1 TO 16
 45 REM w kazdej linii jest do
        wypelnienia 16 pozycji
        dwuznakowych
 50 PRINT ".*";
 60 NEXT j
 70 PRINT
 80 FOR j=1 TO 16
 90 PRINT "+-";
100 NEXT j
110 PRINT
120 NEXT i
```

Przykład 5.4

Napisać program, który numeruje znaki w każdej linii ekranu w następujący sposób

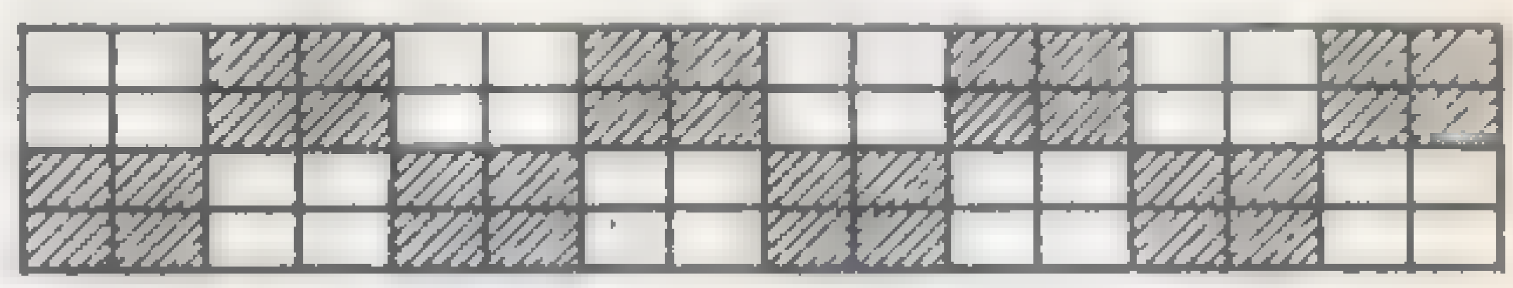
01234567890123456789012345678901 (32 znaki)

Program

```
 10 REM Przyklad 5.4
 20 REM petla typu 4a) -
 trzykrotne zagniezdzenie petli
  25 REM aby program byl bar-
dziej czytelny (przejrzysty)
mozna go zapisac  jak nizej,
uwidaczniajac zagniezdzenie
instrukcji FOR
  30 FOR i=1 TO 22
  40   FOR j=1 TO 3
  50     FOR k=0 TO 9
  60     PRINT k;
  70     NEXT k
  80   NEXT j
  90 PRINT 0;1
 100 NEXT i
```

Przykład 5.5

Napisać program, który „rysuje" na ekranie szachownicę. Każde pole szachownicy ma zajmować na ekranie 4 znaki



```
 10 REM Przyklad 5.5
 15 REM szachownica -
        liczba znakow = 256
        (16 linii*16 znakow
                  w linii)
 20 REM petla typu 4b)
 30 FOR i=1 TO 4
 40 FOR j=1 TO 2
 50 FOR k=1 TO 4
 60 PRINT "  ▀▀";
 65 REM █ - otrzymujemy na kur-
        sorze G,naciskajac jed-
        noczesnie klawisze
        CAPS SHIFT i 8
 70 NEXT k
 80 PRINT
 90 NEXT j
100 FOR j=1 TO 2
110 FOR k=1 TO 4
120 PRINT "▀▀  ";
130 NEXT k
140 PRINT
150 NEXT j
160 NEXT i
```

Przykład 5.6

Napisać program, który drukuje potęgi od 1 do 4 wszystkich liczb od 1 do 10.

Program

```
10 REM Przyklad 5.6
20 FOR i=1 TO 4
30 FOR j=1 TO 10
40 LET k=j↑i
50 PRINT k
60 NEXT j
70 NEXT i
80 STOP
```

W trakcie realizacji naszego programu, gdy został zapełniony liczbami cały ekran (wszystkie linie) na dole ekranu pokazał się napis *s c r o l l?* Naciśnijmy dowolny klawisz np. klawisz A. Liczby napisane w górnych liniach ekranu „chowają" się poza ekran, a od dołu ekranu zaczynają się wprowadzać nowe linie z następnymi liczbami. Po każdym napisie *s c r o l l?* naciśnijmy ten sam klawisz. Przesuwanie się linii do góry zakończy się w tym momencie, gdy na dole ekranu pokaże się napis

9 STOP statement, 80 : 1.

Widać z tego przykładu, że gdy górna część ekranu zapełnia się, to zostaje on przesunięty w górę o jedną linię (itd.). Jeżeli miałoby to pociągnąć za sobą utratę linii, którą chcielibyśmy oglądać, komputer staje i pyta: *s c r o l l?*

Naciśnięcie klawisza **N, SPACE, STOP** lub **BREAK** zatrzyma wykonywanie programu z komunikatem:

D BREAK — CONT repeates nr 1 : nr 2

nr 1 — numer wiersza, w którym został zatrzymany program,

nr 2 — numer instrukcji w tym wierszu.

Natomiast naciśnięcie dowolnego innego klawisza spowoduje przesunięcie ekranu.

Wykonajmy jeszcze raz nasz program (RUN, ENTER). Ekran zapełnia się liczbami i pojawi się nam napis: *s c r o l l?*, naciśnijmy teraz klawisz **N** i **ENTER,** na dole ekranu pojawi się komunikat:

D BREAK — CONT repeates, 40 : 1

Naciśnijmy teraz klawisz **CONT — CONTINUE, komputer „przypomina sobie" numer wiersza podany w ostatnim komunikacie i rozpoczyna wykonywanie programu od tego wiersza (powtarza wiersz).**

Wykonajmy jeszcze raz nasz program. Gdy ekran zapełni się do połowy liczbami naciśnijmy równocześnie klawisze **CAPS SCHIFT** i **SPACE (BREAK)** — przerwiemy wykonywanie programu („zawiesimy" program). W lewym dolnym rogu ekranu pojawi się komunikat

L BREAK — CONT repeates, nr 1 : nr 2

Naciśnijmy teraz klawisz **CONT.** Po wciśnięciu **CONTINUE realizacja programu rozpocznie się od następnego wiersza po wierszu, którego numer został podany w komunikacie.**

Wykonajmy jeszcze raz nasz program. Gdy ekran zapełni się np. do połowy liczbami naciśnijmy klawisz **STOP** — przerwiemy wykonywanie programu, w lewym dolnym rogu ekranu pojawi się komunikat.

9 STOP statement, nr 1 : nr 2

Naciśnijmy teraz klawisz **STOP. Po wciśnięciu CONTINUE realizacja programu rozpocznie się od następnego wiersza, tzn. nie powtarza się instrukcji STOP.**

Przykład 5.7

Napisać program, który drukuje tabliczkę mnożenia w następujący sposób:

1 * 1 = 1   2 * 1 = 2
1 * 2 = 2   2 * 2 = 4

1 * 10 = 10   2 * 10 = 20
itd.

a) program:

```
 10 REM Przyklad 5.7
 20 PRINT "     TABLICZKA  MNOZE
NIA"
 25 PRINT
 30 FOR i=1 TO 9 STEP 2
 40 FOR j=1 TO 10
 50 PRINT i;" * ";j;" = ";i*j,i
+1;" * ";j;" = ";(i+1)*j
 60 NEXT j
 70 PRINT
 80 PAUSE 250
 90 CLS
100 NEXT i
```

Wprowadźmy i wykonajmy ten program.

Poniżej zamieszczamy przykładowe wyniki

b)

```
     TABLICZKA   MNOZENIA

1 * 1 = 1          2 * 1 = 2
1 * 2 = 2          2 * 2 = 4
1 * 3 = 3          2 * 3 = 6
1 * 4 = 4          2 * 4 = 8
1 * 5 = 5          2 * 5 = 10
1 * 6 = 6          2 * 6 = 12
1 * 7 = 7          2 * 7 = 14
1 * 8 = 8          2 * 8 = 16
1 * 9 = 9          2 * 9 = 18
1 * 10 = 10        2 * 10 = 20
```

W czasie realizacji naszego programu zauważamy, że gdy zostaną wydrukowane wyniki dla dwóch kolejnych liczb np. 1 i 2, lub 3 i 4 itd., wyniki te „stoją" na ekranie pewien czas, a następnie ekran zostaje oczyszczony.

W programie napisano instrukcje.

80 PAUSE 250

oraz

90 CLS

Instrukcja

n **PAUSE** m

n — numer wiersza

m — liczba

zatrzymuje działanie programu i wyświetla obraz przez m/50 sek. ramki ekranu (ramek tych wyświetlanych jest 50 na sekundę).

W naszym programie obraz na ekranie będzie „stał" przez 5 sekund, umożliwi to odczytanie wyników umieszczonych na ekranie.

Instrukcja

n CLS

n — numer wiersza

oczyszcza ekran z wszelkich informacji (CLS — clear screen).

CLS można używać w BASICU-u na ZX Spectrum, jako instrukcję, lub jako komendę (omawianą w pierwszym wykładzie).

Można usunąć z programu te dwie instrukcje tzn. 80 PAUSE 250 i 90 CLS, wtedy w czasie realizacji programu pojawi się napis *s c r o l l?* — działanie zostało omówione przy omawianiu przykładu 5.6.

dokończenie na str. 24

dokończenie ze str. 23

Przykład 5.8

Napisać program, który drukuje na ekranie liczby 1,2,3 w następujący sposób.

znaki / linie	1	2	3	....	22	32
1	2	3	3	....	3	
2	1	2	3		3	
3	1	1	2		3	
:	:	:	:		:	
:	:	:	:		3	
22	1	1	1	 1	2	

a) program

```
 10 REM Przyklad 5.8
 15 REM petla typu 7)
 20 FOR i=1 TO 22
 30 FOR j=1 TO 22
 40 IF i=j THEN GO TO 100
 50 IF i>j THEN GO TO 120
 60 PRINT 3;
 70 NEXT j
 75 PRINT
 80 NEXT i
 90 STOP
100 PRINT 2;
110 GO TO 70
120 PRINT 1;
130 GO TO 70
```

b) wyniki

```
2333333333333333333333
1233333333333333333333
1123333333333333333333
1112333333333333333333
1111233333333333333333
1111123333333333333333
1111112333333333333333
1111111233333333333333
1111111123333333333333
1111111112333333333333
1111111111233333333333
1111111111123333333333
1111111111112333333333
1111111111111233333333
1111111111111123333333
1111111111111112333333
1111111111111111233333
1111111111111111123333
1111111111111111112333
1111111111111111111233
1111111111111111111123
1111111111111111111112
```

Tak napisany program ilustruje nam pętlę typu 7)
Instrukcje

40 **IF** i = j **THEN GOTO** 100

oraz

50 **IF** i > j **THEN GOTO** 120

„wyprowadzają" realizację programu poza pętlę, natomiast instrukcje

110 GOTO 70 i 130 GOTO 70

„wprowadzają" realizację programu w obszar pętli (koniecznie w obszar pętli wewnętrznej tzn. pętli ze zmienną sterującą j)

Instrukcja

90 **STOP**

spowoduje, że po zakończeniu pętli program nie wykona następnych instrukcji, które właściwie należą do wnętrza pętli.

W BASIC-u na ZX Spectrum, program ten można napisać bez „wyprowadzania" jego realizacji poza obszar pętli, a tym samym ograniczyć liczbę skoków, a więc w postaci:

```
 10 REM Przyklad 5.8.1
 20 FOR i=1 TO 22
 30 FOR j=1 TO 22
 40 IF i=j THEN PRINT 2; : GO TO
70
 50 IF i>j THEN PRINT 1; : GO TO
70
 60 PRINT 3;
 70 NEXT j
 75 PRINT
 80 NEXT i
 85 REM instrukcja STOP -
        niekonieczna
 90 STOP
```

Przykład 5.9

Dany jest program

```
 10 REM Przyklad 5.9
 20 REM petla typu 5)
 30 FOR i=1 TO 4
 40 FOR j=1 TO 5
 50 PRINT i;" * ";j;" = ";i*j
 60 NEXT i
 70 NEXT j
 75 PRINT
 80 PRINT '"zla konstrukcja pet
li!          petle przecinaja si
e!"
```

Wprowadźmy i wykonajmy ten program. Niżej prezentujemy otrzymane wyniki.

```
1 * 1 = 1
2 * 1 = 2
3 * 1 = 3
4 * 1 = 4
5 * 2 = 10
6 * 3 = 18
7 * 4 = 28
8 * 5 = 40

zla konstrukcja petli!
petle przecinaja sie!
```

Analizując otrzymane wyniki zauważymy, że nie są one poprawne. Program ten miał drukować wyniki mnożenia liczb od 1 do 4 przez liczby od 1 do 5, a więc wyniki postaci.

$1 * 1 = 1$
$1 * 2 = 2$
$1 * 3 = 3$
$1 * 4 = 4$
$1 * 5 = 5$
$2 * 1 = 2$
$2 * 2 = 4$

$4 * 4 = 16$
$4 * 5 = 20$

Natomiast wyniki, które otrzymaliśmy, całkowicie odbiegają od oczekiwanych.

Konstrukcja tego programu jest błędna. Pętle obliczeniowe przecinają się (zachodzą jedna na drugą).
Wynik działania takiego programu jest nieokreślony tzn. w różnych reprezentacjach BASIC-u możemy otrzymać róż-

ne wyniki, ponadto jest zawsze inny od tego, jaki chcieliśmy otrzymać.

Zamieńmy miejscami instrukcje

60 **NEXT** i
70 **NEXT** j

tzn. napiszmy

60 **NEXT** j
70 **NEXT** i

Wykonując tak poprawiony program otrzymamy oczekiwane przez nas wyniki.

Przykład 5.10

Dany jest program

```
10 REM Przyklad 5.10
20 REM petla typu 6)
25 PRINT "zla konstrukcja - n:
          wolno wykonywac skol
          do petli z zewnatrz.
27 PRINT
30 FOR i=1 TO 4
40 FOR j=1 TO 5
50 PRINT i;" * ";j;" = ";i*j
60 NEXT j
70 NEXT i
75 GO TO 50
80 PRINT
90 STOP
```

Wprowadźmy i wykonajmy ten program. Niżej prezentujemy przykładowe wyniki

```
zla konstrukcja - nie
wolno wykonywac skoku
do petli z zewnatrz!

1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
5 * 6 = 30
6 * 7 = 42
7 * 8 = 56
8 * 9 = 72
9 * 10 = 90
10 * 11 = 110
11 * 12 = 132
12 * 13 = 156
13 * 14 = 182
```

Wyników tych byłoby więcej, ale zatrzymano działanie programu naciskając klawisz **BREAK.** Program ten realizuje to samo zadanie co program z przykładu 5.9.

Część wyników jest poprawna (do kreski) i to miał zrealizować program, natomiast (poniżej kreski) inne wyniki są przypadkowe. Zostało to spowodowane umieszczeniem w programie instrukcji:

75 **GOTO** 50

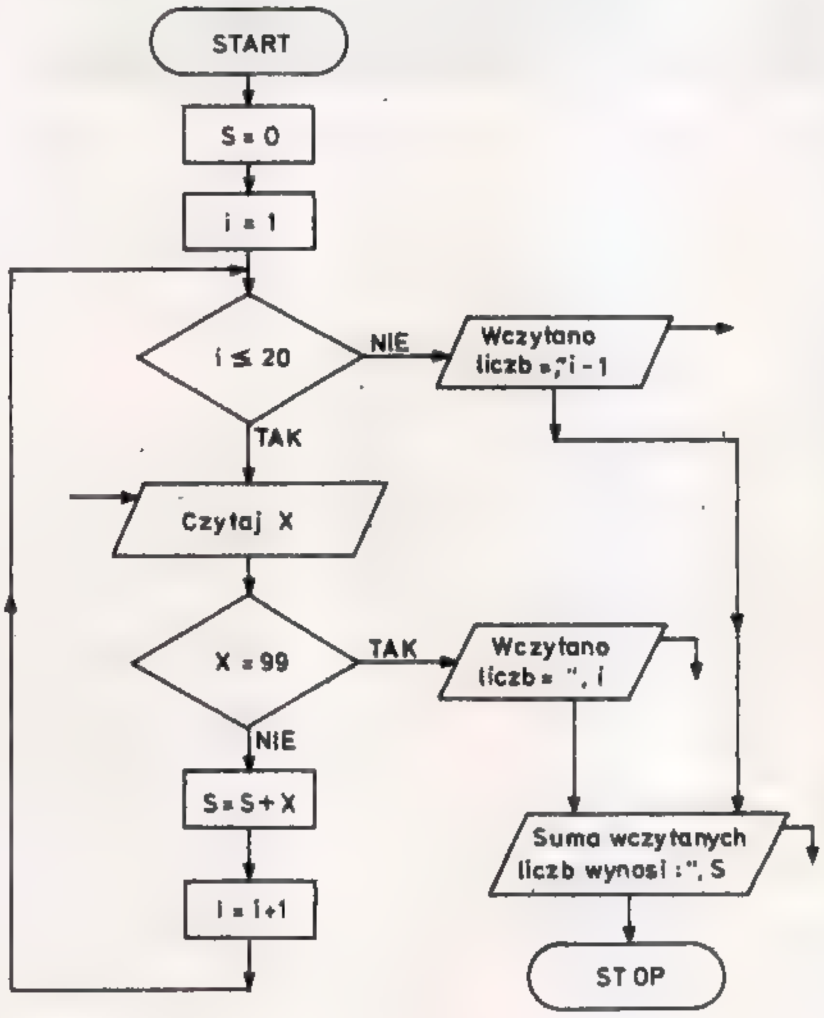
która po zakończeniu pętli wprowadza działanie programu do wnętrza pętli, a tym samym pętle nie rozpoczynają się od podstawienia wartości początkowych i sprawdzenia warunku niewykonywania pętli (patrz wykład 4).

Taki zapis jest błędny, gdyż nie wolno wykonywać skoku do wnętrza pętli z zewnątrz, jeżeli uprzednio z niej nie „wyskoczono".

Przykład 5.11

Napisać program na obliczanie sumy 20 liczb wczytywanych w trakcie realizacji programu. Jeżeli wczytywana liczba jest równa 99, to sumowanie należy zakończyć (nawet gdy wczytano mniej niż 20 liczb).

a) schemat blokowy



b) program

```
 10 REM Przyklad 5.11
 20 REM petla typu 6)
 25 LET s=0
 30 FOR i=1 TO 20
 40 INPUT "x=";x
 50 IF x=99 THEN GO TO 100
 60 LET s=s+x
 70 NEXT i
 80 PRINT "wczytano liczb =";i-
1
 90 GO TO 110
100 PRINT "wczytano liczb =";i
110 PRINT '"suma wczytanych lic
zb        wynosi : ";s
120 STOP
```

c) wyniki

```
wczytano liczb =11

suma wczytanych liczb
wynosi : 10


wczytano liczb =11

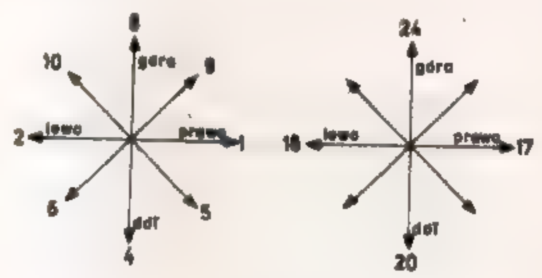
suma wczytanych liczb
wynosi : 10
```

dokończenie na str. 26



Jeśli masz w domu SPECTRUM i popularną przystawkę z interfejsem typu KEMPSTON nazywaną z ang. JOYSTICKIEM i nie wiesz jak wykorzystać ją we własnych programach, poniższy tekst powinien ci pomóc. Poruszając drążkiem powodujesz przesłanie odpowiednich wartości do PORTU 31, który w przypadku KEMPSTONA służy do komunikacji z komputerem. Wystarczy tylko odczytać pojawiające się w porcie informacje i wykorzystać je.

Rysunki przedstawiają wartości dziesiętne odwzorowywane przez JOYSTICK:



Rysunek z prawej podaje zawartość PORTU przy jednoczesnym używaniu przycisku FIRE. Jak widać, wartości pośrednie są sumą wartości przy wychyleniach drążka w pionie i poziomie, zaś FIRE dokłada do PORTU liczbę 16. Oto prosty przykład sterowania ruchem punktu na ekranie.

```
10 CLS
20 LET x=128: LET y=88
30 PLOT x,y
40 IF IN 31=1 OR IN 31=17 THEN
 PLOT INVERSE 1;x,y: LET x=x+1
 : IF x=256 THEN LET x=255
50 IF IN 31=2 OR IN 31=18 THEN
 PLOT INVERSE 1;x,y: LET x=x-1
 : IF x=-1 THEN LET x=0
60 IF IN 31=4 OR IN 31=20 THEN
 PLOT INVERSE 1;x,y: LET y=y-1
 : IF y=-1 THEN LET y=0
70 IF IN 31=8 OR IN 31=24 THEN
 PLOT INVERSE 1;x,y: LET y=y+1
 : IF y=176 THEN LET y=175
80 IF IN 31>=16 THEN BEEP .000
5,50
90 GOTO 30
```

Następny przykład jest mniej poglądowy, lecz krótszy, szybszy (skrócone instrukcje warunkowe) oraz uwzględniający kierunki pośrednie:

```
10 CLS
20 LET x=128: LET y=88
30 PLOT x,y
40 LET x1=x: LET y1=y
50 LET b=IN 31: LET a=b-16*(b>
15)
60 LET x=x+(x<255)*(a=1 OR a=5
 OR a=9)
70 LET y=y+(y<175)*(a>3 AND a<
7)
80 LET x=x-(x>0)*(a=2 OR a=6 O
R a=10)
90 LET y=y-(y>0)*(a>7 AND a<11
)
100 PLOT INVERSE (b<16);x1,y1
110 GOTO 30
```

Spójrzmy teraz na PORT 31, jak na 8--bitowy rejestr. Porównując wartości z rysunków odkryjemy, że każdy tryb pracy JOYSTICKA jest związany z uaktywnieniem odpowiedniego bitu w PORCIE.

bit	7	6	5	4	3	2	1	0
	–	–	–	FIRE	GÓRA	DÓŁ	LEWO	PRAWO

Przykładowo, wychylenie drążka w lewo spowoduje wpisanie jedynki logicznej do bitu 3, zaś odchylenie w kierunku prawo—dół z przyciśnięciem FIRE uaktywni bity 0,2 i 4. Można już spróbować oprogramować JOYSTICK w kodzie wewnętrznym Z 80 (GENS 3M21)

```
10    ....................
20 KEMPS   IN A,(31)    ;odczyt portu
30         BIT 4,A      ;TEST polozenia FIRE
40         CALL NZ,FIRE;jesli bit 4=1 wykonaj podp
rogram FIRE
50         BIT 3,A      ;TEST ruchu "od siebie"
60         CALL NZ,UP   ;jesli bit 3=1 wykonaj podp
rogram UP
70         BIT 2,A      ;TEST ruchu "do siebie"
80         CALL NZ,DOWN;jesli bit 2=1 wykonaj podp
rogram DOWN
90         BIT 1,A      ;TEST ruchu w lewo
100         CALL NZ,LEFT;jesli bit 1=1 wykonaj pod
program LEFT
110        BIT 0,A      ;TEST ruchu w prawo
120        CALL NZ,RIGHT;jesli bit 0=1 wykonaj RIG
HT
130 ....................
```

KRZYSZTOF MAMCARZ

dokończenie ze str. 25

Wyniki (i) reprezentują ciąg 20 liczb, w którym nie było liczby równej 99, natomiast wyniki (ii) reprezentują ciąg liczb, w którym liczba jedenasta równała się 99. W tym przypadku wykonywanie pętli zostało przerwane spełnieniem warunku postaci:

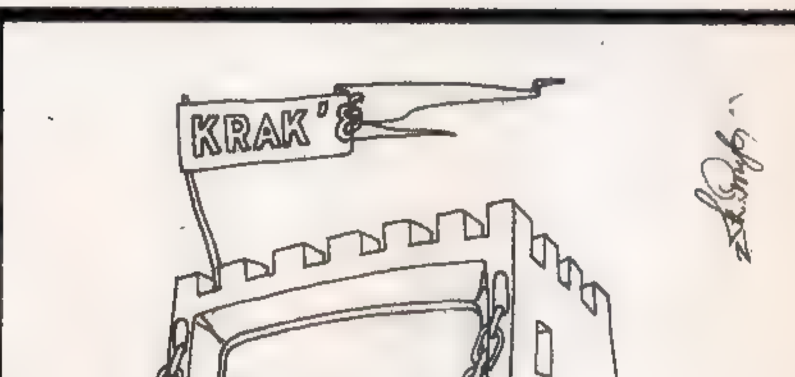
50 **IF** X = 99 **THEN GOTO** 100

Zadania

5.1. Kamienie domina zawierają po dwa pola. Każde pole może być puste lub zawierać od 1 do 6 punktów. Wygenerować wszystkie kamienie jakie można utworzyć z pól o różnej lub tej samej liczbie punktów (takich kamieni jest 28). Każdy kamień przedstawić jako parę liczb, a mianowicie liczb punktów jego pól.

5.2. Napisać program znajdowania takich trzech cyfr, których suma sześcianów jest równa liczbie trzycyfrowej z nich skonstruowanej.
(Uwaga: należy rozwiązać równanie:
$A^3 + B^3 + C^3 = A \times 10^2 + B \times 10 + C$)

Z.Z.W.



Rys. Michał Przybyłowski

Jak to zrobić (3)

Zdarza się, że koncentrując uwagę na jednym problemie, nie dostrzegamy spraw, zdawałoby się mniej ważnych, które innym „rzucają" się w oczy.

W programie zamieszczonym w poprzednim numerze chciałem pokazać, w jaki sposób można wykorzystać grafikę *player-missile*. Pierwsza uwaga, z jaką się spotkałem na jego temat, brzmiała:

„Ludzik może być, ale dlaczego ta gwiazdka, To musi być jakiś inny znak".

Niestety w trybie graficznym używanym w programie nie jest możliwe wypisanie kropki jako znaku mnożenia, jak to się robi w szkole. Spróbujmy jednak zastanowić się, jak należy poprawić program, aby zrealizować ten postulat.

Zacznijmy od garści niezbędnych informacji.

Wszystkie znaki, jakie potrafi napisać nasz komputer zapamiętane są w jego stałej pamięci (ROM) zaczynając od komórki o adresie 57344 — obszar ten bywa nazywany generatorem znaków. Każdy znak jest w nim opisany za pomocą ośmiu bajtów, odpowiadających ośmiu liniom na ekranie monitora. W każdej z tych linii świecą się tylko te punkty, dla których bit, przypisany im we właściwym bajcie ma wartość 1.

Dla dowolnie wybranego znaku ZN bardzo łatwo jest odnaleźć pierwszy bajt jego opisu — wystarczy posłużyć się wzorem

ADDR + (ASC (ZN) — 32) * 8

gdzie: ADDR jest adresem generatora znaków,
ASC jest znaną funkcją BASIC'a podającą kod ASCII znaku będącego jej parametrem.

Charakter pamięci przechowującej wzorce znaków nie pozwala na wprowadzenie w nich jakichkolwiek zmian. Musimy to zrobić inaczej — oszukamy komputer przygotowując nowy zestaw pozmienianych znaków.

Zaczniemy od zarezerwowania pamięci na ich opisy. Posłużmy się instrukcją

```
10 POKE 106, PEEK (106) — 8
```

Możemy teraz przepisać definicje znaków do „naszej" pamięci rozpoczynającej się od adresu obliczonego w instrukcji

```
20 ADDR = PEEK (106) 256
```

Można to zrobić w sposób następujący

```
30 MEM = 57344
40 FOR I = 0 TO 1023
50 POKE ADDR + 1,
PEEK (MEM + I)
60 NEXT I
```

Najtrudniejszą część mamy już za sobą.

Wykorzystując podany wyżej wzór na szukanie bajtów opisujących wybrany znak, w prosty sposób możemy usunąć przyczynę całego zamieszania i zmodyfikować znak „ * ". Poniższe instrukcje zamienią go na kropkę.

```
70 A = ASC /„ * "/
80 M = ADDR + (A — 32) 8
90 FOR I — 0 TO 7
100 READ B : POKE M + I, B
110 NEXT I
120 DATA 0, 0, 0, 24, 24, 0, 0, 0
130 GR. 18
```

Przed nami najważniejsze, musimy „namówić" komputer, by korzystał z naszego zestawu znaków. Jeżeli to zrobimy, osiągniemy cel.

A zrobimy to bez większych problemów wiedząc, że w komórce o adresie 756 przechowywany jest numer bloku, od którego rozpoczyna się pamięć przeznaczona na opisy znaków. Musimy zatem wpisać do niej adres odpowiedniego bloku:

```
140 POKE 756, PEEK (106)
```

UWAGA:

Określenie trybu graficznego powinno nastąpić wcześniej, bowiem instrukcja GRAPHICS spowoduje powrót do starego generatora znaków.

Teraz już możemy napisać wymarzone działanie mnożenia z piękną kropką w miejscu gwiazdki.

```
150 POSITION 2,2
160 PRINT # 6: „x * 3 = 9"
```

(Nie martwmy się, na ekranie pojawi się kropka!).

Idąc „za ciosem" uporać się możemy z podobnymi uchybieniami naszego programu i nauczyć go pisać litery polskiego alfabetu.

Następujące instrukcje:

```
170 DIM BS (1)
180 READ BS : IF BS = „* " THEN
GOTO 300
190 M = ADDR + (ASC (BS) —
32) * 8
200 FOR I = 0 TO 7
210 READ D : POKE M + I,
PEEK (M + I) + D
220 NEXT I
230 GOTO 170
240 DATA C, 8, 0, 0, 0, 0, 0, 0, 0
250 DATA E, 0, 0, 0, 0, 0, 8, 0, 4
260 DATA S, 8, 0, 0, 0, 0, 0, 0, 0,
270 DATA *
300 POSITION 4,4 : PRINT # 6;
„CZEŚĆ"
310 POSITION 5,4 : PRINT # 6:
"PODAJ WARTOŚĆ X"
```

spowodują, że teksty umieszczone w liniach 300 i 310 zostaną wypisane tak, jak oczekiwaliśmy.

Teraz już pozostaje dopisać instrukcję, która zatrzyma obraz na ekranie, np

```
400 GOTO 400,
```

wpisać RUN i przekonać się, czy to, co zostało napisane wyżej jest prawdą.

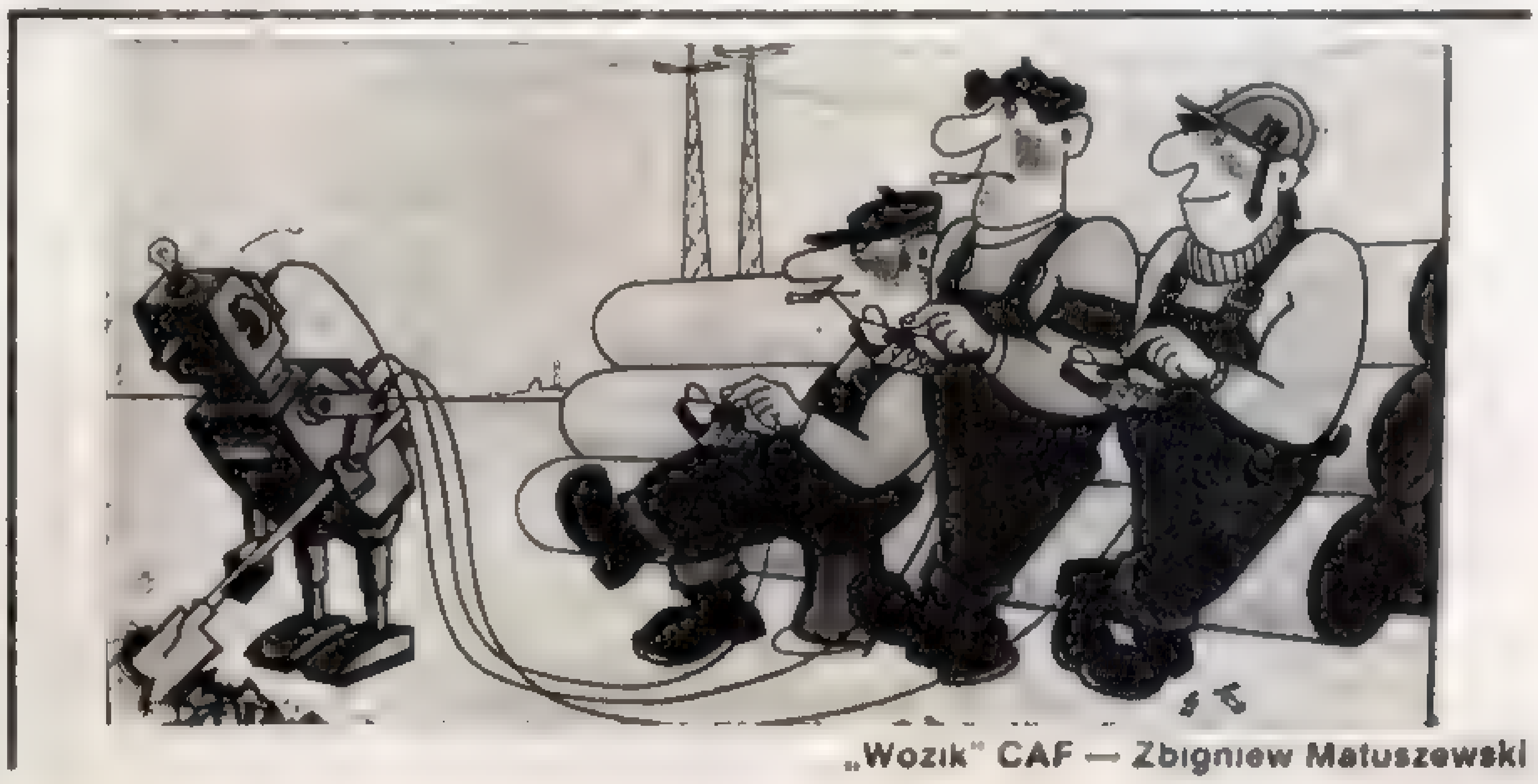
Myślę, że po przeczytaniu tego artykułu, gdy zajdzie potrzeba, każdy będzie potrafił przygotować swój własny zestaw znaków.

Wszystkim, amatorom programowania, którzy zamierzają w praktyce wykorzystać to, o czym dziś mówiliśmy, chciałbym udzielić konkretnej pomocy podając tekst procedury kopiującej komplet 1024 znaków w czasie o wiele krótszym niż użyta instrukcja: 40 FOR..:

```
1000 DIM PROCS (40)
1010 MEM = PEEK (106) — 8 : POKE
106, MEM
1020 CHADR = MEM * 256
1030 FOR I = 1 TO 32 : READ B
PROCS (I) = CHRS (B) : NEXT I
1040 DATA 104, 104, 133, 213, 104,
        133, 212, 104, 133, 215,
        104, 133, 214, 162
1050 DATA 4, 160, 0, 177, 212, 145,
        214, 200, 208, 249, 230,
        213, 230, 215, 202, 208,
        240, 96
1060 A : USR (ADR (PROCS),
224 * 256, CHADR)
1070 RETURN
```

Życzę powodzenia.

Grześ



„Wozik" CAF — Zbigniew Matuszewski

opis str. 2

SHARP

SZYFRACJA

```
 10:"X":FOR K=0TO
    N
 20:B$=K$:IF Q=1
    LET A$=J$(K):
    GOTO 40
 30:A$=J$(K)
 40:FOR L=0TO 16
 50:M=16-L:X$=
    RIGHT$ (A$,L):
    A$=LEFT$ (A$,M
    ):A$=A$+O$
 60:Y$=RIGHT$ (B$,
    L):B$=LEFT$ (B
    $,M):B$=B$+O$
 70:X=ASC X$:Y=ASC
    Y$:W=X+Y:T=W-3
    2:IF Q=1GOTO 1
    00
 80:IF T>91LET T=T
    -91:GOTO 80
 90:GOTO 120
100:W=X-Y:T=W-32
110:IF T<0LET T=T+
    91:GOTO 110
120:W=T+32
130:P$=CHR$ W+P$
140:NEXT L:IF Q=1
    LET N$(K)=P$:
    GOTO 160
150:T$(K)=P$
160:NEXT K:RETURN
170:BEEP 10,100,10
    0
180:"A":CLEAR :
    WAIT 0:USING :
    CLS :RANDOM
190:INPUT "ILE ZNA
    KOW TEKSTU ? Z
    =";Z:Z=ABS Z:Z
    =INT Z:IF Z=0
    GOTO 170
200:INPUT "KLUCZ (
    DO 16 ZNAKOW)
    ";K$
210:N=INT (Z/16):
    DIM J$(N):DIM
    T$(N):DIM N$(N
    ):IF LEN K$=0
    GOTO 170
220:FOR K=0TO N
230:PRINT STR$ (K+
    1):CURSOR 2
240:INPUT "CZLON (
    16 ZNAKOW) ";J
    $(K)
250:CLS :WAIT :
    PRINT J$(K)
260:INPUT "POPRAWN
    IE ? T=1 N=0 "
    ;D:IF D=0GOTO
    230
270:WAIT 0:NEXT K
280:O$="
            "
290:Q=0:GOSUB "X"
300:"B":WAIT :
    PRINT "SZYFRAC
    JA"
310:FOR K=0TO N
320:L=K+1:PRINT "T
    EKST";L:PRINT
    J$(K)
330:PRINT "KRYPTGR
    AM";L:PRINT T$
    (K)
340:NEXT K
350:"K":PRINT " K
    L U C Z":PRINT
    K$:END
370:"C":INPUT "KLU
    CZ (DO 16 ZNAK
    OW) ";K$
380:IF LEN K$=0
    THEN BEEP 10,1
    00,100:GOTO 37
    0
390:Q=1:GOSUB "X"
400:"D":WAIT :
    PRINT "DESZYFR
    ACJA"
410:FOR K=0TO N
420:L=K+1:PRINT "K
    RYPTOGRAM";L:
    PRINT T$(K)
430:PRINT "TEKST";
    L:PRINT N$(K)
440:NEXT K:GOTO "K
    "
450:"S":PRINT "KRY
    PTOGRAM"
460:FOR K=0TO N
470:PRINT T$(K)
480:NEXT K:END
500:"G":USING :
    RANDOM :U$="":
    N=0:WAIT 100
510:PRINT "K L U C
     Z E":PRINT "P
    RZYPADKOWE"
520:INPUT "ILE KLU
    CZY ? N=";N:N=
    ABS N:N=INT N
530:FOR L=1TO N
540:FOR K=1TO 16
550:A=34+RND 88
560:U$=U$+CHR$ A
570:NEXT K:PRINT U
    $:U$=""
580:NEXT L:END
```

Treść kryptogramu podaje wydruk. Wewnątrz programu można wyodrębnić bloki:
Etykieta „A" (wiersze od 170 do 290) dotyczy SZYFRACJI,
„B" (300—340) — wyniki szyfracji,
„C" (370—390) — deszyfracja,
„D" (400—440) — wyniki deszyfracji,
„K" (350) wydruk klucza,
„S" (450—480) — wydruk kryptogramu,
„X" (10—160) — dodawanie i odejmowanie znaków,
„G" (500—580) — generacja losowych kluczy.

Nasze propozycje:

1. Zacznijcie od przykładu z wydruku. Jeśli wynik będzie identyczny to
2. opracujcie własny tekst i własny klucz
3. wykonajcie szyfrację (RUN „A") i deszyfrację (RUN „B")
4. zastosujcie zły klucz („wytrych) i spróbujcie deszyfracji... itd.

Uwaga: przy projektowaniu klucza unikać symbolu cudzysłów "", gdyż (mój) komputer go nie toleruje!

Przykład:

```
DESZYFRACJA
_______________
KRYPTOGRAM 1
R*. -|x5#z (PWXU
TEKST 1
 Li(? &@01:czy
KRYPTOGRAM 2
RU)+""%03G7#bouV
TEKST 2
 wy348.Z@YQ^16=
KRYPTOGRAM 3
okNJ?;JLGA?WvYkg
TEKST 3
=2.RQQSWTSY7E 31
 K L U C Z
29 SIERPNIA 1986
```

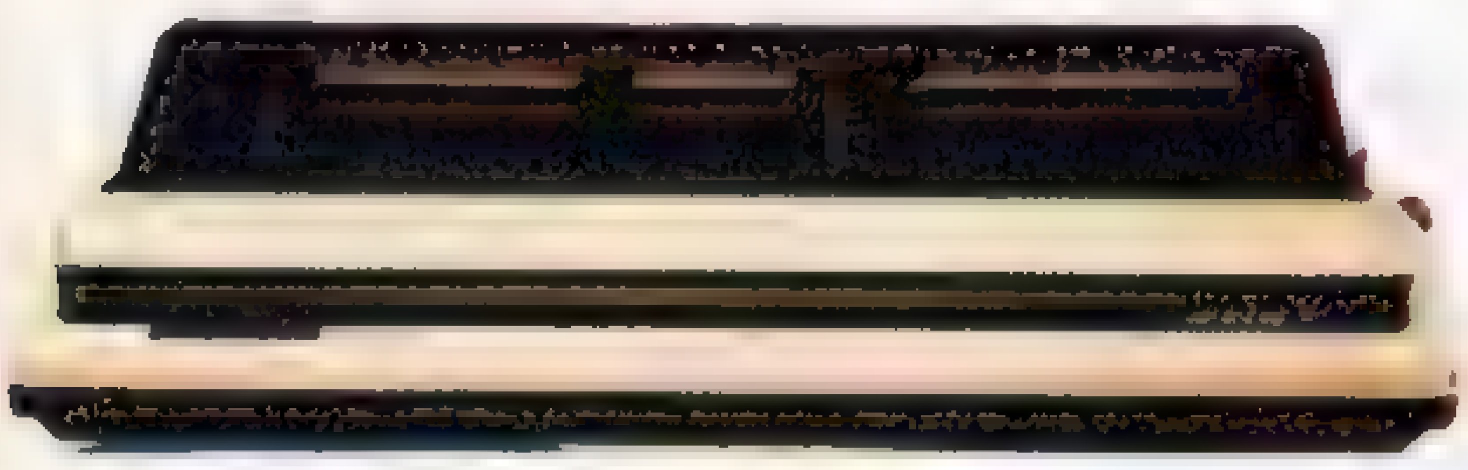




Tej firmy przedstawiać nie trzeba. Oto rodzina urządzeń graficznych produkowanych przez ICL. Najnowocześniejszym wyrobem jest drukarką laserowa — potrafi przez minutę zapełnić drukiem osiem stron formatu A4, zaś ML 193 drukuje 33 znaki na sekundę. Trzecie urządzenie przeznaczone jest do wypisywania druków korespondencyjnych: automatyczne ustawianie papieru, duża staranność i czytelność wydruku. *Thermal printer* to drukarka termiczna — szybkość druku maksymalnie 80 znaków na sekundę (oczywiście znakomita dokładność znaków). I ostatnie urządzenie: plotter kreślący rysunki na papierze formatu A4 i A3.

Ceny? Cóż, nie każdego stać na takie inwestycje.









... na karnawał z komputerem!

Foto: Jan Zelman



Z mikroprocesorem w psychologię

"Zastosowanie mikrokomputerów w psychologii i pedagogice" było tematem konferencji zorganizowanej w Warszawie przez Wojskowy Instytut Medycyny Lotniczej, Instytut Psychologii UAM z Poznania oraz Spółdzielnię Pracy Psychologów z Katowic. Podczas spotkania zaprezentowano rodzime osiągnięcia w zakresie zastosowania mikrokomputerów w psychologii i pedagogice.

Na temat konferencji rozmawiamy z doc. dr. hab. Janem Terelakiem:

— Była to pierwsza konferencja zorganizowana na tak szeroką skalę, obejmująca nie tylko psychologię i pedagogikę, ale również różnorodne systemy eksperymentalne. Za wcześnie jest jeszcze na ogólne podsumowanie. Wypłynęło jednak kilka ważnych problemów, które należy jak najszybciej rozwiązać. Do nich zaliczyłbym sprawę sprzętu i odpowiednich programów.

— *Jak zastosować komputer w psychologii — nauce związanej ze zjawiskami i procesami psychicznymi?*

— Nie liczymy na komputer jako psychologa — może n-ta generacja, za ileś tam lat. Elektronowy „mózg" ma nam pomóc przede wszystkim w przeprowadzeniu obliczeń wielowariantowych, w budowie i doborze właściwych wskaźników, w ich analizowaniu i sprawdzaniu. Komputer znakomicie nadaje się do stawiania różnorodnych diagnoz, prognozowania, może znacznie przyspieszyć opracowanie wyników badań.

— *Czyżby naukowcy już tak bardzo „polubili" bezduszne maszyny, że potrafią....*

—...konferencja wykazała, że stan wiedzy informatycznej wśród psychologów i dydaktyków jest zróżnicowany. Najczęściej jednak sprowadza się do opanowania umiejętności posługiwania się komputerem i programami przygotowanymi przez profesjonalistów.

— *Nie przewiduje Pan, że programy będą pisać również np.: psycholodzy?*

— Już to robią!

— *Dziękuję za rozmowę i życzę dużo satysfakcji z pracy z komputerem.*

Rozmawiał: RYSZARD ROGOŃ

PS Najciekawsze opracowania z konferencji zostaną opublikowane.

opis str. 2

SPECTRUM

```
100 REM "X"
110 LET p$=""
120 FOR k=1 TO n
130 LET b$=k$(1)
140 LET a$=j$(k)
150 IF q=1 THEN LET a$=t$(k)
160 FOR l=1 TO 16
170 LET m=17-l
180 LET x$=a$(m TO )
190 LET a$=a$( TO m-1)
200 LET a$=a$+o$(m TO )
210 LET y$=b$(m TO )
220 LET b$=b$( TO m-1)
230 LET b$=b$+o$(m TO )
240 LET x=CODE x$
250 LET y=CODE y$
260 IF q=1 THEN GOTO 300
270 LET w=x+y: LET t=w-32
280 IF t>=91 THEN LET t=t-91: GOTO 280
290 GOTO 330
300 LET w=x-y: LET t=w-32
310 IF t<0 THEN LET t=t+91: GOTO 310
330 LET w=t+32
340 LET p$=chr$ w+p$
350 NEXT l
360 IF q=1 THEN LET n$(k)=p$: GOTO 380
370 LET t$(k)=p$
380 NEXT k
390 RETURN

1000 REM "A"
1010 BEEP .4,7
1020 CLS: DIM k$(1,16)
1030 INPUT "Ile znaków tekstu? ";z
1040 LET z=INT (ABS z)
1050 IF NOT z THEN GOTO 1000
1060 INPUT "Klucz (do 16 znaków) ";k$(1)
1070 LET n=INT ((z-1)/16)+1
1080 DIM j$(n,16)
1090 DIM t$(n,16)
1100 DIM n$(n,16)
1110 FOR k=1 TO n
1120 INPUT (k);" człon 16 znakowy:";j$(k)
1130 PRINT j$(k)
1140 INPUT "poprawnie ? t/n ";LINE d$
1150 IF d$="n" OR d$="N" THEN GOTO 1120
1160 NEXT k
1170 LET o$="                ": REM 16 spacji
1180 LET q=0: GOSUB 100
1190 REM "B"
1200 CLS: PRINT "SZYFRACJA": PRINT
1210 FOR k=1 TO n
1220 PRINT "tekst ";k
1230 PRINT j$(k)
1240 PRINT "KRYPTOGRAM ";k
1250 PRINT t$(k): PRINT
1260 NEXT k
1270 REM "K"
1280 PRINT "K L U C Z"
1290 PRINT k$(1): STOP

2000 REM "C"
2010 CLS: INPUT "klucz (do 16 znaków) ";k$(1)
2020 LET q=1: GOSUB 100
2030 REM "D"
2040 PRINT "DESZYFRACJA": PRINT
2050 FOR k=1 TO n
2060 PRINT "KRYPTOGRAM ";k
2070 PRINT t$(k)
2080 PRINT "tekst ";k
2090 PRINT n$(k)
2100 NEXT k: GOTO 1270

3000 REM "S"
3010 CLS: PRINT "KRYPTOGRAM": PRINT
3020 FOR k=1 TO n
3030 PRINT t$(k)
3040 NEXT k: STOP

4000 REM "G"
4010 CLS: RANDOMIZE 0
4020 LET u$=""
4030 LET n=0
4040 PRINT "K L U C Z E": PRINT "PRZYPADKOWE": PRINT
4050 INPUT "Ile kluczy? ";n
4060 LET n=INT (ABS n)
4070 FOR l=1 TO n
4080 FOR k=1 TO 16
4090 LET a=34+INT (RND*88)
4100 LET u$=u$+CHR$ a
4110 NEXT k
4120 PRINT u$: LET u$=""
4130 NEXT l: STOP
```

Желаю
больших успехов
молодым
журналистам
молодого IKS
Генерал-майор

Dziękujemy za noworoczne ...

Odpowiedzi do zadań „Ligi Myślących" z 6 nr. „IKS-a"

Na wstępie chcemy wrócić do podanej w 7 nr. „IKS-a" odpowiedzi na zadanie nr 1 z numeru piątego. Otóż większość Czytelników założyła, że do treści zadania wkradł się błąd, a mianowicie w drugim zadaniu zamiast słowa „zużyłby" powinno być „zużył by" i stąd każdy robotnik pracując oddzielnie wykopałby rów odpowiednio w 20 godzin i 30 godzin. Przyjmując, że w zadaniu nie ma błędu otrzymamy odpowiedź, że jeden robotnik kopał rów przez 50 godzin, a drugi przez ok. 15 godz. 48 minut. Obie odpowiedzi zostały uznane za prawidłowe. A teraz przejdźmy do rozwiązań zadań z „szóstki"

Zadanie 1

Wprowadźmy na wstępie następujące oznaczenia
- a — obwód przedniego, a więc mniejszego koła wozu
- A — obwód tylnego koła wozu

Na podstawie treści zadania możemy ułożyć następujący układ równań

Pocztowa giełda

Rozwiązanie krzyżówki z nr. 7 „IKS-a"

Hasło brzmi: „CZYTAJ IKS-a"

Bony pieniężne (1000 zł) wylosowali: Wanda Lipska — Wyszków, Patryk Stokłosa — Warszawa, Artur Turek — Nowa Dęba, Paweł Sroczyński — Sandomierz, Anna Sienkiewicz — Braniewo.

Nagrody książkowe otrzymują: Maciej Makuła — Elbląg, Miron Kłosowski — Gdańsk, Tomasz Polsakiewicz — Elbląg, Arkadiusz Grzesiński — Tczew, Rafał Pasik — Wołomin, Bogdan Trzpit — Rzeszów, Paweł Wojciechowski — Gdynia, Janusz Płonkowski — Przemyśl, Wojciech Kostecki — Gdańsk-Oliwa, Andrzej Pieczonka — Zarzecze.

Najatrakcyjniejsze programy na ZX Spectrum — TANIO. Literatura po polsku na ATARI, SPECTRUM, COMMODORE. Informacja — koperta zwrotna + znaczek, Jarosław Supłacz, ul. Szarych Szeregów 18/20, 09-409 Płock

ATARI — programy kupię, wymienię, sprzedam Marian Fraszczyński 30-611 Kraków ul. J. Nićki 15/34

Sprzedam komputer ATARI 130XE, 2 joysticki, magnetofon oraz programy. Paweł Podgórny, ul. K. Marksa 72/9, 32-100 Proszowice, woj. Kraków

Notowania komputerowe

Giełda		
ZX Spectrum Plus	—	135 tys. zł
ZX Spectrum 48KB	—	80 tys. zł
Drukarka Seikosha GP50S	—	100 tys. zł
ZX Spectrum 128 KB + 2 kasety	—	190 tys. zł
ZX Spectrum 128 KB + magn. + drukarka	—	400 tys. zł
magnetofon	—	28 tys. zł
Atari 800XL + stacja dyskietek 1050	—	250 tys. zł
Atari 800XL + magnetofon	—	150 tys. zł
		130 tys. zł
Atari 800XL + magn. XC11 + joystick + Paddles + Cortridge + 10 kaset z programami	—	185 tys. zł
Timex 2048 + magnetofon + joystick	—	140—150 tys. zł
Commodore 116 + magnetofon + joystick + 10 programów	—	150 tys. zł
Commodore 128 + magnetofon	—	350 tys. zł
Commodore 116	—	65 tys. zł
Commodore Plus 4 + magnetofon	—	140 tys. zł
IBM — Keyboard	—	110 tys. zł
Komis		
Commodore Plus 4 + magnetofon	—	200 tys. zł
Sharp MZ721	—	200 tys. zł
Sharp MZ700	—	170 tys. zł
Dyskietki 5 1/4 cala	—	24 tys. zł
Amstrad CPC 6128 z zielonym monitorem	—	1 200 tys. zł
Studio SYSTEM — Warszawa Wola		
Amstrad PCW 8256 + drukarka	—	1 300 tys. zł
Amstrad CPC 6128 + monitor zielony	—	880 tys. zł
Amstrad CPC 464 + monitor zielony	—	250 tys. zł
Spectrum Plus	—	150 tys. zł
Sharp MZ721	—	180 tys. zł
Commodore 116 + magnetofon	—	91 tys. zł
Joystick Quickshot II	—	8 450 zł
Dyskietki 3.5 cala DS	—	8 760 zł
Dyskietki 3.5 cala SS	—	5 370 zł
Drukarka Seikosha SP-1000PC	—	576 tys. zł
Spectrum 128 KB	—	279 tys. zł
Atari 800XL	—	125 500 zł

$$\frac{720}{a} = \frac{720}{A} + 96$$

$$\frac{720}{a - 0.5} = \frac{720}{A - 0.5} + 128$$

Z rozwiązania powyższego układu równań wynika, że obwód a przedniego koła wozu wynosi 3 m, a obwód A tylnego koła wozu wynosi 5 m.

Zadanie 2

Oznaczamy szukaną liczbę lat ucznia przez x.
Na podstawie treści zadania tworzymy równanie

$$3(x + 3) - 3(x - 3) = x$$

W wyniku rozwiązania równania otrzymamy x = 18, a więc nasz miłośnik łamigłówek ma lat 18

Zadanie 3

Niech x oznacza ilość dni potrzebną do przebycia drogi przez podróżnego maszerującego wolniej
Stąd treść zadania możemy „opisać" następującym równaniem:

$$\frac{520}{x - 3} - 12 = \frac{520}{x}$$

Rozwiązując powyższe równanie otrzymamy: x = 13, a zatem jeden podróżny potrzebuje na przebycie drogi o długości 520 km — 13 dni, a drugi 10 dni

Zadanie 4

Każdą prostą w przestrzeni trójwymiarowej możemy jednoznacznie opisać, podając dwa punkty. Przez dwa dowolne punkty z n-punktów przestrzeni trójwymiarowej możemy poprowadzić tyle prostych ile jest 2-elementowych kombinacji ze zbioru n-elementowego, czyli

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

Podobne rozumowanie przeprowadzamy dla płaszczyzny
Trzy dowolne punkty z n-punktów wyznaczają tyle płaszczyzn ile jest 3-elementowych kombinacji ze zbioru n-elementowego a więc

$$\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$$

Krzyżówka nr 1



POZIOMO: A-1) Pierwiastek chemiczny z rodziny lantanowców, A-7) Degustator oceniający potrawy lub napoje, B-5) Skrót oznaczający komputer analogowy, C-1) Międzynarodowa gwara oparta na francuszczyźnie, C-7) Benzopirol stosowany w przemyśle perfumeryjnym, D-5) Smar maszynowy, E-1) Określa wartość towaru, E-8) Napoleońska wyspa, F-4) Skrót określający moduł dziurkarki taśmy dziurkowanej, F-7) Skrót określający moduł czytnika taśmy dziurkowanej, G-1) Glon, wodorost, G-8) Komputer produkowany przez MERA—ELWRO, H-5) Waluta San Marino, I-1) Rodzina lutników z Cremony, I-7) Niemiecki obóz jeniecki dla oficerów w okresie okupacji, J-5) Oznaczenie sposobu rozwiązania schematowego elementów logicznych lub innych elementów cyfrowych, stanowiących podstawę układów scalonych, używanych przy produkcji krajowych komputerów trzeciej generacji, K-1) W żyłach bogów zamiast krwi, K-7) Numer identyfikujący komórkę pamięci lub zapis na kopercie.

PIONOWO: A-1) Nazwa komputera zbudowanego w Uniwersytecie Cambridge pod kierunkiem M.W. Wilkesa, G-1) Nazwa popularnego komputera, E-2) Opracowany w Polsce język do przetwarzania symboli, A-3) Duchowny żydowski, G-3) Duży budynek, E-4) Skrót nazwy brytyjskiego systemu informatycznego dla dowodzenia marynarką wojenną, A-5) Nazwa metody analizy sieciowej lub nazwa pakietu programów komputerowych, H-5) Jednostka pojemności, D-6) Stała część nazwy modeli komputerów i ich modułów produkowanych w ramach „jednolitego systemu", G-6) Ludolfina, A-7) Stolica nad Nilem, H-7) Przeznaczona dla aktora, E-8) Elektroniczna technika obliczeniowa, A-9) Woźny w zakładzie naukowym, G-9) Między innymi waluta Etiopii, Tajwanu, Malezji, E-10) Gęsty las, A-11) Rulon, zwój, G-11) Czujny strażnik.

HASŁO: C-4, A-7, K-11, D-5, A-4, I-7, K-1, F-5, D-6, C-8, B-7, H-6, A-5, C-7, C-1, I-2, C-10, K-4, A-7, H-9, B-6, G-6, A-2, J-6, B-9, A-11, B-3, E-6, K-3.

Rozwiązanie (tylko hasło) należy przesłać pod adresem redakcji do końca lutego br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI

ICL ONE PER DESK



Oto kilka szczegółów: keyboard — wymiary: 95 × 440 × 250 oczywiście mm, waga 3 kg. Urządzenie może pracować w temperaturze od 10 do 45°C; pamięć RAM 128 Kb. W podstawowej konfiguracji może pracować monitor monochromatyczny i kolorowy (ten drugi niestety ciężki — 14 kg). OPD może pracować jako komputer peryferyjny połączony z jednostką centralną łączami telefonicznymi.



Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8417. Nr ind. 361682 K-70

2 1987 (10)

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" ISSN0860-2794 Cena 50 zł



NIE prowadziliśmy dokładnej buchalterii nadchodzącej do redakcji korespondencji. Wiele odpowiedzi na pytania kierowane do nas przez Czytelników znajdowało się, bądź w najbliższym czasie znajdzie się na łamach „IKS-a". Z otrzymywanych listów wnioskować można o stanie naszej mikroinformatyki — jeżeli pod tym pojęciem rozumieć będziemy stan posiadania, ilość i jakość sprzętu mikrokomputerowego oraz poziom wiedzy na temat jego wykorzystania.

Co zatem wiemy? Otóż największą grupą użytkowników są posiadacze poczciwego *Spectrum* i wszystkich jego wersji. Niewiele mniejsza grupa to właściciele *Atari 800 XL*; systematycznie przybywa nowszych modeli *Atari 130 XE*. Trudno jednak powiedzieć, ile komputerów tego typu jest w Polsce. Na nasz apel o przedstawienie się odpowiedziało kilkanaście tysięcy Czytelników — wśród nich najwięcej było użytkowników zdobywającego obecnie polski rynek *Atari*. To im właśnie jeszcze niedawno dokuczał brak oprogramowania — szczególnie w mniejszych miastach, na warszawskich giełdach bowiem, programów na ten właśnie komputer jest pod dostatkiem (tak jest również w Krakowie). Podobna sytuacja panuje na rynku oprogramowania *Spectrum* — ilość egzemplarzy tego komputera również stale rośnie. Oprogramowanie jest, pojawiają się także programy użytkowe, choć raczej jeszcze nieśmiało torują sobie drogę do indywidualnych użytkowników. Od niedawna przybywa *Amstradów*. *Amstrad 464* jest już co prawda przeszłością konstrukcyjną — do nas jednak nadal napływają jego egzemplarze. Znacznie zmniejszyła się cena tego modelu, a w niektórych komisach komputer ten (wyposażony w zielony monitor) można kupić już za 250—300 tysięcy. Spadła również cena *Amstrada 6128* — choć 500—600 tysięcy dla prywatnych właścicieli, a 800 tysięcy do miliona (z zielonym monitorem) dla odbiorców uspołecznionych, to jednak nadal dużo. Niewiele przybywa natomiast *Commodorów*. Nowe *C-128* są nadal drogie i nie wytrzymują konkurencji z innymi komputerami tej klasy. Mamy też nieco sprzętu egzotycznego: *Sharpy*, *Orici*, *Casio* — stosunkowo tanie, ale za to z ubogim oprogramowaniem, szczególnie z niewielką ilością gier.

Posiadamy zatem przede wszystkim sprzęt zagraniczny. Te niewielkie ilości *Meritum* czy *MK* są bowiem mniej niż kroplą w morzu potrzeb — pracują głównie w klubach, choć i tam nikt ich nie wita z entuzjazmem. Nie należy się spodziewać rychłej poprawy sytuacji — produkcja z wrocławskiej ELWRO przeznaczona dla szkół (*Elwro 800 Junior*) nie wypełni półek w Składnicy Harcerskiej, nie zaspokoi nawet komputerowego głodu w szkołach. Na taką sytuację będziemy musieli jeszcze poczekać. Tymczasem komputer klasy *Juniora* już jest na świecie konstrukcją przestarzałą z co najmniej pięcioletnią historią, a to w przypadku mikrokomputera jest już archeologia!!!

Mamy w Polsce jeszcze prototypy *Mazovii* i *Kraka 86* — jest to sprzęt profesjonalny, ale perspektywy masowej produkcji są raczej mgliste — choć z pewnością jego obecność uzdrowiłaby kaleki rynek komputerowy.

Obecna sytuacja sprzyja udanemu wydawaniu dużej ilości pieniędzy, a chcemy przecież udostępnić komputery młodzieży. Nie każdy może jednak wydać co najmniej sto tysięcy, aby rozwijać swoją osobowość. Pozostają jedynie kluby komputerowe, które też potrzebują niemało grosza na rozruch, ceny obowiązujące przy zakupie na rachunek są bowiem u nas znacznie wyższe niż na giełdach, choć sprzęt bywa ten sam! Zakłady pracy również wydają miliony, kilkakrotnie przepłacając prawdziwą wartość komputera — taka jest cena rachunku i frycowe za obowiązujące przepisy finansowe.

Lepiej jest z wiedzą. Niczym grzyby po deszczu rosną domorośli informatycy, przed którymi komputer nie ma tajemnic. I jeśli byłoby u nas można włamać się do jakiegoś komputerowego banku danych, to z pewnością już by się włamali — na szczęście nam to nie grozi.

Grupa dociekliwych jest zdecydowanie większa od tej, w której poszukuje się rozwiązań gotowych, najchętniej poprawnie działających programów. *„Dlaczego drukujecie program Sharpa, a nie Spectrum?"* — pytano mnie w liceum ogólnokształcącym, na południu Polski, w którym informatyka jest już przedmiotem nauczania! Kochani, jeśli jeszcze nikt Wam tego nie powiedział, to pomyślcie sami, ileż to roboty, aby dialekt Basica obowiązujący w *Sharpie* przerobić na *Spectrum*? Przecież algorytm działania jest ten sam, inne jedynie są instrukcje we/wy. Ale jeśli nie wszyscy potrafią, to słownik podstawowych instrukcji Basica różniących wersję obowiązującą na *Amstradzie* i *Sharpie* od tej, którą dysponuje *Spectrum*, podajemy już w tym numerze.

A teraz życzymy przyjemnej lektury. Czekają na Was programy, nauka Basica, a dla tych, którzy lubią pogrzebać się w ROM-ie kolejny odcinek „Sztuk i sztuczek".

WIESŁAW CETERA

Jak to zrobić (4)

Dziś przedstawiam „Edytor znaków" — program pozwalający na przygotowanie ich według własnego życzenia. W czasie pracy na ekranie widoczne jest pole 8 x 8, na którym za pomocą joysticka (powoduje ruch) i przycisku fire (zmienia kolor punktu wskazującego położenie kursora — pozwalając na zamalowywanie lub czyszczenie punktów) możemy utworzyć rysunek własnego znaku. Dodatkowo, na wysokości każdego wiersza rysunku podawana jest liczba, w znany nam sposób kodująca ten wiersz. W trakcie tej twórczej pracy możemy na bieżąco obserwować, jak nasz znak będzie wyglądał w grafice Ø, 1 i 2.

Program ten jest dodatkowo wzbogacony o :

— odczytywanie stanu klawiszy funkcyjnych,

— elementy samomodyfikacji programu.

Dla lepszego zrozumienia tych zagadnień niezbędne są jednak pewne dodatkowe informacje.

I tak komórka o adresie 53279 pozwala nam stwierdzić, jaki klawisz funkcyjny został wciśnięty. Może w niej znajdować się wartość z przedziału Ø — 7 oznaczająca użycie:

Ø — OPTION, SELECT, START
1 — OPTION, SELECT
2 — OPTION, START
3 — OPTION
4 — SELECT, START
5 — SELECT
6 — START
7 — żaden klawisz nie został wciśnięty

W programie wykorzystane mogą być następujące klawisze:
SELECT — w zależności od postaci

punktu wskazującego położenie kursora, zamienia go na biały prostokąt zaciemniający zajęte pole lub pozostawia wolne miejsce (pozwala spokojnie przyjrzeć się utworzonemu znakowi); jest sygnałem do analizy klawiszy funkcyjnych.

SELECT/START naciśnięte jednocześnie ponawiają proces edycji po wciśnięciu SELECT

START — rozpoczyna od początku edycję znaku

OPTION/START naciśnięte jednocześnie kończą działanie programu.

OPTION — w treści programu zapamiętuje dane do utworzenia przygotowanego znaku; przed użyciem tego klawisza należy wiedzieć, jaki znak chcemy przedefiniować.

W ten sposób doszliśmy do dziwnie brzmiącego, a w gruncie rzeczy bardzo prostego do realizacji hasła samomodyfikacji programu. To dziwne zjawisko polega na kontrolowanej ingerencji programu w swoją treść, w rezultacie czego zmienić możemy jego działanie, a nawet otrzymać całkiem nowy program.

Uwagi na ten temat zacznijmy od pokazania... jak to zrobić. Otóż wstawienie do komórki o adresie 842 wartości 13 powoduje ustawienie naszego komputera w tryb czytania z ekranu. Umiejętne wykorzystanie tej sytuacji zasymuluje proste wciśnięcie klawisza BREAK, wprowadzenie zadanych poprawek i następnie instrukcji CONT. Cały pomysł opierać się musi na tym, aby na ekranie wypisać właściwe instrukcje i zmusić komputer do ich realizacji. Spójrzmy zatem, jak wygląda nic nie robiący schemat rozwiązania naszego problemu:

```
10 GRAPHICS 0
20 POSITION 2,4
30 REM WYPISANIE INSTRUKCJI OD
DRUGIEJ KOLUMNY
40 PRINT „CONT"
50 POSITION 2,2 : REM Instrukcja
                  STOP powoduje wypisanie na
                  ekranie dwóch linii, musimy
                  ustawić kursor o te dwie linie
                  wyżej
60 POKE 842,13 : REM Czytanie instrukcji z ekranu
70 STOP
80 POKE 842,12 : REM Powrót do normalnego trybu
                 pracy
```

W programie powyższe instrukcje mają za zadanie umieścić w odpowiednim jego miejscu instrukcje DATA z danymi potrzebnymi do zdefiniowania aktualnie stworzonego znaku.

Czytelnikom, którym prezentowany program może pomóc w pracy proponuję wykorzystanie tej możliwości do przygotowania procedury zmiany znaków. Po zakończeniu działania programu można go zapisać np. na kasecie (LIST "C:") i umieścić w każdym ze swoich programów (ENTER "C:"). Pierwszy krok jest już dokonany, w przypadku użycia klawiszy OPTION/START, do zakończenia pracy programu usuwana jest jedna z jego linii, by po napisie READY otrzymać zadaną procedurę należy wymazać jeszcze kilka linii.

Ale jak to zrobić...?

Grześ

```
10 REM Prosram demonstruje metode
15 REM tworzenia wlasnych znakow
20 REM
25 REM Rownoczesnie pokazany jest
30 REM sposob wykorzystania klawiszy
      funkcyjnych
35 GRAPHICS 0
40 DIM W(8)
45 MEM=PEEK(106)-8:POKE 106,MEM:POKE 7
56,MEM
50 CHADR=256*MEM:POKE 752,2
55 ZN=ASC("!"):ZN=CHADR+(ZN-32)*8
60 REM Kopiowanie zbioru znakow
65 GOSUB 1000:LAB=1100
69 REM
70 REM Zmiany w  Display List
71 REM
75 DL=PEEK(560)+256*PEEK(561)
80 POKE DL+23,6
85 POKE DL+25,7
89 REM
90 REM Zmiana koloru ekranu
91 REM
95 FOR K=0 TO 2:POKE 710+K,144:NEXT K
99 REM
100 REM Tworzenie obrazu na ekranie
101 REM
105 POSITION 5,0:PRINT "E D Y T O R
Znakow"
110 POSITION 5,2:PRINT ""
115 FOR I=3 TO 10
120 POSITION 4,I:PRINT "
  0   "
125 W(I-2)=0:POKE ZN+I-3,0
130 NEXT I
135 POSITION 5,11:PRINT ""
140 X=5:Y=3:C0=32:C1=32:C2=20
149 REM
150 REM Petla slowna
151 REM
160 S=STICK(0):ST=STRIG(0)
165 L=0:I=0
170 IF ST=0 THEN C1=192-C1:C2=168-C2
175 POSITION X,Y:PRINT CHR$(C2)
180 OX=X:OY=Y:OC=C0
185 IF (S>8) AND (S<12) AND (X>5) THEN
 X=X-1:L=1
190 IF (S>4) AND (S<8) AND (X<12) THEN
 X=X+1:L=1
195 IF (S=10 OR S=14 OR S=6) AND (Y>3)
 THEN Y=Y-1:L=1
200 IF (S=13 OR S=9 OR S=5) AND (Y<10)
 THEN Y=Y+1:L=1
205 IF L=0 THEN GOTO 275
210 POSITION OX,OY:PRINT CHR$(C1)
215 IF C0=C1 THEN GOTO 250
220 REM Zmiana znaku
225 IF C1=160 THEN I=1
230 IF C1<>160 THEN I=-1
235 OX=12-OX:OY=OY-2
240 W(OY)=W(OY)+I*2^OX
245 POKE ZN+OY-1,W(OY):POSITION 20,OY+
2:PRINT W(OY);"        "
250 LOCATE X,Y,C0:POSITION X,Y:PRINT C
HR$(C2)
255 REM Wypisanie wzoru znaku
260 POSITION 6,16:PRINT "!  !  !  !  !
265 POSITION 3,18:PRINT "! ! ! !"
270 POSITION 23,19:PRINT "! ! ! !"
275 IF PEEK(53279)=6 THEN GOTO 105
279 REM Analiza stanu klawiszy
        funkcyjnych
280 POSITION X,Y:PRINT CHR$(C1):IF PEE
K(53279)<>5 THEN GOTO 160
285 WY=W(Y-2):WW=2^(12-X):IF (C1=160 A
ND C0<>160) THEN W(Y-2)=WY+WW
290 IF (C1<>160 AND C0=160) THEN W(Y-2
)=WY-WW:POSITION X,Y:PRINT CHR$(C1)
295 POKE ZN+Y-3,W(Y-2):POSITION 20,Y:P
RINT W(Y-2);"    "
300 REM Ogladanie znaku
305 IF PEEK(53279)<=3 THEN GOTO 350
310 IF PEEK(53279)<>4 THEN GOTO 305
```

dokonczenie na str. 23

Z dużym zainteresowaniem przeczytałem w BAJTKU refleksje na temat „Staruszki ODRY" (S. Polak, R. Wojciechowski, Nie tylko małe, BAJTEK 10/86.) Jestem jednym z wielu, którzy poznawali informatykę, dzięki tej, wówczas bardzo nowoczesnej, maszynie. Mimo upływu lat jeszcze dzisiaj komputery z ELWRO (obecnie MERAELWRO) dostarczają podstawowych mocy obliczeniowych do profesjonalnych zastosowań komputerów w kraju.



ICL 1900... to już historia

Szkoda tylko, że, w bajtkowych refleksjach „zabrano": przetwornicy koło zamachowe, każdemu ze sterowników (kontrolerów lub po prostu PDSów) po dwa napędy dyskowe, a dyskom trochę (300%) pamięci (ostatnio nie 8, a 32 Mega wynosi pojemność każdej stacji)

Najważniejsze jest jednak, że w myśl zasady: „szanujmy wspomnienia, panowie" — temat zasłużonego komputera rodzimej produkcji znalazł trochę miejsca w naszej bardzo poczytnej prasie informatycznej.

ODRA jest w pełni kompatybilna z ICL 1900. Ale... to już niemal historia. Co słychać teraz w firmie ICL?

Okazuje się, że proponowany przez ICL sprzęt i oprogramowanie są nowoczesne i konkurencyjne wobec innych firm zachodnich, także IBMa. Przykładem jest nowa generacja systemów komputerowych „SERIA 39". Są to dwie całkowicie nowe maszyny: LEVEL 80 i LEVEL 30 — pierwszy owoc współpracy ICL z FUJITSU.

Komputery systemu LEVEL 80 wykonują ponad 13 milionów operacji na sekundę przy pojemności pamięci RAM od 16 do 32 Mbajtów. Natomiast LEVEL 30 jest nieco skromniejszy i jest przeznaczony do typowych biur. Komputer ten wykonuje dwa miliony operacji na sekundę, a jego pamięć RAM wynosi od 4 do 16 Mega. Imponującym uzupełnieniem pamięci RAM tych komputerów są dyski o pojemności od 1000 do 10 tysięcy Mega!

Rewelacją w skali światowej są proponowane od ponad roku przez ICL sieci typu MACROLAN. Połączenia sieci — to kable zbudowane ze światłowodów. Przekazują one informacje z ponad 1250 stron maszynopisu w czasie jednej sekundy, to jest ok. 50 Megabitów na sekundę. Technikę tę wykorzystano do sprzężenia szybkich urządzeń peryferyjnych, takich jak na przykład dyski, a także do łączenia zestawów komputerów w duże systemy — sieci komputerowe. Odległość między sprzęganymi elementami nie powinna przekraczać 1500 m.

Kabel światłowodu tworzą dwa włókna: jedno do transmisji informacji, drugie do sterowania transmisją. Łączna średnica wynosi zaledwie 5 mm. Dołączane są one do urządzeń za pośrednictwem przystawki wielkości teczki (Port Switch Unit). Producent radzi, by „teczkę" tę wieszać na ścianie. Łączy ona sześć linii i może wchodzić w skład systemu do 15 stacji MARCOLANa.

Nieco tańszym rozwiązaniem jest OSLAN (Open Systems Local Area Network). Łączy terminale, drukarki, i inne urządzenia komputerowych zestawów. Wykorzystano tu tradycyjny pośrednik przesyłania informacji — kabel koncentryczny. Kabel wewnętrznej sieci OSLANa ma średnicę 10 mm, odcinki po 500 m, co 1500 m wzmacniacz. Nie byłoby w tym nic nadzwyczajnego, gdyby nie wartości parametrów tej sieci — szybkość transmisji wynosząca 10 Megabitów/sek. Jest to ekwiwalentem około 1000 szybkich linii transmisji lub inaczej równoznaczne z przesyłaniem 250 stron A4 w czasie jednej sekundy. OSLAN może „jednocześnie obsłużyć" 250 szybkich drukarek.

Obecnie system OSLAN stosowany jest do łączenia komputerów Serii 39 z DRS 20, System 25, PERQ i SERIA 2900, a także sprzęgania z siecią telefoniczną całego świata.

Nowoczesne komputery firmy ICL wyposażone są w szybkie kontrolery dysków. Ostatnio dostępne są napędy FDS 300 z dyskami o pojemności 337 Mega i szybkości transmisji 2,46 Mb/s oraz FDS 2500 o pojemności 2521 Mbajty i szybkości transmisji 2,86 Mb/s. Zatem każdy kontroler łączący cztery napędy FDS 2500 stanowi dla komputera pamięć dyskową o pojemności ponad 10 GIGA!

Prezentowane parametry, bardzo atrakcyjne parametry, „wnuków, może prawnuków, stryjecznych" ODRY zachęcają do stosowania ich w wielu sytuacjach, gdzie szybkość przetwarzania dużych zbiorów informacji jest celem nadrzędnym. Wiąże się to zazwyczaj z koniecznością zapewnienia bezawaryjnej pracy (elektrownie, fabryki, koleje itp.) maszyn. Osiąga się to przez systemowe dublowanie komputerów lub taniej (w nieco ograniczonym zakresie) przez tak zwany system wczesnego ostrzegania.

Komputery Serii 39 mogą być połączone za pośrednictwem zwykłej telefonicznej linii z centrum diagnozy i kontroli — ICL Support Centres. Linia ta pośredniczy w bezustannej kontroli poprawności pracy sprzętu i oprogramowania komputera bez udziału ludzi. Diagnoza zestawu komputera dotyczy wszystkich elementów: pamięci taśmowych, dyskowych, RAMowskich, logiki, działania systemu operacyjnego itp. W razie, gdy kontrola stanu lub działania komputera wypadnie niepomyślnie, według oceny komputera centralnego zainstalowanego w Support Centres, natychmiast powiadamiany jest o tym użytkownik ze wskazaniem miejsca ewentualnej lub zaistniałej awarii.

Oferta programowa ICLa jest nie mniej atrakcyjna niż sprzęt (bazy danych, edytory, języki itp.). Także UNIX uznawany jest jako przyszłościowy system operacyjny potomków naszej ODRY.

WŁODZIMIERZ GOGOŁEK



PROGRAM 5 COMMODORE

Oprogramowanie dydaktyczne nie nadąża za rozwojem oprogramowania użytkowego, a w porównaniu z tak popularnymi grami to praktycznie margines zastosowań.

Przedstawiony program jest przykładem wykorzystania mikrokomputera właśnie w dydaktyce, zarówno przez uczących się, jak i wykładowców. Jest testem z wyborem wariantu odpowiedzi. Nie dotyczy konkretnego, tematycznego zastosowania. Własne pytania, warianty odpowiedzi, a także dane do oceny (prawidłowe odpowiedzi, liczba dopuszczalnych błędów) koduje użytkownik. Ilość pytań zależy tylko od pojemności pamięci komputera.

Zakodowane pytania zostają zapisane na dyskietce, w zbiorze DANE. Z jednolitą nazwą zbioru zapisaną w wierszach 2010 i 3020 poradzi sobie każdy. Można używać różnych dyskietek lub rozbudować program, wprowadzając alternatywne nazwy zbioru. Można także składować kolejne wersje programu ze zmienioną nazwą (zarówno programu, jak i zbioru danych).

Po wykonaniu tych operacji TEST gotowy jest do zapisu kolejnych pytań (tematów).

Program zapewnia wykonanie następujących operacji;

Z — zapisanie przygotowanych pytań i wariantów odpowiedzi (wiersze 300-500), prawidłowych odpowiedzi dla każdego pytania (510-560) oraz dopuszczalnej liczby błędów dla uzyskania ocen: b.db., db., dst. (600-790). Zatrzymanie programu w wierszu 700 umożliwia przyłączenie dysków. Po wciśnięciu dowolnego klawisza następuje zapis danych (pytań) na dyskietce.

T — realizację testu. Wyprowadzone są kolejne pytania wraz z wariantami możliwych odpowiedzi (800-920). Należy każdorazowo wcisnąć klawisz numeryczny odpowiadając przyjętej (za prawidłową) odpowiedzi. Za każdą prawidłową odpowiedź uzyskujemy jeden punkt.

Wyprowadzenie ilości poprawnych i błędnych odpowiedzi oraz uzyskanej oceny kończy test.

K — korekta zapisów. Wyprowadzane są kolejno wszystkie wprowadzone na etapie „Z" dane. Jeżeli wiersz zapisany jest prawidłowo należy wcisnąć klawisz RETURN, gdy nie — wpisać prawidłowy tekst.

Każde uruchomienie programu wymusza podanie ilości pytań i wariantów odpowiedzi, a operacje „T" i „K" poprzedzane są zawsze wczytaniem zbioru danych z dyskietki.

Program może być wykorzystany w praktyce, ale powstał w zasadzie w celu pokazania idei (jednego ze sposobów) wykorzystania mikrokomputera w dydaktyce. Stąd proste (łatwość adaptacji na innych komputerach) instrukcje oraz sprowadzone do minimum procedury kontroli i korekty danych.

Nowy program można rozbudować o nowe elementy, takie jak: wprowadzenie nazwiska, wyprowadzenie wyników na drukarkę, różną ilość punktów za poszczególne odpowiedzi (w tym punkty ujemne za błędne odpowiedzi).

KRZYSZTOF ŚWIĘTOŃ

```
10 PRINT"♥"
20 PRINT:PRINT:PRINT:PRINT:PRINT"      T E S T"
30 PRINT:PRINT:PRINT:PRINT:PRINT
40 INPUT "PODAJ ILOSC PYTAN    ";A
50 INPUT "PODAJ ILOSC WARIANTOW";B
60 PRINT"♥"
100 PRINT:PRINT:PRINT:PRINT"   K O D Y   P R A C Y"
110 PRINT"   -------------------":PRINT:PRINT:PRINT
120 PRINT"Z  - ZAPIS PYTAN":PRINT:PRINT
130 PRINT"T  - TEST":PRINT:PRINT
140 PRINT"K  - KOREKTA":PRINT:PRINT
150 INPUT "PISZ KOD";A$
200 IF A$="Z" THEN 300
210 IF A$="T" THEN 800
220 IF A$="K" THEN 1300
230 PRINT:PRINT:PRINT:PRINT"BLEDNY KOD PRACY"
240 INPUT A$
250 GOTO 60
300 DIM Z$(A*(B+3)+3)
310 I=0
320 FOR K=1 TO A
330 PRINT"♥":PRINT:PRINT"PYTANIE NR - ";K
340 PRINT:PRINT:PRINT:INPUT Z$(I)
350 FOR J=1 TO B
360 I=I+1
370 PRINT:PRINT:PRINT"WARIANT ODPOWIEDZI NR - ";J
380 INPUT Z$(I)
390 NEXT J
400 I=I+1
410 NEXT K
490 J=1
500 FOR I=A*(B+1) TO A*(B+2)-1
510 PRINT"♥"
520 PRINT:PRINT:PRINT"WPROWADZ PRAWIDLOWE ODPOWIEDZI"
530 PRINT:PRINT J;" - "
540 INPUT Z$(I)
550 J=J+1
560 NEXT I
600 PRINT"♥" PRINT PRINT PRINT"DOPUSZCZALNA ILOSC BLEDOW DLA OCEN
610 PRINT PRINT
620 INPUT "B. DB. - ";Z$(A*(B+3))
630 PRINT PRINT
640 INPUT "DOBRY  - ";Z$(A*(B+3)+1)
650 PRINT PRINT
660 INPUT "DOST.  - ";Z$(A*(B+3)+2)
700 PRINT"♥" PRINT PRINT
710 INPUT"ZAPISZ DANE";B$
720 GOSUB 2000
730 PRINT PRINT PRINT PRINT PRINT
790 STOP
800 I=0 GOSUB 3000
810 L=A*(B+2)
820 FOR K=1 TO A
830 PRINT"♥" PRINT PRINT PRINTK;" - ";Z$(I)
850 FOR J=1 TO B
860 I=I+1
870 PRINT PRINT PRINT"   ";J;" - ";Z$(I)
880 NEXT J
890 I=I+1 PRINT PRINT
900 INPUT "ODPOWIEDZ ";Z$(L)
910 L=L+1
920 NEXT K
940 S=0 K=A*(B+2)
950 FOR I=A*(B+1) TO A*(B+2)-1
960 IF Z$(K)<>Z$(I) THEN 980
970 S=S+1
980 K=K+1
990 NEXT I
1000 PRINT"♥" Z=A-S
1010 PRINT PRINT PRINT"ILOSC PYTAN          - ";A
1020 PRINT PRINT PRINT"POPRAWNYCH ODPOWIEDZI - ";S
1030 PRINT PRINT PRINT"BLEDNYCH ODPOWIEDZI   - ";Z
1040 PRINT PRINT PRINT"---------------------"
1100 T=VAL(Z$(A*(B+3)))+1
1110 U=VAL(Z$(A*(B+3)+1))+1
1120 V=VAL(Z$(A*(B+3)+2))+1
1130 S$="NDST.  (2)"
1150 IF Z<V THEN S$="DST.   (3)"
1160 IF Z<U THEN S$="DOBRY  (4)"
1170 IF Z<T THEN S$="B. DB. (5)"
1200 PRINT PRINT PRINT"   O C E N A        - ";S$
1210 PRINT PRINT PRINT PRINT PRINT
1290 STOP
1300 GOSUB [illegible]
1310 I=0 [illegible]
1320 PRINT"♥ PRINT PRINT PYTANIA"
1330 FOR I=1 TO [illegible]
1335 PRINT PRINT PRINT PRINTI;"  ";Z$(I)
1340 B$="" INPUT B$ IF B$="" THEN 13[illegible]
[illegible]
```

dokończenie na str. 6

dokończenie ze str. 5

```
1360 I=I+1:L=L+1
1370 IF K>A THEN1500
1380 FOR J=1TOB
1390 PRINT"    ";J;".= ";Z$(I)
1400 B$="":INPUT B$
1410 IF B$="" THEN 1430
1420 Z$(I)=B$
1430 I=I+1
1440 NEXT J
1500 IF K=A THEN GOSUB 1800
1520 IF K=2*A THEN GOSUB 1900
1530 NEXT K
1540 GOSUB2000
1550 PRINT:PRINT:PRINT:STOP
1800 L=1
1810 PRINT"♥":PRINT:PRINT:PRINT"ODPOWIEDZI"
1820 RETURN
1900 L=1
1910 PRINT"♥":PRINT:PRINT:PRINT"DOPUSZCZALNA ILOSC BLEDOW"
1920 PRINT"DLA OCEN : B,DB. ,DB. ,DST.":PRINT:PRINT
1930 RETURN
2000 OPEN15,8,15
2010 OPEN3,8,3,"@0:DANE,S,W"
2020 FOR K=0 TO A*(B+3)+3
2030 PRINT#3,Z$(K)
2040 NEXT K
2050 A$="****"
2060 PRINT#3,A$,CHR$(13);
2070 CLOSE 3:CLOSE15
2080 RETURN
2420 Z$(I)=B$
3000 OPEN 15,8,15
3010 DIMZ$(A*(B+3)+3)
3020 OPEN3,8,3,"0:DANE,S,R"
3030 FOR K=0 TO A*(B+3)+3
3040 INPUT#3,A$
3045 Z$(K)=A$
3050 IF A$="****" THEN 3070
3060 NEXT K
3070 CLOSE3:CLOSE15
3080 RETURN
```

UWAGA! ATARI

Komputerowy teleexpress

Zapewne nieraz widzieliście efektowny napis świetlny biegnący wzdłuż telewizyjnego ekranu. Bardzo często efekt ten wykorzystuje się w celach reklamowych. Może on mieć również zastosowanie w redagowaniu gazety świetlnej lub też możecie go wykorzystać np. do przesłania bardzo atrakcyjnych edytorsko życzeń koleżance lub koledze, którzy dysponują minikomputerem „Atari". Program ten jest nadzwyczaj prosty i łatwo można go rozbudować. A więc, przystępujemy do zredagowania naszej pierwszej telegazety. A może... na naszym domowym ekranie wyświetlimy komputerowe życzenia noworoczne?... Spróbujcie!

```
10 DIM TEKST$ (200)
20 GRAPHICS 2
30 PRINT "PROSZĘ WPROWADZIĆ TEKST"
40 DIM SP$ (200)
50 INPUT TEKST$
60 SP$= " "
70 SP$ (LEN (SP$) + 1) = TEKST$
80 TEKST$ = SP$
90 TEKST$ (LEN (TEKST$) + 1) = " "
100 ML=LEN (TEKST$)
110 POKE 752,1
120 PRINT " ) "
130 SETCOLOR 2,0,0
140 FOR P=1 TO ML
150 El=P+19:IF EL>ML THEN EL=ML
160 POSITION 0,5
170 PRINT ≠ 6; TEKST$ (P, EL)
180 FOR W=1 TO 99:NEXT W
190 NEXT P
200 GOTO 140
```

„IKS UCZY": Projektujemy stronę tytułową programu

Każdy z Was zdążył już zebrać sporo programów na „Atari" — chociażby ułożonych na podstawie iksowych listingów. Warto więc zadbać i o to, by miały one ładną stronę tytułową. Prezentowany tu krótki program jest jedynie wstępem do tej pożytecznej zabawy. Można go wzbogacić np. o miłe dla ucha efekty dźwiękowe, czy też swój własny piktogram lub hasło; datę; można wpisać jakąś dedykację itp. Przyjemnej zabawy!



Atari 260ST

```
10 GRAPHICS 2
20 SETCOLOR 2,0,0
30 POSITION 6,4
40 PRINT ≠ 6; „IKS UCZY"
50 FLIP=224
60 PRINT „NACIŚNIJ START"
70 SETCOLOR 0,0,0
80 FOR WAIT=1 TO 500: NEXT WAIT
90 SETCOLOR 0, HUE,6
100 HUE=HUE+1
110 POKE 756,FLIP:FLIP=450—FLIP
120 FOR WAIT=1 TO 500:NEXT WAIT
130 IF PEEK (53279) <> 6 THEN GOTO 70
140 POKE 756,224
```

„Atari" udaje, że kreśli koła

Użytkownicy „Atari" — szczególnie najmłodsi, nie bardzo wiedzą, jak posługując się wyłącznie instrukcjami „BASIC-u" wykreślić koło. Podsuwamy więc pomysł na wykreślenie kilku kół koncentrycznych względem siebie. Wprawdzie do elegancji daleko im, jednak zawsze to już coś.

```
10 GRAPHICS 7
20 DEG
30 FOR J=1 TO 360
40 COLOR J
50 FOR I=1 TO 360
60 PLOT 80+INT (J · 10 · COS (I)), 40 + INT (J · 10 · SIN (I))
70 NEXT I
80 NEXT J
```

opracowała ANITA FRYSZKIEWICZ

KRESKA

(Basic CPC 464) (1)



Każdy z nas w mniejszym lub większym stopniu potrafi posługiwać się instrukcjami interpretera BASIC rezydującego w komputerze. Istniejące rozkazy interpretera, umożliwiają w praktyce narysowanie każdej płaskiej figury geometrycznej, a także realizację nieskomplikowanej grafiki przestrzennej.

Zamieszczony i opisany niżej program KRESKA pozwala na tworzenie prostych rysunków na ekranie monitora. Jest jednocześnie ilustracją określonych rozwiązań możliwych do zrealizowania przy użyciu języka BASIC. W niniejszym numerze „IKS-a" zamieszczona została podstawowa część tego programu. W kolejnych odcinkach będzie się on rozrastał o procedury realizujące kolejne funkcje. Chciałbym, aby jego końcowa postać była przydatnym narzędziem do tworzenia rysunków. Nie będzie to jednak grafika komputerowa, gdyż nie program będzie jej autorem. W naszym przypadku ekran monitora zastąpi papier i farby, a klawisze kursora lub joystick — pędzel i kredki. Ponieważ w kolejnych artykułach zamieścimy fragmenty programu, które dołączać będziemy do części podstawowej, mamy więc szansę wspólnie tworzyć, przez kilka kolejnych miesięcy, program graficzny. Jego ja·ość będzie zależała od naszych pomysłów i rozwiązań. Przygotowałem dwa pierwsze artykuły. Mam koncepcję odnośnie dalszej rozbudowy programu. Chciałbym jednak, abyśmy wspólnie spróbowali dokończyć „dzieła" — oczekuję Waszych propozycji i uwag.

* * *

Podstawowym parametrem decydującym o jakości widzianego przez nas obrazu jest jego rozdzielczość, czyli maksymalna ilość punktów świetlnych, na jakie jest podzielony ekran. Niezależnie od trybu pracy każdy punkt możemy opisać parą współrzędnych x, y. Pozycję punktu wzdłuż osi Y można określić za pomocą liczb od 0 do 399, przy czym dwie kolejne wartości odpowiadają jednej pozycji na ekranie. Oś X posiada numerację od 0 do 639. Zatem ekran jest podzielony na 640 punktów w poziomie i 400 punktów w pionie. Rozwiązanie takie przyjęto dla lepszego zachowania proporcji obrazu. CPC 464 może wyświetlać znaki w trzech trybach:

MODE 0 — tryb wielobarwny. Mamy do dyspozycji jedynie 20 kolumn i 25 wierszy tekstu oraz 160 x 200 punktów w przypadku grafiki. Możliwe jest jednoczesne wykorzystywanie 16 kolorów z palety 27 barw ponumerowanych do 0 do 26.

MODE 1 — tryb standardowy. Mieści on 40 kolumn i 25 wierszy tekstu, co odpowiada 320 x 200 punktom dla grafiki. Ale do dyspozycji mamy już tylko cztery kolory z 27.

MODE 2 — tryb wysokiej rozdzielczości. Pozwala na równoczesne użycie jedynie dwóch z 27 barw, lecz wynagradza nam to dostępność 80 kolumn i 25 wierszy tekstu, lub 640 x 200 punktów w przypadku obrazów.

Jak widać parametrem związanym ze zmianą trybu pracy jest wielkość punktów kreślonych wzdłuż osi X. Ponieważ numeracja nie zależy od trybu i wynosi, jak już wspomniałem, od 0 do 639, tak więc, jedna pozycja na ekranie w poziomie odpowiada kolejnym trzem, dwóm lub jednej wartości.

BORDER (ramka) to obszar otaczający PAPER (tło). Znaki wyświetlane na ekranie mogą ukazywać się tylko wewnątrz ramki. PEN (pióro) pisze sam znak. INK (atrament) możemy sobie wyobrazić jako zbiorniczek wypełniony atramentem o wybranym kolorze. Relacje między komendami PAPER, PEN, INK są ustawiane standardowo w chwili włączenia komputera. Możemy je zmieniać poprzez użycie instrukcji INK.

O miejscu zapalenia punktu na ekranie decyduje położenie kursora. Zazwyczaj w programach wykorzystujemy dwa kursory: znaku i graficzny. Różnią się one przeznaczeniem oraz wielkością — działają w sposób rozłączny. Kursor znaku ma wielkość matrycy znaku i dopóki nie użyjemy rozkazu LOCATE zaczyna on ruch od lewego górnego rogu ekranu co odpowiada współrzędnym (x,y)=(1,1), gdzie x oznacza numer kolejny, a y numer wiersza. Wielkość maksymalna współrzędnej x determinowana jest przez tryb pracy (MODE) i przybiera wartości 20,40 lub 80. W przeciwieństwie do kursora znaku współrzędne kursora graficznego odnoszą się do lewego dolnego rogu ekranu (x,y)=(0,0). Współrzędne opisujące położenie tego kursora nie zależą od trybu. Kursor graficzny nie jest wyświetlany na ekranie. Do pokazania jego położenia może być użyty rozkaz PLOT.

Pewne instrukcje języka BASIC wprowadzające wyniki na określone urządzenie wyjścia używają pojęcia stream — strumień, przy czym stream opisywany jest cyfrą od 0 do 9. Pojęcie stream określa, jakie urządzenie wyjścia powinno być wykorzystane. Znak # to znacznik strumienia. Skierowanie danych do strumieni o numerach od # 0 do # 7 powoduje wyświetlenie ich na ekranie monitora. Strumienie o powyższych numerach są przypisane do całego ekranu lub tzw. okienek (agn. WINDOWS), o ile zostaną one zdefiniowane. Skierowanie danych do strumienia # 8 powoduje ich przesłanie na drukarkę. Strumień # 9 związany jest natomiast z pamięcią kasetową.

Po tych ogólnych uwagach przejdźmy do prezentowanego przykładu.

Ekran w naszym programie został zorganizowany w trybie standardowym (MODE 1). Podzieliłem go na dwie części. Górna, wiersze 1 i 2, w trybie znakowym, spełniają funkcję informacyjną. W lewym górnym rogu (WINDOW # 5) wyświetlane są bieżące współrzędne kursora graficznego. Część środkowa (WINDOW # 2) przeznaczona została do prezentacji funkcji możliwych do zrealizowania w danej chwili (tzw. menu), zaś górny prawy róg (WINDOW # 3) wykorzystałem do ukazania wybranego koloru pióra. Pozostała część ekranu (WINDOW # 1) — wiersze 3 do 25 — służyć nam będzie do tworzenia rysunków. Kreślenia dokonuje się za pomocą klawiszy sterujących ruchem kursora lub joystickiem.

Program KRESKA realizuje trzy podstawowe funkcje:

— kreślenie — opcja wybierana klawiszem z literą K;
— czyszczenie części ekranu przeznaczonej do rysowania — opcja C;
— zakończenie pracy — opcja Z.

dokończenie na str. 8

dokończenie ze str. 7

Wybór opcji K — kreślenie, daje nam następujące możliwości:
— przemieszczanie kursora graficznego bez śladu — opcja P;
— wybór koloru pióra — opcja K;
— rysowanie „punkt po punkcie" opcja W;
— wycieranie niepotrzebnych fragmentów rysunku — opcja G.

Podstawowymi komendami języka BASIC wykorzystanymi w programie KRESKA są:

BORDER <nr INK> — określa kolor ramki

CLS [#<stream>] — wypełnia określony zakres ekranu odpowiednim dla tła kolorem.

DRAW x,y, <nr INK> — rysuje linię od aktualnej pozycji kursora graficznego, (x',y') do nowego punktu (x,y).

INK <nr atr>,<nr> — przypisuje atramentowi o numerze nr atr kolor o numerze nr, jaki zostanie użyty przez rozkazy PEN lub PAPER.

LOCATE [#<stream>],x,y — określa miejsce położenia kursora znakowego na ekranie w okienku określonym numerem stream.

MOVE x,y — przesuwa kursor graficzny do punktu określonego współrzędnymi (x,y).

MOVER a,b — przesuwa kursor graficzny z punktu (x,y) do punktu (x',y') = (x+a, y+b).

PAPER [#<stream>],<nr INK> — ustawia kolor tła w okienku ekranu o numerze # stream, zadeklarowanym komendą WINDOW.

PEN [#<stream>],<nr INK> — określa kolor atramentu pióra, który jest używany do pisania w okienku o numerze # stream.

PLOT x,y, <nr INK> — rysuje punkt o współrzędnych (x,y). Kolor punktu określony jest numerem atramentu.

PLOTR a,b, <nr INK> — rysuje punkt względem bieżącego położenia kursora graficznego (x,y), o współrzędnych (x+a, y+b) kolorem określonym przez numer atramentu.

WINDOW [#<stream>],1,p,g,d — określa rozmiary okienka tekstowego nadając mu numer strumienia — # stream. Parametry l — numer pierwszej — lewej kolumny, p — ostatniej — prawej kolumny, g — numer górnego wiersza i d — dolnego wiersza definiują punkty graniczne okienka zgodnie z zadanym trybem (MODE).

TEST x,y — bada numer atramentu punktu o współrzędnych (x,y).

XPOS YPOS — podają współrzędne odpowiednio x i y kursora graficznego.

Powyższe komendy związane były bezpośrednio z tworzeniem rysunku. W programie używam także innych instrukcji:

CALL adres — w naszym przypadku umożliwia wywołanie podprogramu znajdującego się w pamięci ROM.

INKEY (nr) — funkcja przyjmuje wartość 0, jeśli zostanie wciśnięty na klawiaturze klawisz o numerze nr.

INKEY $ — przyjmuje wartość pierwszego znaku wprowadzonego z klawiatury lub tzw. łańcucha pustego — " ", gdy żaden klawisz nie został wciśnięty.

GOSUB nr linii — skok do podprogramu określonego numerem linii. Koniec podprogramu jest oznaczony instrukcją RETURN. Po zakończeniu jego realizacji następuje powrót do programu głównego, do następnego rozkazu GOSUB.

GOTO nr linii — skok do określonej numerem linii programu.

FOR wsk = a TO b
:
NEXT b — realizuje fragment programu między FOR a NEXT — b-a razy.

IF wyr. log. THEN w1 ELSE w2 — sprawdza czy wyrażenie logiczne jest prawdziwe. Jeżeli tak, to realizowany jest ciąg instrukcji w1 zapisany w tej samej linii programu. Jeżeli jest fałszywe, realizowana jest druga część wyrażenia.

Omówię dalej funkcje realizowane przez poszczególne fragmenty programu KRESKA. Linie od — do zawierają:

50 — 100 zdefiniowanie okienek, określenie kolorów tła i pióra. Wyznaczają punkt startowy kreślenia.

140 — 170 wybór sposobu kreślenia z jednoczesnym początkowym nadaniem wartości parametrom G%, D%,L%, P%,F%. Przyporządkowanie tym parametrom numerów klawiszy kursora i klawisza COPY (odpowiednio: 0,2,8, 1,9), ewentualnie styków joysticka i przycisku FIRE (72,73,74,75,76).

360 linia ta zawiera rozkaz CALL B1A9. Wykonanie tego polecenia powoduje wyzerowanie oraz wystartowanie procedur inicjujących pracę (wynik analogiczny, jak przy równoczesnym naciśnięciu SHIFT, CONTROL i ESC).

420 — 440 wypełnienie fragmentu ekranu przeznaczonego do tworzenia rysunków kolorem tła. Jednocześnie zostaje narysowany punkt o współrzędnych (320, 200). W ten sposób zostaje określone nowe położenie kursora graficznego.

480 — 590 w liniach tych tworzone jest menu związane z wyborem opcji K — kreślenie. Po dokonaniu wyboru sterowanie przenoszone jest do odpowiedniej linii związanej z wybraną opcją. Akceptowane znaki to: M,P,K,G,W,↑,↓.

630 — 700 wybór koloru pióra. Możliwy jest wybór jednego z trzech kolorów (kolor tła jest niedostępny). W wypadku naciśnięcia klawisza kursora następuje zmiana numeru atramentu (nr INK) związanego z piórem (PEN). Zmiana powyższa jest sygnalizowana w górnym prawym rogu ekranu (WINDOW#3).

750 — 820 w liniach tych zawarte zostało sterowanie przesunięciem kursora graficznego (bez kreślenia). Samo przemieszczenie odbywa się przez odwołanie do odpowiednich procedur.

870 — 920 ustalenie numeru atramentu (INK) punktu o współrzędnych (x%,y%) i podstawienie jego wartości pod zmienną p.

930 — 970 wypełnienie punktu (x%, y%) kolorem atramentu (INK) o numerze p.

980—1020 sprawdzenie czy kursor graficzny mieści się w obszarze przeznaczonym do kreślenia. Punkty graniczne tego obszaru to: dół (0,0) i (639,0), góra (0,366) i (639,366). W wypadku, gdy nowe współrzędne wynikające z przesunięcia wychodzą poza obszar dozwolony, punkt jest cofany na granicę okna. Próby wyjścia poza ekran są sygnalizowane dźwiękiem (CHR$ 7).

1070—1120 nadanie nowych wartości współrzędnym (x%,y%) w zależności od naciśnięcia

klawisza kursora lub wychylenia joysticka.

- **1160—1300** kreślenie „punkt po punkcie" - z jednoczesnym sprawdzeniem czy kursor nie opuszcza dozwolonego obszaru.
- **1340—1480** wycieranie fragmentów rysunku. Mechanizm wycierania polega na wypełnianiu obszaru ograniczonego rozmiarami gumki kolorem tła (INK 0).
- **1530—1550** wyświetlanie aktualnych współrzędnych kursora graficznego. Należy podkreślić, że w języku BASIC wszystkie wyprowadzane liczby uzupełniane są spacją kończącą. Stąd np. liczba trzycyfrowa zajmuje cztery pozycje na ekranie. W naszym przypadku wymusza to deklarację okna tekstowego o szerokości czterech znaków (3+1).
- **1600—1610** zapamiętanie numeru atramentu (INK) i bieżących współrzędnych kursora graficznego.

W programie używam następujących zmiennych:

- x%,y% — współrzędne bieżące kursora graficznego,
- xp%,yp% — współrzędne pamiętanego punktu;
- x,y — współrzędne służące do wydruku;
- G% — numer klawisza lub styku joysticka wywołującego ruch kursora graficznego w górę
- D% — j.w w dół;
- L% — j.w. w lewo;
- P% — j.w. w prawo;
- F% — numer klawisza COPY lub przycisku FIRE;
- sk% — informuje, czy punkt przykryty kursorem graficznym został zamalowany;
- pr% — informuje, czy nastąpiła zmiana współrzędnych kursora graficznego;
- p — pamiętany numer atramentu (INK);
- rb% — oznacza stany gumki (0-wyłączona 1-włączona)

```
10 REM####################################
20 REM    PROGRAM GRAFICZNY - KRESKA
30 REM autor: Jan Romanowski        1986
40 REM####################################
50 MODE 1:BORDER 13:INK 0,26:INK 1,0:INK 2,6:INK 3,14:kolor=1:x%=320:y%=200
60 WINDOW #0,1,5,1,2:PAPER #0,2:PEN #0,1:CLS #0:PRINT #0,"X=320";
70 LOCATE #0,1,2:PRINT #0,"Y=200";
80 WINDOW #2,6,6,1,2:PAPER #2,2:PEN #2,1:PRINT #2,CHR$(211)+CHR$(211);
90 WINDOW #2,7,40,1,2:PAPER #2,2:CLS #2:PEN #2,1
100 WINDOW #1,1,40,3,25:PAPER #1,0:CLS #1:PLOT x%,y%,0:x=XPOS:y=YPOS:GOSUB 1530
110 REM####################################
120 REM          MENU GLOWNE
130 REM####################################
140 PRINT #2,"   (K)LAWIATURA CZY (J)OYSTICK."
150 IF INKEY(37)=0 THEN G%=0:D%=2:L%=8:P%=1:F%=9:GOTO 180
160 IF INKEY(45)=0 THEN G%=72:D%=73:L%=74:P%=75:F%=76:GOTO 180
170 GOTO 150
180 CLS #2:PRINT #2," (Z)AKONCZENIE PRACY (K)RESLENIE";
190 LOCATE #2,2,2:PRINT #2,"(C)LS ";
200 REM####################################
210 REM           KRESLENIE
220 REM####################################
230 IF INKEY(37)=0 THEN CLS #2:GOTO 480
240 REM####################################
250 REM    CLS - CZYSZCZENIE EKRANU
260 REM####################################
270 IF INKEY(62)=0 THEN CLS#2:PRINT #2," CLS - CZY JESTES PEWIEN  T/N":GOTO 420
280 REM####################################
290 REM       ZAKONCZENIE PRACY
300 REM####################################
310 IF INKEY(71)=0 THEN CLS #2:PRINT #2," CZY JESTES PEWIEN ?   T/N":GOTO 360
320 GOTO 230
330 REM####################################
340 REM    ZAKONCZENIE PRACY  T/N
350 REM####################################
360 IF INKEY(51)=0 THEN MODE 0:PRINT "K O N I E C":FOR n=1 TO 2000:NEXT n:CALL BIA9
370 IF INKEY(46)=0 THEN GOTO 180
380 GOTO 360
390 REM####################################
400 REM     CZYSZCZENIE EKRANU
410 REM####################################
420 IF INKEY(51)=0 THEN CLS #1:x%=320:y%=200:PLOT x%,y%,0:x=XPOS:y=YPOS:GOSUB 1530:GOTO 180
430 IF INKEY(46)=0 THEN GOTO 180
440 GOTO 420
450 REM####################################
460 REM    KRESLENIE - MENU GLOWNE
470 REM####################################
480 WINDOW #2,7,38,1,2:PRINT #2,"(M)ENU GLOWNE (P)RZEMIESZCZENIE (K)OLOR        LUB ("+CHR$(241)+")" ;
490 WINDOW #3,39,40,1,2:PAPER #3,kolor:CLS #3:PEN #3,1:PRINT #3,CHR$(211)+CHR$(32)+CHR$(211);
500 IF INKEY(38)=0 THEN WINDOW #2,7,40,1,2:GOTO 180
510 IF INKEY(2)=0 THEN GOTO 550
520 IF INKEY(37)=0 THEN GOSUB 630
530 IF INKEY(27)=0 THEN GOSUB 1600:GOTO 750
540 GOTO 500
550 CLS #2:PRINT #2,"(G)UMKA (W)OLNE KRESLENIE LUB("+CHR$(240)+")"
560 IF INKEY(0)=0 THEN GOTO 480
570 IF INKEY(59)=0 THEN GOTO 1160
580 IF INKEY(52)=0 THEN GOSUB 1340
590 GOTO 560
600 REM####################################
610 REM          WYBOR KOLORU
620 REM####################################
630 CLS #2:PRINT #2,"WYBIERAJAC KOLOR UZYWAJ - "+CHR$(241)+"     POWROT DO MENU     - P"
640 i$=INKEY$:IF i$="" THEN 640
650 IF INKEY(2)=0 THEN kolor=kolor+1:SOUND 1,100,10,7,3,5,1
660 IF INKEY(27)=0 THEN GOTO 480
670 IF kolor<4 THEN PAPER #3,kolor:CLS #3:PEN #3,1:PRINT #3,CHR$(211)+CHR$(32)+CHR$(211)+CHR$(32);
680 IF kolor<4 THEN FOR i=1 TO 100:NEXT i:GOTO 640
690 IF kolor>3 THEN kolor=1:PAPER #3,kolor:CLS #3:FOR i=1 TO 100:NEXT i:GOTO 640
700 GOTO 650
710 REM####################################
720 REM     PRZEMIESZCZANIE KURSORA
730             GRAFICZNEGO
740 REM####################################
750 CLS #2:PRINT #2," UZYWAJ :"+CHR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+" LUB JOYSTICK"
760 GOSUB 1600
770 LOCATE #2,1,2:PRINT #2," POWROT DO MENU - P"
780 IF INKEY(27)=0 THEN 480
790 GOSUB 930:GOSUB 1070:GOSUB 870:GOSUB 930:GOSUB 980:GOSUB 870
800 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp%,kol:MOVE x%,y%:sk%=1
810 GOTO 780
820 RETURN
830 REM####################################
840 REM         ANALIZA KURSORA
850 REM           GRAFICZNEGO
860 REM####################################
870 IF TEST(x%,y%)=1 THEN p=1
880 IF TEST(x%,y%)=2 THEN p=2
890 IF TEST(x%,y%)=3 THEN p=3
900 IF TEST(x%,y%)=0 THEN p=0
910 PLOT x%,y%,1
920 RETURN
930 IF p=1 THEN PLOT x%,y%,1
940 IF p=2 THEN PLOT x%,y%,2
950 IF p=3 THEN PLOT x%,y%,3
960 IF p<>1 AND p<>2 AND p<>3 THEN PLOT x%,y%,0
970 RETURN
980 IF y%<1 THEN y%=1:PRINT #3,CHR$(7)
```

dokonczenie na str. 11

Sztuki i sztuczki (4)

Zmienne systemowe ROM Spectrum

EKRAN ZNAKOWY

S-POSN	23688/9
SPOSNL	23690/1
DF-CC	23684/5
DF-CCL	23686/7
SCR-CT	23692
DF-SZ	23659
CHARS	23606/7
K-CUR	23643
MODE	23617

Układ bajtów w obszarze ekranu nie jest sekwencyjny, co najlepiej sprawdzić obserwując pojawianie się obrazka z dowolnej gry. Adresy dostosowane są do wydruku znaków. Każdy standardowy znak ASCII na ekranie składa się z 64 punktów w konfiguracji 8 × 8. Definiuje to ekran znakowy na 24 linie i 32 kolumny. 22 linie zawiera ekran główny, a dwie część edycyjna. Te ustalenia można oczywiście zmienić. Rozszerzenie edytora nastąpi po: POKE 23659, a gdy a > 2, co jednocześnie zmniejszy ekran główny do 24-a linii. Ustalając DF-SZ na zero pozbywamy się edytora, choć jest to pomysł trochę niebezpieczny. Nie można wtedy stosować komend INPUT i SAVE w progamie, wymuszać pytanie scroll? i pieczołowicie omijać BREAK (z drugiej strony to niezła blokada). Do dyspozycji są 24 linie ekranu głównego, ale o ile TAB w instrukcjach z PRINT działa normalnie, to AT nie wolno używać dla linii ostatniej (23). Poniżej przykład wydruku w liniach edycji:

```
10 BORDER 2: PAPER 1:CLS
20 POKE 23659,0: REM brak linii edycji
30 PRINT PAPER 2;AT 22,2;"WYNIK";TAB 10;"PALIWO";
   TAB 20;"CZAS";TAB 4;0;TAB 12;100;TAB 20;"0:00"
40 POKE 23659,2: REM dwie linie edycji
50 GOTO 50
```

Zmienna SCR-CT jest wskaźnikiem przewijania ekranu. Zawiera liczbę o 1 większą, niż liczba linii, o jaką przesunie się obraz ku górze

```
10 POKE 23692,10
20 PRINT AT 10,1;"TEKST"
30 PRINT AT 21,31''
40 IF PEEK 23692=1 THEN STOP
50 PAUSE 4
60 GOTO 30
```

Proszę zauważyć, że przesuwanie zostaje wymuszone w linii 30 przez symulację wydruku poza ekranem, co normalnie spowodowałoby reakcję scroll? Efekt podobny można osiągnąć przez bezpośrednie wywołanie procedur ROM w adresach 3190 lub 3582:

```
10 CLS:PRINT AT 21,0;
20 IF LEN INKEY$ THEN GOTO 20
30 IF NOT LEN INKEY$ THEN GOTO 30
40 IF CODE INKEY$=12 THEN PRINT CHR$ 8;" ";CHR$ 8;:GOTO 60
50 PRINT INKEY$;
60 BEEP .02,-10
70 IF PEEK 23689=3 AND PEEK 23688=1 THEN RANDOMIZE USR 3190:
   POKE 23688,33
80 GOTO 20
```

Przy okazji wykorzystaliśmy zmienną S-POSN. Określa ona pozycję wydruku kolejnego znaku. Oznacza to, że jeśli wydruk był w polu np. (x,y), to następny znak pojawi się w polu (24-PEEK 23689, 33-PEEK 23688). Podobnie odczytujemy pozycję w liniach edycyjnych wykorzystując SPOSNL. Poniższy program „nie lubi" przenosić części wy- [illegible]

```
10 LET a$="Najlepszym sposobem wydruku na ekranie jest
        pojawianie się kolejnych liter od lewej do prawej
        strony z szybkością czytania wzrokiem."
20 LET k=1
30 PRINT a$(k);
40 BEEP .05,20
50 IF NOT (k-LEN a$) THEN STOP
60 LET k=k+1
70 IF PEEK 23688=1 AND a$(k-1)<>" " THEN GOSUB 200
80 GOTO 30
200 IF a$(k)=" " THEN GOTO 250
210 PRINT CHR$ 8;: REM cofacz
220 LET k=k-1
230 IF a$(k)<>" "THEN GOTO 210
240 PRINT TAB 0;
250 LET k=k+1: RETURN
```

Wymuszaliśmy już przesuw ekranu, więc spróbujmy go teraz uniknąć.

```
10 PRINT "mikrokomputer"
20 IF PEEK 23689=2 THEN POKE 23689,3: REM ekran "stoi"
30 GOTO 10
```

Innym sposobem wydruku jest bezpośrednie korzystanie z adresów obszaru ekranu. Zmienne DF-CC i DF-CCL odpowiednio dla ekranu głównego i edycyjnego wskazują adres kolejnego wydruku. Wykorzystujemy to do zrobienia [illegible]

```
10 CLS: PRINT "H";
20 GOSUB 60
30 PRINT "O";
40 STOP
50 DATA 0,0,0,28,4,24,32,60
60 LET adres=PEEK 23684+256*PEEK 23685
70 LET licz=0
80 READ k
90 POKE adres,k
100 LET adres=adres+256
110 LET licz=licz+1
120 IF licz<8 THEN GOTO 80
130 POKE 23684,PEEK 23684+1
140 RETURN
```

Znaki, które ukazują się na ekranie zdefiniowane są w ROM-ie. W zmiennej CHARS znajduje się ich adres początkowy 15616 pomniejszony o 256. Własny zbiór grafik dla znaków o kodach 32..127 można wprowadzić do RAM najlepiej pod adres modulo 256 pamiętając, że każdą grafikę określa 8 bajtów. Przejście odbywa się przez modyfikację CHARS

```
POKE 23606,0
POKE 23607,adres/256-1
```

Oto dwa przykłady wykorzystania CHARS

```
5 LET chars=PEEK 23606+256*PEEK 23607
10 INPUT "znak? ";LINE a$
20 IF a$="" OR LEN a$>1 OR CODE a$<32 THEN GOTO 10
30 LET n=CODE a$
40 FOR a=0 TO 7
50 LET x=PEEK (chars+8*n+a)
60 LET t$=""
70 FOR b=1 TO 8
80 LET c=INT(x/2):LET d=x-2*c
90 LET t$=("$" AND d)+(":" AND NOT d)+p$
100 LET x=c: NEXT b
110 PRINT p$: NEXT a
```

```
10 DIM c(8,8)
20 LET chars=PEEK 23606+256*PEEK 23607
30 INPUT LINE a$
40 LET s=LEN a$: IF s>8 THEN LET s=8
50 FOR r=1 TO s
```

```
 60 LET i=chars+8*CODE a$(r)
 70 FOR t=0 TO 7: LET m=PEEK (i+t)
 80 FOR p=0 TO 7
 90 LET c(t+1,8-p)=m-2*INT(m/2)
100 LET m=INT(m/2)
110 NEXT p:NEXT t
120 FOR u=1 TO 4: LET t$=""
130 FOR z=1 TO 4
140 LET k=c(2*u-1,2*z)+2*c(2*u-1,2*z-1)+4*c(2*u,2*z)+
    8*c(2*u,2*z-1)
150 LET t$=t$+ CHR$(128+k)
160 NEXT z: PRINT AT u,4*r-4;t$
170 NEXT u: NEXT r
```

Bajt 23606 można wykorzystać do innych efektów:

```
 10 DATA "Zmieniając","wartość","bajtu","23606",
"osiągamy","bardziej","spektakularny","sposób",
"wydruku","na","ekranie","$"
 20 RESTORE: PRINT AT 0,0;
 30 READ a$
 40 IF a$="$" THEN GOTO 20
 50 FOR a=7 TO 0 STEP -1
 60 POKE 23606,a
 70 PRINT a$
 80 POKE 23689,PEEK 23689+1
 90 NEXT a: PRINT
100 GOTO 30
```

Szczególnym znakiem drukowanym na ekranie jest kursor edycyjny. Miejsce jego aktualnego pobytu w obszarze między E-LINE, a WORKSP przechowuje zmienna K-CUR, natomiast znak, jaki będzie drukowany zależy od bajtu MODE. Manipulując tą ostatnią zmienną można otrzymać różne rozwiązania graficzne kursora w trakcie egzekucji komendy INPUT, gdyż jest to jedyna sposobność, gdy bufor edycji przedstawiany jest na ekranie. Tu zajmę się tylko wymuszaniem trybów pracy podczas INPUT. Tryb graficzny osiągamy przez:

```
10 POKE 23617,2
20 INPUT LINE i$
30 IF CODE i$>164 THEN GOTO 10
```

Tryb EXTENDED będzie obowiązywał tylko dla pierwszego znaku, co może posłużyć do napisania podręcznego kalkulatora obliczającego wartości zaimplementowanych funkcji:

```
10 POKE 23617,1
20 INPUT LINE i$
30 PRINT VAL i$
```

Po wczytaniu np. BIN 10 otrzymamy 2, a po SIN (PI/2) zobaczymy 1.
Możliwe jest również wymuszenie trybu K:

```
10 PRINT "Która z komend służy do
   niszczenia obszaru zmiennych
   BASIC ?"
20 POKE 23611,128: POKE 23617,158
30 INPUT LINE i$
40 IF CODE i$<>253 THEN GOTO 20
50 PRINT i$(1)'INVERSE 1;"DOBRZE"
```

W następnym numerze o atrybutach ekranu.

Krzysztof MAMCARZ

dokończenie ze str. 9

```
990 IF y%>366 THEN y%=366:PRINT #3,CHR$(
7)
1000 IF x%<1 THEN x%=1:PRINT #3,CHR$(7)
1010 IF x%>639 THEN x%=639:PRINT #3,CHR$
(7)
1020 RETURN
1030 REM**********************************
1040 REM       ZMIANA  WSPOLRZEDNYCH
1050 REM         KURSORA GRAFICZNEGO
1060 REM**********************************
1070 IF INKEY(G%)=0 THEN y%=y%+4:pr%=1
1080 IF INKEY(D%)=0 THEN y%=y%-4:pr%=1
1090 IF INKEY(L%)=0 THEN x%=x%-4:pr%=1
1100 IF INKEY(P%)=0 THEN x%=x%+4:pr%=1
1110 x=XPOS:y=YPOS:GOSUB 1530
1120 RETURN
1130 REM**********************************
1140 REM          WOLNE KRESLENIE
1150 REM**********************************
1160 CLS #2:PRINT #2,"UZYWAJ :"+CHR$(240
)+CHR$(242)+CHR$(243)+CHR$(241)+" LUB JO
YSTICK"
1170 LOCATE #2,1,2:PRINT #2,"POWROT DO M
ENU - P"
1180 IF INKEY(27)=0 THEN GOTO 480
1190 IF INKEY(G%)=0 THEN PLOTR 0,1,kolor
1200 IF INKEY(D%)=0 THEN PLOTR 0,-1,kolo
r
1210 IF INKEY(L%)=0 THEN PLOTR -1,0,kolo
r
1220 IF INKEY(P%)=0 THEN PLOTR 1,0,kolor
1230 x=XPOS:y=YPOS:GOSUB 1530
1240 x%=x:y%=y
1250 IF x<1 THEN MOVER 1,0:PRINT #3,CHR$
(7)
1260 IF x>639 THEN MOVER -1,0:PRINT #3,C
HR$(7)
1270 IF y>366 THEN MOVER 0,-1:PRINT #3,C
HR$(7)
1280 IF y<1 THEN MOVER 0,1:PRINT #3,CHR$
(7)
1290 GOTO 1180
1300 RETURN
1310 REM**********************************
1320 REM             GUMKA
1330 REM**********************************
1340 CLS #2:PRINT #2,"GUMKA - WYLACZANA
 P - POWROT"
1350 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
1360 rb%=0
1370 GOSUB 930:GOSUB 1070:GOSUB 980:GOSU
B 870:GOSUB 930
1380 IF INKEY(27)=0 THEN GOTO 480
1390 IF INKEY(F%)=0 AND rb%=0 THEN rb%=1
:FOR i=1 TO 200:NEXT i:LOCATE #2,9,1:PRI
NT #2,"WLACZONA ":GOTO 1400 ELSE IF INKE
Y(F%)=0 AND rb%=1 THEN rb%=0:LOCATE #2,9
,1:PRINT #2,"WYLACZONA":FOR i=1 TO 200:N
EXT i:GOTO 1400
1400 IF rb%=1 THEN GOTO 1410 ELSE GOSUB
870:GOTO 1350
1410 FOR v%=-2 TO 2
1420 MOVE x%-3,y%+v%:DRAW x%+3,y%+v%,1
1430 NEXT v%
1440 FOR v%=-3 TO 3
1450 MOVE x%-4,y%+v%:DRAW x%+4,y%+v%,0
1460 NEXT v%
1470 GOTO 1370
1480 RETURN
1490 REM**********************************
1500 REM    WYSWIETLANIE WSPOLRZEDNYCH
1510 REM       KURSORA GRAFICZNEGO
1520 REM**********************************
1530 WINDOW #5,3,6,1,2:PAPER #5,2:PEN #5
,1:PRINT #5,USING"###";x
1540 PRINT #5,USING"###";y
1550 RETURN
1560 REM**********************************
1570 REM     CHARAKTERYSTYKA PUNKTU
1580 REM           BIEZACEGO
1590 REM**********************************
1600 xp%=XPOS:yp%=YPOS:kol=TEST(xp%,yp%)
:sk%=0:pr%=0
1610 RETURN
```

W następnym numerze będziemy rozbudowywać prezentowany program o szereg nowych funkcji, w tym m.in.: kreślenie linii, okręgu, czworokąta, składowanie pamięci ekranu na kasecie magnetofonowej, wczytywanie rysunku z kasety do pamięci obrazu.

Janek

Możliwości Commodore 64

Commodore 64 jest jednym z mikrokomputerów, które oferują znaczne możliwości graficzne. Niestety dostarczana użytkownikowi wraz z mikrokomputerem instrukcja obsługi nie omawia wszystkich szczegółów związanych ze sposobem wykorzystania C-64 do tworzenia grafiki komputerowej. Zacznijmy zatem od początku.

Jak wygląda pamięć Commodore 64?

Mikrokomputer posiada dwa rodzaje pamięci półprzewodnikowej: pamięć stałą (ROM) i pamięć o dostępie swobodnym (RAM).

Po włączeniu mikrokomputera C-64 wyświetlany jest komunikat:

64 RAM SYSTEM   38911 BASIC BYTES FREE

Informuje on nas o tym, że system dysponuje pamięcią 64Kbajtów, lecz dla progamu w Basicu dostępne jest tylko 38911 bajtów.

ADRES		OPIS
od	do	
0	1	rejestry mikroprocesora
2	1023	pamięć wykorzystywana przez system operacyjny
1024	2039	pamięć (matryca) obrazu wyświetlanego za pomocą znaków — PAMIĘĆ EKRANU
2040	2047	wskaźniki na bloki pamięci, w których zdefiniowano postacie sprite
2048	40959	pamięć wykorzystywana przez program w Basicu, dane lub w inny sposób; jest to nasza dyspozycyjna pamięć
40960	49151	interpreter języka Basic
49152	53247	dodatkowy obszar pamięci RAM do wykorzystania przez użytkownika
53248	53249	rejestry układu kontroli obrazu (VIC-a)
54272	55295	rejestry syntetyzera dźwięków (SID-a)
55296	56319	pamięć (matryca) kolorów wyświetlanych znaków — PAMIĘĆ KOLORU
56320	57343	rejestry wejścia/wyjścia
57344	65535	procedury systemu operacyjnego (KERNAL)

Rys. 1. Jedna z możliwych (podstawowa) konfiguracji pamięci C-64.

Uwaga: Generator znaków ROM (zbiór standardowych postaci znaków C-64) można uzyskać w obszarze od 53248 do 57343 bajtu po ustawieniu odpowiednich przełączników.

Na rys. 1. przedstawiono jedną z możliwych, najczęściej wykorzystywaną konfigurację pamięci.

Jak powstaje obraz?

Na ekranie monitora widzimy różne obiekty (np. litery, znaki graficzne, sprite) wytwarzane przez mikrokomputer. Każdy obiekt powstaje po zinterpretowaniu określonych obszarów pamięci definiujących jego postać, kolor, położenie na ekranie. Można powiedzieć, że mikrokomputer wyświetla swoją pamięć.

Całą grafiką mikrokomputera C-64 steruje układ scalony zwany układem kontroli obrazu. W dalszej części będziemy posługiwać się jego skrótową nazwą VIC (ang. Video Interface Controller). Przetwarza on określone obszary pamięci mikrokomputera w obiekty oglądane na ekranie. VIC posiada następujące podstawowe tryby pracy:

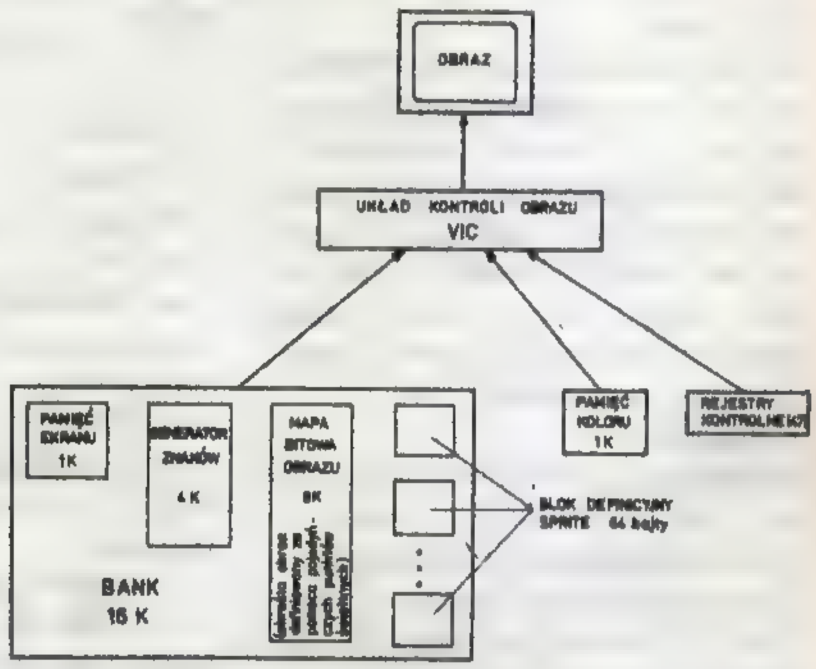
a) tryb znakowy — możliwość wyświetlania znaków w 40 kolumnach i 25 wierszach;

b) tryb graficzny — możliwość wyświetlania obrazów graficznych o rozdzielczości 320 na 200 punktów świetlnych;

c) sprite — możliwość wyświetlania do 8 ruchomych obrazków graficznych (24 na 21 punktów świetlnych każdy).

Wymienione tryby pracy mogą być realizowane w grafice wielobarwnej lub standardowej. Można je również łączyć konstruując bardzo barwne obrazy graficzne.

Ważną własnością VIC jest niezatrudnianie do swojej pracy mikroprocesora. VIC jest sam specjalizowanym procesorem mającym możliwość niezależnej adresacji pamięci. Na formę obrazu oglądanego na ekranie, a wytwarzanego przez VIC wpływa zawartość wielu obszarów pamięci. W schematyczny sposób pokazano to na rys. 2.



Rys. 2 Podstawowe elementy biorące udział przy tworzeniu obrazu graficznego przez C-64.

Do sterowania pracą i komunikowania się z VIC służy 47 rejestrów rozmieszczonych między 53248 i 53294 bajtem pamięci. Ustalając odpowiednie zawartości tych rejestrów możemy wpływać na grafikę uzyskiwaną na ekranie. Niektóre z rejestrów pełnią kilka funkcji. Realizację wybranej funkcji uzyskuje się zapalając odpowiednie bity (ustawiając na wartość 1), bądź też gasząc odpowiednie bity (ustawiając na wartość 0).

Wykonajmy następującą instrukcję w trybie bezpośrednim:

POKE 53265, PEEK(53265)AND 239

W rejestrze zlokalizowanym w 53265 bajcie zgasiliśmy bit 4. Spowodowało to zniknięcie obrazu na ekranie. Cały ekran jest w kolorze obrzeża, a znaki nie są wyświetlane. Przyciśnijmy dowolny klawisz. Na ekranie nie pojawiają się żadne znaki. Naciśnijmy klawisz RETURN, a następnie wprowadźmy kolejną instrukcję (ostrożnie, gdyż wprowadzane znaki są niewidoczne):

POKE 53265, PEEK(53265)OR 16

W tym samym rejestrze zapaliliśmy obecnie bit 4 i otrzymaliśmy normalny obraz na ekranie.

Do tworzenia obrazu wykorzystywany jest również obszar pamięci zwany PAMIĘCIĄ KOLORU zlokalizowany między 55296 i 56319 bajtem. PAMIĘĆ KOLORU określa kolor znaku wyświetlanego w każdej pozycji na ekranie.

Oprócz wymienionych obszarów pamięci (rejestry i PAMIĘĆ KOLORU) VIC wykorzystuje do tworzenia obrazu blok pamięci wielkości 16Kbajtów, zwany BANKIEM. W BANKU VIC posiada formy wszystkich obiektów wyświetlanych na ekranie monitora:

- PAMIĘĆ EKRANU zawierającą kody znaków wyświetlanych w 40 kolumnach i 25 wierszach;
- GENERATOR ZNAKÓW zawierający zbiór form wyświetlanych znaków; każdy kod znaku ma odpowiadającą mu dokładnie jedną formę;
- bloki definiujące postać każdego sprite'a wyświetlanego na ekranie.

Rozmieszczenie tych obiektów wewnątrz BANKU może ulegać zmianie. Sterujemy tym za pomocą rejestrów VIC.

C-64 dysponuje pamięcią 64K. Można wyodrębnić w niej cztery bloki o wielkości 16K (rys. 3). Każdy z wydzielonych bloków może zostać interpretowany przez VIC jako BANK. VIC widzi tylko 16K naraz. Każdy z BANKÓW jest adresowany od 0 do 16383 w procesie sterowania pracą VIC, choć w rzeczywistości mogą to być inne adresy.



Rys. 3. Koncepcja BANKU.

Jak powstają znaki na ekranie?

Każdy znak wyświetlony na ekranie monitora można scharakteryzować następującymi składowymi: forma, kolor i umiejscowienie. Z punktu widzenia mikrokomputera dodatkowym i niezbędnym elementem charakteryzującym znak staje się jego kod. Przyspiesza bowiem realizację wielu operacji na znakach np. porównanie.

Spróbujmy w bardzo ogólnym zarysie prześledzić procedurę tworzenia znaku na ekranie monitora. Naciskając klawisz z dowolnym znakiem powodujemy generację odpowiedniego kodu dla tego znaku. Na podstawie zawartości 648 bajtu pamięci system operacyjny określa rzeczywiste położenie PAMIĘCI EKRANU i przesyła do odpowiedniej pozycji tej pamięci kod znaku. Kolor znaku (kod koloru) jest zapisywany do odpowiedniej pozycji PAMIĘCI KOLORU. Aktualny kolor znaków wprowadzanych jest zawarty w 646 bajcie. Mając te dane VIC pobiera z GENERATORA ZNAKÓW na podstawie kodu znaku jego formę, a z PAMIĘCI KOLORU identyfikuje jego kolor i wyświetla odpowiedni znak w odpowiednim miejscu i kolorze na ekranie monitora.

GENERATOR ZNAKÓW standardowych pochodzących z pamięci ROM jest dostępny dla VIC tylko w BANKU 0 i 2. Znajduje się on między 4096 i 8191 bajtem wg adresacji VIC. Zawartość generatora nie zmienia zawartości pamięci RAM w obu odpowiadających mu rzeczywistych obszarach. Z punktu widzenia VIC pamięć ROM przesłania pamięć RAM.

Jak gospodarować pamięcią?

Nie licząc rejestrów VIC i PAMIĘCI KOLORU wszystkie pozostałe obszary pamięci definiujące formę obrazu są ruchome tzn. można je tworzyć w różnych rejonach pamięci RAM.

Stwarza to duże możliwości graficzne np.:

- możliwość definiowania własnych znaków graficznych;
- możliwość tworzenia animacji;
- możliwość optymalnego wykorzystania posiadanej pamięci.

Dla tych, którzy pragną korzystać z tych możliwości zamieszczamy tabelę, która w prosty sposób pozwala gospodarować wybranymi obszarami pamięci. Za pomocą tej tabeli możemy wybrać BANK, a wewnątrz BANKU określić położenie PAMIĘCI EKRANU i GENERATORA ZNAKÓW. Wykonujemy te czynności ustawiając odpowiednie zawartości trzech bajtów:

- 56576 — najmłodsze dwa bity określają BANK dołączany do VIC;
- 53272 — rejestr VIC-a określający położenie GENERATORA ZNAKÓW (3 najmłodsze bity) i PAMIĘCI EKRANU (4 najstarsze bity);
- 648 — wskaźnik położenia PAMIĘCI EKRANU używany przez system operacyjny; oblicza się go według wzoru.

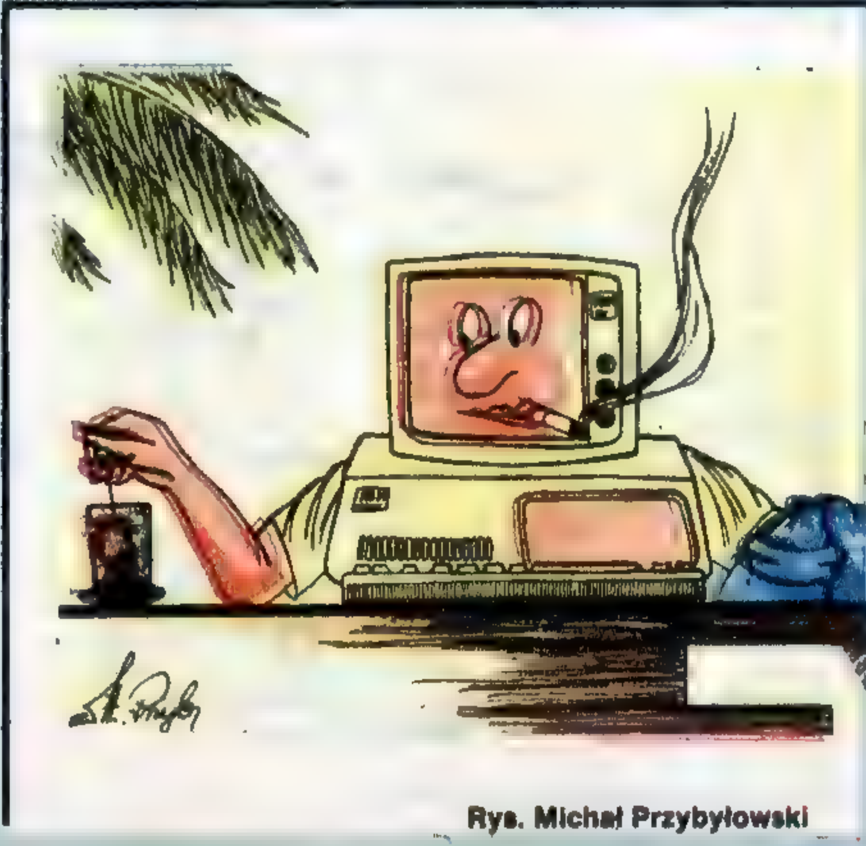
$$\text{wskaźnik} = \frac{\text{adres początkowy PAMIĘCI EKRANU}}{256}$$

Przykłady użycia tabeli:

a) Wykonując sekwencję instrukcji:

POKE 56576,149: POKE 648,140: POKE 53272,53

dokończenie na str. 28



Rys. Michał Przybyłowski

INTERFEJS

Najwygodniejszym sposobem sterowania grami jest zastosowanie joysticka. Najpopularniejszy w naszym kraju mikrokomputer ZX Spectrum nie posiada gniazda, dającego taką możliwość. W tym celu musimy zastosować odpowiedni interfejs.

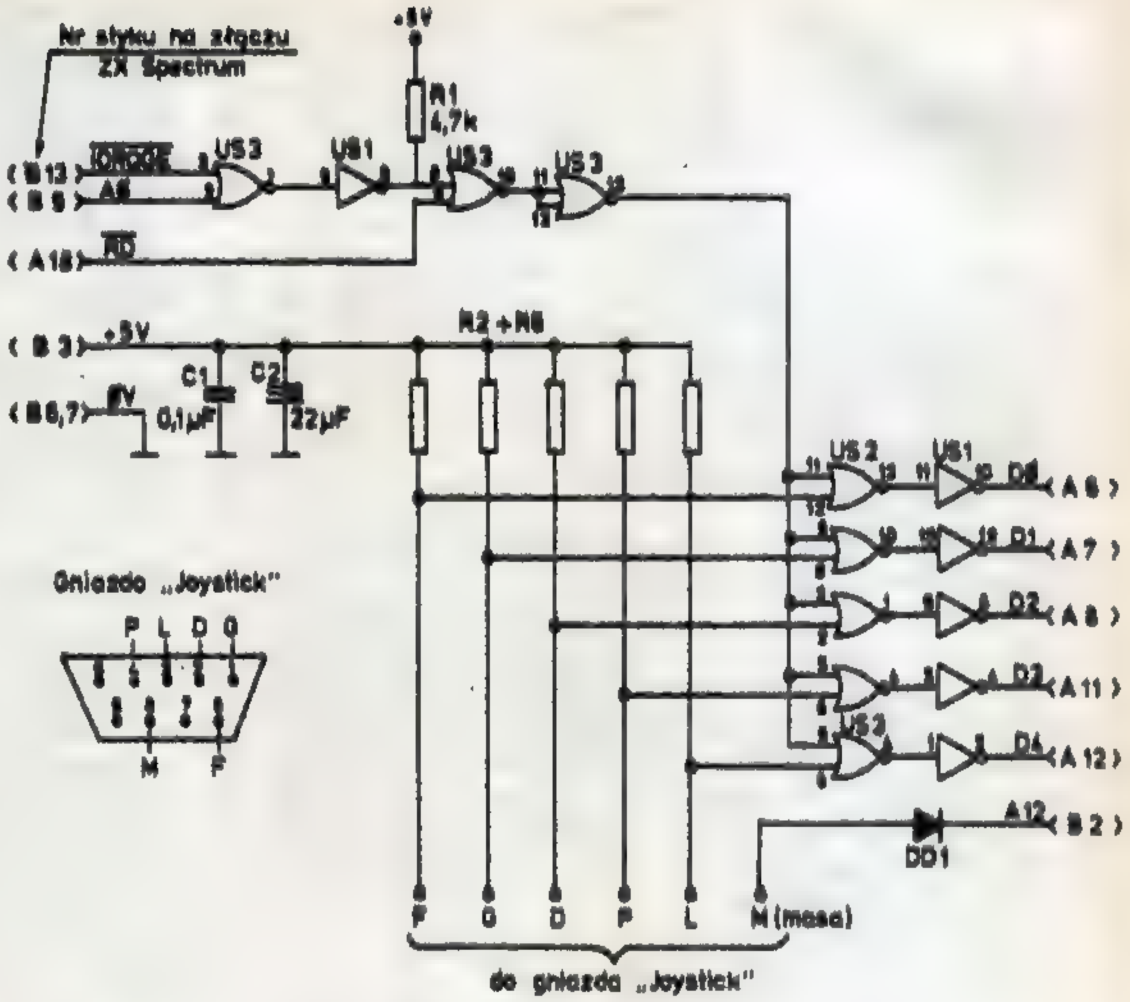
W artykule opisano konstrukcję interfejsu pracującego w standardzie SINCLAIR umożliwiającego dołączenie jednego joysticka. Wybrano ten standard ze względu na możliwość wykorzystania do budowy interfejsu elementów tylko krajowej produkcji.

Schemat ideowy urządzenia przedstawiono na rys. 1. Zastosowanie cyfrowych układów scalonych rodziny TTL-LS pozwala na zasilanie interfejsu i zasilacza +5V mikrokomputera.

Z punktu widzenia oprogramowania odczyt funkcji joysticka jest odczytem stanu klawiszy: 6, 7, 8, 9, 0.

Odczyt stanu manipulatora następuje w momencie wybierania układu ULA do odczytu (sygnały IORQGE, A0, RD mają niski stan) oraz po podaniu na linię adresową A 12 niskiego stanu. Niski poziom napięcia z tej linii adresowej może być podany, poprzez styki joysticka, na jedną z linii F, G, D, P, L. Stan zwarcia (niski stan) podawany jest poprzez bramki NOR i NOT na linie danych D0 ÷ D43. Informacja o funkcji manipulatora pojawia się na odpowiednich bitach tej linii w postaci niskiego stanu. W położeniu neutralnym joysticka wszystkie bity linii danych mają (muszą mieć) wysoki stan.

Osiągnięto to przez zastosowanie rezystorów R2 ÷ R6 utrzymujących wyjście bramek NOR w stanie niskim. Po zanegowaniu przez bramki NOT układu US1, daje nam to wysoki stan na bitach D0 ÷ D4 linii danych mikrokomputera. Układ US1 ma wyjścia z otwartym kolektorem. Dzięki temu układy mikrokomputera mogą ustawić linie danych w stanie niskim. Zapewnia to poprawną pracę komputera



Rys. 1 Schemat interfejsu

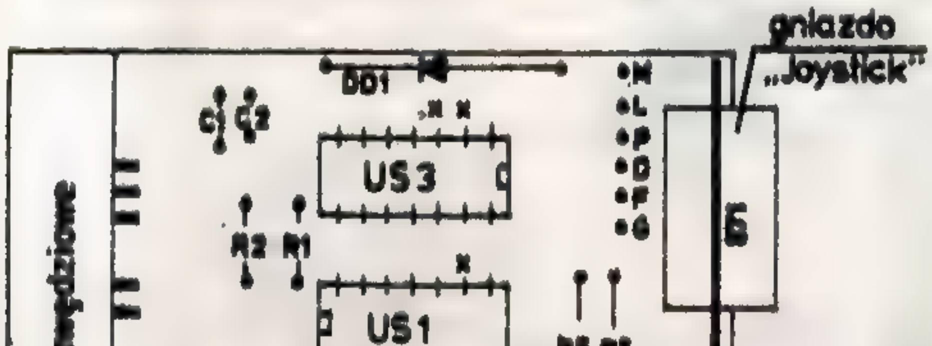
Szerszą informację o wykorzystywanych sygnałach możemy znaleźć w pracach [1] i [2].

Układ zmontowano na dwustronnej płytce drukowanej o wymiarach 62×54 mm. Na rysunku 2 pokazano rozmieszczenie elementów, a na kolejnych rysunkach (3 i 4) widok mozaiki połączeń po obu stronach płytki.

Do połączenia interfejsu ze złączem komputera, wykorzystano gniazdo krawędziowe, a do podłączenia joysticka 9-stykowe gniazdo szufladowe. Oba gniazda umocowano do płytki drukowanej. Gniazdo krawędziowe przylutowano (rys. 5), a gniazdo manipulatora przykręcono za pośrednictwem dwóch wsporników (kątowników).

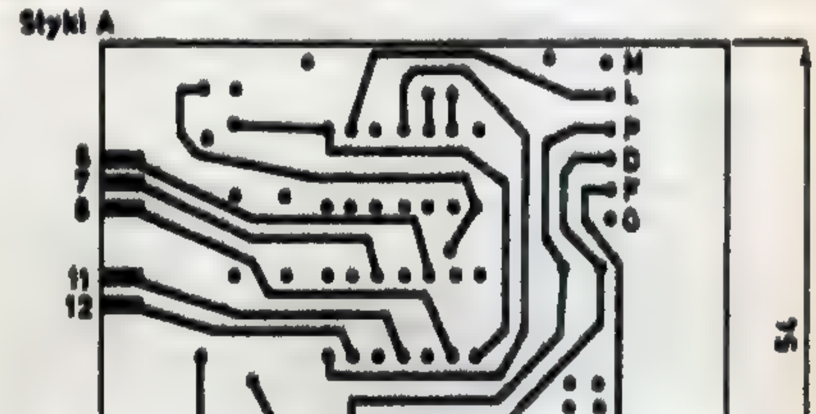
Gniazdo krawędziowe wymaga przed montażem przygotowania. Należy je przyciąć na długość 17 styków. Styki niewykorzystane należy usunąć i w odpowiednim miejscu (rys. 6) umocować wykonaną z izolacyjnego materiału grubości 1,5 mm przegródkę ustalającą.

Po umocowaniu gniazd lutujemy kolejno elementy dyskretne i układy sca-

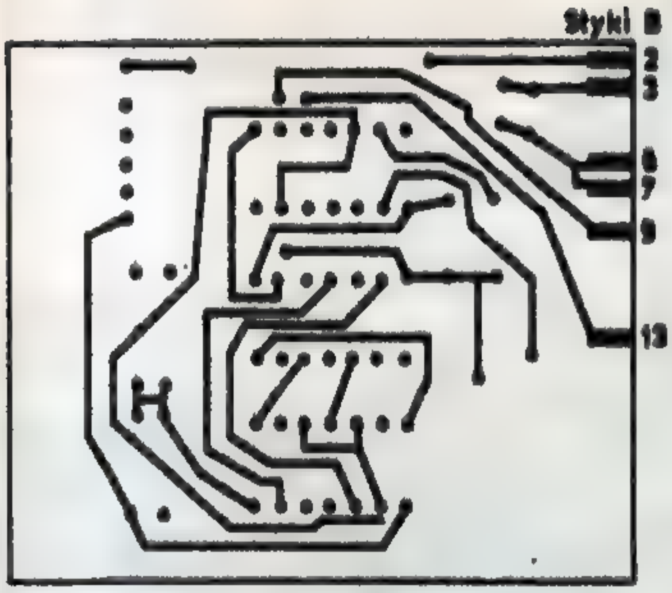


x - punkt lutowniczy (zwora)

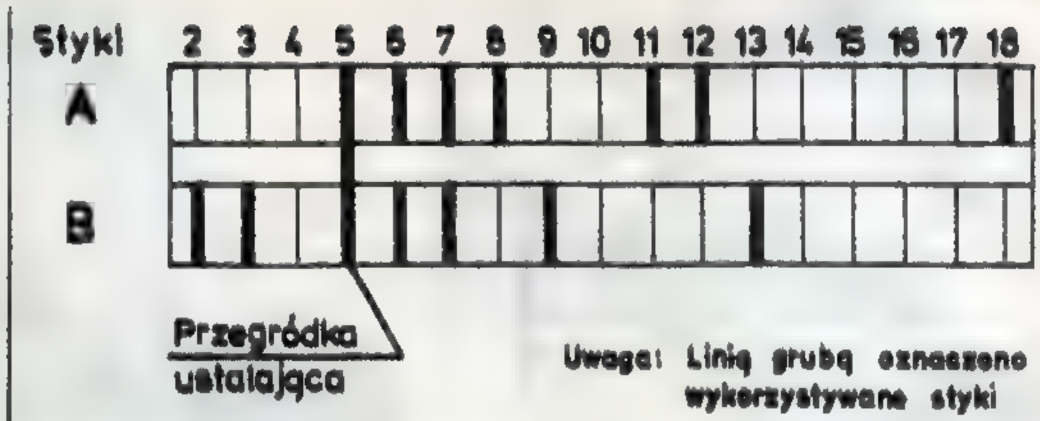
Rys. 2 Rozmieszczenie elementów



Rys. 3 Płytka drukowana — widok od strony części



Rys. 4 Płytka drukowana widok od strony lutowania



Rys. 6. Wykorzystanie styków złącza krawędziowego

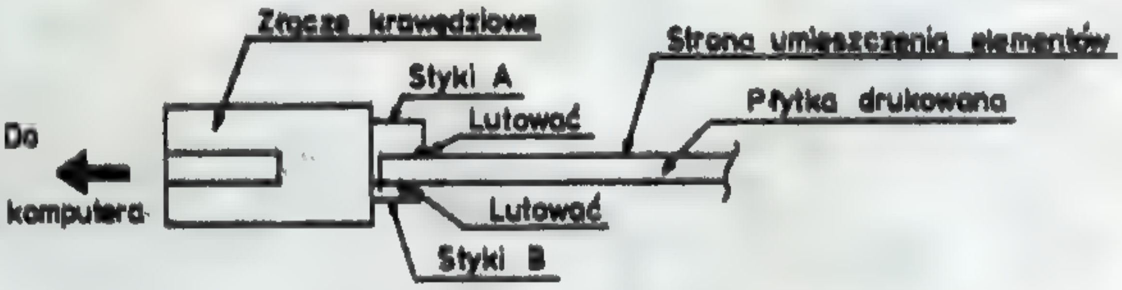
lone. Tak wykonany interfejs nie wymaga żadnych regulacji. Po wizualnej kontroli montażu możemy interfejs dołączyć do ZX Spectrum.

UWAGA!

Dołączenie interfejsu do komputera należy przeprowadzić przy wyłączonym zasilaniu komputera. Joystick możemy dołączyć do interfejsu zarówno przy wyłączonym, jak i włączonym zasilaniu komputera.

Po dołączeniu wg powyższych uwag interfejsu i joysticka włączamy zasilanie komputera. Uaktywnienie dowolnej funkcji manipulatora powinno powodować wyświetlanie na ekranie telewizora cyfr. Poszczególnym funkcjom joysticka powinny odpowiadać następujące cyfry:

Funkcja joysticka	Cyfra
w lewo	5
w prawo	7
w dół	8
w górę	9
strzał	0



Rys. 5 Szkic montażu złącza krawędziowego do płytki drukowanej

Przy zastosowaniu sprawnych elementów oraz prawidłowym montażu układ powinien działać od razu po podłączeniu.

(B. L.)

Wykaz elementów:

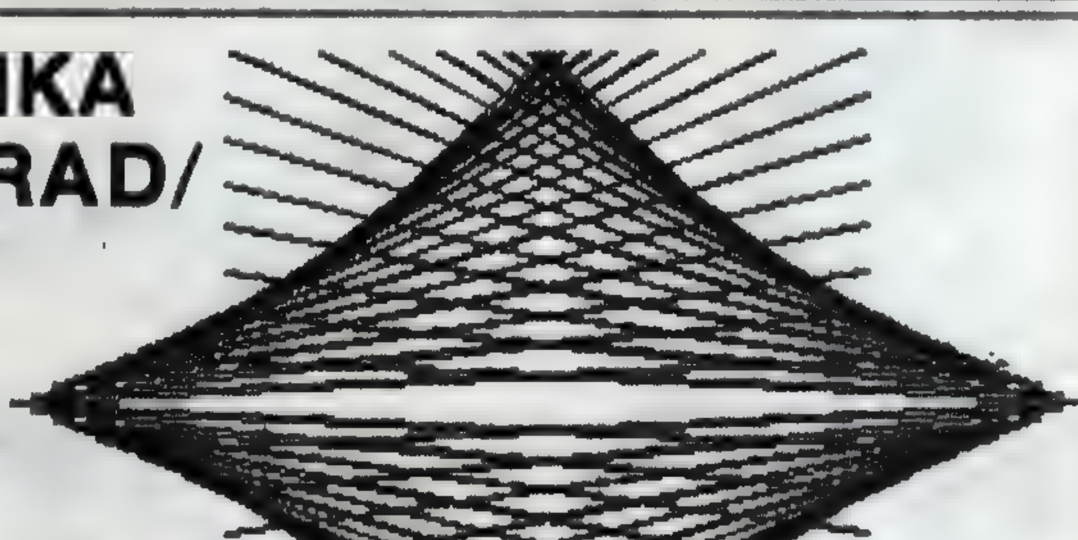
Układy scalone:
US1 — UCY 74LS05,
US2,U53 — UCY 74LS02;
Dioda:
DD1 — BAVP 19 (lub dowolna impulsowa);
Rezystory (wszystkie 0,125 W):
R1 — 4,7 kΩ,
R2 ÷ R6 — 6,8 k ÷ 10 kΩ;
Kondensatory:
C1 — 0,1 μF/163 V typu KFPm (lub MKSE),
C2 — 22 μF/6,3 V elektrolityczny;
Inne:
Gniazdo krawędziowe typu Eltra 801 (2×24 styki),
Gniazdo joysticka — wtyk szufladowy typu Eltra 871 — 9 styków.

Literatura:

[1] Bednarski P.: Dodatkowa klawiatura do mikrokomputera ZX Spectrum, Radioelektronik 7185
[2] Dickens A.: Spectrum Hardware Manual, Melbourne House, 1983

GRAFIKA AMSTRAD/

```
2 CLS
3 ORIGIN 70,200
5 FOR x=1 TO 15
11 DRAW 400,25*x
20 DRAW 10*x,5*x
25 NEXT x
30 ORIGIN 570,200
35 FOR y=1 TO 15
40 DRAW -400,25*y
50 DRAW -10*y,5*y
60 NEXT y
70 ORIGIN 70,200
80 FOR z=1 TO 15
85 DRAW 400,-25*z
90 DRAW 10*z,-5*z
100 NEXT z
110 ORIGIN 570,200
120 FOR k=1 TO 15
130 DRAW -400,-25*k
140 DRAW -10*k,-5*k
150 NEXT k
```













Wo

WOJSKOWY



Foto: Stanisław Iwan

czoraj i dziś...

INSTYTUT INFORMATYKI

AMSTRAD — BASIC

Pomimo wielu prób standaryzacji komend BASIC-a ciągle istnieje tendencja do pojawiania się kolejnych wersji tego języka. Każdy nowy typ mikrokomputera zawiera zbiór unikalnych słów kluczowych, a z oryginału pozostaje nie więcej jak kilkanaście. Podobnie jest z maszyną **AMSTRAD**.

Poniżej opisane zostały te jej słowa kluczowe, które nie występują w innych dialektach BASIC-a.

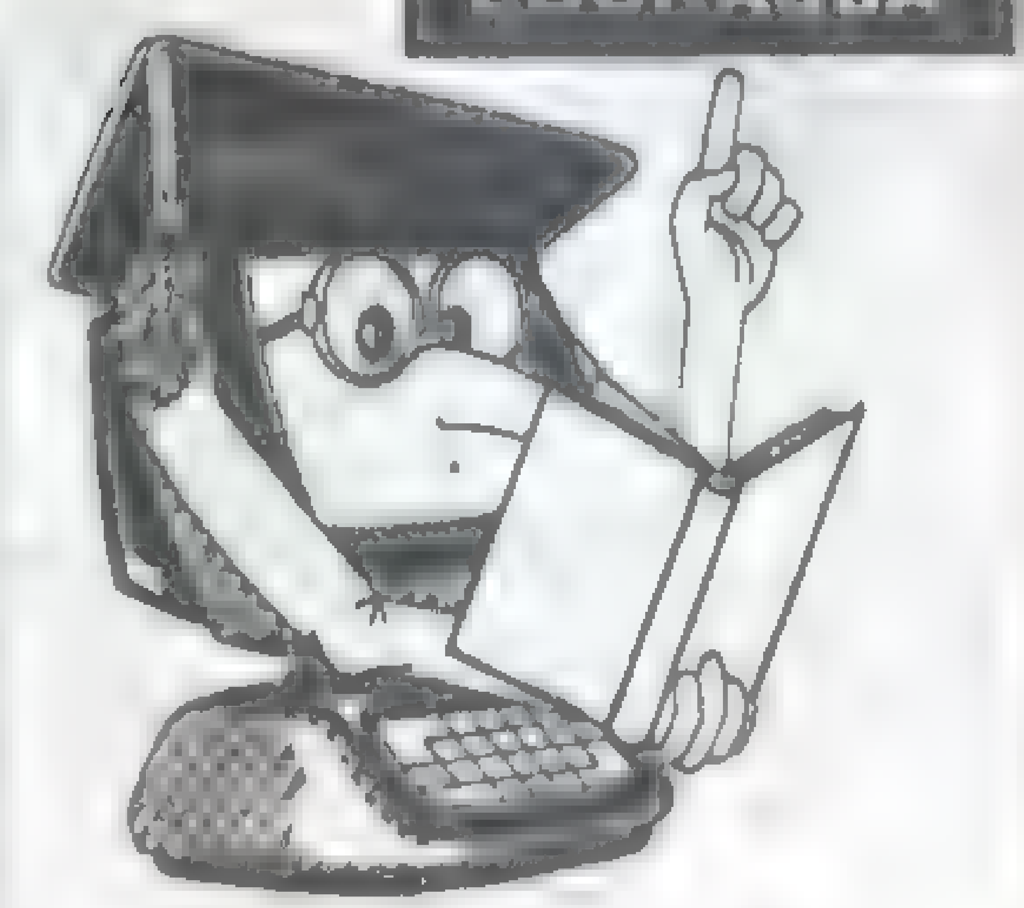
Komendy:

MEMORY — ustala wielkość pamięci dostępnej dla BASIC-a przez podanie najwyższego bajtu (analogia do CLEAR w Spectrum);
CALL — wywołuje podprogram w języku wewnętrznym;
FRAME — powoduje czasowe zgranie wydruków na ekranie upłynniając ruch;
AFTER...GOSUB — obsługa przerwań (!); po pewnym, określonym przedziale czasowym wywołuje podprogram w BASIC-u;
EVERY...GOSUB — obsługa przerwań (!); regularnie, co pewien okres wywołuje podprogram w BASIC-u;
ERROR — wywołuje błąd o podanym kodzie;
RESUME — wznawia normalną egzekucję programu po powrocie z podprogramu obsługi błędów wywołanego przez ON ERROR GOTO lub ON BREAK GOTO;
CHAIN — ładuje (LOAD) i uruchamia (RUN) program od podanego numeru linii;
OPENIN — otwiera plik ASCII do odczytu z dyskietki;
OPENOUT — otwiera plik ASCII do zapisu na dyskietkę;
CLOSEIN — zamyka odczyt pliku ASCII;
CLOSEOUT — zamyka odczyt pliku ASCII;
ENT — definiuje parametry wysokości dźwięku;
ENV — definiuje parametry głośności dźwięku;
RELEASE — uaktywnia kanały dźwiękowe;
WAIT — czeka na określoną wartość w porcie;
WIDTH — ustala długość linii dla drukarki;
SPEED KEY — określa szybkość samopowtarzania klawisza;
SPEED INK — określa częstotliwość zmian kolorów w trakcie migania (Spectrum FLASH);
SPEDD WRITE — określa szybkość zapisu do pamięci zewnętrznej;
CLG — czyści ekran graficzny;
ORIGIN — ustala miejsce graficznego punktu (0,0) na ekranie graficznym;
MASK — podaje parametr dla instrukcji DRAW powodujący możliwość modyfikowania linii (np. przerywana, punktowa);
FILL — wypełnia kolorem figurę zamkniętą;
LOCATE — pozycja wydruku znaku (Spectrum AT);
MODE — ustala tryb ekranu tekstowego na 20,40 lub 80 znaków w linii;
MOVE — przesuwa kursor graficzny;
SYMBOL — definiuje grafikę użytkownika;
TAG — włącza tryb wydruku znaków na ekranie graficznym;
TAGOFF — wyłącza powyższy tryb;
ZONE — określa liczbę znaków, o jaką przesunie się kursor tekstowy napotykając przecinek;
DEG — włącza tryb obliczania kątów w stopniach;
RAD — włącza tryb obliczania kątów w radianach;
DEFINT — deklaracja zmiennych typu integer;
DEFREAL — deklaracja zmiennych rzeczywistych;
DEFSTR — deklaracja zmiennych tekstowych;
ERASE — wymazanie zmiennej z pamięci;
KEY — definicja klawisza funkcyjnego;
KEYDEF — zmienia znaczenie klawisza;
CLEAR INPUT — powoduje wyczyszczenie bufora klawiatury;
CURSOR — odpowiada za wydruk kursora podczas symulacji instrukcji INPUT;
DI — wyłącza system przerwań;
EI — włącza system przerwań;

Funkcje:

CINT — zmienia liczbę rzeczywistą na całkowitą;
CREAL — zmienia liczbę całkowitą na rzeczywistą;
FIX — wyodrębnia część całkowitą liczby;
MAX — podaje najwyższą wartość z wyszczególnionych argumentów numerycznych;
MIN — podaje najniższą wartość;
ROUND — zaokrągla wartość do podanej liczby miejsc po przecinku;
UNT — podaje dziesiętnie adres wyrażony binarnie lub heksadecymalnie;
ASC — podaje kod znaku w systemie ASCII;
BIN$ — podaje wartość binarną liczby całkowitej;
HEX$ — przedstawia liczbę w postaci heksadecymalnej;
DEC$ — przedstawia liczbę w postaci dziesiętnej;
LOWER$ — zmienia w łańcuchu tekstowym duże litery na małe;
UPPER$ — zmienia w łańcuchu tekstowym małe litery na duże;
STRING$ — tworzy łańcuch tekstowy złożony z podanej ilości wybranego znaku;
SPACE$ — tworzy łańcuch tekstowy złożony z podanej ilości spacji;
COPYCHR$ — odczytuje znak z ekranu (Spectrum SCREEN$);
INSTR — szuka łańcucha tekstowego w innym i podaje pozycję, jeśli znajdzie;
INKEY — sprawdza, w jakiej konfiguracji naciśnięto podany klawisz (shift, control);
JOY — odczytuje stan dołączonego joystick'a;
POS — określa poziome położenie kursora tekstowego, głowicy drukarki lub ilość przesłanych znaków na dyskietkę;
REMAIN — wyłącza zegar obsługujący przerwania podając jego dotychczasowy stan;
SQ — podaje stan jednego z trzech kanałów dźwiękowych;
VPOS — określa pionowe położenie kursora tekstowego;
EOF — wskaźnik końca zbioru podczas pracy z urządzeniami WE/WY;
ERR — podaje numer ostatniego błędu;
DERR — podaje numer linii ostatniego błędu AMSDOD;
ERL — podaje numer linii ostatniego błędu;
FRE — wskazuje liczbę wolnych bajtów pamięci dla BASIC-a;
HIMEM — podaje najwyższy adres dostępny dla BASIC-a;
TEST — podaje kolor wyszczególnionego punktu na ekranie graficznym;
TIME — zwraca czas rzeczywisty od momentu wyzerowania komputera.

K. MAMCARZ



SŁOWNIK SHARP — PC 1500

Japońska firma SHARP produkuje mikrokomputery posługujące się językiem BASIC. Obowiązuje tu typowy dla BASIC słownik i gramatyka BASIC.

Na tej podstawie poszczególne firmy, a w tym i SHARP opracowały własne WYRAŻENIA.

Ponieważ Czytelnicy posługują się RÓŻNYMI komputerami podajemy skrócony „słownik", właściwy dla PC 1500 SHARP. To ułatwi „przetłumaczenie" z SHARP na SPECTRUM.

Wyrazenia (software) wynikają z architektury (hardware). Zatem krótko o PC 1500: ma on wbudowany WSKAŹNIK 26-pozycyjny z ciekłych kryształów LCD. Na nim wydawane są informacje literowo-cyfrowe. Przez konsolę komputer jest połączony z minipisakiem czterokolorowym. Moze on drukować (wypisywać) teksty literowo-cyfrowe, lub kreślić minirysunki na taśmie 59 mm zwykłego papieru.

PAMIĘCI SHARP

Rejestracja LICZB: Jest tu cała mnogość sposobów znakowania pamięci: Oto odmiany zapisu:
- — jednoliterowa: A,B,...,Z
- — dwuliterowa: AA, AB...,AZ, BA, BB,...,BZ, ZA, ZB,...,ZZ
- — literowo-cyfrowa: A0, A1,...,A9,...,Z0 Z1,...,Z9

Rejestracja WYRAZÓW: Operuje TAKIMI SAMYMI ZBIORAMI SYMBOLI z tą uwagą, iż dla WYRAZÓW dodajemy sufix $.

Np. CZ$ = „czwartek", A$ = „Jan", W1$ = „12 lipca".

Pamięci ZBIOROWE: Wyrażenie DIM A(3) oznacza CZTERY pamięci liczbowe A(0), A(1), A(2), A(3). Wyrażenie DIM T$ (7) oznacza OSIEM pamięci wyrazów.

Zapis typu DIM A(1,2) oznacza pamięć liczbową MACIERZOWĄ, mającą DWA wiersze i TRZY kolumny, zaś zapis typu DIM IN$ (5,10).

Przykład ŁĄCZENIA pamięci słów:
W$ = „WAC":W$ = W$ + „ — " + STR$1234

Daje W$ = „WAC—1234"

ROZKAZY SHARP

RUN 397, RUN „A" — startuj od wiersza 397, etykiety „A"
NEW — kasowanie całego programu
LIST 870, LIST „Z" — podaj wiersz 870, etykietę „Z" programu

DEKLARACJE SHARP

INPUT — wejście (wprowadzenie) danych
PRINT — wyjście: wyświetlenie na wskaźniku.
CURSOR 13 — zacznij od 13 pozycji wskaźnika
PAUSE — krótkotrwałe wyświetlenie na wskaźniku
USING — deklaracja formy zapisu danych
WAIT 50 — występuje łącznie z PRINT. Jest to czas wyświetlenia na wskaźniku
CLS — wygaszenie wskaźnika
BEEP 7,123,300 — generacja dźwięku o 7 powtórzeniach, częstotliwości (umownej) 123, impulsu trwającego 300 jednostek czasu
CLEAR — zerowanie wszystkich pamięci
RANDOM — generacja liczb losowych
DEGREE — kąt w stopniach
RADIAN — kąw w radianach

FUNKCJE SHARP

SIN, COS, TAN — sinus, cosinus, tangens
ASN, ACS, ATN — arcusy: sinusa, cosinusa, tangensa
LN, LOG — logarytmy naturalny i dziesiętny
EXP — exponenta
DEG — stopnie i ich ułamki dziesiętne
DMS — zapis typu STOPNIE, MINUTY, SEKUNDY
RND 7 — generowanie liczb losowych od 1 do 7
SQR — pierwiastek kwadratowy (lub $\sqrt{\ }$)
ABS — wartość bezwzględna
INT — część całkowita

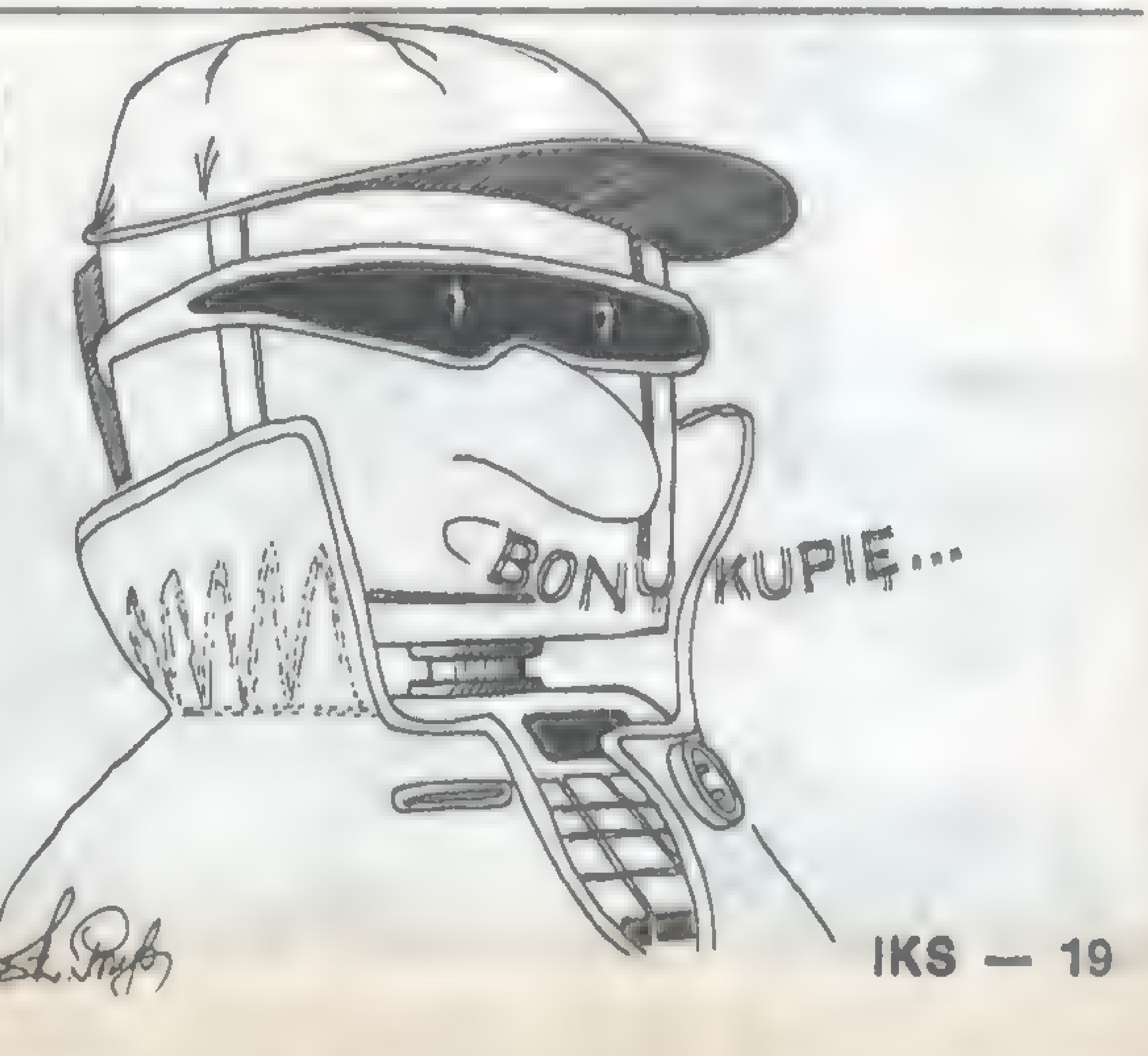
OPERACJE SHARP

+, —, *, / — cztery działania
(,) — nawiasy
<, <=, >, >= — nierówności
=, <> — równość, nierówność
∧ — „do potęgi"
AND — funkcja logiczna ORAZ
OR — funkcja logiczna LUB
NOT — logiczne ZAPRZECZENIE
INKEY$ — startowanie dowolnym klawiszem
TIME — wywołanie zegara-kalendarza

STEROWANIE PISAKIEM SHAPR

LLIST 230, 1500 — wydruk programu od wiersza 230 do 1500 włącznie
TEST — próba pisaka. Rysuje cztery kolorowe kwadraciki
LPRINT — wydruk tekstu literowo-cyfrowego
COLOR 3 — pisak czerwony (0 — czarny, 1 — niebieski, 2 — zielony)
GLCURSOR 138 — przesunięcie pisaka o 138 „punktów"
SORGN — ustawienie początku w zadanym punkcie
LINE — (0,0) — (20,—53) — wyrysowanie linii od współrzędnych x,y = 0,0 do 20,—53
RLINE — (10,—12) — wyrysowanie odcinka o rzutach 10,—12
TEXT — stan typowy — gotów do druku tekstów literowo-cyfrowych
GRAPH — przestawienie na rysowanie linii (LINE, RLINE...)

Wybrał: Janusz MILLER



Rys. Michał Przybyłowski



Klub mikrokomputerowy „Bit” **Foto: Jan Zelman**

Elementy systemów mikrokomputerowych (4)

ARYTMETYKA I ADRESOWANIE

Odcinek ten rozpoczynamy od dokończenia, omawianej w poprzednim numerze, logiki i arytmetyki systemów cyfrowych.

Zapis w systemie dwójkowym, inaczej naturalny zapis binarny (dwójkowy), ma pewne wady, które są nie bez znaczenia dla mikroprocesora. Wymaga on przeznaczenia najstarszego (najbardziej znaczącego) bitu liczby na zapis jej znaku oraz nie bardzo wiadomo co robić z faktem, że istnieją, jakby dwa zera (+0 i —0) zapisywane jako 0 0000 i 1 0000. W 1 bajcie za pomocą naturalnego zapisu binarnego, w skrócie NKB (natural binary code), można zapisywać liczby całkowite z przedziału (+0,...,255).

Z uwagi na niedogodność zapisu w NKB w kodzie znak-moduł, powodującym skomplikowanie operacji arytmetycznych, arytmetyka mikroprocesorów oparta jest na zapisie w kodzie uzupełnień do 2, oznaczanym U2. Kod ten umożliwia zapisanie w 1 bajcie liczby całkowitej ze znakiem z przedziału (—128, +127). Liczba w kodzie U2 zapisywana jest następująco:

$$(22)\quad L_{U2} = -a_n \times 2^n + \sum_{i=0}^{n-1} a_i \times 2^i$$

Najważniejsze cechy charakterystyczne zapisu w kodzie uzupełnień do 2 to: zmiana znaku liczby poprzez zanegowanie bajtu i dodanię jedynki; wykonywanie dodawania i odejmowania jak na liczbach w NKB, traktowanie całkowitych liczb dodatnich jak zapisanych w NKB

Mikroprocesory 8-bitowe nie są w stanie realizować wprost operacji mnożenia i dzielenia. Wyjątek stanowią mnożenie i dzielenie przez 2. Dzielenie przez 2 wykonywane jest poprzez przesunięcie liczby w prawo i powtórzenie na najbardziej znaczącej pozycji bitu, jaki się tam znajdował przed wykonaniem przesunięcia. Mnożenie przez 2 wykonywane jest poprzez przesunięcie liczby w lewo i dopisanie na najmniej znaczącej pozycji zera.

Z uwagi na używanie przez mikrokomputery zapisu dwójkowego, a przez człowieka zapisu dziesiętnego dobrym rozwiązaniem problemu komunikacji między człowiekiem i mikrokomputerem mógłby być zapis dwójkowo-dziesiętny, zwany BCD *(binary code decimal notation)*. W zapisie BCD każda cyfra dziesiętna kodowana jest oddzielnie za pomocą 4 bitów.

Przykładowo zapiszmy liczbę dziesiętną w kodzie BCD 8421:

$$1957_{10} = 0001'1001'0101'0111$$

Z uwagi na architekturę mikroprocesora nie opłaca się stosować tego kodu w mikrokomputerach, nabiera on znaczenia w dużych i średnich systemach komputerowych.

Niejako rozwiązaniem pośrednim problemu komunikacji jest zastosowanie zapisu szesnastkowego, znacznie ograniczającego długość zapisu liczb, łatwego do konwersji na zapis dwójkowy.

Znane są już nam pojęcia: cykl rozkazowy mikroprocesora *(instruction set)* i cykl maszynowy *(machine cycle)*, zwany cyklem procesora. Z uwagi na sygnały sterujące, wytwarzane przez system mikroprocesorowy, można wyróżnić kilka typów cykli procesora:

— cykl odczytu z pamięci;
— cykl zapisu do pamięci;
— cykl odczytu z układów wejścia-wyjścia;
— cykl zapisu do układów wejścia-wyjścia;
— cykl przyjęcia przerwania.

W czasie tych cykli mikroprocesor generuje sygnały sterujące, w przypadku mikroprocesora INTEL 8080 są nimi:

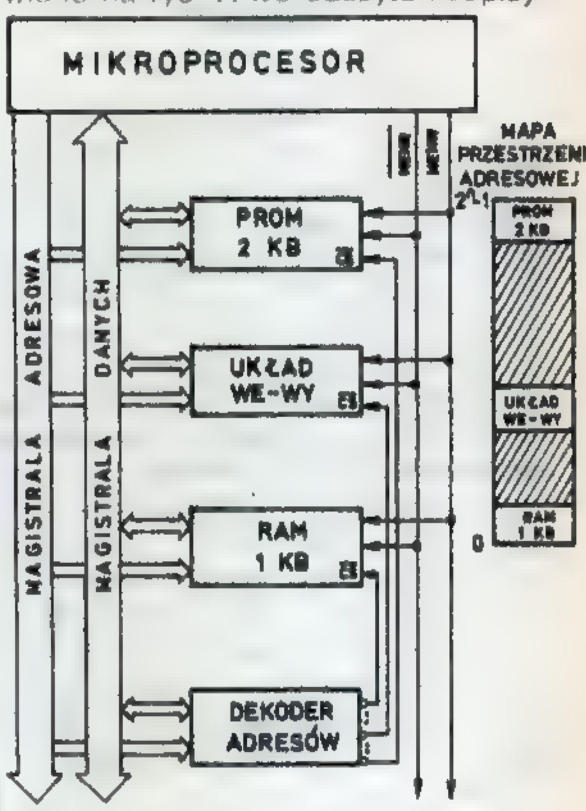
— **MEMR** *(memory read)*, sygnał odczytu z pamięci;
— **MEMW** *(memory write)*, sygnał zapisu do pamięci;
— **IOR** *(input/output read)*, sygnał odczytu z układów wejścia-wyjścia;
— **IOW** *(input/output write)*, sygnał zapisu do układów wejścia-wyjścia;
— **INTA** *(interrupt acknowledge)*, sygnał potwierdzenia przyjęcia przerwania.

Za pomocą takich sygnałów mikroprocesor 8080 komunikuje się z pamięcią i układami wejścia-wyjścia, stanowiącymi jego moduły zewnętrzne. Wybieranie modułów, z którymi mikroprocesor wymienia dane, następuje poprzez podanie magistralą adresową kombinacji zer i jedynek logicznych, stanowiącej adres modułu.

W zależności do realizacji układowej (sprzętowej) można wyróżnić dwie metody adresowania zewnętrznych modułów mikroprocesora:

— **jednolite adresowanie** pamięci i układów wejścia-wyjścia *(memory maped i/o)*;
— **odrębne adresowanie** pamięci i układów wejścia-wyjścia *(isolated i/o)*.

W pierwszym przypadku istnieje jednolita przestrzeń adresowa, o 2^N adresach (gdzie N — liczba linii adresowych). W przestrzeni tej pamięci jak i układy wejścia-wyjścia mogą być umieszczone w dowolny sposób. Rejestry układów wejścia-wyjścia traktowane są tu jak komórki pamięci. Metodę jednolitego adresowania przedstawiono na rys. 7. Do odczytu i zapisywania danych między mikroprocesorem a modułami zewnętrznymi wykorzystywane są dwa sygnały sterujące: MEMR i MEMW. Ten sposób adresowania posiada bardzo ważną zale-

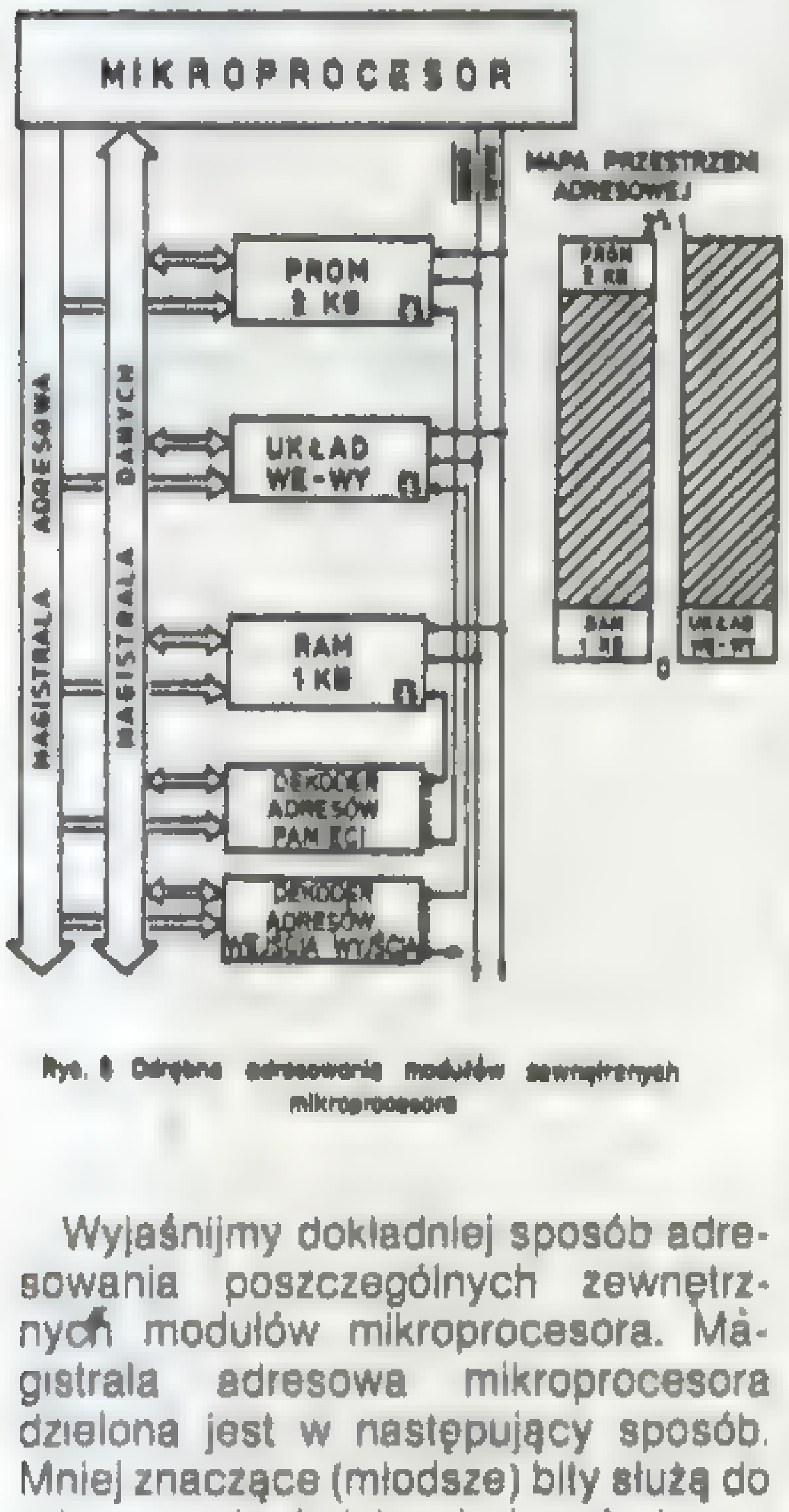


Rys. 7 Jednolite adresowanie modułów zewnętrznych mikroprocesora

dokończenie na str. 22

dokończenie ze str. 21
tę, do komunikacji mikroprocesora z układami wejścia-wyjścia można wykorzystać bogatszy zbiór rozkazów i trybów adresowania, odnoszących się w przypadku odrębnego adresowania tylko do pamięci.

W drugim przypadku, przy odrębnym adresowaniu, istnieją dwie odrębne przestrzenie adresowe. Do komunikacji z modułami zewnętrznymi mikroprocesor wykorzystuje cztery sygnały: MEMR, MEMW, IOR, IOW. Do przesyłania danych z/do układów wejścia-wyjścia służą dwa rozkazy IN *(input)* i OUT *(output)*. Metodę odrębnego adresowania przedstawiono na rys. 8.

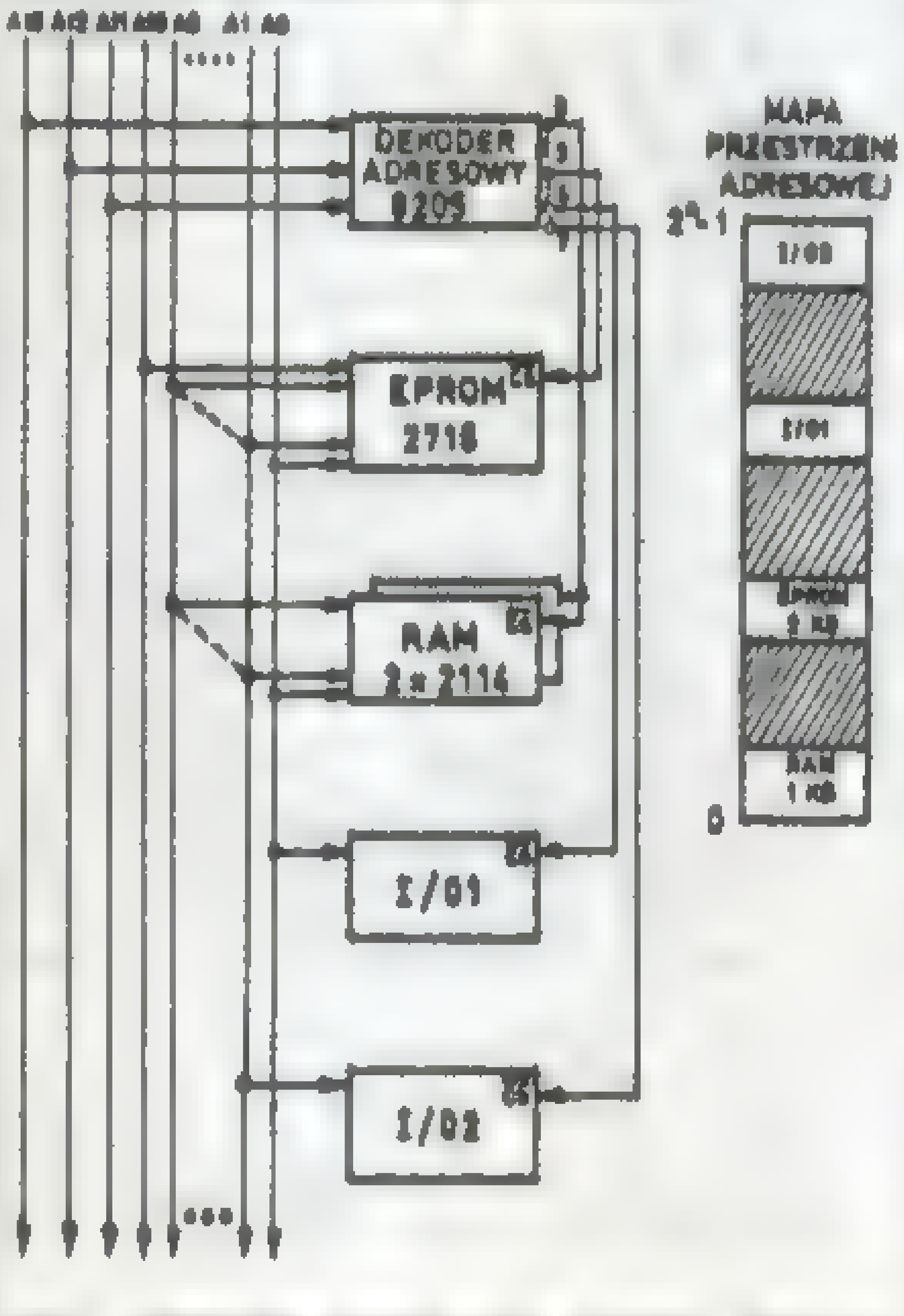


Rys. 8 Odrębne adresowanie modułów zewnętrznych mikroprocesora

Wyjaśnijmy dokładniej sposób adresowania poszczególnych zewnętrznych modułów mikroprocesora. Magistrala adresowa mikroprocesora dzielona jest w następujący sposób. Mniej znaczące (młodsze) bity służą do adresowania kolejnych komórek pamięci lub układów wejścia-wyjścia. Bardziej znaczące (starsze) bity służą do wybierania, za pomocą **dekoderów adresów**, kolejnych modułów, w których uaktywniane są **wejścia wybierające CS** *(chip select)*. W danej chwili, w zależności od sygnałów na wejściach dekodera czyli od starszych bitów magistrali adresowej, tylko na jednym z wyjść dekodera jest logiczne zero, wybierające konkretny moduł zewnętrzny. Przedstawia to poglądowo rys. 9.

Jako dekodery mogą być wykorzystywane również pamięci PROM lub EPROM. Na ich wyjściach danych można zaprogramować odpowiednie sekwencje zer i jedynek logicznych, przełączające — w zależności od bitów na wejściach adresowych pamięci — kolejne moduły mikroprocesora.

W systemach mikroprocesorowych stosuje się zwykle **dwustopniowe dekodowanie adresu**. Pierwszy dekoder służy do zdekodowania grupy modułów drugi natomiast umożliwia zaadreso- [illegible] odpowiedniego modułu.



Rys. 9 Wybieranie modułów przy jednolitym adresowaniu

Za pomocą [illegible] dzielić dysponowaną w systemie przestrzeń adresową, utworzyć **mapę przestrzeni adresowej**. W zależności od potrzeb użytkowych przestrzeń ta może być **ciągła, z dekodowaniem pełnym** lub **nieciągła, z dekodowaniem częściowym**, jak na rys. 9.

Zanim przejdziemy do trybów adresowania *(addressing mode)* czyli do sposobów określenia miejsca umieszczenia argumentu danego typu operacji mikroprocesora, przypomnijmy określenie — słowo mikroprocesora. Jest to elementarna jednostka informacji, jaka jest przesyłana w systemie mikroprocesorowym. Słowo może mieć długość 4 bity, 8 bitów (tzw. bajt), 16 bitów, 32 bity. W słowie może być zawarty kod rozkazu (operacji), kod rozkazu wraz z pośrednim adresem argumentu, liczba binarna lub w kodzie BCD, kod znaku alfanumerycznego. Rozkazy mogą mieć postać jednosłowową, lub wielosłowów, gdy zawierają argument będący liczbą lub numer komórki pamięci (adres), w której zapisana jest liczba.

W **liczniku rozkazów** mikroprocesora zawsze zawarty jest adres komórki pamięci, w której zapisano kod wykonywanego rozkazu. W zależności od zastosowanego w rozkazie trybu adresowania argumenty można określać w różny sposób. Wśród sposobów adresowania, wykorzystywanych w mikroprocesorach, wyróżnia się następujące podstawowe tryby:
— adresowanie natychmiastowe *(immediate addresing)*;
— adresowanie bezpośrednie *(direct addressing)*;
— *adresowanie pośrednie (indirect addressing)*;
— adresowanie względne *(relative addressing)*;
— adresowanie indeksowe *(indexing addressing)*;

Adresowanie natychmiastowe dotyczy rozkazów wielosłowowych, 2- lub 3-słowowych. W pierwszym słowie rozkazu umieszczony jest kod operacji, w słowach następnych znajduje się argument (operand). Tryb ten stosowany jest do umieszczania w pamięci danych stałych.

Adresowanie bezpośrednie odnosi się do rozkazów co najmniej 3-słowowych. W rozkazie, bezpośrednio za kodem operacji, umieszcza się adres komórki pamięci, z której należy pobrać argument, lub do której należy ten argument przesłać. Tryb ten wykorzystywany jest również przy wykonywaniu operacji na rejestrach wewnętrznych mikroprocesora. Są to zazwyczaj rozkazy 1-słowowe. Programiści stosują ten tryb wówczas, gdy znany jest adres argumentu, dotyczy to np. rozkazów skoków.

Adresowanie pośrednie dotyczy rozkazów, w których poza kodem operacji, umieszczony jest adres pamięci, pod który zapisany jest adres argumentu operacji. Do adresowania pośredniego można również zaliczyć adresowanie rejestrowe pośrednie *(pointer addressing)*, w którym adres argumentu umieszczony jest w rejestrze (rejestrach) roboczym mikroprocesora. Rejestr taki nazywany jest wówczas licznikiem danych *(data counter)* lub wskaźnikiem danych *(data pointer)*. Tryb adresowania pośredniego umożliwia korzystanie z argumentów znajdujących się w całej przestrzeni adresowej mikroprocesora.

Adresowanie względne odnosi się do rozkazów, w których adres argumentu jest wynikiem dodania liczby umieszczonej w rozkazie (za kodem operacji) i zawartości licznika rozkazów. Tryb ten stosuje się w rozkazach skoków warunkowych, adresowanie to jest stosowane w programach, które mogą pracować w dowolnym miejscu przestrzeni adresowej.

Adresowanie indeksowe jest to odmiana adresowania względnego. Adres argumentu jest sumą adresu zawartego wewnątrz rozkazu, bezpośrednio za kodem operacji, a zwanego przesunięciem oraz adresu umieszczonego w rejestrze indeksowym mikroprocesora. Adresowanie to jest oczywiście stosowane w mikroprocesorach posiadających taki rejestr. Tryb wykorzystywany jest głównie, gdy konieczny jest dostęp do danych umieszczonych kolejno w pamięci jako tablica

Omówione tryby adresowania są podstawowymi, spotykanymi w większości mikroprocesorów. Jak wiele jest sposobów adresowania niech świadczy to, że 16-bitowy mikrokomputer INTEL 8086 posiada ich aż 24.

Źródłem danych dla mikroprocesora jest świat zewnętrzny, z którym to komunikuje się on za pomocą układów wejścia-wyjścia, a są nimi urządzenia zewnętrzne takie jak: klawiatura, monitor lub wyświetlacz cyfrowy, drukarka, pamięć masowa (np. na dyskach elastycznych) etc.

Wymiana danych z urządzeniami zewnętrznymi może być realizowana m.in. poprzez:
— system przerwań (interrupt system);
— bezpośredni dostęp do pamięci (direct memory access DMA)

System przerwań umożliwia przerwanie wykonywania przez mikroprocesor **programu głównego** i przejście do **podprogramu** obsługującego urządzenie zgłaszające przerwanie. Sygnały żądania przerwania (każdy z nich posiada przyporządkowany odpowiednio priorytet) można potraktować jako sygnały sterujące pracą mikroprocesora. System przerwań umożliwia więc reakcje mikroprocesora na zdarzenia w jego otoczeniu zewnętrznym. Aby było to możliwe mikroprocesor, pod koniec każdego cyklu rozkazowego, sprawdza czy nie pojawiły się sygnały żądania przerwania. Jeśli w systemie zastosowano **jednopoziomową strukturę przerwań** to mikroprocesor obsłuży zgłoszenie przerwania, które pojawiło się jako pierwsze. Ingorowane są inne zgłoszenia — nawet o wyższym priorytecie, aż do końca obsługi pierwszego. Jeśli pojawiło się kilka sygnałów zgłoszeń przerwania to obsłużone zostanie to z nich, które posiada najwyższy priorytet. W ten sposób mikroprocesor obsłuży wszystkie zgłoszenia i dopiero powróci do wykonywania programu głównego. Jeśli w systemie zastosowano wielopoziomową strukturę przerwań, to podczas wykonywania obsługi przerwania, gdy pojawi się zgłoszenie przerwania o wyższym priorytecie, następuje zawieszenie obsługi danego przerwania, aż do momentu zakończenia obsługi zgłoszenia ważniejszego. Dopiero wtedy następuje powrót do obsługi mniej ważnego zgłoszenia (posiadającego niższy priorytet).

Bezpośredni dostęp do pamięci DMA umożliwia przeprowadzanie szybkich przesłań danych (tzw. transmisji blokowych) między pamięcią systemu mikroprocesorowego a urządzeniami zewnętrznymi. Transmisje te wykonywane są bez udziału mikroprocesora, poprzez specjalizowany układ wejścia-wyjścia tzw. sterownik DMA. Gdy ze sterownikiem współpracuje wiele urządzeń zewnętrznych, to podobnie jak w systemie przerwań określany jest priorytet urządzenia.

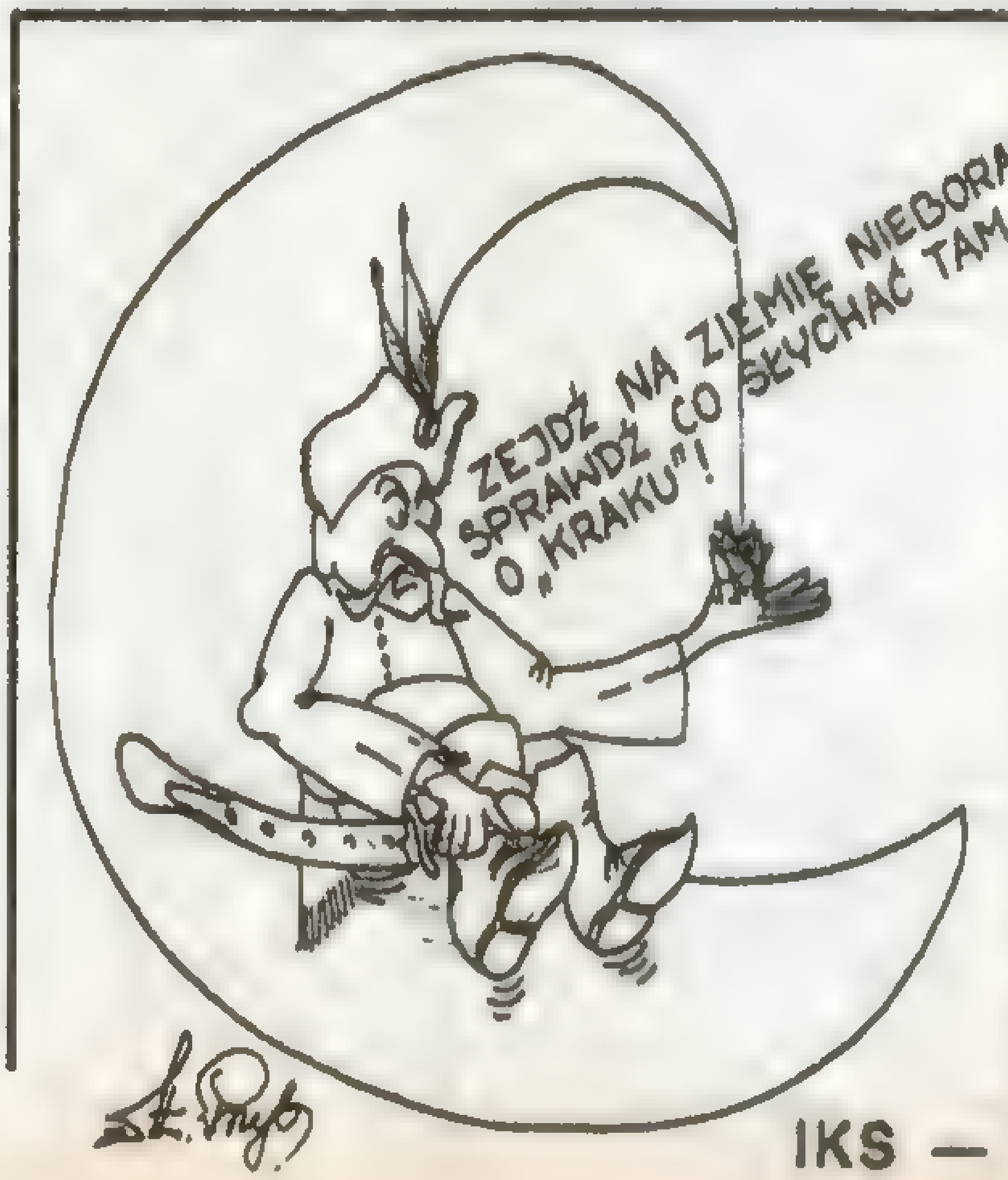
Terminologia i pojęcia wprowadzone w pierwszych czterech odcinkach cyklu staną się lepiej zrozumiałe już w następnym numerze „IKS-a", od którego rozpoczynamy omawianie budowy i programowania konkretnych systemów mikroprocesorowych.

Jacek WOJTALA

Jak to zrobić — dokończenie ze str. 3

```
315 W(Y-2)=WY
320 POKE ZN+Y-3,W(Y-2):POSITION 20,Y:P
RINT W(Y-2);"      "
325 GOTO 160
345 REM Zapamietanie danych dla
        definicji znaku
350 IF PEEK(53279)<3 THEN GOTO 500
355 W(Y-2)=WY:POKE ZN+Y-3,W(Y-2):POSIT
ION 20,Y:PRINT W(Y-2);"      "
360 GRAPHICS 0:POSITION 2,2
365 PRINT "Podaj kod ASCII definiowane
g          znaku  ";
370 INPUT CH:PRINT "Prawidlowa postac
 instrukcji  DATA"
375 POSITION 2,7
380 PRINT LAB;" DATA ";CH;:LAB=LAB+5
385 FOR I=1 TO 8
390 PRINT ",";W(I);
395 NEXT I
400 REM Wprowadzenie poprawki w
        tresci programu
405 POSITION 2,9:PRINT "CONT":POSITION
 2,4
410 POKE 842,13:STOP
415 POKE 842,12
420 GRAPHICS 0
425 REM Odtworzenie srodowiska
430 MEM=PEEK(106)-8:POKE 106,MEM:POKE
756,MEM
435 CHADR=256*MEM:POKE 752,2
440 ZN=ASC("!"):ZN=CHADR+(ZN-32)*8
445 GOSUB 1055:GOTO 75
500 IF PEEK(53279)<2 THEN GOTO 305
505 REM OPTION/START -Koniec programu
510 GRAPHICS 0
515 POSITION 2,4:PRINT "1056"
520 PRINT "CONT":POSITION 2,0
525 POKE 842,13:STOP
530 POKE 842,12
540 END
1000 DIM PROC$(32)
1010 REM Definiowanie procedury
         kopiujaCej znaki
1015 FOR I=1 TO 32
1020 READ B:PROC$(I)=CHR$(B)
1025 NEXT I
1030 DATA 104,104,133,213,104,133,212,
104,133,215,104,133,214,162,4,160,0,17
7,212,145,214
1035 REM
1040 DATA 200,208,249,230,213,230,215,
202,208,240,96
1045 REM Kopiowanie zbioru znakow
1050 REM
1055 A=USR(ADR(PROC$),224*256,CHADR)
1060 READ J:IF J=0 THEN RETURN
1065 RETURN
1070 READ B:POKE CHADR+(J-32)*8+I,B
1075 NEXT I
1995 DATA 0
2000 GOTO 1060
```

GRZEŚ



Rys. Michał Przybyłowski

PASCAL (3)

Dotychczas poznane instrukcje wyczerpują zbiór najczęściej wykorzystywanych instrukcji języka PASCAL. Jednak o użyteczności języka decydują nie tylko dostępne instrukcje, ale także, w znacznej mierze, dostępne dla programisty typy danych. Występujące w przedstawionych wcześniej przykładach typy INTEGER i REAL należą do grupy typów prostych. Oprócz typów prostych w języku PASCAL istnieją typy strukturalne, umożliwiające opis obiektów, traktowanych jako zbiory wartości typów prostych i strukturalnych. Do typowych obiektów tej klasy należą znane z matematyki wektory i macierze. W języku PASCAL odpowiednikiem wektorów i macierzy są tzw. tablice, którym poświęcamy niniejszą część.

1. Deklaracje i wykorzystanie tablic

Jednowymiarowa tablica to ciąg ponumerowanych elementów, których wartości są tego samego typu. Dwuwymiarowa tablica to ciąg wierszy lub kolumn, z których każda liczy tyle samo elementów tego samego typu. Rozmiary dwuwymiarowej tablicy to liczba wierszy i liczba elementów w wierszu (czyli liczba kolumn). Za pomocą jednowymiarowych tablic opisujemy zwykle takie obiekty jak wektory, natomiast za pomocą tablic dwuwymiarowych — macierze. (PASCAL pozwala na definiowanie tablic o większej ilości wymiarów. Ponieważ nie przewidujemy przedstawiania problemów wymagających takich tablic poprzestaniemy na omówieniu tablic co najwyżej dwuwymiarowych).

Naturalnym sposobem opisu tablic jest więc notacja.

a) **ARRAY [**1...*n***] OF** *integer*,

dla opisu tablicy jednowymiarowej o „*n*" elementach typu *integer*,

b) **ARRAY [**1...*m*, 1...*n***] OF** *real*,

dla tablicy dwuwymiarowej o rozmiarach $m \times n$ i elementach typu *real*.

Sposoby deklarowania i wykorzystania tablic w programach przedstawiamy na przykładach.

Rozważmy następujący problem: należy uporządkować rosnąco 6 zadanych liczb całkowitych. Jeden ze sposobów rozwiązania tego problemu opiera się na zasadzie porównywania i zamiany par sąsiadujących ze sobą liczb dopóty, dopóki wszystkie liczby nie zostaną ustawione w kolejności od najmniejszej do największej. Z przedstawionego sposobu wynika, że warunkiem jego zrealizowania jest jednoczesne udostępnienie wszystkich sześciu liczb. Jeżeli potraktujemy zadane liczby jako elementy „pionowej" tablicy jednowymiarowej to, przy pewnej dozie wyobraźni, możemy je uważać za znajdujące się w naczyniu z wodą bąbelki w „wagach" proporcjonalnych do ich wartości. Zatem każde przejście przez tablicę da w wyniku wypchnięcie bąbelka na poziom odpowiadający jego wadze. Metoda ta jest znana pod nazwą sortowania bąbelkowego. Przykładowy przebieg sortowania bąbelkowego ilustruje poniższa tabelka

Tabela nr 1

Ustawienie początkowe	Pierwsze przejście	Drugie przejście	Trzecie przejście	Czwarte przejście	Piąte przejście
574	8	8	8	8	8
303	574	23	23	23	23
34	303	574	34	34	34
125	34	303	574	125	125
8	125	34	303	574	303
23	23	125	125	303	574

Program realizujący porządkowanie dowolnych sześciu liczb całkowitych metodą sortowania bąbelkowego zamieszczono w przykładzie 8.

Przykład 8

```
PROGRAM prog8;
  {porzadkowanie zadanych 6 liczb rosnaco}
 VAR liczby:ARRAY[1..6] OF integer;
    x,i,j:integer;
 BEGIN
  FOR i:=1 TO 6 DO  {wczytanie liczb}
    READ(liczby[i]);
  WRITELN;
  WRITELN('wprowadzone liczby:');
  FOR i:=1 TO 6 DO
   WRITE(liczby[i]:5,' ':3);
  FOR i:=2 TO 6 DO {porzadkowanie liczb}
   FOR j:=6 DOWNTO i DO
     IF liczby[j-1]>liczby[j] THEN
       BEGIN
        x:=liczby[j-1];
        liczby[j-1]:=liczby[j];
        liczby[j]:=x
       END;
  WRITELN;
  WRITELN('liczby uporzadkowane:');
  FOR i:=1 TO 6 DO
   WRITE(liczby[i]:5,' ':3)
END.
```

Zadane liczby wczytywane są do jednowymiarowej sześcioelementowej tablicy „liczby". Działania na elementach tablicy przeprowadza się tak, jak na zmiennych typu zgodnego z typem elementów tablicy. Każdy element tablicy jest identyfikowany jednoznacznie przez nazwę tablicy oraz tzw. indeks, określający położenie elementów w tablicy, np

liczby [i] — i-ty wskazywany przez wartość zmiennej „i" element jednowymiarowej tablicy „liczby",

liczby [j — 1] — [j — 1]-szy (wskazany przez wartość wyrażenia „j—1" element jednowymiarowej tablicy „liczby".

W przypadku, gdy indeks jest wyrażeniem, wartość wyrażenia jest obliczona przed udostępnieniem elementu tablicy. Należy zauważyć, że bieżąca wartość indeksu musi mieścić się w przedziale wartości dopuszczalnych dla tego indeksu (w naszym programie <1,6>). Zmiana elementów tablicy miejscami odbywa się za pośrednictwem pomocniczej „x" (dlaczego?)

Program wykonano dla danych z przedstawionej tabelki uzyskując następujące wyniki

```
wprowadzone liczby:
 574     303     34     125     8     23
liczby uporzadkowane:
   8      23     34     125   303    574
```

Wprowadzając dane do programu „prog 8" należy pamiętać, aby po każdej wprowadzanej liczbie nacisnąć klawisz ENTER (co spowoduje wykonanie instrukcji READ wczytującej w tym przypadku jedną liczbę). Wyjaśnienie, dlaczego jest to konieczne, przedstawimy w odpowiednim czasie

Nowo wprowadzonym typom — w tym przypadku tablicom — można przypisywać nazwę, która będzie traktowana jako synonim danego typu. Nazwa ta może być następnie wykorzystana między innymi w deklaracji zmiennych. Taki sposób deklarowania tablic zastosowano w programie dodawania dwóch macierzy o ustalonych wymiarach (przykład 9). Przypomnijmy, że sumą dwóch macierzy **A** i **B** tego samego wymiaru jest trzecia macierz **C**, której ele-

menty wyznaczane są według zależności:

$c_{ij} = a_{ij} + b_{ij}$

gdzie:

i — indeks wierszy,

j — indeks kolumny

Przykład 9

```
PROGRAM prog9;
  (dodawanie macierzy o ustalonych wymiarach)
 TYPE
  macierz=ARRAY[1..3,1..5] OF real;
 VAR
  a,b,          (macierze wejsciowe)
  c:macierz;    (macierz wyjsciowa)
  i,j:integer;(zmienne pomocnicze)
 BEGIN
  (wczytanie elementow macierzy a)
   FOR i:=1 TO 3 DO
    BEGIN
     FOR j:=1 TO 5 DO
      READ(a[i,j]);
     READLN
    END;
  (wczytanie elementow macierzy b)
   FOR i:=1 TO 3 DO
    BEGIN
     FOR j:=1 TO 5 DO
      READ(b[i,j]);
     READLN
    END;
  (dodawanie macierzy a i b}
   FOR i:=1 TO 3 DO
    FOR j:=1 TO 5 DO
      c[i,j]:=a[i,j]+b[i,j];
  WRITELN;
  WRITELN('dane wejsciowe:');
  (wydruk macierzy a)
  WRITELN(' ':10,'macierz a');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     WRITE(' ':3,a[i,j]:10);
    WRITELN
   END;
  (wydruk macierzy b}
  WRITELN(' ':10,'macierz b');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     WRITE(' ':3,b[i,j]:10);
    WRITELN
   END;
  WRITELN;
  WRITELN('wynik:');
  (wydruk macierzy c)
  WRITELN(' ':10,'macierz c=a+b');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     WRITE(' ':3,c[i,j]:10);
    WRITELN
   END
```

W programie, w deklaracji zmiennych typu tablica (a,b,c) wykorzystano wcześniej zdefiniowany typ „macierz" opisujący macierze o wymiarach 3 × 5 i elementach typu REAL. Typ ten został zdefiniowany zgodnie z następującą, przyjętą w języku PASCAL, ogólną postacią definicji typu:

TYPE t = T,

gdzie:

TYPE — *słowo kluczowe oznaczające definicję typu,*

t — *nazwa definiowanego typu („macierz"),*

T — *opis definiowanego typu*

(**ARRAY** [1...3,1...5] **OF** *real*)

Program wykonano dla macierzy:

$$A = \begin{bmatrix} 10 & -6 & 15 & 1 & 33 \\ 25 & -12 & 31 & 6 & 89 \\ 16 & 4 & 8 & 19 & -7 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

uzyskując następujące wyniki:

```
dane wejsciowe:
          macierz a
   1.0000E+01   -6.000E+00   1.5000E+01   1.0000E+00   3.3000E+01
   2.5000E+01   1.2000E+01   -3.100E+01   6.0000E+00   8.9000E+01
   1.6000E+01   4.0000E+00   8.0000E+00   1.9000E+01   -7.000E+00
          macierz b
   1.0000E+00   2.0000E+00   3.0000E+00   4.0000E+00   5.0000E+00
   1.0000E+00   2.0000E+00   3.0000E+00   4.0000E+00   5.0000E+00
   1.0000E+00   2.0000E+00   3.0000E+00   4.0000E+00   5.0000E+00

wynik:
          macierz c=a+b
   1.1000E+01   -4.000E+00   1.8000E+01   5.0000E+00   3.8000E+01
   2.6000E+01   1.4000E+01   -2.800E+01   1.0000E+01   9.4000E+01
   1.7000E+01   6.0000E+00   1.1000E+01   2.3000E+01   -2.000E+00
```

Elementy obu macierzy zostały wprowadzone wierszami przy czym, w odniesieniu do każdego wiersza, obowiązywały te same zasady wprowadzania danych, jak w programie poprzednim. Przejście do nowego wiersza następowało po dodatkowym wciśnięciu klawisza ENTER, po wprowadzeniu elementów danego wiersza (odpowiada to wykonaniu instrukcji READLN).

Zastanówmy się obecnie, jakie zmiany byłyby konieczne w przedstawionych programach w przypadku zmiany wymiarów tablic. Nietrudno zauważyć, że zmiana wymiarów „rozprzestrzeni" się na cały program. Należy bowiem zmodyfikować również wszystkie instrukcje programu, w których występują wartości graniczne wymiarów tablic. I tak, zmiana wymiarów tablicy „liczby" z przykładu 8 wymaga wprowadzenia zmian w sześciu wierszach programu a zmiana wymiarów tablicy „macierz", z przykładu 9 — aż w trzynastu wierszach!

Wad tych jest pozbawiony program mnożenia dwóch macierzy, z przykładu 10. Przypomnijmy, że iloczyn **A** * **B** dwóch macierzy **A** i **B** jest taką macierzą **C**, w której element i-tego wiersza i j-tej kolumny jest sumą iloczynów elementów i-tego wiersza macierzy **A** przez odpowiednie elementy j-tej kolumny macierzy **B**. Liczba kolumn macierzy **A** musi być przy tym równa liczbie wierszy macierzy **B**. Jeżeli więc macierz **A** jest stopnia p * q i macierz **B** — stopnia q * r, to macierz **C** jest stopnia p * r:

$$C_{p*r} = A_{p*q} \times B_{q*r}$$

gdzie:

$$c_{ij} = \sum_{k=1}^{q} a_{ik} * b_{kj}$$

Przykład 10

```
PROGRAM prog10;
 (mnozenie macierzy)
 CONST
  dw=1; (dolna wartosc indeksu wierszy)
  gw=7; (gorna wartosc indeksu wierszy)
  dk=1; (dolna wartosc indeksu kolumn)
  gk=7; (gorna wartosc indeksu kolumn)
 TYPE
  tablica=ARRAY[dw..gw,dk..gk] OF real;
 VAR
  p,           (liczba wierszy macierzy a)
  q1,          (liczba kolumn macierzy a )
  q2,          (liczba wierszy macierzy b)
```

dokończenie na str. 26

dokończenie ze str. 25

```
  r:integer;  {liczba kolumn macierzy b }
  a,b,        {macierze wejsciowe}
  c:tablica;  {macierz wynikowa  }
  i,j,k:integer; {zmienne robocze}
  s:real;
 BEGIN
  READLN(p,q1);{wczytanie wymiarow macierzy a}
  IF (p>gw) OR (q1>gk). THEN
   BEGIN
    WRITELN;
    WRITELN('niepoprawne wymiary macierzy a')
   END
  ELSE
   BEGIN
    FOR i:=dw TO p DO  {wczytanie elementow}
     BEGIN             {macierzy a         }
      FOR j:=dk TO q1 DO
       READ(a[i,j]);
      READLN
     END;
    READLN(q2,r);{wczytanie wymiarow macierzy b}
    IF (q2>gw) OR (r>gk) THEN
     BEGIN
      WRITELN;
      WRITELN('niepoprawne wymiary macierzy b')
     END
    ELSE
     BEGIN
      FOR i:=dw TO q2 DO  {wczytanie elementow}
       BEGIN              {macierzy b         }
        FOR j:=dk TO r DO
         READ(b[i,j]);
        READLN
       END;
      WRITELN;{wypisanie wczytanych macierzy }
      WRITELN(' ':5,'macierz a');
      FOR i:=dw TO p DO
       BEGIN
        FOR j:=dk TO q1 DO
         WRITE(a[i,j]:10,' ');
        WRITELN
       END;

      WRITELN(' ':5,'macierz b');
      FOR i:=dw TO q2 DO
       BEGIN
        FOR j:=dk TO r DO
         WRITE(b[i,j]:10,' ');
        WRITELN
       END;
      IF q1=q2 THEN {sprawdzenie zgodnosci  }
       BEGIN        {wymiarow macierzy a i b}
        FOR i:=dw TO p DO {mnozenie macierzy}
         FOR j:=dk TO r DO
          BEGIN
           s:=0;
           FOR k:=dk TO q1 DO
            s:=s+a[i,k]*b[k,j];
           c[i,j]:=s
          END;
        WRITELN;
        WRITELN(' ':5,'macierz c=a*b');
        FOR i:=dw TO p DO
         BEGIN
          FOR j:=dk TO r DO
           WRITE(c[i,j]:10,' ');
          WRITELN
         END
       END
      ELSE
       WRITELN('niezgodne wymiary macierzy a i  b')
     END
   END
 END.
```

Wymiary tablic zostały określone poprzez nazwy stałych (dw, gw, dk, gk), zdefiniowanych w części definiującej programu po słowie kluczowym CONST. Ponieważ w części operacyjnej programu również używamy zdefiniowanych stałych, nie jest ona wrażliwa na zmiany wymiarów tablic. Ewentualna zmiana wymiarów tablic wymaga jedynie przedefiniowania odpowiednich stałych.

Analizując program „prog 10" można zauważyć, że aktualnie wymagane wymiary tablic są wczytywane w czasie wykonywania programu. Pozwala to na zmianę wymiarów tablic bez konieczności modyfikacji tekstu programu. Oczywiście zakres zmian jest ograniczony zdefiniowanymi wymiarami tablic. Zatem po wczytaniu bieżących wartości wymiarów tablic należy sprawdzić, czy mieszczą się one w przedziale wartości dopuszczalnych. Program wykonano dla macierzy:

$$A = \begin{bmatrix} 1 & 34 & 0 \\ 22 & -14 & -5 \end{bmatrix}$$

$$B = \begin{bmatrix} -7 & 2 & 1 & 2 & 0 \\ 12 & 2 & 5 & 1 & -1 \\ 3 & 2 & -9 & 1 & 0 \end{bmatrix}$$

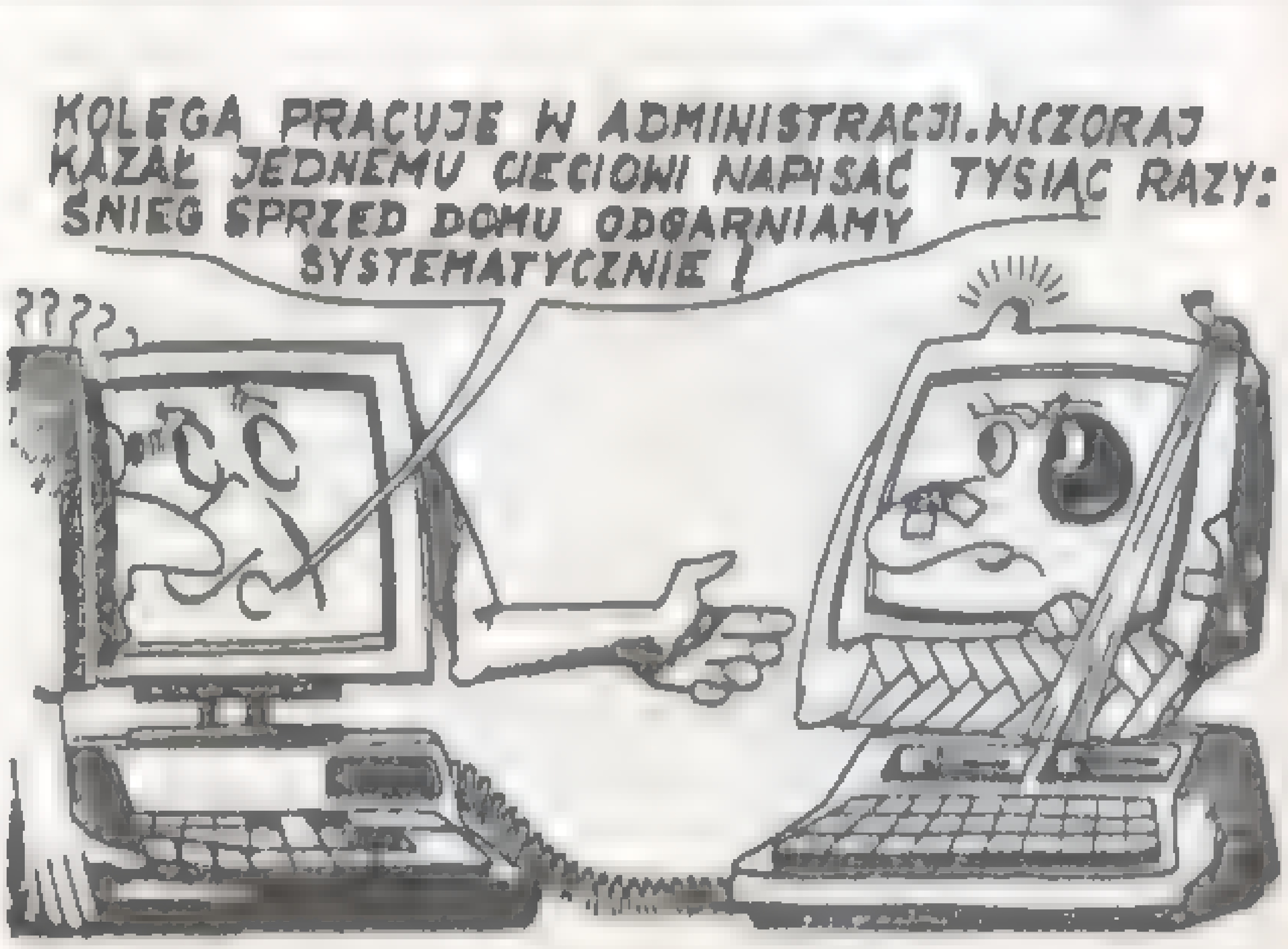
uzyskując [illegible]

```
         macierz a
1.0000E+00 3.4000E+01 0.0000E+00
2.2000E+01 -1.400E+01 -5.000E+00
         macierz b
-7.000E+00 1.0000E+00 2.0000E+00 0.0000E+00
1.2000E+01 5.0000E+00 1.0000E+00 -1.000E+00
3.0000E+00 -9.000E+00 1.0000E+00 0.0000E+00

         macierz c=a*b
4.0100E+02 1.7100E+02 3.6000E+01 -3.400E+01
-3.370E+02 -3.000E+00 2.5000E+01 1.4000E+01
```

2. Składowanie zbiorów roboczych

Czytelnicy wykorzystujący praktycznie system **TURBO PASCAL** zauważyli zapewne, że żądanie utworzenia nowego zbioru roboczego, gdy w pamięci istniał już zbiór (np. poddawany edycji), powodowało wyprowadzenie pytania dotyczącego składowania istniejącego zbioru. Odpowiedź twierdząca (Y) powoduje w tym przypadku składowanie zbioru w pamięci dyskowej, przecząca zaś — jego zniszczenie poprzez otworzenie nowego zbioru roboczego. Widzimy zatem, że składowanie zbioru zapobiega jego zniszczeniu. Składowanie programu można uzyskać również przez wykonanie komendy S. Jeżeli istnieje poprzednia wersja zbioru o tej samej nazwie, co składowany zbiór roboczy, to typ zbioru w tej wersji zostanie przemianowany na **BAK**. Należy podkreślić, że wprowadzając nowy program o dużej ilości wierszy źródłowych warto dokonywać jego składowania po wprowadzeniu kolejnej porcji wierszy



Rys. Michał Przybyłowski

Zapobiegnie to utracie całego wprowadzonego programu w przypadku jakiejkolwiek awarii. Wczytanie składowanego programu umożliwia znana już komenda **W**.

3. Zadania

Zadanie 7
Napisać program obliczający iloczyn skalarny dwóch wektorów o n współrzędnych całkowitych.

Zadanie 8
Napisać program porządkowania n-elementowego ciągu liczb rzeczywistych metodą „przez proste wybieranie". W metodzie tej przyjmuje się następujące zasady:
- — wybieramy liczbę najmniejszą,
- — zamieniamy ją z pierwszą liczbą ciągu,
- — operacje powyższe powtarzamy dla pozostałych n-1 liczb, następnie dla n-2 liczb itd. aż pozostanie tylko liczba największa.

Zadanie 9
Napisać program odejmowania dwóch macierzy o zadanych wymiarach i elementach rzeczywistych.

Zadanie 10
Napisać program, który dla zadanej macierzy kwadratowej oblicza sumy elementów leżących:
- a) nad przekątną główną,
- b) na przekątnej głównej,
- c) pod przekątną główną.

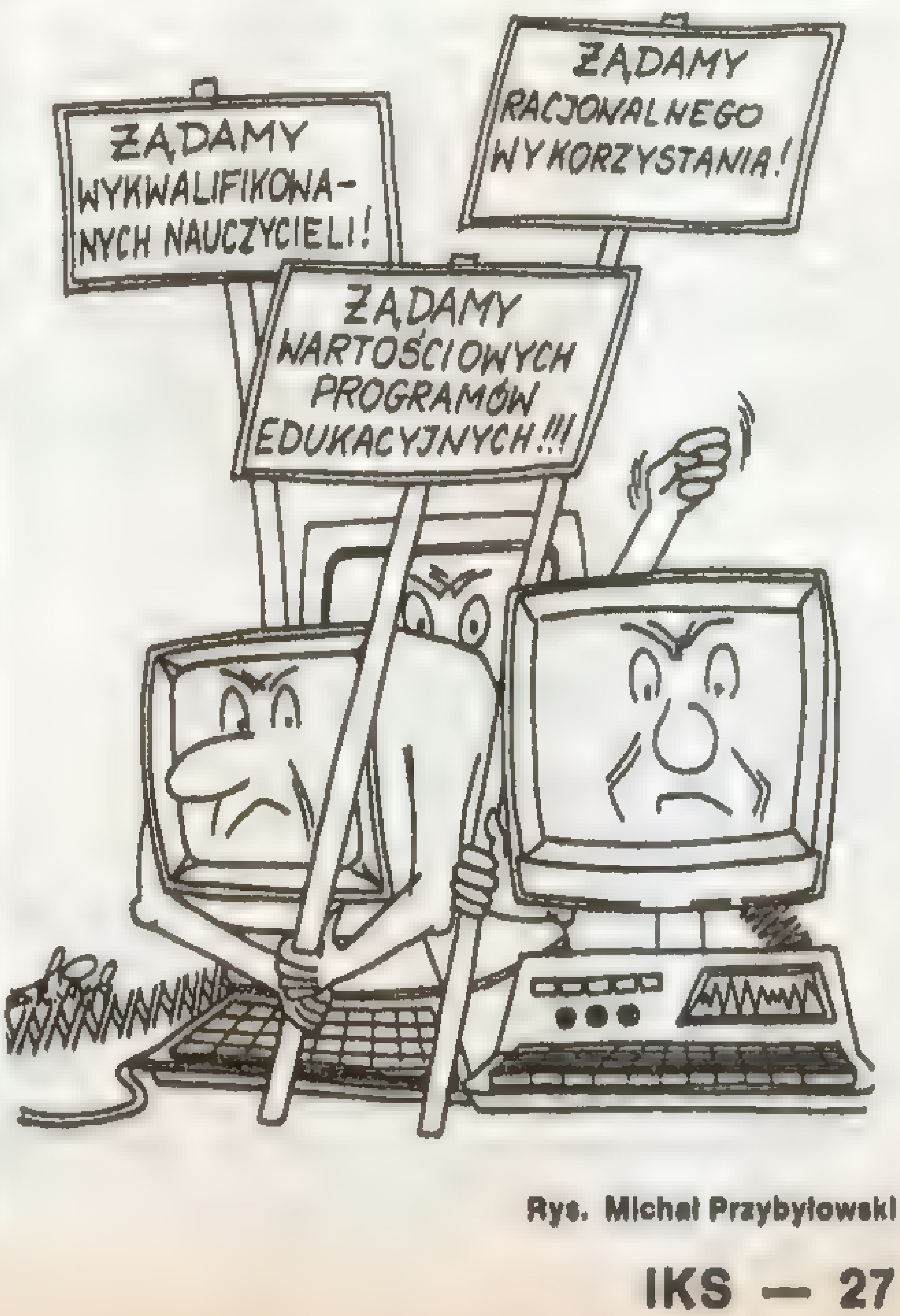
Stefan ROZMUS

Przepraszamy...
zamieszczając poprawne, dobrze napisane programy z pierwszej części PASCALA.

```
PROGRAM prog2;
  VAR x,y,suma,roznica,iloczyn:integer;
      iloraz:real;  {deklaracje zmiennych}
  BEGIN
   READ(x,y);
   WRITELN;
   WRITELN('pierwsza liczba = ',x:2,
          '   druga liczba = ',y:2);
   suma:=x+y;
   roznica:=x-y;
   iloczyn:=x*y;
   iloraz:=x/y;
   WRITELN('suma = ',suma:3,'  roznica = ',
         roznica:2,'  iloczyn = ',iloczyn:4,
         '   iloraz = ',iloraz:2:5)
 END.
```

```
 PROGRAM prog3;
  VAR a,b,c,  {wspolczynniki trojmianu}
      x1,x2,  {pierwiastki trojmianu}
      delta:real; {zmienna pomocnicza}
  BEGIN
   READLN(a);
   IF a <= 0 THEN
     WRITELN('  niedopuszczalna wartosc ',
             'wspolczynnika a')
   ELSE
    BEGIN
     READ(b,c);
     WRITELN;
     WRITELN('dla wspolczynnikow : a = ',a);
     WRITELN('                     b = ',b);
     WRITELN('                     c = ',c);
     delta:=b*b-4*a*c; {obliczenie wyroznika}
                       { trojmianu          }
     IF delta < 0 THEN
      WRITELN('brak pierwiastkow rzeczywi',
              'stych')
     ELSE
      IF delta = 0 THEN
       BEGIN
        x1:=-b/(2*a);
        WRITELN('trojmian ma jeden pierwi',
                'astek podwojny x = ',x1)
       END
      ELSE
       BEGIN
        x1:=(-b-SQRT(delta))/(2*a);
        x2:=(-b+SQRT(delta))/(2*a);
        WRITELN;
        WRITELN('trojmian ma dwa pierwiastki',
                ' : x1 = ',x1);
        WRITELN(' ':27,' : x2= ',x2)
       END
    END
END.
```

```
 PROGRAM prog4;
   VAR x,y:integer;
   BEGIN
    READ(x,y);
    WRITELN;
    WRITE('najwiekszy wspolny podzielnik',
          ' liczb ',x,' i ',y,' wynosi ');
    WHILE x<>y DO
      IF x>y THEN
        x:=x-y
      ELSE
        y:=y-x;
    WRITELN(x)
   END.
```



Rys. Michał Przybyłowski

osiągniemy następujące efekty:

— VIC korzysta z BANKU drugiego,

— PAMIĘĆ EKRANU rozpoczyna się od 35840,

— GENERATOR ZNAKÓW jest standardowy (z ROM) od adresu 36864,

— system operacyjny poinformowany jest o aktualnym położeniu PAMIĘCI EKRANU (35840:256=140)

b) Wykonując sekwencję instrukcji:

POKE 56576,150: POKE 648,120: POKE 53272,25

POKE 56,28672/256: CLR lub POKE 56,112: CLR

osiągamy następujące efekty:

— VIC korzysta z BANKU pierwszego,

— PAMIĘĆ EKRANU rozpoczyna się od 30720,

— GENERATOR ZNAKÓW rozpoczyna się od 28672,

— system operacyjny poinformowany jest o aktualnym położeniu PAMIĘCI EKRANU (30720:256=120),

— zabezpieczenie GENERATORA ZNAKÓW i PAMIĘCI EKRANU przed ingerencją Basica w tym obszarze. Wskaźnik (bajt 56) określający największy adres używany przez Basic ustala go na 28671.

Po włączeniu mikrokomputera VIC pracuje w BANKU 0. PAMIĘĆ EKRANU zlokalizowana jest począwszy od 1024 bajtu, a GENERATOR ZNAKÓW (standardowych) od 4096 bajtu. Jeżeli zmienimy położenie PAMIĘCI EKRANU, to powrót do standardowego obszaru uzyskamy przyciskając jednocześnie klawisze (energicznie): RUN/STOP i RESTORE. Brak kursora na ekranie będzie świadczył o niezgodności rzeczywistego położenia PAMIĘCI EKRANU ze wskazaniami bajtu 648. Standardowa wartość wynosi 4.

Jak programować własne znaki i jak korzystać z graficznego trybu pracy VIC w następnej części.

Program demonstracyjny:

```
0 REM C-64 ZADEMONSTRUJE UZYWANIE DWOCH EKRANOW
10 POKE 56576,148: REM WYBIERAMY BANK 3
20 B$="NERT": REM DOWOLNE 4 ZNAKI
99 REM WYSWIETLAMY JEDEN EKRAN JEDNOCZESNIE ZAPISUJAC NASTĘPNY
100 GOSUB 1000: POKE 53272,17: POKE 648,200: REM EKRAN 1
110 GOSUB 1000: POKE 53272,33: POKE 648,196: REM EKRAN 2
120 GOTO 100
999 REM BUDUJEMY LOSOWY LANCUCH ZNAKOW DO WYSWIETLENIA
1000 X$="": FOR J=1 TO 9: L=RND(1)*4+1
1010 X$=X$+MID$(B$,L,1): NEXT
1020 PRINTCHR$(147): FOR J=1 TO 111: PRINTX$;: NEXT
1030 RETURN
```

BANK 0: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252) OR 3

(zwykle wartość 151)

Adresy początkowe PAMIĘĆ EKRANU	POKE 648.	GENERATOR ZNAKÓW ROM 2048	4096	6144	8192	10240	12288	14336
		POKE 53272.						
1024	4	19	21	23	25	27	29	31
2048	8	35	37	39	41	43	45	47
3072	12	51	53	55	57	59	61	63
PAMIĘĆ EKRANU nie moze być zlokalizowana między 4096--8191, ponieważ VIC widzi tu GENERATOR ZNAKÓW standardowych z ROM-u								
8192	32	131	133	135	137	139	141	143
9216	36	147	149	151	153	155	157	159
10240	40	163	165	167	169	171	173	175
11264	44	179	181	183	185	187	189	191
12288	48	195	197	199	201	203	205	207
13312	52	211	213	215	217	219	221	223
14336	56	227	229	231	233	235	237	239
15360	60	243	245	247	249	251	253	255

BANK 1: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252) OR 2

(zwykle wartość 150)

Adresy początkowe PAMIĘĆ EKRANU	POKE 648	GENERATOR ZNAKÓW 16384	18432	20480	22528	24576	26624	28672	30720
		POKE 53272.							
16384	64	1	3	5	7	9	11	13	15
17408	[illegible]	17	19	21	23	25	27	29	31
18432	72	33	35	37	39	41	43	45	47
19456	76	49	51	53	55	57	59	61	63
20480	80	65	67	69	71	73	75	77	79
21504	84	81	83	85	87	89	91	93	95
22528	[illegible]	97	99	101	103	105	107	109	111
23552	92	113	115	117	119	121	123	125	127
24576	96	129	131	133	135	137	139	141	143
25600	100	145	147	149	151	153	155	157	159
26624	104	161	163	165	167	169	171	173	175
27648	108	177	179	181	183	185	187	189	191
28672	112	193	195	197	199	201	203	205	207
29696	116	209	211	213	215	217	219	221	223
30720	120	225	227	229	231	233	235	237	239
31744	124	241	243	245	247	249	251	253	255

BANK 3: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252)

(zwykle wartość 148)

UWAGA: pamięć o adresach 49152-53247 może być tylko używana jako PAMIĘĆ EKRANU (patrz komentarz w BANKU 2).

Adresy początkowe PAMIĘĆ EKRANU	POKE 648.	GENERATOR ZNAKÓW 49152	51200	53248	55296	57344	59392	61440	63488
		POKE 53272.							
49152	192	1	3	5	7	9	11	13	15
50176	196	17	19	21	23	25	27	29	31
51200	200	33	35	37	39	41	43	45	47
52224	204	49	51	53	55	57	59	61	63

BANK 2: wybieramy za pomocą instrukcji

POKE 56576, (PEEK(56576) AND 252) OR 1
(zwykle wartość 149)

Adresy początkowe PAMIĘĆ EKRANU	POKE 648.	GENERATOR ZNAKÓW ROM							
		32[illegible]	3481[illegible]	368[illegible]	38[illegible]	40[illegible]	430[illegible]	450[illegible]	[illegible]
		POKE 53272							
32768	128	1	3	5	7	9	11	13	15
33792	132	17	19	21	23	25	27	29	31
34816	136	33	35	37	39	41	43	45	47
35840	140	49	51	53	55	57	59	61	63

PAMIĘĆ EKRANU nie może być zlokalizowana między 36864-40959, ponieważ VIC widzi tu GENERATOR ZNAKÓW standardowych z ROM.

Obszaru pamięci o adresach 40960-49151 nie należy używać jako PAMIĘCI EKRANU. VIC ma dostęp w tym obszarze do pamięci RAM, natomiast mikroprocesor widzi pamięć ROM. Pamięć RAM i ROM jest w tym rejonie zrównoleglona i stanowi dwie odrębne części. Chcąc użyć tych obszarów jako PAMIĘCI EKRANU należy wyłączyć pamięć ROM. VIC i mikroprocesor widzą wtedy ten sam rzeczywisty obszar pamięci.

W naszym komputerlandzie

Wśród robotów z kierowanego przez siebie zespołu Spektruś uchodził zawsze za szefa dobrego, wyrozumiałego lecz pryncypialnego, wymagającego i dominującego nad wszystkimi. W jego domu od dawna jednak panuje równouprawnienie. Może właśnie dlatego trzepanie dywanu, pastowanie podłogi i zmywanie naczyń znużyło go już trochę. Jego żona, Elwra, parokrotnie zwracała na to uwagę. Najpierw delikatnie, potem nieco ostrzej. Spektruś próbował się mobilizować, ale bezskutecznie. Stosunki partnerskie polegają jednak na tym, że gdy jedna strona przeżywa kryzys — druga wyciąga do niej pomocną dłoń.

Obliczywszy, że trzepanie dużego dywanu może kosztować 2 tys. zł, a pastowanie i mycie podłóg jakieś 5 tys., żona postanowiła zastosować elementy reformy.

— Daj mi najlepiej te pieniądze — rzekła kiedyś. Przecież, gdybyś był obcym robotem, musiałabym ci właśnie tyle zapłacić. A przecież jakoś ostatnio praca w domu wcale cię nie interesuje.

Trudno było Spektrusiowi zaprzeczyć. Chwycił za portfel i nieśmiało zapytał:

— A jak będzie z porządkami?

— Nie martw się — odrzekła. Za taką sumę można wypucować mieszkanie do połysku.

Po dwóch dniach żona pojawiła się w nowej sukni. Do tego zmieniła uczesanie i klipsy. Z zadowoleniem przeglądała się w lustrze.

— No, jeśli chcesz, żebym była elegancka, zabieraj się do roboty. Chyba nie chcesz, żebym pobrudziła tę elegancką sukienkę.

Spektruś bez namysłu złapał za ścierkę. Tak więc dzięki drobnym reformatorskim posunięciom wróciło współpartnerstwo i harmonia.

Podglądał: Eugeniusz MLECZAK

PROGRAM 6

SHARP

BITWA MORSKA

Proponowana bitwa morska odbywa s ę na ograniczonym AKWENIE: ma on 25 PÓL uszeregowanych w 5 wierszy i 5 kolumn. (Rys. 1)

Moja pierwsza salwa została oddana w pole o współrzędnych „wiersz" 1, „kolumna" 5, czyli skrótowo (1,5)

Komputer zameldował pudło.

Moja druga salwa ma współrzędne (4,4). Komputer: W CELU ZATOPIONY.

Mam satysfakcję, ponieważ zniszczyłem pierwszy cel. Prócz tego dostałem cenną informację: ponieważ cel został „zatopiony", zatem wokół niego są obszary, w które nie warto celować.

Moja trzecia salwa wywołała odpowiedź „w celu nie tonie" (2,2).

Mam drugie trafienie i bardziej niż istotną informację: ponieważ „nie tonie", zatem jest to „duża jednostka" i należy ją ostrzelać z zachodu, ze wschodu, z północy lub z południa.

Ja w moich rozgrywkach posługuję się kratkowanym „akwenem", rejestruję każdą salwę i starannie planuję współrzędne kolejnej salwy.

Zdaję sobie sprawę, że taki mały akwen może być zbyt skromnym „teatrem wojny". Łatwo dokonać modyfikacji dla DUŻEGO AKWENU (100 pól, 10 wierszy i 10 kolumn, 40 celów i 50 salw).

Przed każdą bitwą generator liczb losowych rozmieszcza cele. Są one „nieznane" w czasie bitwy. Ich lokalizację, oraz raport walki poda meldunek „po bitwie"

Program wywołuje RUN „B".

Życzymy rozgromienia armady.

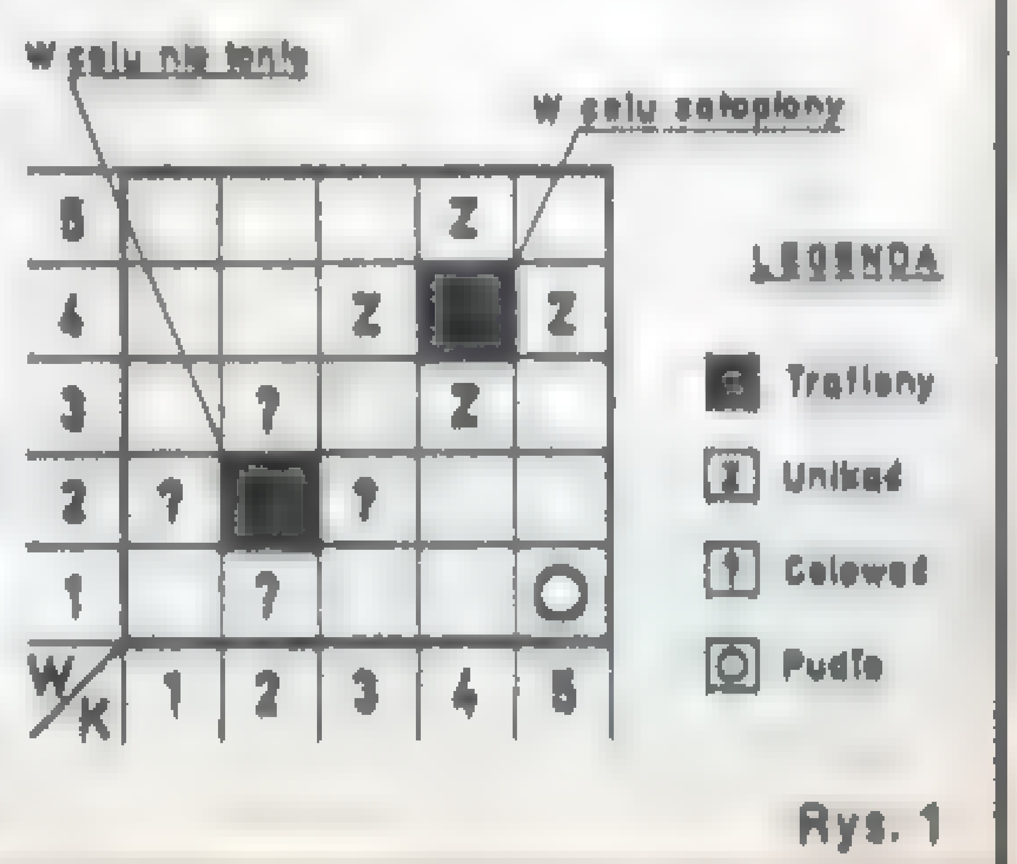
```
1050:"B":CLEAR :
     WAIT :USING
     :DIM T(6,6)
1060:FOR L=1TO 10
1070:W=RND 5:K=
     RND 5
1080:IF T(W,K)=1
     GOTO 1070
1090:T(W,K)=1
1100:NEXT L:PRINT
     "BITWA MORSK
     A"
1110:FOR L=1TO 10
1120:INPUT "CELUJ
      (1,2,3,4,5)
      W=";W:B=0
1130:W=INT W:W=
     ABS W:IF W<1
     OR W>5LET B=
     1
1140:INPUT "CELUJ
      (1,2,3,4,5)
      K=";K:WAIT
     ▒
1150:K=INT K:K=
     ABS K:IF K<1
     OR K>5OR B=1
     THEN PRINT "
     1,2,3,4,5":
     BEEP 1,100,5
     00:GOTO 1120
1160:IF T(W,K)=0
     THEN PRINT "
     P U D L O":
     BEEP 3,100,2
     00:GOTO 1230
1170:PRINT "W CEL
     U":CURSOR 7:
     Q=Q+1:C=0:D=
     0:T(W,K)=-1
1180:IF T(W-1,K)=
     1OR T(W+1,K)
     =1LET C=1
1190:IF T(W,K+1)=
     1OR T(W,K-1)
     =1LET D=1
1200:IF (C+D)>0
     THEN PRINT "
     NIE TONIE":
     GOTO 1220
1210:PRINT "ZATOP
     IONY"
1220:BEEP 10,50,1
     00
1230:WAIT :PRINT
     "SALWA";L;"
     TRAFIEN";Q
1240:CLS :NEXT L:
     WAIT :PRINT
     "PO BITWIE"
1250:FOR W=1TO 5
1260:FOR K=1TO 5
1270:IF T(W,K)=0
     GOTO 1310
1280:Z$="("+STR$
     W+","+STR$ K
     +")"
1290:IF T(W,K)=-1
     THEN PRINT "
     ZATOPIONY ";
     Z$:GOTO 1310
1300:PRINT "WALCZ
     Y ";Z$
1310:NEXT K
1320:NEXT W:END
```

Wydruk „Przykładu" dotyczy innego przypadku niż przedstawiony na rysunku

```
PRZYKLAD


PO BITWIE
ZATOPIONY (1,1)
WALCZY (1,5)
ZATOPIONY (2,3)
WALCZY (2,4)
ZATOPIONY (3,2)
ZATOPIONY (3,3)
WALCZY (3,4)
WALCZY (4,1)
WALCZY (4,5)
WALCZY (5,3)


JANUSZ MILLER
SHARP PC-1500
```



Rys. 1

Pocztowa giełda

Rozwiązanie krzyżówki z „IKS-a" nr 8

Hasło brzmi: „DO SIEGO ROKU". Bony pieniężne (1000 zł) wylosowali: Garlak Piotr — Legnica, Jolanta Mycek — Ruda Śl., Beata Konieczna — Leszno, Mariusz Kordys — Rzeszów, Dariusz Tokarew — Złotów. Nagrody książkowe otrzymują: Dorota Masłowska — Aleksandrów Kuj., Magdalena Grześk — Bydgoszcz, Mirosław Cwalina — Łomża, Tomasz Malus — Ostrołęka, Aneta Czartoryska — Zambrów, Jan Cholewa — Ustroń, Tomasz Radziewicz — Braniewo, Dorota Tyszka — Tychy, Andrzej Kwiatkowski — Szczecin, Sławomir Czerwiecki — Bytom.

Stawką w tej grze jest komputer

Aby złamać szyfr, trzeba odnaleźć klucz. Oto zapis, który należy odczytać:

```
QYs 2*)"zNr)x$5s
dU!K1r!#zN"#o!7N
dPnJ$$Yu4;"/ PsN
d0pU+$zT*"!-yqcN
```

Przypominamy: obowiązują znaki ASCII zastosowane na Sharpie. Klucz jest datą wydarzenia historii współczesnej zapisaną w międzynarodowym systemie:

DATA: RRRR. MM. DD.

Jakie to wydarzenie? Szukajcie go w naszym stuleciu, dodajmy, że suma cyfr roku, w którym do niego doszło, wynosi 19.

O tym, jak łamać szyfry pisaliśmy w poprzednim numerze „IKS-a" (nr 1/87). A teraz do dzieła, rozwiązanie konkursu już w maju.

Poszukuję książki:

1. Układy scalone serii UCA 64/VC 74. Parametry i zastosowania — Włodzimierz Sasal WKL Warszawa 1985.
2. Mikrokomputer — elementy, budowa, działanie Andrzej Rydzewski, Krzysztof Socha. Sigma Warszawa 1985.
3. Mikrokomputery — programowanie w języku Basic. Bogdan Frelek, Andrzej Lewandowski. Sigma Warszawa 1986.

JANUSZ JANASIK, ul. Zawadzkiego 22/68 63-100 Śrem.

Giełda	
Commodore 128 + magnetofon	340 tys. zł
Commodore 128 + joystick + stacja dyskietek	500 tys. zł
Commodore 16	65 tys. zł
Commodore 16 + magnetofon 1531	100 tys. zł
Commodore Plus 4 + magnetofon + 4 programy	150 tys. zł
ZX Spectrum 128	175 tys. zł
ZX Spectrum 48 KB + magnetofon + joystick	195 tys. zł
Sharp MZ-700	160 tys. zł
Sharp MZ-721	155 tys. zł
Drukarka Serial 8056	115 tys. zł
Drukarka Seikosha GP 505	105 tys. zł
Drukarka Seikosha GP 500	180 tys. zł
Drukarka MPS 802	220 tys. zł
Atari — kaseta	800 — 1500 zł
Atari — kaseta zawierająca 5 gier i 3 programy kopiujące	1700 zł
Atari — gry	100 — 300 zł/szt.
Spectrum — kaseta	1200 zł
Spectrum — gry	100 — 300 zł/szt.
Spectrum — programy użytkowe	od 150 zł/szt.
Commodore — gry, programy użytkowe	150 — 300 zł/szt.
Studio SYSTEM — W-wa Wola	
Schneider Amstrad CPC 6128	885 tys. zł
Schneider Amstrad PCW 8256	1 200 tys. zł
Schneider Amstrad PCW 8512	1 540 tys. zł
Komputer kompatybilny XT: — 640 KB RAM — 2 × FD 360 KB — 20 MB HD — Hercules — RS 232	5 400 tys. zł
Komputer kompatybilny AT: — 640 KB RAM (512) — 20 MB HD — 1,2 MB FD (5 360 KB) — (Coprocessor 80287) — Hercules (EGA) — RS 232	11 — 13 mln zł
Drukarki: — SG 15 — 15 XI — SG 10 — 10 XI — NL 10	 988 tys. zł 920 tys. zł 700 tys. zł 630 tys. zł 840 tys. zł
Manipulatory kulowe (myszki) do IBM	od 280 tys. zł
Interface'y KEMPSTON	10 200 zł
Interface'y SINCLAIR	10 620 zł
Dyskietki 5.25" 2D	1 071 zł
Dyskietki 5.25" 2D firmowe	3 500 zł
Dyskietki 5.25" 2HD	6,5 — 9,5 tys. zł
Dyskietki 3"	6,2 — 8,4 tys. zł
Zestaw programów SPECTRUM (min 8)	1 600 zł
Bogata biblioteka programów ATARI	400 zł/szt.

Krzyżówka nr 2



POZIOMO: A-1) Może być na przykład aperturowa, dualna, indeksowa itp. A-7) Sekwencja rozkazów, która została celowo określona przez programistę i jest wykonywana przez procesor w sposób powtarzalny, aż do momentu spełnienia założonego warunku B-5) Jednostka gęstości zapisu na magnetycznym nośniku danych C-1) Określona porcja np. promieniowania C-7) Uniwersalny język programowy stosowany w zagadnieniach ekonomicznych i zarządzania D-5) Jedno z wcieleń Janusza Gajosa E-1) Robocza nazwa komputerów „jednolitego systemu" E-8) Grecki Mars F-4) Jednostka oporności elektrycznej F-7) Stała część nazwy systemów operacyjnych dla komputerów rodziny Riad lub firmy IBM G-1) Skrótowa nazwa angielskiego specjalistycznego systemu przetwarzania danych przeznaczonych do informowania kierownictwa G-8) Część ciała ludzkiego H-5) Potrawa na ratunek I-1) Osoba żyjąca przy klasztorze nie składająca ślubów I-7) Odwrotność różniczki J-5) Imię żeńskie K-1) Metropolia Jordanii K-7) W starożytnym Rzymie pieśń żałobna

PIONOWO: A-1) Urządzenie zmieniające symbole pewnego alfabetu na symbole innego alfabetu G-1) Obraz religijny w sztuce Wschodnio-Chrześcijańskiej E-2) Skrótowa nazwa firmy, będącej największym producentem sprzętu komputerowego na świecie A-3) Pokaz, parada G-3) Mahometanizm E-4) Ogólna nazwa systemów operacyjnych dla komputerów rodziny RIAD lub firmy IBM A-5) Starożytne liczydła H-5) Pistolet maszynowy skonstruowany przez Shepparda i Turpina D-6) Przyjęta z języka angielskiego nazwa spójnika LUB G-6) Klasyczny japoński balet dramatyczny A-7) Nazwa systemu informatycznego przeznaczonego do kierowania produkcją H-7) Nazwa pakietu programów przeznaczonych dla komputerów serii ODRA 1300 oraz ICL-1900 i stosowanych do kontroli zapasów magazynowych E-8) Stosowana w języku rosyjskim skrótowa nazwa zautomatyzowanego systemu zarządzania A-9) Środki transportowe G-9) Może być operacyjny (mnożenia, sumujący itp.) E-10) Skrót oznaczający komputer Hybrydowy A-11) Najpopularniejszy instrument muzyczny w starożytnej Grecji, Atrybut Euterpe G-11) Lęk, strach przed czymś

HASŁO: (3-B, 3-I, 3-C, 6-E, 7-F)(4-A, 8-C)(10-I, 9-J, 2-K, 7-A, 8-G, 9-A, 8-K, 3-A)(1-C, 11-C, 2-A)(7-C, 3-D, 5-J, 2-J, 1-G, 10-E)

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji do końca marca 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI



SAM 3001 AT produkowany jest przez koreańską firmę Samsung Semiconductor Telecommunications Co. Ltd. Jest kompatybilny z komputerem IBM PC AT. Pracuje w oparciu o mikroprocesor 80286 z możliwością dołączenia dodatkowego procesora arytmetycznego 80287. Pamięć operacyjna ma 640 Kbajtów z możliwością powiększenia jej do 16 Mbajtów. Pojemność pamięci na dyskach elastycznych 1,2 Mbajta i na dyskach sztywnych 20 Mbajtów z możliwością powiększenia do 80 Mbajtów. Cena podstawowa 4395 dolarów.



Rys. Michał Przybyłowski

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8501. Nr. ind. 361682. K-62

3

INFORMATYKA
KOMPUTERY
SYSTEMY

Dodatek „Żołnierza Wolności" ISSN 0860-2094 Cena 50



W numerzie:

- W szponach ATARI (2) Praca w trybie znakowym — str. 3
- Kreska CPC 464 (2) — str. 7
- Grafika (Amstrad) — str. 8
- Grafika Commodore 64 (2) — str. 9
- Mikrokomputery IBM PC — str. 11
- Biblioteka — str. 12
- „Home, Office, Personal Computer 1987" — Tradycja to nasz obowiązek — str. 13 — Na ekranie „Komputera'87" — str. 14 — Fotoreportaż — str. 16—17
- Udawanie rzeczywistości — str. 18
- Mikroserwis — str. 20
- Wyszukiwanie informacji w zbiorze — str. 21
- Sztuki i sztuczki (5) — str. 22
- Szkoleniowy system graficzny — Oprogramowanie znacznika świetlnego — str. 24
- Sami piszemy gry — str. 27
- W naszym komputerlandzie — str. 29
- OLIVETTI — str. 29
- Liga Myślących — str. 31
- Notowania komputerowe — str. 31

Podobnie jak przed rokiem parada komputerów wzbudziła duże zainteresowanie. Ponad siedemdziesiąt firm, w tym znaczące w świecie marki, ściągnęło do dusznych sal Pałacu Kultury i Nauki fachowców, i pasjonatów nowoczesnej techniki. A było na co popatrzeć.

Dla handlowców oczywiście niewielkie mają znaczenie okrzyki zachwytu, bo o nie w naszej sytuacji raczej nietrudno, ale konkretne kontrakty, których realność jest sprawą być, albo nie być na polskim rynku.

Oczywiście „być". I jeśli nawet teraz trudno się z nami handluje, szczególnie tak nowoczesną techniką, którą nie zawsze jesteśmy w stanie skonsumować, to przecież w przyszłości, dla światowych potentatów możemy być dobrym partnerem. Tymczasem komputerowe firmy muszą zadbać, aby ich przyszły klient miał za co kupić ich wyroby, czyli po prostu wcześniej trzeba mu dolary do kieszeni włożyć, żeby je później wyciągnąć. Z tą drugą czynnością nie ma kłopotów — gorzej z pierwszą.

Z kim handlować warto? Na to trzeba odrobiny doświadczenia i przede wszystkim wiedzy. Z dnia na dzień przybywa nam firm komputerowych, oferujących sprzęt i oprogramowanie. Handlowanie oprogramowaniem jest w wielu przypadkach czystym piractwem, w którym interes polega na oferowaniu cudzej pracy, za duże pieniądze, z jednoczesnym pozbawieniem wszelkich praw autorskich i majątkowych. Ot, po prostu kto pierwszy znajdzie dobry program przerzyna go natychmiast i puszcza w obieg.

Oczywiście dotyczy to nie tylko oprogramowania profesjonalnego, którego ceny sięgają często setek tysięcy, ale również i naszych programów drukowanych na łamach „IKS-a". Spotkać je można na komputerowych giełdach, starannie nagrane na kasety w niezmienionej wersji, albo przerobione na inny dialekt BASIC-a. W tej sytuacji autor jest nieważny — tu liczy się pieniądz.

Niestety dla wielu zwiedzających liczy się przede wszystkim efekt: błysk kamery, obficie rozdawane kolorowe foldery i drobne upominki.

Atrakcją komputerowych pokazów były oczywiście gry i kolorowe grafiki. Nikogo to nie dziwi. Tak jest na całym świecie. Taki jest właśnie pierwszy kontakt z komputerem. A że zainteresowanie tą techniką u nas nie spada — możemy sądzić, że za kilka lat komputer stanie się nie tylko kosztowną zabawką, ale powszechnie stosowanym narzędziem pracy. Czego i nasza redakcja życzyłaby sobie.

W. CETERA

Uwaga ośmioklasiści

Wojskowe licea ogólnokształcące czekają

Uczniowie klas ósmych muszą zdecydować o dalszej nauce. Jedną z ciekawszych możliwości są dla nich wojskowe licea ogólnokształcące. Aktualnie oczekuje na swoich przyszłych uczniów pięć wojskowych liceów w: Częstochowie, Lublinie, Olsztynie, Toruniu i Wrocławiu.

Ogólnokształcące licea wojskowe realizują pełny program średniej szkoły w klasie matematyczno-fizycznej. Dodatkową atrakcją są w procesie nauczania elementy politechnizacji, komputeryzacji, cybernetyki i informatyki. Licea dysponują bowiem nowoczesną bazą sprzętu komputerowego. Nauka w szkole trwa 4 lata. Kandydatów obowiązuje egzamin wstępny z matematyki i języka polskiego oraz sprawdzian sprawności fizycznej.

Podania o przyjęcie do wybranego wojskowego liceum ogólnokształcącego zainteresowani składają w terminie do 15 kwietnia br. w najbliższej wojskowej komendzie uzupełnień.

Uczniowie w czasie nauki w liceum wojskowym otrzymują bezpłatnie: zakwaterowanie w internacie, ubiór szkolny (wyjściowy i codzienny); wyżywienie, wojskową pomoc leczniczą, podręczniki i inne materiały szkolne; przejazdy (4 razy w roku) do miejsca zamieszkania najbliższej rodziny.

W okresie letnich wakacji organizuje się dla uczniów bezpłatne, trzytygodniowe obozy szkoleniowe, w ramach których można uzyskać między innymi kartę pływacką, patent żeglarza jachtowego, prawo jazdy, specjalność skoczka spadochronowego itp.

Podstawowym problemem każdego komputera, używającego odbiornika telewizyjnego dla celów zobrazowania informacji, jest to, że wyświetlanie jest procesem dynamicznym; z tego powodu telewizor nie pamięta obrazu. W rezultacie musi go pamiętać komputer i stale wysyłać sygnał do telewizora, mówiący co jest do wyświetlania. Wysyłanie informacji jest procesem ciągłym i wymaga stałej uwagi. Z tego powodu większość mikrokomputerów ma specjalne układy, które obsługują telewizor. W Atarii funkcje te spełnia ANTIC.

W szponach ATARI (2)

Praca w trybie znakowym

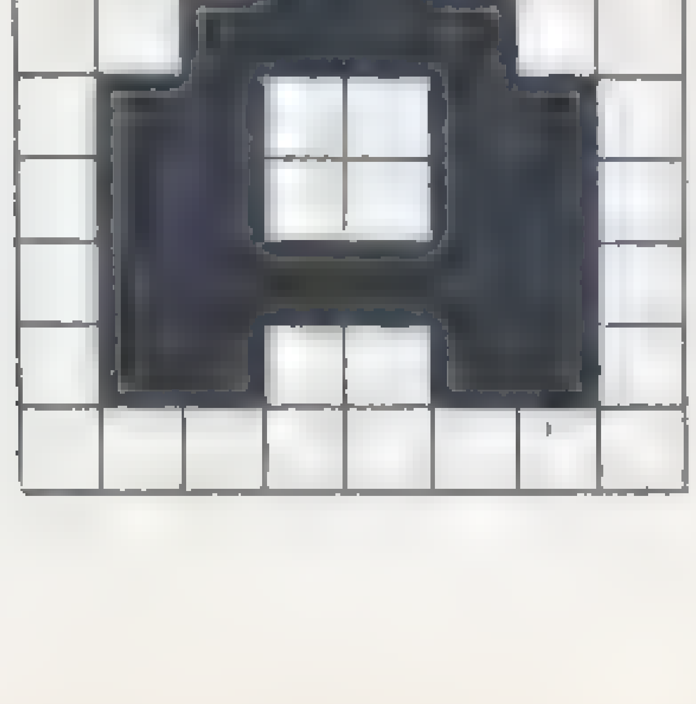
Tomasz MROWIEC
Ludwik PIELA

Mikroprocesor zapisuje informację w pamięci RAM. ANTIC stale kontaktuje się z tym obszarem pamięci, pobierając dane, które przekształca (z pomocą GTIA) na sygnały telewizyjne. Sygnały te idą do telewizora, który wyświetla informację. Pamięć ekranu odwzorowywana jest na ekranie w takiej samej kolejności, w jakiej występuje w RAM. Oznacza to, że pierwszy bajt pamięci ekranu odwzoruje lewy górny narożnik obrazu, drugi bajt odwzoruje jedną pozycję w prawo, następnie trzeci, czwarty, aż do ostatniego bajtu, który jest odwzorowany w prawym dolnym rogu ekranu.

ANTIC może pracować w trybach tekstowych lub graficznych. Najistotniejsza różnica między nimi polega na źródle danych, które tworzą obraz. W trybach graficznych dane z pamięci ekranu są bezpośrednio wyświetlane. W trybach tekstowych każdy bajt traktowany jest jako numer znaku. Na podstawie tego numeru ANTIC wybiera odpowiedni wzorzec i umieszcza go na ekranie. Dzięki temu każdemu znakowi wyświetlonemu na ekranie odpowiada jeden bajt w pamięci ekranu.

Od razu należy wyjaśnić, że numer znaku przechowywany w pamięci ekranu, nie jest kodem *ATASCII* tego znaku. Jest to po prostu numer porządkowy, który wskazuje wzorzec znaku przechowywany w pamięci ROM.

Aby zrozumieć, w jakiej postaci wzorce znaków przechowywane są w pamięci komputera, przyjrzyjmy się bliżej, w jaki sposób znak wyświetlany jest na ekranie. Wyobraźmy sobie siatkę 8x8 oczek, w której umieszczono 64 miniaturowe żarówki. Każda z nich reprezentuje elementarny punkt ekranu (pixel) i może być zapalona lub nie. Aby uzyskać kształt litery „A", muszą być zapalone żarówki jak na poniższym rysunku:



ATARI COMPUTERS

Każdy wiersz siatki może być opisany za pomocą bajtu. „Żarówka" zapalona może być oznaczona jako 1, a zgaszona jako 0. Zatem poszczególne wiersze mogą być zapisane w postaci liczb dwójkowych, jakie umieszczone są z prawej strony siatki. Na prawo od nich umieszczone są odpowiedniki dziesiętne. Ci, którzy nie mają wprawy w posługiwaniu się liczbami binarnymi, mogą przekształcić każdy wiersz w następujący sposób:

1. Powyżej każdej kolumny umieszczone są liczby — są to potęgi liczby 2
2. Jeżeli „żarówka" jest zapalona, bierzemy liczbę umieszczoną nad nią i dodajemy do zmiennej „suma". Sumujemy wszystkie zapalone „żarówki" w wierszu (np. czwarta linia „A" ma wartość 64 + 32 + 4 + 2 = 102)
3. Wykonujemy to dla ośmiu wierszy.

W ten sposób, za pomocą ośmiu bajtów możemy opisać kształt dowolnego znaku, utworzonego w powyższej siatce. W takiej właśnie postaci przechowywane są wzorce znaków w pamięci komputera. Zestaw znaków przechowywanych jest w ciągłym obszarze pamięci ROM w adresach 57344 do 58367 ($E000 do $E777 szesnastkowo): od $E000 zaczynają się znaki specjalne, interpunkcyjne i cyfry; od $E100 (57600) zaczynają się wielkie litery; od $E200 (57856) — znaki graficzne i od $E300 (58112) — małe litery. Łącznie zajęty jest obszar 1024 ($400) bajtów, 128 znaków po 8 bajtów każdy. Negatywy znaków nie są przechowywane, lecz uzyskiwane przez odpowiednie manipulowanie bajtami.

W komórkach pamięci o adresach 756 (CHBAS) i 54281 przechowywany jest adres początku obszaru, w którym przechowywane są wzorce znaków, a dokładniej, numer stronicy (256-bajtowego bloku). Dzięki temu istnieje możliwość zmodyfikowania kształtów istniejących symboli lub wręcz zdefiniowania własnego ich zestawu. Możliwe jest zatem posiadanie znaków polskich, greckich, cyrylicy lub innych specjalnych. Można, na przykład, utworzyć zestaw symboli elektronicznych, zawierający symbole tranzystorów, diod, oporników, kondensatorów i przewodów, dzięki którym możliwe byłoby kreślenie schematów elektronicznych.

Jeszcze ciekawsze efekty wynikają z możliwości zmiany zestawów znaków podczas działania programu. Można wyróżnić trzy tryby czasowe dla takiego multipleksowania zestawu znaków: wolny (rzadziej niż co 1 sek.), szybki (1/50 sek. do 1 sek.) i bardzo szybki (częściej niż co 1/50 sek.).

Wolne multipleksowanie znaków przydatne jest do „zmiany scenerii". Na przykład, program podróży w kosmosie może używać jednego zestawu znaków graficznych dla jednej planety, drugiego dla kosmosu i trzeciego dla innej planety. W miarę przemieszczania się podróżnika, program zmienia zestaw znaków dla otrzymania nowej, egzotycznej scenerii.

Szybkie przełączanie zestawów znaków może być przydatne do animacji. Efekty ruchu można uzyskać zmieniając znaki wewnątrz jednego zestawu, albo zmieniając całe zestawy znaków.

Bardzo szybka animacja cykliczna całego ekranu możliwa jest dzięki przygotowaniu kilku zestawów znaków, wykreśleniu obrazu, a następnie cyklicznemu wybieraniu zestawów. Jeśli każdy znak ma różne wcielenia w innych zestawach znaków, to przejdzie przez animowany ciąg w miarę zmian zestawów. W ten sposób ekran zapełniony obiektami może być cyklicznie przesuwany za pomocą bardzo prostej pętli.

Używanie zestawów znaków do grafiki i animacji ma wiele zalet i pewne ograniczenia. Największą zaletą jest zajmowanie małego obszaru pamięci RAM do utworzenia szczegółowego obrazu. Na przykład, w trybie zerowym możliwe jest umieszczenie na ekranie 960 znaków (24 linie po 40 znaków) i uzyskanie rozdzielczości takiej jak w trybie 8. Dla porównania, zajętość pamięci w trybie 0—960 bajtów, a w trybie 8—7680 bajtów, czyli osiem razy więcej. Obraz używający trybu 2 może dać tyle samo szczegółów i o jeden kolor więcej niż obraz w trybie 7. Dodatkowo będzie on zajmował 200 bajtów, podczas gdy w trybie 7 zajmuje 4000 bajtów. Ze względu na mniejszą objętość pamięci manipulowanie ekranem z grafiką znakową jest dużo szybsze.

Jednak grafika znakowa nie jest zbyt elastyczna. Nie można ustawić tego co chcemy w dowolnym miejscu ekranu. Ograniczenie to uniemożliwia używanie grafiki znakowej w pewnych zastosowaniach. Pozostaje jednak wiele zastosowań graficznych, w których potrzebujemy wyświetlić tylko ograniczoną liczbę wcześniej zdefiniowanych kształtów w ustalonych miejscach. W tych wypadkach grafika znakowa jest bardzo wygodna.

Interesującym przykładem wykorzystania grafiki znakowej jest program „Eastern front 1941", w którym uzyskano bardzo ciekawe efekty dzięki zdefiniowaniu odpowiednich znaków graficznych terenu.

Tworzenie własnych znaków

Komputery Atari, jak zresztą wiele innych, odznaczają się interesującą i pożyteczną własnością — możliwością tworzenia własnych znaków. Nie ma jednak do tego żadnych specjalnych instrukcji. Wszystko trzeba wykonać za pomocą **PEEK** i **POKE**, instrukcji umożliwiających pobieranie i modyfikowanie zawartości pamięci, w bardzo uciążliwy sposób.

Tworzenie własnych znaków rozpoczynamy od zaprojektowania ich kształtu. Można to zrobić za pomocą ołówka, kalkulatora i kilku arkuszy papieru milimetrowego. W celu utworzenia znaku w odpowiedni sposób wypełniamy siatkę 8 x 8 oczek, w której każde oczko reprezentuje elementarny punkt ekranu. Musimy uwzględnić to, że linie poziome wyświetlane są jaśniej niż pionowe, dlatego konieczne jest pogrubienie tych ostatnich, aby stały się widoczne.

Po zaprojektowaniu kształtów znaków musimy je przetłumaczyć na liczby, które są bardziej zrozumiałe dla komputera. Sposób zamiany kształtu znaku na liczby opisany został powyżej. Następnie zestawiamy liczby w instrukcje **DATA**. Gdy już to zrobimy, musimy podmienić liczby wybranego znaku naszymi. Nie jest to takie proste. Zestaw znaków przechowywany jest w pamięci ROM i nie może być zmieniany. Ale i na to jest rada. Wystarczy skopiować ten obszar do RAM, gdzie może być zmieniany, i poinformować komputer (a dokładnie ANTIC) o miejscu położenia nowego zestawu znaków. W tym celu musimy znaleźć odpowiednie miejsce w pamięci do ich przechowywania.

Jednym z rozwiązań jest umieszczenie wzorców znaków w górnych adresach pamięci RAM dostępnej dla użytkownika, a następnie przekonanie komputera, że ma mniej pamięci niż w rzeczywistości. Chroni to zestaw znaków przed przypadkowym uszkodzeniem. W tym celu używamy komórki o adresie 106 (RAMTOP), która przechowuje ilość dostępnych stronic pamięci (bloków po 256 bajtów). Atari 800 XL zawsze umieszcza program wyświetlania (display list) i dane do wyświetlania bezpośrednio poniżej RAMTOP. Jeśli zmienimy zawartość komórki 106, system operacyjny przesunie program wyświetlania i dane do niższych adresów RAM. Zmniejsza to obszar dostępny dla programu, lecz uwalnia część pamięci powyżej nowej wartości RAMTOP. Obszar ten nie będzie używany przez program ani system operacyjny i może być traktowany jako „zarezerwowany".

Poniższy program przedstawia, w jaki sposób można zmniejszyć RAMTOP o cztery strony (1024 bajty). Należy pamiętać o użyciu rozkazu GRAPHICS bezpośrednio po zmianie zawartości komórki 106, w celu przesunięcia programu wyświetlania i danych.

```
10 ? "WOLNE RAM = ";FRE(0)
20 RAMTOP=PEEK(106):? "RAMTOP =
 ";RAMTOP;" STRONIC":? "OSTATNI
 ADRES = ";RAMTOP*256
30 FOR W=1 TO 2000:NEXT W
40 MALERAM=RAMTOP-4
50 POKE 106,MALERAM
60 GRAPHICS 0
70 ARAMTOP=PEEK(106)
80 ? "AKTUALNIE WOLNE RAM = ";F
RE(0)
85 ? "STARE RAMTOP = ";RAMTOP;"
 STRONIC"
90 ? "NOWE RAMTOP = ";ARAMTOP;"
 STRONIC":? "OSTATNI ADRES = ";
ARAMTOP*256
100 ? "ZAREZERWOWANY OBSZAR ZAC
ZYNA SIE OD ";ARAMTOP*256+1
```

Używanie tak zarezerwowanego obszaru jest wygodne, ponieważ zawartość komórki 106 odtwarzana jest tylko w dwóch wypadkach: włączenia zasilania i naciśnięcia klawisza RESET. Należy go jednak używać ostrożnie, a w chwili rezerwowania tego obszaru uwzględnić:

1) każde użycie rozkazu GRAPHICS zeruje ekran oraz pierwsze 64 bajty powyżej RAMTOP,
2) każde użycie rozkazu CLEAR (lub PRINT CHR$(125)) zeruje również pierwsze 64 bajty powyżej RAMTOP.

Dla bezpiecznej pracy najlepiej będzie pozostawić wolne pierwsze 800 bajtów powyżej RAMTOP. Wymaga to zarezerwowania większego obszaru pamięci, niż to jest niezbędne, a tym samym zmniejszenia obszaru dostępnego dla programu.

Innym rozwiązaniem jest umieszczenie znaków w dolnych adresach pamięci RAM, a następnie przekonanie systemu operacyjnego, że wolna pamięć zaczyna się w innym miejscu. W tym celu używamy wskaźnika początku obszaru wolnej pamięci — MEMLO. Przechowywany on jest w komórkach o adresach 743 i 744. Wskazuje adres pierwszej wolnej komórki RAM dostępnej dla programu. Bez DOS-u MEMLO wskazuje 1792, a z DOS-em 7420. Jego zawartość aktualizowana jest po włączeniu zasilania lub naciśnięciu klawisza RESET.

Możemy zmienić zawartość MEMLO w celu zarezerwowania obszaru pamięci poniżej programu. Musi to być jednak zrobione na samym początku, zanim BASIC przejmie sterowanie, ponieważ rozpoczyna on przechowywanie wprowadzanego programu od najniższych adresów. Wymaga to użycia prostego podprogramu w języku maszynowym, który działa następująco: przejmuje kontrolę nad komputerem i wstawia nową wartość do MEMLO, a następnie ponownie inicjuje BASIC — tak jak po naciśnięciu klawisza RESET. Kontrolę nad komputerem przejmuje BASIC nieświadomy, że ma teraz mniej wolnej pamięci. Poniżej zamieszczamy program, który umożliwia zmianę zawartości MEMLO.

```
10 DIM SUBR$(24)
20 FOR I=1 TO 24:READ PRG
30 SUBR$(I)=CHR$(PRG):NEXT I
40 ? "Ile bajtow chcesz zarezer
wowac ";:INPUT REZ
50 HI=INT(REZ/256):LO=REZ-256*H
I
60 SUBR$(6,6)=CHR$(LO):SUBR$(14
,14)=CHR$(HI)
70 Z=USR(ADR(SUBR$))
80 DATA 24,173,231,2,105,0,141
90 DATA 231,2,173,232,2,105,0,1
41
100 DATA 232,2,169,0,133,8,76,0
,160
```

W celu określenia adresu, od którego zaczyna się zarezerwowany obszar, musimy wprowadzić przed uruchomieniem tego programu następującą linię

?PEEK(743) + 256 * PEEK(744)

Otrzymana liczba może być używana jako adres przeznaczenia dla dowolnych danych, które chcemy przechować w zarezerwowanym obszarze.

Najczęściej jednak obywamy się bez specjalnych sztuczek i umieszczamy własne znaki w obszarze pamięci przeznaczonym na program, zakładając optymistycznie, że nie zajmie pamięci całkowicie. Dlatego najchętniej umieszczamy znaki tuż przed pamięcią ekranu. Istnieje jednak pewna komplikacja. Wielkość pamięci ekranu zależy od używanego trybu graficznego, od 272 bajtów w GRAPHICS 3, do prawie 8000 bajtów w GRAPHICS 8. Możemy wykorzystać zawartość komórki 106 (RAMTOP) i cofnąć się wystarczająco daleko, aby zostawić miejsce na pamięć ekranu

Zatem dla znalezienia miejsca do przechowania zestawu znaków w trybach 0, 1, 2, 3, 4 i 5 cofnijmy się o 4 stronice od RAMTOP — miejsce na pamięć ekranu, a następnie o 4 dodatkowe do przechowania znaków, razem 8. GRAPHICS 6 będzie wymagać 12, GRAPHICS 7 — 20, a GRAPHICS 8 — 36 stron pamięci.

Po wybraniu odpowiedniego obszaru pamięci możemy przesłać zestaw znaków z ROM do RAM. Mogą to wykonać następujące instrukcje

```
10 CHBAS=57344
20 CHSET=(PEEK(106)-8)*256
30 FOR I=0 TO 1023
40 POKE CHSET+I,PEEK(CHBAS+I)
50 NEXT I
60 SUBR$(6,6)=CHR$(LO):SUBR$(14
,14)=CHR$(HI)
70 Z=USR(ADR(SUBR$))
80 DATA 24,173,231,2,105,0,141
90 DATA 231,2,173,232,2,105,0,1
41
100 DATA 232,2,169,0,133,8,76,0
,160
```

Przesłanie w Basic'u zajmuje dosyć dużo czasu, dla 1024 bajtów trwa około 15 sekund. Dlatego lepiej jest użyć znacznie szybciej działającej procedury w języku maszynowym. Na przykład podobnej do poniższej:

```
10 DIM BYTE$(80)
20 CHSET=(PEEK(106)-8)*256
30 FOR L=1 TO 32:READ B:BYTE$(L
,L)=CHR$(B):NEXT L
40 DATA 104,104,133,213,104,133
,212
50 DATA 104,133,215,104,133,214
,162
60 DATA 4,160,0,177,212,145,214
70 DATA 200,208,249,230,213,230
,215
80 DATA 202,208,240,96
90 Z=USR(ADR(BYTE$),224*256,CHS
ET)
100 POKE 756,CHSET/256
```

Kolejna linia w każdym z powyższych programów powinna informować ANTIC, gdzie umieściliśmy zestaw znaków:

60 POKE 756. CHSET/256

Normalnie w komórce o adresie 756 przechowywana jest wartość 224 (224 X 256 = 57344 — adres standardowego zestawu znaków w ROM).

Teraz, mając już zestaw znaków w RAM, możemy przystąpić do ich modyfikacji. W tym celu wybieramy znak, który chcemy zmienić, wyznaczamy jego adres i podmieniamy odpowiednie bajty. Po zmianie kształtów wszystkich interesujących nas znaków musimy przechować nowy zestaw znaków w pamięci zewnętrznej, dla wykorzystania w przyszłości do innych programów.

Jak widać z przedstawionego opisu, tworzenie własnych znaków jest czynnością niełatwą i bardzo żmudną. Na szczęście, istnieją specjalne programy, zwane edytorami kształtu znaków lub generatorami znaków, które pomagają nam zmieniać wygląd znaków, pokazują, jak będą wyglądać, wykonują niezbędne obliczenia i przechowują cały zestaw znaków na taśmie lub dyskietce. Do najbardziej znanych należą: The Next Step z Online, Instedit z APX, Font Edit z Code Works czy Ultrafont z Antic. Ze względu na kłopoty związane ze zdobyciem tych programów proponujemy własną wersję generatora znaków. Być może nie ma on wszystkich funkcji oferowanych przez wymienione, ale chcieliśmy, aby był napisany w języku Basic i nie był zbyt trudny do wyprowadzenia

Generator znaków

Prezentowany program jest ilustracją materiału dotyczącego generowania własnych znaków. Umożliwia on zmodyfikowanie kształtu dowolnego, wybranego przez nas znaku lub zaprojektowanie go od nowa. Działanie rozpoczynamy od wybrania znaku, którego postać chcemy zmienić. W tym celu naciskamy klawisz SELECT i za pomocą manipulatora przesuwamy kursor na wybrane miejsce prezentowanego repertuaru znaków. Po naciśnięciu przycisku manipulatora znak, w dużym powiększeniu, pojawi się w dolnej części ekranu. W przedstawionej siatce możemy zmienić dowolne pole. Naciśnięcie przycisku zmienia jego stan na przeciwny. Jednocześnie, z prawej strony powiększonego znaku widoczna jest jego postać o wymiarach dwa razy większych niż normalnie (tak jak w GRAPHICS 2). Jeżeli uznamy, że uzyskany kształt znaku jest odpowiedni, naciskamy klawisz START. Spowoduje to zamianę znaku.

Po dokonaniu wszystkich modyfikacji musimy przechować zestaw znaków w pamięci zewnętrznej. W tym celu naciskamy klawisz OPTION. Na ekranie pojawi się pytanie o nazwę zbioru. Jeśli chcemy zapisać nowe znaki na taśmie magnetycznej, to podajemy **„C:"**, w wypadku zapisu na dyskietkę podajemy **„D: NAZWA ZB. LST"**. Zapis ma postać linii DATA o numerach od 32000 do 32042.

Jeśli chcemy zacząć pracę od nowa lub wygenerować inny zestaw znaków, to wystarczy nacisnąć klawisz HELP. Spowoduje to odtworzenie standardowego zestawu znaków i przejście do początku programu.

Zapisany w pamięci zewnętrznej zestaw znaków można dołączyć do własnego programu instrukcją ENTER. Włączenie tych linii do programu nie wystarcza. Należy wprowadzić te dane do tablicy znaków i przekazać komputerowi adres tej tablicy. Poniższy program, który może być włączony do własnego programu po ewentualnej zmianie numerów linii, rezerwuje obszar pamięci, niezbędny do przechowania zestawu własnych znaków oraz wpisuje do niego dane o znakach.

```
30000 MEM=PEEK(106)-8:POKE 106,
MEM-1
30010 CHACT=MEM*256
30020 RESTORE 32000
30030 FOR I=0 TO 1023
30040 READ D:POKE CHACT+I,D
30050 NEXT I
30060 POKE 756,MEM
30070 RETURN
```

Dla lepszego zrozumienia działania programu oraz ewentualnej jego rozbudowy podajemy w skrócie, co realizują poszczególne linie:

Linie 10—20 — rezerwowanie obszaru pamięci

dla zestawu znaków i grafiki P/M.

- Linie 60—80 — przygotowanie podprogramów w języku wewnętrznym.
- Linie 100—120 — ustawienie parametrów dla grafiki P/M.
- Linie 140—190 — rozpoznawanie funkcji.
- Linie 200—220 — przepisanie zestawu znaków z ROM do RAM
- Linie 300—690 — podprogram wyboru znaku do poprawiania.
- Linie 700—795 — ustawienie kursora w środku ekranu.
- Linie 800—860 — podprogram przemieszczania kursora
- Linie 900—940 — podprogram rozpoznawania położenia kursora na ekranie
- Linie 1000—1090 — podprogram wyświetlania powiększonego znaku.
- Linie 1100—1195 — podprogramy w języku wewnętrznym.
- Linie 1500—1550 — podprogram wyświetlania strony tytułowej programu
- Linie 1600—1700 — podprogram wyświetlania instrukcji programu.
- Linie 2000—2090 — podprogram umieszczania napisu w środku ekranu
- Linie 2100 — podprogram wygaszania kursora
- Linie 3000—3170 — podprogram zapisu zestawu znaków w pamięci zewnętrznej.

Generator znaków

```
4 CLOSE #6:OPEN #6,4,0,"S:"
5 CLR :DIM A$(10),BUF$(32),UP$(21),DOL$(21)
10 MEM=PEEK(106)-16
20 AD=PEEK(106)-8:WSK=0:POKE 106,MEM-1
30 CHACT=MEM*256
40 GOSUB 1500:REM SKOK DO STRONY TYTULOWEJ
50 RESTORE
60 FOR L=1 TO 32:READ PGM:POKE 708,L:BUF$(L,L)=CHR$(PGM):NEXT L
70 FOR L=1 TO 21:READ PGM:POKE 708,L:UP$(L,L)=CHR$(PGM):NEXT L
80 FOR L=1 TO 21:READ PGM:POKE 708,L:DOL$(L,L)=CHR$(PGM):NEXT L
85 GRAPHICS 0
90 GOSUB 1600
95 GOSUB 2100
100 POKE 54279,AD:POKE 53277,3:POKE 559,46
110 START=AD*256+512+68:POKE 53248,142
120 POKE 704,64:POKE 53256,0:POKE 752,1
130 GOSUB 200
140 KF=PEEK(53279):KH=PEEK(732)
150 IF KF=7 AND KH=0 THEN 140
160 IF KH=17 THEN POKE 732,0:GOSUB 1600:GOSUB 200
170 IF KF=3 THEN GOSUB 3000
180 IF KF=5 THEN GOTO 300
190 GOTO 140
200 REM PRZEPISANIE TABLICY ZNAKOW
210 Z=USR(ADR(BUF$),224*256,CHACT)
220 POKE 756,MEM:RETURN
300 REM WYBOR ZNAKU I POPRAWA
310 ? "}":POSITION 4,1:FOR L=0 TO 26
320 ? CHR$(L);:NEXT L
330 POSITION 4,2:FOR L=32 TO 64:? CHR$(L);:NEXT L
340 POSITION 4,3:FOR L=65 TO 96:? CHR$(L);:NEXT L
350 POSITION 4,4:FOR L=97 TO 124:? CHR$(L);:NEXT L
360 GOSUB 700
370 A=STICK(0):GOSUB 800
380 IF STRIG(0)<>0 THEN 370
390 GOSUB 900
400 B=M
410 IF B>127 THEN B=B-127
420 IF B<32 THEN B=B+64:GOTO 440
430 IF B<97 THEN B=B-32
440 POSITION 10,10:FOR I=0 TO 7
450 W=PEEK(CHACT+I+B*8)
460 POKE START+I,W
470 KK=256
480 FOR K=0 TO 7
490 KK=KK/2
500 IF W<KK THEN ? "+";:GOTO 520
510 ? " ";:W=W-KK
520 NEXT K
530 POSITION 10,11+I
540 NEXT I
550 GOSUB 700
560 A=STICK(0):GOSUB 800
565 IF PEEK(53279)=6 THEN 640
570 IF STRIG(0)<>0 THEN 560
580 GOSUB 900
590 IF M=160 THEN A$="+"
600 IF M=ASC("+") THEN A$=CHR$(160)
610 POSITION XP,YP:? A$;
620 GOSUB 1000
630 IF PEEK(53279)<>6 THEN 560
640 FOR I=0 TO 7
650 W=PEEK(START+I)
660 POKE CHACT+I+B*8,W
670 NEXT I
680 GOSUB 700
690 GOTO 140
700 REM KURSOR W SRODKU
720 POKE 53249,142
730 X=142:Y=80:POKE 705,88
740 RESTORE 1200
750 AD1=AD*256+640
760 FOR I=AD1 TO AD1+127:POKE I,0:NEXT I
770 FOR I=AD1+Y TO AD1+Y+7
780 READ A:POKE I,A
790 NEXT I
795 RETURN
800 REM RUCH KURSORA
810 IF (A=10 OR A=9 OR A=11) THEN X=X-1
820 IF (A=6 OR A=7 OR A=5) THEN X=X+1
830 POKE 53249,X
840 IF (A=10 OR A=14 OR A=6) THEN U=USR(ADR(UP$),AD1+Y):Y=Y-1
850 IF (A=9 OR A=13 OR A=5) THEN U=USR(ADR(DOL$),AD1+Y):Y=Y+1
860 RETURN
900 REM LOKALIZACJA KURSORA
910 XP=(X-46)/4:YP=(Y-16)/4
920 A$=""
930 LOCATE XP,YP,M
935 POSITION XP,YP:? CHR$(M)
940 RETURN
1000 REM ZMIANA ZNAKU P/M
1010 FOR I=0 TO 7
1020 KK=256:W=0
1030 FOR J=0 TO 7
1040 LOCATE 10+J,10+I,M
1050 KK=KK/2:IF M=160 THEN W=W+KK
1060 NEXT J
1070 POKE START+I,W
1080 NEXT I
1090 RETURN
1100 DATA 104,104,133,213,104,133,212
1110 DATA 104,133,215,104,133,214,162
1120 DATA 4,160,0,177,212,145,214
1130 DATA 200,208,249,230,213,230,215
1140 DATA 202,208,240,96
1150 DATA 104,104,133,204,104,133,203
1160 DATA 160,1,177,203,136,145,203
1170 DATA 200,200,192,10,208,245,96
1180 DATA 104,104,133,204,104,133,203
1190 DATA 160,9,177,203,200,145,203
1195 DATA 136,136,192,255,208,245,96
1200 DATA 0,0,24,60,24,0,0,0
1500 GRAPHICS 2+16
1510 A$="GENERATOR":P=3
1520 GOSUB 2000
1530 A$="ZNAKOW":P=6
1540 GOSUB 2000
1550 RETURN
1600 REM INSTRUKCJA
1610 ? "}":POSITION 10,5:? "  M E N U      ":? :? :? :POKE 752,1
1620 ? "SELECT - wybor znaku"
1630 ? "START  - akceptacja znaku"
1640 ? "HELP   - powrot do poczatku"
1650 ? "OPTION - zapis znakow na kasete"
1660 ? "               lub dysk"
1690 GOSUB 2100
1700 RETURN
2000 Z=INT(0.5+(20-LEN(A$))/2)
2010 FOR I=1 TO LEN(A$)
2020 FOR J=19 TO Z STEP -1
2030 SOUND 0,J*2*I,10,10
2040 POSITION J,P:? #6;A$(I,I);" ";
2050 FOR W=1 TO 10:NEXT W
2060 NEXT J
2070 Z=Z+1
2080 NEXT I
2090 SOUND 0,0,0,0:RETURN
2100 FOR I=AD*256+512 TO AD*256+767:POKE I,0:NEXT I:RETURN
3000 REM ZAPIS ZNAKOW NA C: LUB D:
3010 CHR=PEEK(756)*256
3020 NUM=32000
3030 ? "} Podaj nazwe zbioru";:INPUT A$
3040 FOR I=0 TO 1023 STEP 24
3050 ? "}"
3060 POSITION 2,4:? NUM;" ";"DATA ";
3070 ? PEEK(CHR+I);
3080 FOR J=1 TO 23
3090 ? ",";PEEK(CHR+I+J);
3100 IF I+J=1023 THEN 3120
3110 NEXT J
3120 ? :? "CONT"
3130 POSITION 2,0:POKE 842,13:STOP
3140 POKE 842,12:NUM=NUM+1
3150 NEXT I
3155 ? "}"
3160 LIST A$,32000,32042
3170 A$="":RETURN
```

Colossus chess

Jest to program szachowy, dlatego nie ma potrzeby omawiania zasad gry. Zainteresowani znają je z pewnością lepiej od nas. A ci, którzy nie znają, mogą je znaleźć w stosownej literaturze. Dlatego ograniczymy się do przedstawienia mozliwości oferowanych przez program Colossus chess 3.0.

W czasie gry dysponujemy dwoma przełączanymi obrazami: jeden z nich przedstawia szachownicę, a drugi sześć ostatnich posunięć każdego z partnerów, nazwy graczy (Colossus i Opponent), zegary szachowe oraz sposób analizowania ruchów przez program. Przełączania dokonujemy klawiszem „SPACE BAR". Po zakończeniu analizowania pozycji program wyświetla kilka najlepszych posunięć.

dokończenie na str. 23

Zgodnie z zapowiedzią, w poprzednim odcinku obecnie przystąpimy do rozszerzenia naszego programu. Dołączymy do niego elementy, które realizują następujące funkcje:

— składowanie pamięci obrazu na kasecie magnetofonowej,
— wczytanie rysunku z pamięci kasetowej do pamięci obrazu,
— kreślenie odcinka, okręgu i czworokąta.



KRESKA

(Basic CPC 464) (2)

Pierwsza z tych funkcji realizowana jest w linii 198, przy użyciu komendy **SAVE**. Ogólny format instrukcji:
SAVE nazwa — kartoteki — zbioru (typ danych), adres początku zbioru w pamięci RAM, (długość zbioru)

W przypadku naszego programu, kartotekę nazwałem RYSUNEK. Typ danych — litera B oznacza, że tworzony jest na taśmie zbiór binarny. Taki sposób zapisu minimalizuje czas jego zapisywania i odczytywania. Adres C000 pokrywa się z adresem początkowym pamięci obrazu, która w przypadku CPC 464 wykorzystuje 16 Kbajtów RAM (od adresu C000 do FFFF). Długość zbioru określona jest liczbą szesnastkową 4000, co odpowiada 16 K
4000 szesnastkowo = 16384 dziesiętnie ($4 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16^1 + 0 \cdot 16^0$). W rezultacie na taśmie kasetowej tworzony jest zbiór składający się z 8 bloków po 2Kbajtów każdy. Szybkość przesyłania danych w trakcie zapisu określana jest przez instrukcję:

SPEED WRITE a

dla a=0 szybkość transmisji wynosi 1000 bodów (bitów na sekundę) dla a=1 przesyłanie odbywa się z szybkością 2000 bodów. Wczytywanie rysunku z pamięci kasetowej do pamięci obrazu — linia 194 — realizowana jest w oparciu o komendę LOAD
Format

LOAD nazwa — kartoteki

Kreślenie odcinka prostej, okręgu oraz czworokąta dokonywane jest odpowiednio w liniach 4220—4400, 4590—4800, 4840—5090. Wspólną cechą przyjętych rozwiązań jest określanie punktów charakterystycznych poprzez naciskanie klawisza **COPY** bądź przycisku FIRE. I tak w wypadku rysowania odcinka pierwsze naciśnięcie **COPY/FIRE** spowoduje zaznaczenie jego punktu początkowego, ponowne zaś, wykreślenie linii do punktu aktualnie wskazywanego przez kursor graficzny. W przypadku kreślenia okręgu, pierwsze naciśnięcie **FIRE/COPY** wyznaczy jego środek. Dalsze przemieszczanie kursora graficznego, po ekranie określa zmianę długości promienia. Ponowne wciśnięcie **FIRE/ /COPY** spowoduje wyliczenie tej długości wg wzoru:

$$R = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

gdzie
(x_1, y_1) — współrzędne środka okręgu
(x_2, y_2) — współrzędne punktu aktualnie wskazywanego przez kursor

Samo rysowanie okręgu polega na zapalaniu na ekranie monitora wszystkich punktów (x,y) spełniających równania

$$x = x_1 + R \cdot \cos(a)$$
$$y = y_1 + R \cdot \sin(a)$$

gdzie a zawiera się w przedziale od 0° do 360°. Kreślenie czworokąta (równoległoboku) sprowadza się do wyznaczenia trzech jego wierzchołków. Czwarty wyliczany jest wg następujących zależności.

$$x_4 = x_1 - (x_3 - x_2)$$
$$y_4 = y_3 - (y_2 - y_1)$$

Po obliczeniu współrzędnych punktu (x_4, y_4) następuje wykreślenie czworokąta za pomocą instrukcji DRAW. Połączenia wszystkich elementów programu KRESKA dokonujemy następująco:

Po wprowadzeniu II części programu do pamięci komputera, składujemy ją na taśmie kasetowej komendą **SAVE** np.:

SAVE „PLUS"

Następnie kasujemy zawartość pamięci komputera. Po wykonaniu tej czynności możemy przejść do utworzenia rozszerzonej wersji programu. W tym celu należy:
1.. wczytać program **KRESKA** komendą **LOAD „KRESKA"**
2. dołączyć część II instrukcją **MERGE „PLUS"**
3. usunąć zbędne linie rozkazem **DELETE**. Dotyczy to linii: 820, 1300 oraz 1480.
4. przenumerować nowo otrzymany program komendą **RENUM**
Nową wersję programu **KRESKA** należy nagrać na taśmie magnetofonowej komendą:

SAVE „KRESKA V.2"

Na zakończenie uwaga praktyczna. Przy wprowadzeniu programu zaleca się pisanie komend małymi literami. Umożliwia to łatwe odnajdywanie błędów formalnych, gdyż wszystkie niepoprawnie wpisane instrukcje zostaną przez interpreter BASICa wylistowane na ekranie w niezmienionej postaci (interpreter poprawne komendy wypisuje wielkimi literami).

W kolejnych numerach zostaną przedstawione kolejne funkcje:
— wypełnianie kolorem figur zamkniętych
— aerograf
— kreślenie prostopadłościanu.

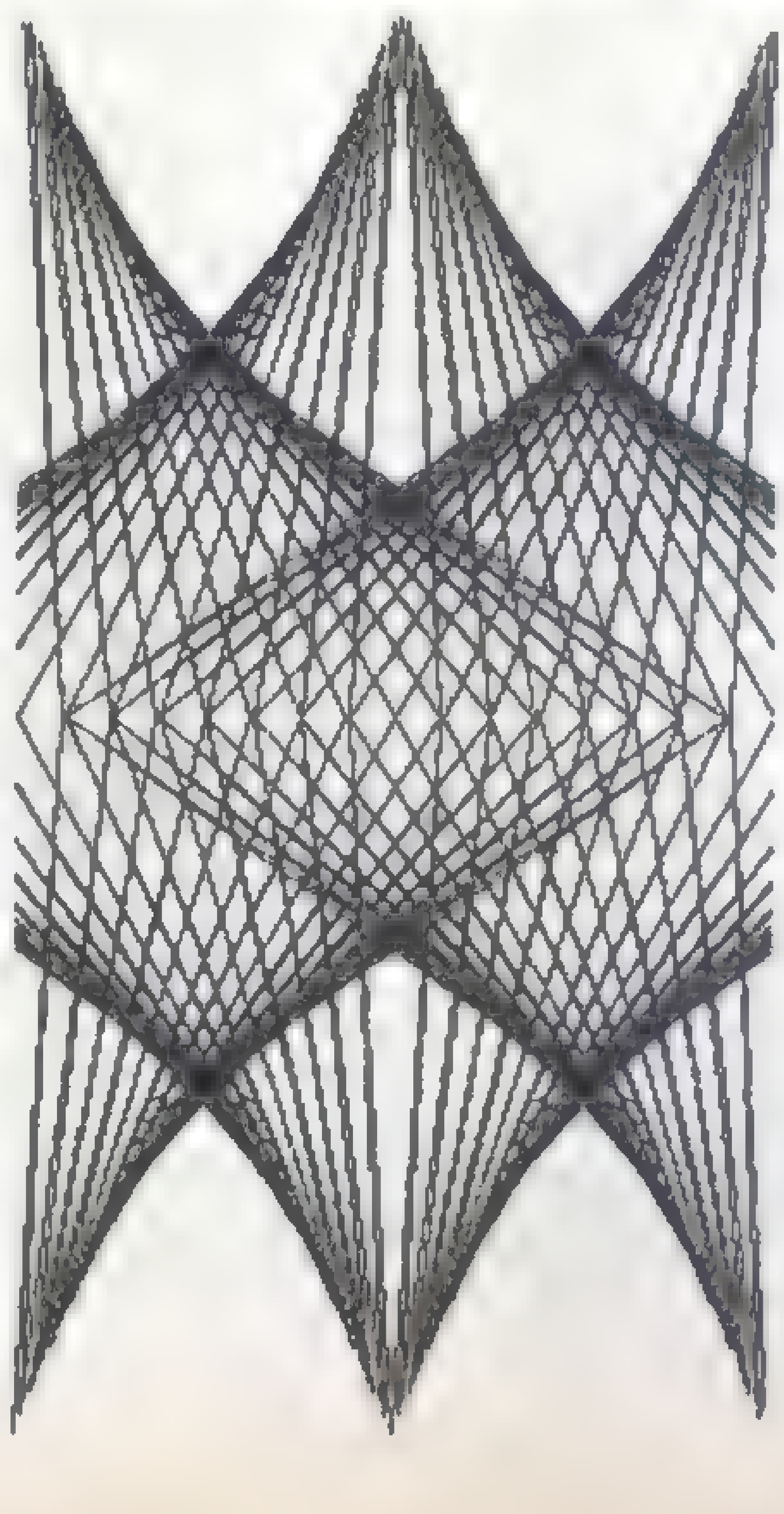
Janek

```
180 CLS #2:PRINT #2," (L)OAD (S)AVE (K)R
ESLENIE (C)LS";
190 LOCATE #2,2,2:PRINT #2,"(Z)AKONCZENI
E PRACY";
191 REM**********************************
192 REM      CZYTANIE RYSUNKU Z TASMY
193 REM**********************************
194 IF INKEY(36)=0 THEN CLS #2:PRINT #2,
" PROSZE WCISNAC <PLAY>":LOAD"!RYSUNEK":
GOTO 180
195 REM**********************************
196 REM      ZAPISANIE RYSUNKU NA TASMIE
197 REM**********************************
198 IF INKEY(60)=0 THEN CLS #2:PRINT #2,
" PROSZE WCISNAC <PLAY> I <REC> ":FOR i=
1 TO 2000:NEXT i:SPEED WRITE 1:SAVE"!RYS
UNEK",B,&C000,&4000:GOTO 180
320 GOTO 194
480 WINDOW #2,7,38,1,2:PRINT #2,"(M)ENU
GLOWNE (P)RZEMIESZCZENIE (K)OLOR (L)INIA
 (F)IGURY LUB("+CHR$(241)+")"
501 IF INKEY(36)=0 THEN 4220
502 IF INKEY(53)=0 THEN 4440
520 IF INKEY(37)=0 THEN GOTO 630
580 IF INKEY(52)=0 THEN GOTO 1340
4190 REM*********************************
4200 REM          KRESLENIE LINII
4210 REM*********************************
```

```
4220 CLS #2:PRINT #2," WYZNACZ PIERWSZY
PUNKT":LOCATE #2,2,2:PRINT #2,"UZYWAJ :"
+CHR$(240)+CHR$(242)+CHR$(243)+CHR$(241)
+" COPY LUB JOYSTICK"
4230 o%=0
4240 GOSUB 1600:GOSUB 930:GOSUB 1070
4250 GOSUB 870:GOSUB 930:GOSUB 980
4260 GOSUB 870
4270 IF o%=1 THEN PLOT a%,b%,1:MOVE x%,y
%
4280 IF INKEY(F%)=0 THEN GOTO 4310
4290 IF INKEY(27)=0 THEN 480
4300 GOTO 4240
4310 IF o%=1 THEN GOTO 4360
4320 a%=x%:b%=y%:o%=1
4330 LOCATE #2,1,1:PRINT #2," WYZNACZ DR
UGI PUNKT              "
4340 FOR i=1 TO 100:NEXT i
4350 GOTO 4240
4360 MOVE a%,b%
4370 DRAW x%,y%,kolor:o%=0
4380 LOCATE #2,1,1:PRINT #2," WYZNACZ PI
ERWSZY PUNKT  LUB P   "
4390 FOR i=1 TO 100:NEXT i
4400 GOTO 4240
4410 REM**********************************
4420 REM              FIGURY
4430 REM**********************************
4440 CLS #2:PRINT #2," (O)KRAG (C)ZWOROK
AT "
4450 LOCATE #2,2,2:PRINT #2,"(P)OWROT DO
 MENU "
4460 IF INKEY(34)=0 THEN 4590
4470 IF INKEY(27)=0 THEN 480
4475 IF INKEY(62)=0 THEN 4810
4480 GOTO 4460
4560 REM**********************************
4570 REM        KRESLENIE OKREGU
4580 REM**********************************
4590 CLS #2:PRINT #2," WYZNACZ SRODEK OK
REGU   LUB P"
4600 LOCATE #2,2,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241);"
 COPY LUB JOYSTICK"
4610 GOSUB 980:GOSUB 930:GOSUB 1070:GOSU
B 870:GOSUB 930
4620 IF INKEY(27)=0 THEN 480
4630 IF INKEY(F%)=0 THEN cx%=x%:cy%=y%:L
OCATE #2,2,1:PRINT #2,"WYZNACZ PROMIEN O
KREGU":FOR i=1 TO 150:NEXT i:GOTO 4660
4640 GOSUB 870
4650 GOTO 4610
4660 GOSUB 930:GOSUB 1070:GOSUB 980:GOSU
B 870:GOSUB 930
4670 PLOT cx%,cy%,1:MOVE x%,y%
4680 IF INKEY(F%)=0 THEN ox%=x%:oy%=y%:L
OCATE #2,2,1:PRINT #2,"WYZNACZ SRODEK OK
REGU   LUB P":GOTO 4710
4690 GOSUB 870
4700 GOTO 4660
4710 d1%=cx%-ox%:d2%=cy%-oy%
4720 d1%=ABS(d1%):d2%=ABS(d2%)
4730 r=SQR((d1%^2)+(d2%^2))
4740 FOR a%=1 TO 360
4750 DEG
4760 PLOT cx%+r*COS(a%),cy%+r*SIN(a%),ko
lor
4770 x=XPOS:y=YPOS:GOSUB 1530
4780 NEXT a%
4790 PLOT cx%,cy%,0:x%=cx%:y%=cy%
4800 GOTO 4590
4810 REM**********************************
4820 REM      KRESLENIE CZWOROKATA
4830 REM**********************************
4840 CLS #2:PRINT #2," WYZNACZ PIERWSZY
PUNKT   LUB P":LOCATE #2,2,2:PRINT #2,"U
ZYWAJ :"+CHR$(240)+CHR$(242)+CHR$(243)+C
HR$(241)+" COPY LUB JOYSTICK"
4850 q%=0
4860 GOSUB 1600:GOSUB 930:GOSUB 1070
4870 GOSUB 870:GOSUB 930:GOSUB 980
4880 GOSUB 870
4890 IF sk%=0 AND pr%=1 AND q%=0 THEN PL
OT xp%,yp%,kol:MOVE x%,y%:q%=1
4900 IF INKEY(F%)=0 THEN 4930
4910 IF INKEY(27)=0 THEN 480
4920 GOTO 4860
4930 PLOT x%,y%,1
4940 x1%=x%:y1%=y%
4950 LOCATE #2,1,1:PRINT #2," WYZNACZ DR
UGI PUNKT              "
4960 GOSUB 930:GOSUB 1070:GOSUB 870:GOSU
B 930:GOSUB 980:GOSUB 870
4970 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
4980 IF INKEY(F%)=0 THEN 5000
4990 GOTO 4960
5000 PLOT x%,y%,1:x2%=x%:y2%=y%:q%=0
5010 LOCATE #2,1,1:PRINT #2," WYZNACZ TR
ZECI PUNKT             "
5020 GOSUB 1600:GOSUB 930:GOSUB 1070:GOS
UB 870:GOSUB 930:GOSUB 980:GOSUB 870
5030 IF sk%=0 AND pr%=1 AND q%=0 THEN PL
OT xp%,yp%,kol:MOVE x%,y%:sk%=1:q%=1
5040 IF INKEY(F%)=0 THEN 5060
5050 GOTO 5020
5060 x3%=x%:y3%=y%:MOVE x1%,y1%
5070 x4%=x1%+(x3%-x2%):y4%=y3%-(y2%-y1%)
5080 DRAW x2%,y2%,kolor:DRAW x3%,y3%,kol
or:DRAW x4%,y4%,kolor:DRAW x1%,y1%,kolor
5090 x=XPOS:y=YPOS:GOSUB 1530:GOTO 4840
```

Grafika (AMSTRAD)

```
10 CLS
20 ORIGIN 0,0
30 FOR x=1 TO 15
40 DRAW 320,25*x
50 DRAW 10*x,5*x
60 NEXT x
70 ORIGIN 640,0
80 FOR y=1 TO 15
90 DRAW -320,25*y
100 DRAW -10*y,5*y
110 NEXT y
120 ORIGIN 0,400
130 FOR z=1 TO 15
140 DRAW 320,-25*z
150 DRAW 10*z,-5*z
160 NEXT z
170 ORIGIN 640,400
180 FOR k=1 TO 15
190 DRAW -320,-25*k
200 DRAW -10*k,-5*k
210 NEXT k
220 ORIGIN 0,200
230 FOR x=1 TO 15
240 DRAW 320,25*x
250 DRAW 10*x,5*x
260 NEXT x
270 ORIGIN 0,200
280 FOR y=1 TO 15
290 DRAW 320,-25*y
300 DRAW 10*y,-5*y
310 NEXT y
320 ORIGIN 640,200
330 FOR z=1 TO 15
340 DRAW -320,25*z
350 DRAW -10*z,5*z
360 NEXT z
370 ORIGIN 640,200
380 FOR x=1 TO 15
390 DRAW -320,-25*x
400 DRAW -10*x,-5*x
410 NEXT x
```



Możliwości Commodore 64 (2)

W tej części zajmiemy się znakowym trybem pracy mikrokomputera Commodore 64. W szczególności opisany zostanie GENERATOR ZNAKÓW oraz metoda programowania własnych znaków w trybie pracy dwukolorowej i wielokolorowej. Dla przybliżenia omawianych problemów posłużymy się dwoma programami:

1. **zobrazowanie bitowej reprezentacji znaków rezydujących w GENERATORZE ZNAKÓW;**
2. **dwukolorowa i wielokolorowa realizacja podstawowych elementów obwodu elektrycznego.**

Generator znaków

Wszystkie standardowe znaki graficzne widziane na ekranie otrzymywane są z GENERATORA ZNAKÓW (GZ) mieszczącego się w pamięci ROM. Bezpośredni dostęp do tej pamięci posiada VIC pracując w BANKU Ø lub w BANKU 2. W BANKU Ø standardowy GZ zlokalizowany jest od adresu 4Ø96 do 8191, a w BANKU 2 od adresu 36864 do 4Ø959. W obszarach występowania standardowego GZ pamięć RAM jest przesłaniana pamięcią ROM zawierającą GZ — z punktu widzenia VIC-a. System operacyjny i programy korzystają w tych miejscach z normalnej pamięci RAM o identycznej adresacji

Dostęp do standardowego GZ system operacyjny i programy mają w obszarze od 53248 bajtu do 57343 bajtu. Napotykamy tu jednak pewną trudność: ten sam obszar pamięci jest wykorzystywany do dwóch różnych celów. Pamięć RAM zawiera w tym miejscu rejestry we/wy niezbędne do obsługi przerwań. O tym, która pamięć fizycznie występuje pod wspomnianymi adresami decyduje aktualne ustawienie przełącznika rodzaju pamięci zlokalizowanego w trzecim bicie pierwszego bajtu pamięci.

W celu uzyskania dostępu do standardowego GZ musimy poinformować system operacyjny, aby nie reagował na pojawiające się sygnały przerwań.

```
3Ø POKE 56334, PEEK (56334) AND 254
```

Posiadacze mikrokomputerów C-64 mogą jednocześnie wprowadzać do pamięci pierwszy z przykładowych programów, który posłuży nam do ilustracji omawianych zagadnień związanych z GENERATOREM ZNAKÓW.

Ustawmy teraz przełącznik rodzaju pamięci tak, aby w obszarze 53248 — 57343 mieć GZ z pamięci ROM.

```
4Ø POKE 1, PEEK (1) AND 251
```

Powrót do stanu poprzedniego tj. odłączenie pamięci ROM, przyłączenie pamięci RAM i umożliwienie obsługi przerwań uzyskamy sekwencją instrukcji

```
6Ø POKE 1, PEEK (1) OR 4
7Ø POKE 56334, PEEK (56334) OR 1
```

GENERATOR ZNAKÓW podzielony jest na dwa zbiory znaków (po 256 znaków każdy) wielkości 2K. W procesie wyświetlania znaków na ekranie VIC pracuje zawsze tylko na jednym ze zbiorów, który możemy wybrać:

- przyciskając jednocześnie klawisze SHIFT i COMMODORE;
- ustawiając odpowiednią wartość 53272 bajtu według tabeli zamieszczonej w części pierwszej artykułu.

Każdy znak graficzny ma określony KOD EKRANOWY (KE), który jest liczbą porządkową tego znaku w zbiorze GZ. Ponieważ istnieją dwa zbiory znaków, ten sam KE może odnosić się do dwóch różnych znaków (np. kod 1 odpowiada znakom A i a). Zestawy znaków standardowych i ich KODY EKRANOWE zamieszczone są w każdej instrukcji do mikrokomputera C-64.

W naszym przykładowym programie znak będziemy lokalizować podając dane:

```
1Ø INPUT "KOD EKRANOWY ZNAKU Ø — 255"; KE
2Ø INPUT "ZBIÓR ZNAKÓW 1 — 2"; ZB
```

Każdy znak występujący w GENERATORZE ZNAKÓW definiowany jest na ośmiu bajtach. Rzeczywisty adres (RA) znaku w pamięci można wyznaczyć według reguły:

```
5Ø RA=512ØØ+ZB*2Ø48+KE*8
```

Po zlokalizowaniu znaku pozostaje tylko wyprowadzić zawartość pamięci definiującej znak. Każdy z ośmiu bajtów wyprowadzany ma być w postaci zerojedynkowej w kolejnych wierszach ekranu, zachowując uporządkowanie z pamięci. Przedstawiona poniżej grupą instrukcji wykonuje opisaną procedurę

```
6Ø PRINT: PRINT "POSTAĆ BITOWA ZNAKU:".PRINT
7ØFOR I=Ø TO 7: LP=PEEK (RA+I): FOR J=7
TO Ø STEP —1
8Ø LK=INT(LP/2): T(J)=LP—LK*2: LP=LK
NEXT: Z$=""
9Ø FOR J=ØTO 7: Z$=Z$+MID$(STR$(T(J)).
2,1): NEXT
1ØØ PRINT"        ";Z$: NEXT
```

Spróbujmy uruchomić nasz program dla następujących danych: KE=1, ZB=1 oraz KE=1, ZB=2. Uzyskamy na ekranie bitową reprezentację litery A i litery a. Litera A ma postać:

```
Ø Ø Ø 1 1 Ø Ø Ø
Ø Ø 1 1 1 1 Ø Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø 1 1 1 1 1 1 Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø 1 1 Ø Ø 1 1 Ø
Ø Ø Ø Ø Ø Ø Ø Ø
```

Zatem, każdy znak zbudowany jest z 64 punktów (8 wierszy na 8 kolumn); a wartość bitu 1 odpowiada świecącemu punktowi na ekranie, zaś wartość Ø odpowiada kolorowi. Oczywiście jest to słuszne w trybie pracy dwukolorowej

Dla ciekawskich pozostawiam dokładniejsze przebadanie GENERATORA ZNAKÓW za pomocą zaprezentowanego programu. Możemy wykryć dodatkowe cechy standardowego GZ (np. jakie znaki graficzne przedstawiają KE, które nie zostały pokazane w instrukcji do C-64?).

Programowanie własnych znaków

Ogólną metoda wykorzystania w grafice komputerowej niestandardowych znaków sprowadza się do budowy własnego GENERATORA ZNAKÓW i przekazania go VIC-owi. Przy realizacji tego typu programów należy wykonać następujące czynności:

- zaprojektować własne znaki (uwzględniając ewentualne ich wykorzystanie w trybie wielobarwnym);

— wybrać obszar pamięci RAM, w którym powstanie nowy GENERATOR ZNAKÓW (korzystaj z tabeli zamieszczonej w pierwszej części artykułu) i skopiować do tego obszaru niezbędne znaki ze standardowego GZ,

— wprowadzić zaprojektowane znaki w odpowiednie miejsca do nowego GZ;

— zlecić VIC-owi korzystanie z nowego GZ;

— przeprowadzić dodatkowe operacje związane z wykorzystywaniem własnego GZ (np. zabezpieczenie się przed niekontrolowanym zniszczeniem zawartości GZ przez program w Basicu)

Powyzsze czynności zilustrowane zostały w kolejnym programie, którego zadaniem jest zobrazowanie na ekranie podstawowych elementów obwodu elektrycznego: rezystancji, pojemności, indukcyjności. W zestawie znaków standardowych C-64 nie ma symboli graficznych wymienionych elementów. Przyjęte zostały następujące założenia dotyczące projektowanych znaków

- rezystancja zaprojektowana jest jako dwa znaki o KE 35 i 36 (# i $);
- pojemność reprezentuje jeden znak o KE 37 (%);
- indukcyjność przedstawiana jest za pomocą trzech znaków o KE 38, 39 i 4Ø/&, ' i (/.

Poszczególne grupy instrukcji opatrzone zostały komentarzem ułatwiającym zrozumienie programu

```
5 REM KOPIOWANIE STANDARDOWEGO GZ DO WYBRANEGO OBSZARU PAMIĘCI RAM
1Ø POKE56334, PEEK (56334) AND 254
2Ø POKE 1,PEEK(1) AND 251
3Ø FOR I=Ø TO 4Ø95: POKE 4Ø96Ø+I, PEEK (53248+I) NEXT
4Ø POKE 1,PEEK(1) OR 4
5Ø POKE 56334, PEEK (56334) OR 1
55 REM WPROWADZENIE WŁASNYCH ZNAKÓW DO GZ W ODPOWIEDNIE MIEJSCA
6Ø FOR I=4124Ø TO 41287: READ L: POKEI,L: NEXT
7Ø DATA Ø,17Ø,128,128,128,128,17Ø,Ø: REM REZYSTANCJA
8Ø DATA Ø,17Ø,2,2,2,2,17Ø,Ø
9Ø DATA 13Ø,13Ø,13Ø,13Ø,13Ø,13Ø,13Ø,13Ø: REM POJEMNOŚĆ
1ØØ DATA 4Ø,13Ø,13Ø,13Ø,13Ø,Ø,Ø,Ø: REM INDUKCYJNOŚĆ
11Ø DATA 162,8,8,8,8,Ø,Ø,Ø
12Ø DATA 128,32,32,47,47,Ø,Ø,Ø
125 REM PRZEKAZANIE NOWEGO GZ VIC-owi
13Ø POKE 56576,149: POKE 648,14Ø: POKE 53272,57
14Ø POKE 56,3584Ø/256: CLR: REM OGRANICZENIE OD GÓRY PAMIĘCI RAM DLA BASICA
15Ø POKE 5328Ø,1: POKE 53281,1: REM BIAŁY KOLOR OBRZEŻA I EKRANU
16Ø PRINTCHR$(147) : PRINT: PRINT„PODSTAWOWE ELEMENTY OBWODU ELEKTRYCZNEGO" : PRINT
17Ø POKE 646,11: REM ZNAKI KOLORU SZAREGO
18Ø PRINT" o——#$—%—&'(——o": STOP: REM WYŚWIETLENIE OBWODU ELEKTRYCZNEGO
215 REM POWROT DO SYTUACJI POCZĄTKOWEJ
22Ø POKE 56,4Ø96Ø/256: CLR
23Ø POKE 56576,151: POKE 53272,21: POKE 648,4
```

Występująca w wierszu 18Ø instrukcja STOP pozwala na pozostanie w BANKU 2. Kontynuacja programu spowoduje powrót VIC-a do BANKU Ø.

W programie wykorzystana jest właściwość mikrokomputera C-64 polegająca na takiej samej adresacji dwóch różnych pamięci. Nowy GENERATOR ZNAKÓW zlokalizowany jest w obszarze 4Ø96Ø — 45Ø58 pamięci RAM. Pod tymi samymi adresami występuje interpreter Basica w pamięci ROM.

Kolorowanie znaków

W przedstawionych programach wykorzystywany był dotychczas dwukolorowy tryb pracy VIC-a. Pozwala on na wyświetlenie znaku mieszczącego się na 64 punktach ekranu (w ośmiu wierszach i kolumnach) odwzorowujących 64 bity pamięci (jeden bit — jeden punkt ekranu). W dowolnej chwili mieliśmy możliwość określenia koloru ekranu (jednego tła dla wszystkich znaków) i koloru każdego ze znaków występujących na ekranie. Kolor ekranu wybieramy wprowadzając do bajtu 53281 liczbę odpowiadającą wybranej barwie (jednej z szesnastu możliwych). Kolor znaku określany jest na podstawie zawartości najmłodszych trzech bitów odpowiedniego bajtu PAMIĘCI EKRANU (patrz cz.1).

W trybie wielokolorowym wyświetlany znak mieści się na 32 punktach ekranu (w ośmiu wierszach i czterech kolumnach) odwzorowujących również 64 bity pamięci, przy czym parze bitów odpowiada jeden punkt ekranu. Znak na ekranie w obu trybach zajmuje tę samą powierzchnię ekranu. Kolory punktów tworzących znaki są zależne od odpowiadających im par bitów. Para bitów wskazuje skąd należy pobrać kod barwy wyświetlanych punktów według poniższego zestawienia

ØØ — bajt 53281;
Ø1 — bajt 53282;
1Ø — bajt 53283,
11 — najmłodsze trzy bity w bajcie odpowiadają położeniu znaku w PAMIĘCI KOLORU.

Zapisując odpowiednie kody barw w wymienionych bajtach można realizować wielobarwną grafikę w trybie znakowej pracy VIC-a

Wykorzystanie trybu wielokolorowego zilustrujemy wprowadzając do ostatniego programu dodatkowe wiersze:

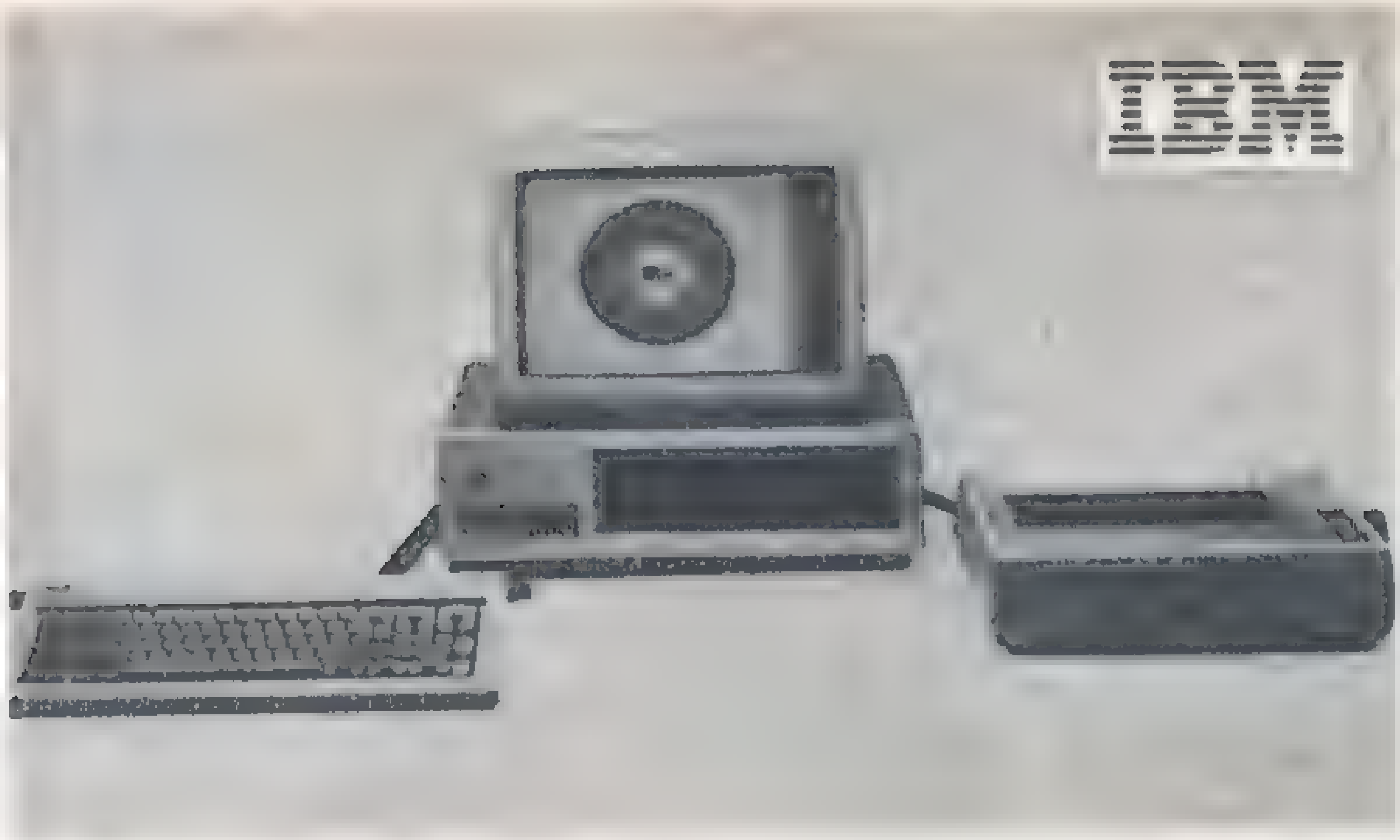
```
19Ø POKE 53282,15: POKE 53283,Ø: REM WYBÓR KOLORÓW
2ØØ POKE 5327Ø, PEEK (5327Ø) OR 16: STOP: REM PRZEŁĄCZENIE VICA NA TRYB WIELOKOLOROWY
21Ø POKE 5327Ø, PEEK(5327Ø) AND 239: STOP: REM POWRÓT DO TRYBU DWUKOLOROWEGO
```

O programowaniu grafiki o dużej rozdzielczości przeczytać będzie można w następnej części.

Sławomir WASIELEWSKI



Rys. Michał Przybyłowski



(1) # Mikrokomputery IBM PC

Cykl artykułów o mikrokomputerach personalnych firmy IBM stanowi próbę zaprezentowania ich architektury, parametrów technicznych i oprogramowania. Próba ta poparta jest pewnym doświadczeniem autora, zdobytym w czasie pracy z tym ciekawym systemem.

Określenie komputer personalny (Personal Computer), lub komputer osobisty używane jest w odniesieniu do sprzętu, który może być wykorzystywany profesjonalnie. Jak dynamicznie kształtuje się światowy standard komputera personalnego możemy prześledzić właśnie na przykładzie firmy IBM

Komputery **IBM PC** pojawiły się na rynku pod koniec 1981 roku. Jako jednostkę centralną wykorzystywano w nim 16-bitowy mikroprocesor INTEL 8088. Komputer posiadał pamięć operacyjną RAM 64 Kb, pamięć stałą ROM 40 Kb oraz pamięć zewnętrzną na dyskach elastycznych, zwanych dyskietkami (Diskette), o wymiarach 5 1/4 cala (133 mm) i czasie dostępu do informacji rzędu kilkudziesięciu milisekund

Na początku 1983 roku pojawiła się wersja **IBM PC/XT**. Mikrokomputer wyposażono w większą pamięć operacyjną, o pojemności 128 Kb oraz umożliwiono rozbudowanie jej do pojemności 640 Kb. Obok pamięci masowej na dyskach elastycznych, o gęstości zapisu 184 320 bajtów (znaków informacji) lub 368 640 bajtów, zastosowano tzw. dysk stały (Hard Disk) o pojemności 10 Mb, później 20 Mb. W mikrokomputerze wykorzystano, obok jednostki centralnej 8088, koprocesor arytmetyczny INTEL 8087. Połączenie mikroprocesora 8088 z koprocesorem daje programiście jeden wspólny procesor tzw. interprocesor, o automatycznie skoordynowanych funkcjach. Koprocesor pracuje równolegle z procesorem głównym, wykonując te rozkazy, pobierane z pamięci programu które są dla niego przeznaczone

W połowie 1984 roku pojawiła się kolejna wersja **IBM PC/AT**, wyposażona w nową jednostkę centralną INTEL 80286 i koprocesor arytmetyczny INTEL 80287. Pamięć operacyjna ma pojemność 256 Kb i posiada możliwość rozszerzenia do 1 Mb. Mikrokomputer posiada dysk stały 20 Mb (lub większy) oraz mechanizmy dyskietkowe; jeden pozwalający na osiągnięcie na dyskietce 5 1/4 cala gęstości zapisu 1,2 Mb, a drugi o gęstości 368 Kb. IBM PC/AT posiada inne udoskonalenia

W bieżącym roku zaprezentowano najnowszą wersję **IBM PC/RT** (lub **IBM 6150**). W mikrokomputerze zastosowano 32-bitowy procesor o architekturze RISC (Redunded Instruction Set Computer), dużo szybszy od jednostek centralnych stosowanych w poprzednich wersjach oraz koprocesor AT, umożliwiający wykorzystanie programów z IBM PC/AT. Pamięć operacyjna może mieć pojemność od 1 do 4 Mb. Pamięć masowa to dysk stały 210 Mb oraz dyski elastyczne, o gęstości zapisu 1,2 Mb

Porównując wersję IBM PC/RT z pierwszą wersją IBM dochodzimy do wniosku, że jest to olbrzymi skok możliwości technicznych sprzętu, niewiarygodnie wysoko podnoszący standard komputera personalnego. Rozwiązania sprzętowe tworzą z IBM PC/RT komputer o bardzo dużych mocach obliczeniowych.

O wartości komputerów IBM PC świadczą nie tylko ich parametry techniczne. Można uznać, że sprzęt (Hardware) jest „drugoplanową" cechą tych komputerów. Cechą najważniejszą, decydującą o tym, że IBM PC opanowały już praktycznie wszystkie dziedziny życia, jest ich bardzo bogate oprogramowanie (software).

Mikrokomputery IBM PC/XT dysponując w typowej konfiguracji pamięcią operacyjną 640 Kb, pamięcią zewnętrzną na dysku stałym i dyskach elastycznych, stanowią uniwersalny system obliczeniowy, o dość dużej mocy obliczeniowej. Moc ta może być zwiększana poprzez utworzenie sieci, z reguły 16- lub 32- komputerowej. Na komputerach IBM PC/XT skupimy właśnie naszą uwagę w dalszych numerach „IKS-a".

W mikrokomputerach PC/XT zastosowano dyskowy system operacyjny (DOS — Disk Operating System), będący bogatym i uniwersalnym systemem, pozwalającym na budowę finezyjnych programów użytkowych. Wykorzystywane są dwie implementacje tych systemów: MS-DOS (wersje 2.0, 2.11, 3.10) oraz PC-DOS (wersja 3.0) Mikrokomputer może pracować również pod systemem operacyjnym CP/M

Wersja PC/RT posiada system operacyjny IBM AIX, umożliwiający pracę wielodostępną i wieloprogramową wzorowaną na wersji V systemu UNIX Innym bardzo ciekawym systemem jest MS-WIN DOS, zorientowany graficznie pakiet oprogramowania, komunikujący się z użytkownikiem za pomocą okien z zestawem funkcji reprezentowanych przez piktogramy

Oprogramowanie do PC/XT zawiera, oprócz interpretera BASIC-u, kompilatory BASIC-u, C, COBOL-u, FORTRAN-u, PASCAL-a. Z języków, posiadających implementację na PC/XT, kolejne to: MODULA-2 LOGO, FORTH. Do tworzenia oprogramowania w języku procesora służy MACRO ASSEMBLER. Istnieje wiele programów wspomagających tworzenie oprogramowania użytkowego, są to edytory czy programy, jak np. SIDEKICK, na stałe rezydujące po ich wpisaniu do pamięci operacyjnej, a służące do podręcznych kalkulacji i notatek

Istnieje wiele zintegrowanych pakietów programowych, umożliwiających całą gamę zastosowań PC/XT — zaczynając od komputerowego wspoma-

gania projektowania inżynierskiego (CAE — Computer Aided Engineering czy CAD — Computer Adided Desing), wspomagania produkcji, a kończąc na programach do prowadzenia ewidencji i statystyk w gabinetach lekarskich, czy obsługujących zawody sportowe

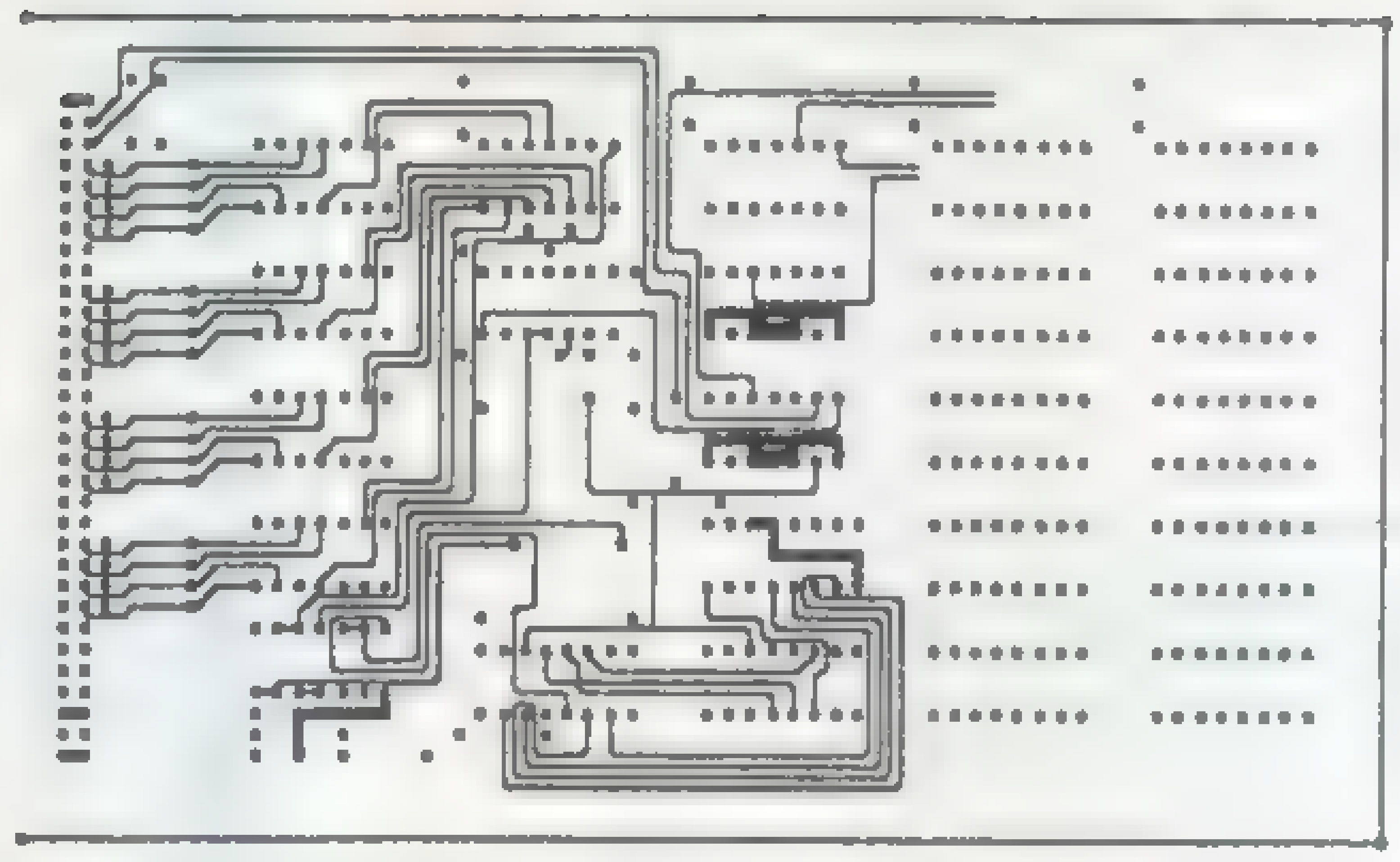
Wśród programów wspomagających projektowanie warto wymienić: sam — ARTWORK, program umożliwiający projektowanie dwustronnych płytek elektronicznych — przykładowe wydruki przedstawiono na rys. 1 i rys. 2, ANALOP i NODAL, programy do analizy obwodów mikrofalowych. Do obliczeń całek lub rozwiązywania równań różniczkowych może posłużyć mu — SIMP (mu — MATH)

Inna grupa to programy typu baza danych, wśród nich: DBASE II, DBASE III, RBASE 4000 lub im podobne, takie jak: SYMPHONY czy LOTUS

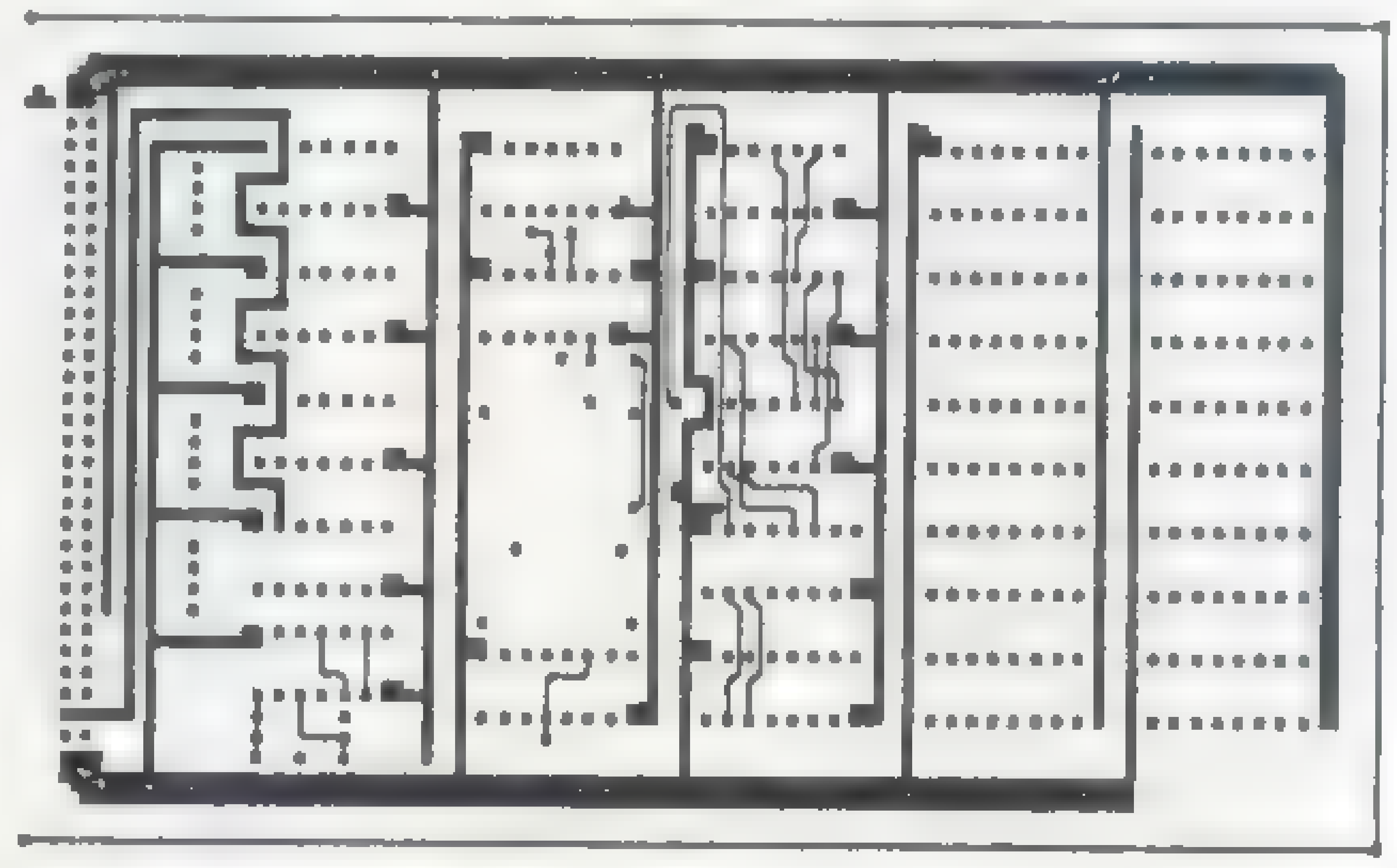
Następna grupa to programy służące do edycji tekstów np. WORDSTAR i WORDSTAR 2000 + (do pracy w sieci) lub programy wspomagające sporządzanie dokumentacji technicznej

Kolejna grupa to programy graficzne, takie jak: ACAD, PAINT BRUSH, GEM, ENERGRAPHICS, 3 DESIGN

Tak w skrócie wygląda najważniejsza, dostępna dla autora, część oprogramowania mikrokomputerów IBM PC/XT. Oczywiście zgodnie z zasadą „kompatybilności w górę" większość programów może pracować na wersji PC/AT i prawdopodobnie na PC/RT. Wersja IBM PC/RT, dopiero po opracowaniu nowego oprogramowania, zdecydowanie zwielokrotni swoje możliwości w stosunku do modeli XT i AT.



Rys.1 i 2 — Przykładowe wydruki płytek elektronicznych projektowane za pomocą programu SM-ARTWORK. (Skala 1:2)



Nie powinniśmy zapomnieć, że mikrokomputery IBM PC nie mogłyby znaleźć zastosowań profesjonalnych bez odpowiednich urządzeń graficznych, takich jak drukarki matrycowe czy plottery, służące do sporządzania rysunków, wykresów itp

Do charakterystyki oprogramowania będziemy wracać w kolejnych numerach, natomiast w następnym numerze „IKS-a" poznamy architekturę mikrokomputera IBM PC/XT.

(sierpień 1986 r.)

Jacek WOJTALA

BIBLIOTEKA

Z listów do redakcji wynika, że „IKS-a" czytają: uczniowie, studenci i specjaliści — krótko: „IKS-a" czytają Ci, których łączy jedna wspólna cecha: ciekawość (Co komputer może? Jak działa? Jaki będzie?...).

Wychodząc naprzeciw tym, którym nie wystarcza „tylko palcowanie" rozpoczynamy nowy dział.

Mikrorecenzje będą opisami książek nadesłanych nam. Zaczynamy od Wydawnictw Komunikacji i Łączności.

1) **„24 proste układy elektroniczne do samodzielnego wykonania".** (cena 200 zł). Poziom hobbysty-majsterkowicza Para autorska: Maria i Wojciech Nowakowscy szczegółowo opisują domowe urządzenia elektroniczne: alarm włamaniowy, interkom, minutnik... zasilacz do akumu-



zalety urządzeń, natomiast bardziej niż dokładnie konstrukcję, a w szczególności nowoczesną technologię (świadectwem ich dokładności jest powtórzenie na końcu rysunków obwodów drukowanych w skali 1:1)

* * *

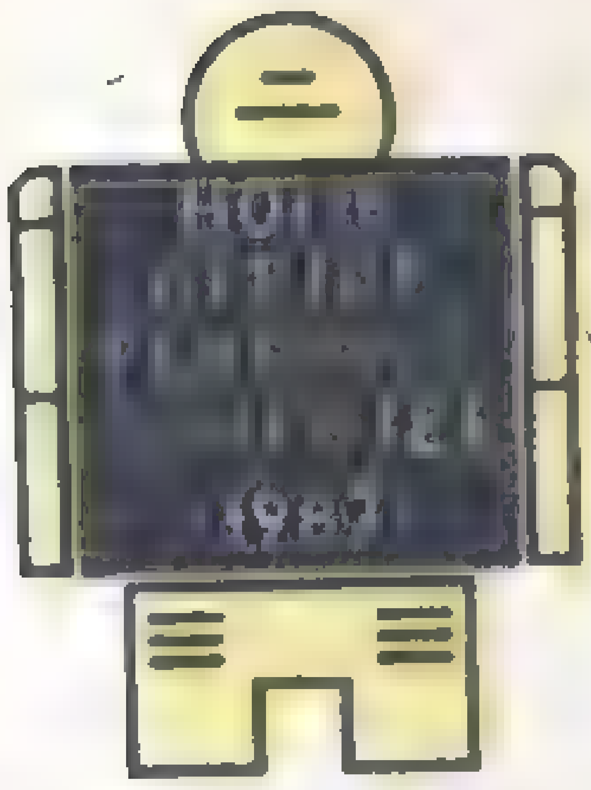
2) **„PL/I dla inżynierów"**

„Elektronizacja — przewodnik zawodowy (zeszyt 22) autorstwa Jana Bieleckiego — przeznaczony jest jedynie dla profesjonalistów. PL/I zdobywa sobie coraz więcej użytkowników. W Polsce pojawił się wraz z komputerem IBM/369/379 i systemem RIAD, który jako Jednolity System M C. zdobywa sobie rynek krajów socjalistycznych.

ciąg dalszy na str. 31



Rys. Michał Przybyłowski



TRADYCJA TO NASZ OBOWIĄZEK



Rozmawiamy z dyrektorem Oddziału INTERNATIONAL COMPUTERS LIMITED w Polsce — JANEM J. KLUKIEM

— Mówią o Was potentaci...

Jan J. Kluk: — Kto mówi?

— Fachowcy i CI, dla których ICL oznacza po prostu dobry komputer.

J.J. Kluk: — Na tę opinię pracujemy już ponad osiemdziesiąt lat.

— I chyba nie tylko na nią.

J.J.K.: — Jesteśmy obecnie największą brytyjską firmą komputerową, posiadamy swoje oddziały w 80 krajach, na pięciu kontynentach. Zatrudniamy 23 500 osób, a obroty firmy przekraczają miliard funtów rocznie.

— W Polsce ICL obecny jest od 20 lat — jakie są tego efekty?

J.J.K.: — Dokładnie od 25 lat działa Biuro Handlowe ICL, a efektem jego pracy jest działanie ponad 400 instalacji systemów komputerowych ICL i ICL — pochodnych.

— Całe zastępy polskich fachowców wychowały się na ODRACH.

J.J.K.: — ODRA powstała na bazie ICL 1900. I oczywiście w tym przypadku liczył się nie tylko osprzęt (hardware) ale równie bogate oprogramowanie (software).

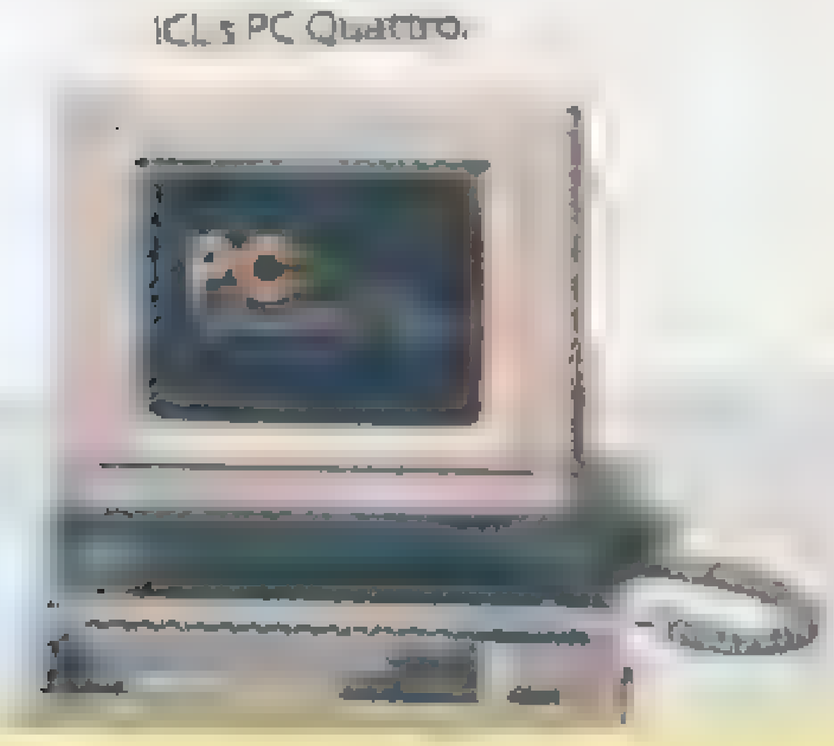
— Czy nie sądzi Pan, że właśnie ten pierwszy kontakt większości naszych informatyków z ICL, powoduje, że tak często decydują się w przyszłości na wasz sprzęt, który jednak nie poddał się światowemu standardowi IBMa?

J.J.K.: — W interesach nikt nie kieruje się sentymentami. Jeśli na nasze systemy zdecydowały się stocznie w Gdańsku, Gdyni, Szczecinie i Ustce, Huta im. Lenina, Zakłady Metalowe im. Cegielskiego, banki, centrale handlu zagranicznego — lista ta jest oczywiście niepełna — i ciągle realizowane są nowe kontrakty, to nie z sentymentu — tu liczy się jakość.

— I co jeszcze?

J.J.K.: — Nasze atuty to wiarygodność i elastyczność działania. Wymusza to stosowanie najnowszych technologii przy konstruowaniu sprzętu i tworzeniu oprogramowania. Musimy również budować urządzenia kompatybilne zarówno z IBM, jak i ze sprzętem krajowym, takim jak na przykład ODRA. Nowy komputer ICL nie może wprowadzać organizacyjnego zamieszania i nie może tworzyć sytuacji, w której działające dotychczas systemy informatyczne będzie trzeba przeprogramowywać. Poza tym coraz częściej proponujemy rozwiązania kompleksowe. To znaczy sprzęt i specjalistyczne ▶

ICL's PC Quattro.



Rodzina mikrokomputerów ICL PC serii QUATTRO+ obejmuje cztery modele M19, M39, M49, M59 różniące się pojemnością pamięci zewnętrznej. Do jednostki centralnej ICL PC QUATTRO+ można podłączyć jednocześnie cztery stanowiska (monitor monochromatyczny lub kolorowy, klawiatura, myszka). Mikrokomputery te mogą współpracować z drukarkami mozaikowymi, laserowymi i termicznymi, ploterami, czytnikami taśm, czytnikami kodów paskowych, kasami sklepowymi, modemami i innymi urządzeniami dołączanymi poprzez interfejs RS232C/V24. QUATTRO+ pracuje pod systemem operacyjnym Multiuser Concurrent CP/M-86 3.1.8 lub Concurrent DOS 4.1. Systemy te umożliwiają emulację MS-DOS — a to pozwala na realizację wielu programów z IBM PC.



ICL CLAN jest 32-bitowym, szesnastostanowiskowym mikrokomputerem pracującym pod UNIXem. Trzy jego modele: 35, 45 i 55 różnią się pojemnością pamięci zewnętrznej, ilością stanowisk, które mogą obsłużyć i oczywiście ceną. Może współpracować podobnie jak PC QUATTRO z drukarkami mozaikowymi, termicznymi i laserowymi. System operacyjny zawiera ponad 150 programów standardowych realizujących między innymi: sortowanie, wyszukiwanie informacji. Oprogramowanie narzędziowe stanowią: język C, Fortran 77, SMC Basic, SVS, Pascal, Level II Cobol.

oprogramowanie, takie jak systemy magazynowe czy finansowe w oparciu o komputery osobiste do wielkich systemów komputerowych Serii 39

— Czy nie jest to zbyt optymistyczne?

J.J.K.: — Jeśli coś zwyczajnie nawali, a to choć rzadko, to przecież się zdarza, mamy serwis gwarancyjny, pogwarancyjny i niezłą kadrę specjalistów. W Polsce mamy również własny skład konsygnacyjny co znacznie ułatwia pracę. Poza tym nasz klient wie, że w przyszłości kupi u nas coś nowoczesnego, utrzymanego w dotychczasowym standardzie. Nasze wyroby idą do przodu — nasi kontrahenci mają taką pewność.

— Ale nie mają pewności, że będzie ich na to stać.

J.J.K.: — ICL nie jest przeznaczony dla amatorów

— Zawodowcy również nie zawsze mogą sobie pozwolić na takie inwestycje.

J.J.K.: — Zdajemy sobie z tego sprawę

— I co?

J.J.K.: — I coraz częściej musimy dbać o to, aby nasz potencjalny klient miał za co nas kupić. To jest cena utrzymania się na tym rynku. Pracownicy naszego biura muszą poszukiwać towarów, które dadzą się sprzedać w drugim obszarze

— Tymczasem coraz więcej firm decyduje się na popularny sprzęt tańszy od ICLa.

J.J.K.: — Ale nie są to rozwiązania kompleksowe. Obecnie np. AMSTRAD to rozwiązanie doraźne. Nie dorównuje nam ani sprzętowo, ani programowo Jest jedynie tańszy.

— I to jest argument.

J.J.K.: — Przynajmniej na razie. Na Zachodzie firmy przeszły już przez ten etap. Tam dobre są jedynie rozwiązania takie jakie my proponujemy. Przez Polskę przechodzi fala masowej komputeryzacji, ale są to głównie zastosowania indywidualne. Ludzie kupują sprzęt dlatego, że mogą go kupić — a nie z konieczności wprowadzania informatyki. Komputer w domu to rozrywka. Ale kiedy trzeba będzie naprawdę stosować informatykę potrzebny będzie lepszy sprzęt.

— Zatem uważa Pan, że i tak do ICLa wrócą?

J.J.K.: — Nie chcę być zarozumiały ale uważam, że tak

— Co Pan im zaproponuje?

J.J.K.: — To zależy oczywiście od zastosowania. Dziś proponujemy ICL — Clan — 16-stanowiskowy mikrokomputer pracujący pod systemem operacyjnym UNIX. PC Quattro — jedno- lub wielostanowiskowe i najnowocześniejszy nasz wyrób — DRS 300.

— Nie wszystko to jednak może trafić na nasz rynek...

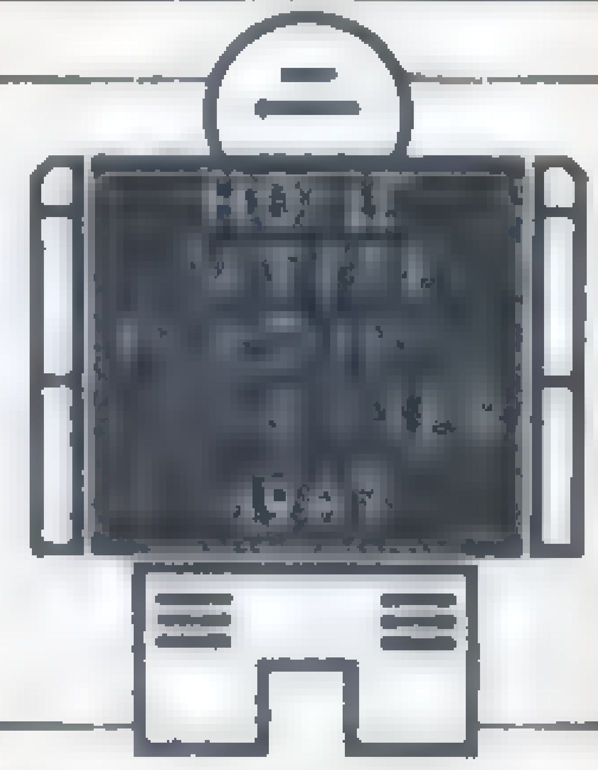
J.J.K.: — Funkcjonują jeszcze zakazy — konsekwencje amerykańskiego embarga. Dotyczy ono między innymi komputerów o pojemności pamięci większej 2.4 Mb. ICL próbuje walczyć w Waszyngtonie o zniesienie tych ograniczeń. To leży w naszym interesie — przecież tradycyjnie gwarantujemy klientom najwyższy poziom — to nasz obowiązek.

— Dziękuję za rozmowę.

Rozmawiał: Wiesław CETERA

Po raz drugi AGPOL zorganizował międzynarodową wystawę „Home Office Personal Computer 1987". W dniach 10—13 lutego br. w Pałacu Kultury i Nauki w Warszawie 70 wystawców z 11 krajów zaprezentowało największą w krajach RWPG ekspozycję techniki komputerowej. Mimo że wystawa miała charakter handlowy i była przeznaczona dla specjalistów, cieszyła się ogromnym powodzeniem wśród wszystkich zwiedzających.

Tym, którzy nie mogli zapoznać się z wystawą „Komputer '87", dedykujemy nasze rozmowy i spostrzeżenia zobrazowane błyskiem flesza.

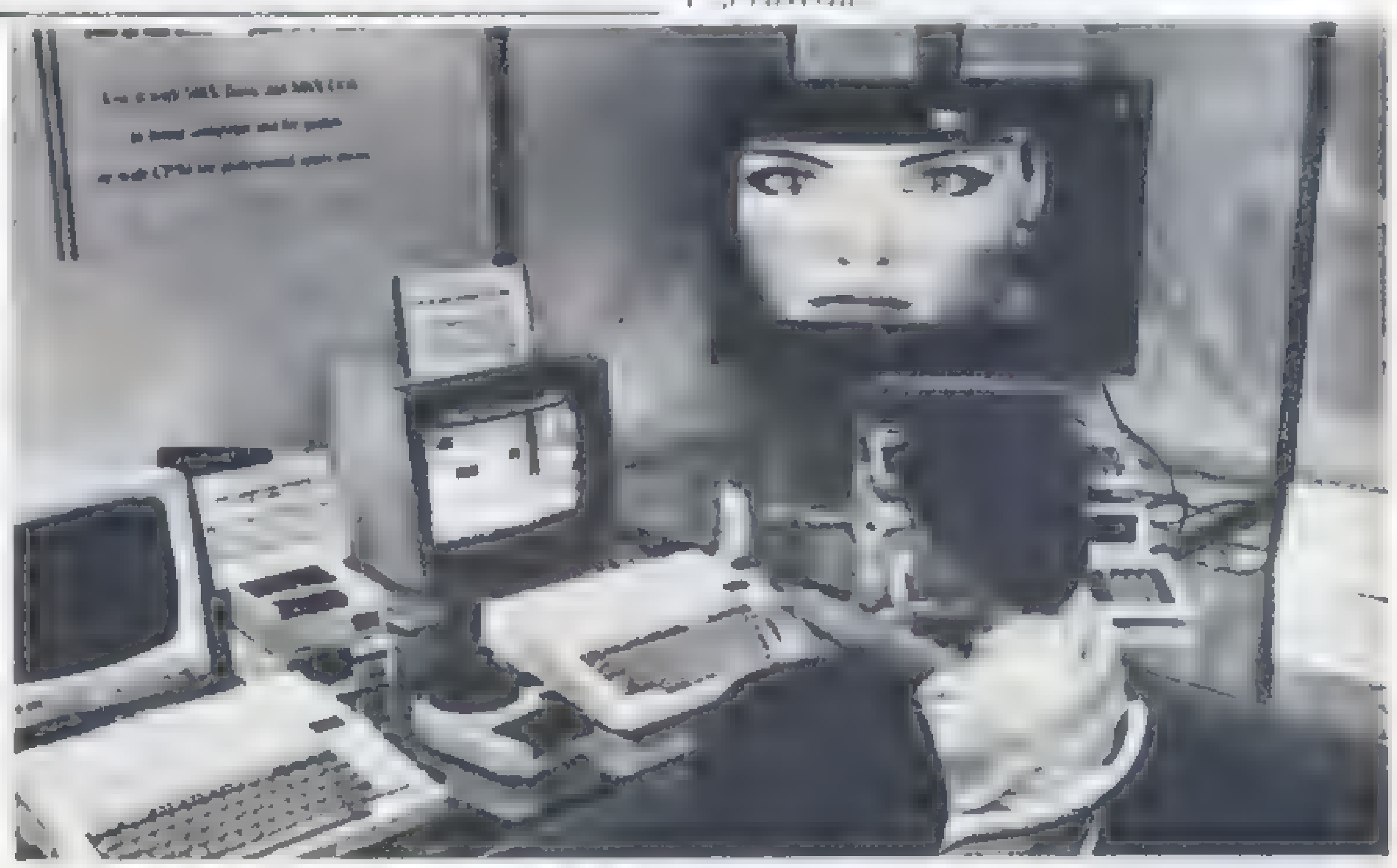


Na ekranie „Komputera'87"

„Prawda" łamana komputerowo...

MERA-BŁONIE
Krzysztof Woliński

— To przyszłość, miejmy nadzieję, że niedaleka. Obecnie w naszej produkcji dominują drukarki. Po odpowiednim przystosowaniu mogą współpracować ze Spectrum, z Commodore i Atari. Mamy kłopoty z tzw. wsadem dewizowym. Dlatego też nasza produkcja nie przekracza 30 tys. sztuk rocznie. Drukarki prezentują przyzwoity europejski poziom. Ponad 80 proc. jest wysyłane na eksport. W kraju łatwiej dostać je np. za lepik, (jeśli akurat jest potrzebny zakładowi) niż za pieniądze. Współpracujemy z zakładami elektronicznymi w Związku Radzieckim. MRAMOR powstał właśnie z tej kooperacji. Pod tym kryptonimem kryje się system komputerowego składu i łamania gazety. I będziemy to produkować. Projektem interesuje się redakcja dziennika „Prawda".





Fotografował: Jan Zełman

„Krówki"... i mikroprocesory...

ALMA — Przedsiębiorstwo Polonijno-Zagraniczne Zakład Produkcji i Oprogramowania Systemów Komputerowych
Wojciech Lipko

— Handlujemy słodyczami, ale nasza oferta obejmuje również komputerowe systemy użytkowe dla szkół i zakładów pracy. Dlaczego aż taka duża rozpiętość — od cukierka po mikroprocesor? Za dolary zdobyte na eksporcie łakoci możemy kupić potrzebny sprzęt i części w dalekiej Azji. Komputery montujemy w Polsce, tworząc na ich bazie sieci lokalne i wielodostępne. Oferujemy systemy przetwarzania informacji dotyczące gospodarki magazynowej oraz wspomagania prac biurowych. Naszym klientom proponujemy sprzęt i usługi w granicach od 4 do 8 milionów złotych. Wszystko zależy od wielkości zamówienia i zakresu wykonywanych prac. Doradzamy, szkolimy obsługi, wykonujemy projekty techniczne i oprogramowanie. Mamy własny serwis. Ceny umowne.

Chcemy wtargnąć do szkół...

ACORN
Komputerowa firma angielska
M. Neugarten

— Firma w Polsce pojawiła się... w grudniu 1986 roku. Zdążyła się już zaaklimatyzować na Śląsku. Na wystawie prezentuje komputery Master 128 i Compact 128. Jest to jedyna ekspozycja ogólnodostępna dla zwiedzających (dzieci tłuką w klawiaturę kilku komputerów od rana do wieczora). Ciekawostka, jaką po raz pierwszy prezentujemy, to płyta wizyjna z odczytem laserowym. Obraz Anglii przedstawiono w postaci 200 tys. fotografii, planów i schematów oraz 150 tys. stron tekstu. A wszystko to na jednej niewielkiej lśniącej płycie. Wyszukiwanie informacji trwa sekundy.

Nasze komputery można porównywać do wyrobów tak znanej firmy jak Amstrad. Nie przegrywają w tej rywalizacji, raczej są lepsze. W Anglii jesteśmy najlepsi w produkcji komputerów edukacyjnych. Zrobimy wszystko, aby jak najszybciej pojawiły się w polskich szkołach.

Nie musimy się reklamować...

GERPOL
Przedsiębiorstwo Polonijno-Zagraniczne
Marek Siwiński

— Nasza oferta jest obszerna. Obejmuje zarówno samostery do jachtów pełnomorskich, jak i elektronikę dla rolnictwa. Współpracujemy z ośrodkami naukowymi we Wrocławiu i Gdańsku. Największym sukcesem firmy jest opracowanie systemu METEOSAD, w którym dane meteorologiczne określa się za pomocą satelity. Zajmujemy się również komputeryzacją w przedsiębiorstwach. Dotyczy to takich działów jak gospodarka magazynowa, finanse, kadry itp. Ofert jest coraz więcej.

Komputery wyciskane z porzeczek...

INTEGRATED MICRO COMPUTERSETS
Szwajcarska Spółka Akcyjna
Elżbieta Duchowicz

— Handlujemy w zasadzie dużymi komputerami profesjonalnymi. Na polski rynek staramy się dostarczyć najnowsze konstrukcje, m.in. posiadamy licencje sprzedaży Fortune System (komputery 32- i 16-bitowe). Współpracujemy tylko z przedsiębiorstwami państwowymi, wymieniając towar za towar. Wiele firm patrzyło na nas z politowaniem, gdy komputery wymienialiśmy na soki porzeczkowe. Po pomyślnym zrealizowaniu przez nas tego typu umowy, inni też rozglądają się za podobnymi kontraktami. Przełamaliśmy pewien stereotyp myślenia. Uważam to za swój osobisty sukces. Chcemy rozszerzyć listę towarów, które można wymieniać na komputery. Jestem pełna optymizmu i wierzę, że pomogą skomputeryzować kraj, w którym się urodziłam.

Notował:
Ryszard ROGOŃ







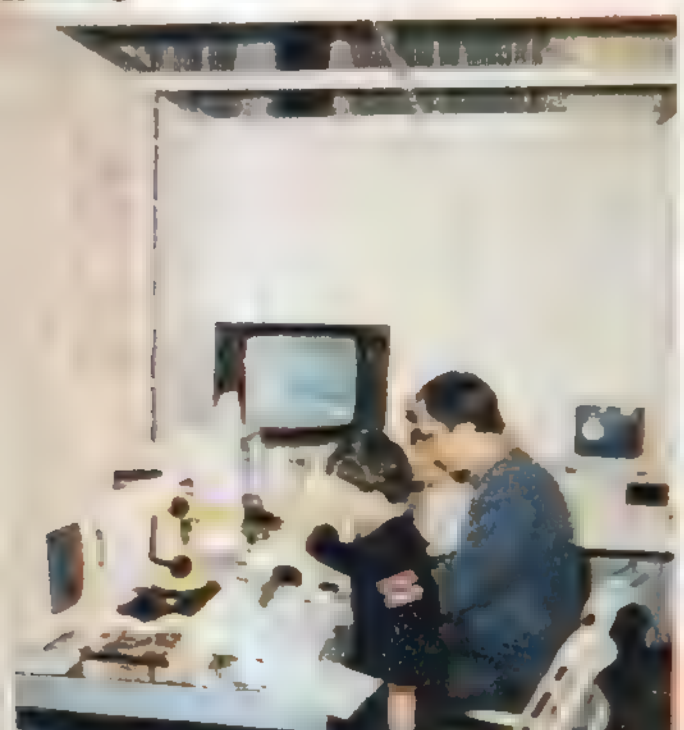


„Komputer '87"

Foto: Jan Zelman



















Początek był nieciekawy



ale w miarę przybywania eksponatów ...było na co popatrzeć



...wykorzystać do pracy, zabawy a przede wszystkim ...porozmawiać z przedstawicielami firm, uzgodnić warunki i... kupić!







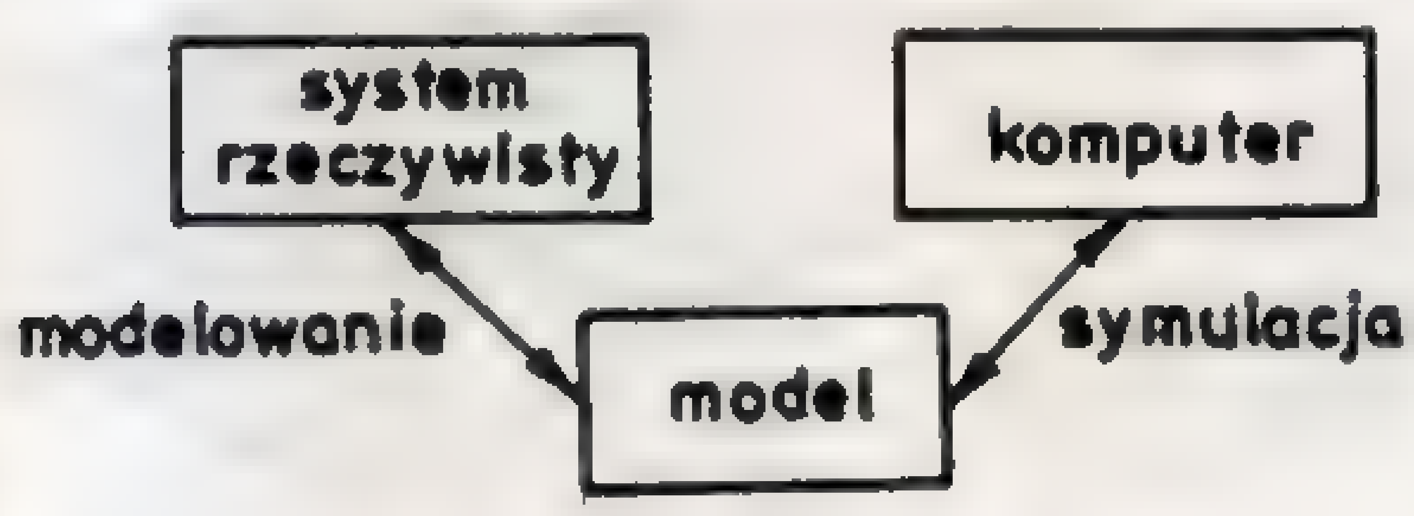


Tekst i foto: Ryszard Rogoń

UDAWANIE RZECZYWISTOŚCI

SYMULACJA jest jedną z ważniejszych i ciekawszych możliwości wykorzystania mikrokomputera. Pozwala ona na przeanalizowanie zjawisk, które w rzeczywistym systemie przebiegają zbyt szybko lub wolno lub też wtedy, kiedy eksperymentowanie z rzeczywistym systemem byłoby zbyt kosztowne, niebezpieczne czy w ogóle niemożliwe (np. niemożliwe jest dowolne kształtowanie podaży i popytu w rzeczywistym systemie ekonomicznym, co łatwo można wykonać przy pracy z modelem, względy moralne uniemożliwiają manipulowanie mózgiem człowieka, natomiast eksperymentowanie z modelem nie stwarza takich ograniczeń). Ponadto komputerowe eksperymenty są całkowicie niedestrukcyjne (nie niszczą systemu rzeczywistego), zapewniają powtarzalność i umożliwiają łatwą zmianę parametrów modelu.

Aby przeprowadzić symulację należy skonstruować model systemu rzeczywistego. **Proces konstruowania modelu nazywamy modelowaniem**. Pojęcie „modelowanie i symulacja" oznacza zespół działań związanych z konstruowaniem modelu systemu rzeczywistego i symulowania go na komputerze. Z definicji tej wynika, że możemy wyróżnić trzy elementy: system rzeczywisty, model i komputer oraz dwie relacje pomiędzy tymi elementami: modelowanie i symulację. Związek ten przedstawiony jest na rys. 1.



Systemem rzeczywistym nazywamy interesującą nas część świata rzeczywistego.

Modelem jest zbiór informacji o systemie zebranych w celu jego zbadania. Model powinien nie tylko odtwarzać zaobserwowaną reakcję systemu rzeczywistego, ale również wiernie odzwierciedlać sposób powstawania tej reakcji w systemie rzeczywistym. Jest to postulat tzw. zasadności strukturalnej modelu.

Istnieje kilka klasyfikacji modeli. Najważniejsze z nich związane są:

— z podstawą czasu, w której występują zdarzenia (modele ciągłe w czasie, modele dyskretne w czasie);

— z zakresami wartości przyjmowanych przez zmienne opisowe (modele ciągłe w stanie, modele dyskretne w stanie);

— ze zmiennymi losowymi (modele determinityczne, modele stochastyczne).

— **Model jest ciągły** w czasie, jeżeli opis czasu dokonuje się w sposób ciągły, tzn. zegar czasu modelu zwiększa swoją wartość o stałe przyrosty.

— **Model jest dyskretny** w czasie, jeżeli czas płynie skokowo, tzn. zegar czasu modelu zwiększa się okresowo o różne wielokrotności umownej jednostki czasu.

— **Model jest ciągły w stanie**, jeżeli jego zmienne przyjmują wartości z pewnego przedziału liczb rzeczywistych (zmieniają się w sposób ciągły).

— **Model jest dyskretny w stanie**, jeżeli jego zmienne przyjmują wartości dyskretne.

— **Model nazywamy deterministycznym**, jeżeli w jego opisie nie występują zmienne; w przeciwnym przypadku model nazywamy stochastycznym lub probabilistycznym.

W niniejszym artykule przedstawiony zostanie prosty przykład symulacji rozwoju społeczeństwa. Jest to model deterministyczny, ciągły w czasie i dyskretny w stanie. Determinizm tego modelu polega na tym, że prawa rozwoju społeczeństwa opisane są ścisłymi matematycznymi formułami, bez zmiennych losowych.

Załóżmy, że w pewnym małym społeczeństwie struktura ludności jest następująca

grupa	wiek	liczba osób
A	0-19	4000
B	20-39	3600
C	40-59	3100
D	60 i więcej	2450

Przyjmijmy również, że wskaźniki śmiertelności i reprodukcji w okresach dwudziestoletnich dla każdej z grup wynoszą

grupa	wiek	WS(%)	WR(%)
A	0-19	5	32
B	20-39	12	67
C	40-59	18	1
D	60 i więcej	80	0

WS-wskaźnik śmiertelności

WR-wskaźnik reprodukcji

Na podstawie takiego systemu rzeczywistego można łatwo skonstruować model matematyczny. Model ten opisuje w postaci czterech równań liczbę ludności w poszczególnych grupach w kolejnych okresach czasowych, n=1, 2, 3, ... N. Przyjmujemy czas trwania jednego cyklu 20 lat.

Równanie (1)÷(4) tworzą matematyczny model systemu

1) $A_{n+1} = 0.32\ A_n + 0.67\ B_n + 0.01\ C_n$
(liczba narodzin, czyli liczba osób, które w następnym cyklu zasilą grupę A)

2) $B_{n+1} = 0.95\ A_n$
(95% populacji z grupy A przeżywa i przechodzi do grupy B)

3) $C_{n+1} = 0.88\ B_n$
(88% populacji z grupy B przeżywa i przechodzi do grupy C)

4) $D_{n+1} = 0.82\ C_n + 0.2\ D_n$
(82% populacji z grupy C przeżywa i przechodzi do grupy D, ponadto 20% populacji grupy D przeżywa do następnego cyklu, tj. więcej niż 20 lat).

Warunki początkowe w chwili T = 0 są następujące:
$A_0 = 4000$, $B_0 = 3600$, $C_0 = 3100$, $D_0 = 2450$

```
2995 REM program "Symul" ########################### (c) Janusz MORBITZER
3000 CLS : BRIGHT 1: PRINT AT 8,7;"Symulacja dynamiki": PRINT AT 11,0;"rozwoju społeczeństwa w zadanej": PRINT AT 14,0;"ilości cykli dwudziestoletnich.": BRIGHT 0: BEEP .5,10: GO SUB 3500
3020 CLS : BRIGHT 1: PRINT AT 0,2;"Załóżmy, że struktura wieku": PRINT AT 2,2;"pewnego małego społeczeństwa": PRINT AT 4,2;"jest następująca (model de-": PRINT AT 6,8;"terministyczny):": BRIGHT 0
3025 PRINT AT 8,0;"********************************": PRINT AT 9,0;"*grupa*    wiek    * liczba osób*": PRINT AT 10,0;"********************************"
3030 PRINT AT 11,0;"*  A  *  0-19  *    4000     *": PRINT AT 12,0;"*------------------------------*"
3035 PRINT AT 13,0;"*  B  *  20-39  *    3600     *": PRINT AT 14,0;"*------------------------------*"
3040 PRINT AT 15,0;"*  C  *  40-59  *    3100     *": PRINT AT 16,0;"*------------------------------*"
3045 PRINT AT 17,0;"*  D  *60 i więcej*    2450     *": PRINT AT 18,0;"********************************"
3050 GO SUB 3500
3055 CLS : PRINT AT 1,0;"  Załóżmy, że zdolność do repro-": PRINT AT 2,0;"dukcji i śmiertelność w okresie": PRINT AT 3,0;"20-letnim dla każdej z grup wy-"
3060 PRINT AT 4,0;"nosi (w %% w odniesieniu do li-": PRINT AT 5,0;" czby ludności z danej grupy):"
3065 PRINT AT 8,0;"********************************": PRINT AT 9,0;"*grupa*    wiek    *WS(%) *WR(%)*": PRINT AT 10,0;"********************************"
3070 PRINT AT 11,0;"*  A  *  0-19  * 5  * 32  *": PRINT AT 12,0;"*  B  *  20-39  * 12  * 67  *": PRINT AT 13,0;"*  C  *  40-59  * 18  * 1  *"
3075 PRINT AT 14,0;"*  D  *60 i więcej* 80  * 0  *": PRINT AT 15,0;"********************************"
3080 PRINT BRIGHT 1;AT 17,4;"WS-wskaźnik umieralności": PRINT BRIGHT 1;AT 19,4;"WR-wskaźnik reprodukcji"
3085 GO SUB 3500
3120 REM struktura ludności
3130 READ a,b,c,d
3140 DATA 4000,3600,3100,2450
3150 REM wskaźniki śmiertelności
3160 READ d1,d2,d3,d4
3170 DATA .05,.12,.18,.8
3180 REM wskaźniki urodzeń
3190 READ b1,b2,b3,b4
3200 DATA .32,.67,.01,0
3210 BEEP .5,10: CLS : PRINT AT 9,2;"Symulacja może być przeprowa-": PRINT AT 11,2;"dzona dla n cykli dwudziesto-": PRINT AT 13,12;"letnich.": PRINT AT 17,3; BRIGHT 1;"Podaj żądaną liczbę cykli:"
3212 INPUT "                n="; LINE n$: LET l=LEN n$: IF l=0 THEN GO TO 3212
3214 FOR i=1 TO l: IF n$(i)>="0" AND n$(i)<="9" THEN NEXT i: LET n=VAL n$: GO TO 3220
3216 GO TO 3212
3220 IF n<=0 THEN GO TO 3212
3230 LET t=0
3250 BEEP .5,10: CLS : PRINT AT 0,0;"________________________________": PRINT AT 1,0;"|cykl| 0-19 |20-39 |40-59| 60+ |": PRINT AT 2,0;"________________________________"
3270 PRINT "|";TAB (4-LEN (STR$ t));t;TAB 5;"|";TAB 7;a;TAB 12;"|";TAB 14;b;TAB 19;"|";TAB 21;c;TAB 25;"|";TAB 27;d;TAB 31;"|"
3275 IF a=0 AND b=0 AND c=0 AND d=0 THEN PRINT "________________________________": BEEP .8,0: PAUSE 50: GO SUB 3500: CLS : PRINT BRIGHT 1;AT 12,6;"Społeczeństwo wymarło": GO SUB 3500: GO TO 3450
3280 IF t>=n THEN GO TO 3360
3290 LET a1=b1*a+b2*b+b3*c+b4*d
3300 LET d=INT ((1-d3)*c)
3310 LET c=INT ((1-d2)*b)
3320 LET b=INT ((1-d1)*a)
3330 LET a=INT a1
3340 LET t=t+1
3350 GO TO 3270
3360 PRINT "________________________________": PRINT : BEEP .8,10: GO SUB 3500
3370 GO TO 3450
3450 RESTORE : CLS : PRINT AT 13,8; BRIGHT 1;"Koniec symulacji": BEEP .8,10: GO SUB 3500: GO TO 3000
3500 REM podprogram
3505 PRINT #1;AT 0,4; BRIGHT 1; FLASH 1;"Naciśnij dowolny klawisz": PAUSE 0: RETURN
```

Przedstawiony program napisany w języku BASIC na ZX Spectrum pozwala na przeprowadzenie symulacji dla dowolnie długiego okresu. Instrukcje 3000 ÷ 3085 służą do wydrukowania informacji ogólnych o problemie i mogą zostać pominięte. Użytkownik ma możliwość zmiany parametrów modelu społeczeństwa:

(a) początkowej struktury ludności (instrukcja DATA w linii 3140);

(b) wskaźników śmiertelności (instrukcja DATA w linii 3170);

(c) wskaźników urodzeń (instrukcja DATA w linii 3200).

W ten sposób można przeprowadzić prognozy demograficzne dla różnych wartości poszczególnych parametrów modelu. Dla parametrów podanych w artykule otrzymujemy następujący przebieg symulacji:

cykl	0-19	20-39	40-59	60+
0	4000	3600	3100	2450
1	3723	3800	3168	2541
2	3769	3536	3344	2597
3	3606	3580	3111	2742
4	3584	3427	3150	2551
5	3474	3404	3015	2582
6	3422	3300	2995	2472
7	3335	3250	2904	2455
8	3273	3168	2860	2381
9	3198	3109	2787	2345
10	3134	3038	2735	2285
11	3065	2977	2673	2242
12	3002	2911	2619	2191
13	2937	2851	2561	2147
14	2875	2790	2508	2100
15	2814	2731	2455	2056
16	2754	2673	2403	2013
17	2696	2616	2352	1970

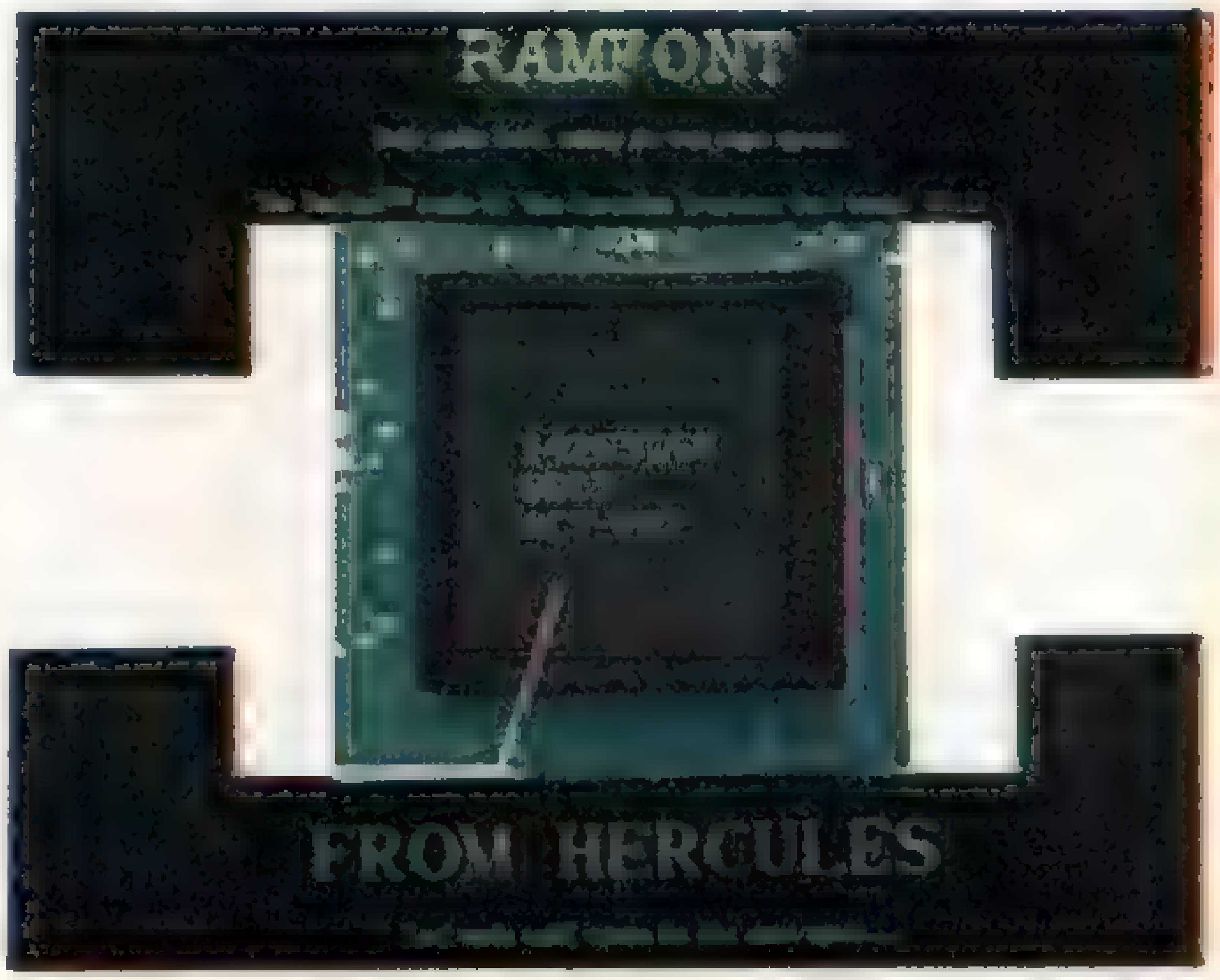
Dokładniejszy model można uzyskać, przyjmując krótszy czas trwania jednego cyklu, np. 5 lat.

Janusz MORBITZER

HERCULES PLUS

Cztery lata temu pojawiła się na rynku karta graficzna firmy Hercules do mikrokomputera IBM PC , która w krótkim czasie stała się światowym standardem . Karta *HERCULES* posiada dwa tryby pracy: tekstowy i graficzny . Tryb tekstowy charakteryzuje się dużą szybkością , lecz ograniczony jest do 256 znaków . Tryb graficzny pozwala na dowolną liczbę znaków , ale do przetwarzania tekstów jest zbyt wolny . Nowy produkt firmy ,karta *HERCULES PLUS*, posiada dodatkowy , trzeci tryb pracy **RAMFont** . Łączy on w sobie szybkość trybu tekstowego z elastycznością trybu graficznego i umożliwia definiowanie 3072 znaków o różnej szerokości i wysokości . Prace nad nową kartą trwały dwa lata . Głównym ich celem było zaprojektowanie procesora V112 . Cena karty *HERCULES PLUS* 299 $.

TANDY 102



Firma TANDY wyprodukowała nowy mikrokomputer przenośny ,oznaczony symbolem 102 .Posiada on 8-bitowy mikroprocesor 80C85 , pracujący z częstotliwością 2,4 MHz , pamięć zapisywalną RAM 24KB i pamięć stałą 32KB . TANDY 102 posiada wskaźnik na ciekłych kryształach LCD ,pozwalający na pracę : w trybie alfanumerycznym w 8 liniach po 80 znaków , w graficznym 240x64 znaki adresowane bezpośrednio . Mikrokomputer posiada klawiaturę QWERTY ,złożoną z 56 klawiszy ,rozbudowaną o 8 klawiszy funkcyjnych . TANDY wyposażony jest w port szeregowy RS-232 i równoległy oraz interfejs do magnetofonu i modemu . Istnieje możliwość podłączenia przenośnej stacji dysków i rozszerzenia RAM-u o 8KB.Cena TANDY 102 wynosi 299 $.

COMPAQ DESKPRO 386

DESKPRO 386 jest komputerem osobistym kompatybilnym z IBM PC . Zastosowano w nim 32-bitowy mikroprocesor *INTEL 80386*,pracujący z częstotliwością 16 MHz .DESKPRO 386 może posiadać pamięć zapisywalną RAM o pojemności do 14 MB . W mikrokomputerze zastosowano kartę graficzną EGA . DESKPRO 386 posiada stacje dysków elastycznych 5 1/4 cala o gęstości zapisu 1,2 MB . Mikrokomputer może mieć dysk stały 40 MB lub 130 MB . DESKPRO wyposażono w dyskowy system operacyjny *DOS* firmy *MikroSoft* . Cena mikrokomputera DESKPRO 386 z dyskiem stałym 130 MB wynosi 9 000 £ .



STATGRAPHICS

Jest to nowy pakiet do IBM PC pozwalający na wizualizację analizowanych danych w formie atrakcyjnych wykresów i diagramów . Zawiera 255 procedur z zakresu analizy regresji, analizy danych eksperymentalnych oraz analizy szeregów czasowych. Współpraca z DBASEIII, LOTUS SYMPHONY, można wykorzystywać zbiory ASCII . Cena 449 $.

opracowali: Jacek Wojtala i Mariusz Jarzębowski

Wyszukiwanie informacji w zbiorze

Wykorzystanie komputera do gromadzenia dużej ilości danych pociąga za sobą konieczność opracowania metod wyszukiwania informacji w zbiorze.

Wyszukiwanie odbywa się zazwyczaj wg pewnej wybranej cechy (lub cech) rekordu, np. nazwiska, tytułu książeczki czy numeru katalogowego. Cechę, wg której realizowane jest wyszukiwanie nazywamy **deskryptorem** lub **kluczem wyszukiwania.**

Zbiór, w którym wyszukujemy informacje może być uporządkowany wg kluczy (np. alfabetycznie) lub nieuporządkowany.

W zbiorze nieuporządkowanym jedyną rozsądną strategią wyszukiwania informacji jest wyszukiwanie proste. Polega ono na sekwencyjnym przeglądaniu zbioru (pobieraniu rekordów) i sprawdzaniu czy natrafiliśmy na żądany rekord (klucz wyszukiwania zgodny z kluczem danego rekordu).

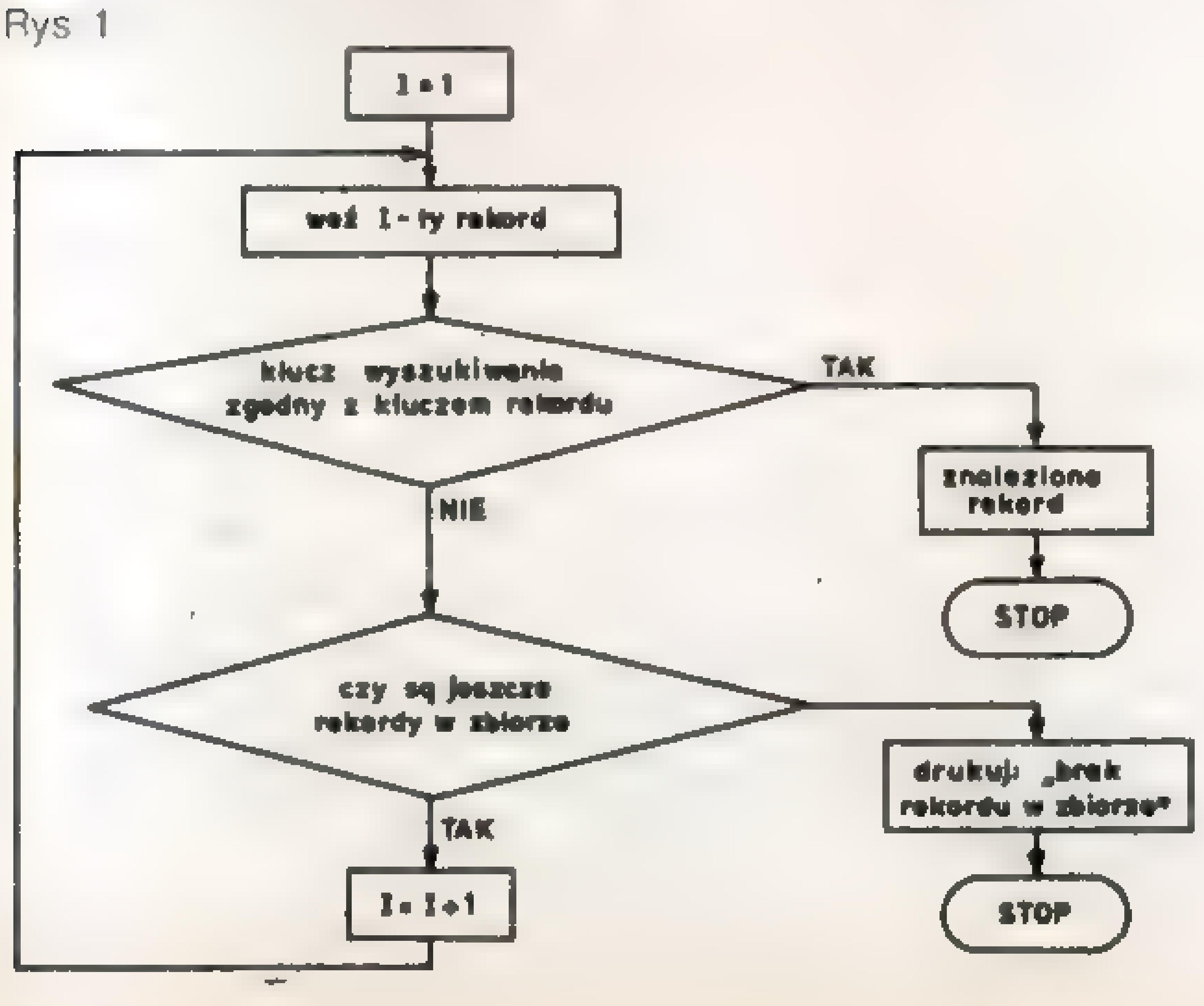
Schemat blokowy takiego wyszukiwania przedstawia rys. 1:

Powyższy schemat przedstawia sytuację, gdy wystarcza nam znalezienie jednego rekordu spełniającego określony warunek. Jeżeli takie rozwiązanie nas nie zadowala, należy sekwencyjnie przeglądnąć wszystkie rekordy w zbiorze.

Pobranie jednego rekordu ze zbioru w celu porównania jego klucza z kluczem wyszukiwania nazywamy **cyklem.** Oszacujmy średnią liczbę cykli niezbędną do odnalezienia informacji w zbiorze metodą wyszukiwania sekwencyjnego. Niech N będzie liczbą rekordów w zbiorze. W najlepszym razie poszukiwany rekord może być odnaleziony w pierwszym cyklu, w najgorszym zaś — w ostatnim, czyli N-tym. Średnia liczba cykli wynosi więc

$$C = (1 + N)/2$$

Rys. 1



Wadą wyszukiwania sekwencyjnego jest jego powolność, zaletą — oprócz prostoty — łatwość aktualizacji zbioru. Zbiór można uzupełniać nowymi rekordami, dopisując je na końcu lub w wolne miejsca; można też usuwać rekordy bez scalania zapisów zakładając, że przy wyszukiwaniu rekordy proste są pomijane.

Jeżeli znane jest uporządkowanie rekordów w zbiorze, możliwe jest zastosowanie szybszych metod wyszukiwania informacji np. metody podziałów połówkowych, zwanej również metodą binarną.

Wyszukiwanie binarne jest przykładem wyszukiwania **dychotomicznego,** tj. procesu, w którym każdy cykl wyszukiwania związany jest z podziałem zbioru na dwie części, z których jedna może być odrzucona. W metodzie tej porównywany jest z kluczem wyszukiwania najpierw klucz rekordu środkowego. Jeżeli wartości kluczy są równe rekord został odnaleziony. Jeżeli wartość klucza rekordu jest większa od wartości klucza szukania, odrzucana jest dolna część zbioru, jeżeli jest ona mniejsza, odrzucana jest górna część zbioru. Odrzuconą część zbioru nazywany „martwą", pozostałą zaś — „żywą". Opisane postępowanie prowadzone jest z kolejnymi „żywymi" częściami zbioru, aż do znalezienia rekordu lub stwierdzenia, że poszukiwanego rekordu nie ma w zbiorze (część „żywa" jest pusta). W każdym cyklu obszar poszukiwań zmniejszany jest o połowę względem poprzedniej jego objętości — stąd też wywodzi się nazwa metody.

Należy zwrócić uwagę, że dokładniej rzecz ujmując w kolejnych cyklach zbiór zmniejszany jest nie o połowę, lecz o połowę —1, gdyż klucz rekordu środkowego był już sprawdzony i w dalszych działaniach może zostać pominięty. Liczba cykli potrzebna do znalezienia rekordu wynosi (dla zbioru o N rekordach) $\log_2 N \leq c \leq 1 + \log_2 N$

Stosowanie metody binarnej jest opłacalne przy większych zbiorach; dla małych zbiorów szybszą może się okazać metoda wyszukiwania sekwencyjnego (należy wziąć również pod uwagę koszt czasowy określenia kolejnego rekordu do sprawdzenia, który jest dużo większy w metodzie binarnej oraz koszt utrzymania rekordów w stanie uporządkowania). Przykładowo, znalezienie rekordu w zbiorze liczącym 1000 rekordów metodą wyszukiwania sekwencyjnego wymaga średnio 500 cykli, podczas gdy zastosowanie metody wyszukiwania binarnego — tylko 9 do 10 cykli. Poniżej przedstawiony został napisany w języku BASIC dla ZX Spectrum algorytm realizujący wyszukiwanie binarne.

Zaletą tego algorytmu jest to, że może on być używany nie tylko do wyszukiwania rekordów, ale również do sortowania zbioru, przy czym sortowanie odbywa się bezpośrednio po wprowadzeniu nowego rekordu tzn. rekordy zapisywane są od razu we właściwym miejscu. Sortowanie takie odbywa się szybko. Wykorzystana jest tu naturalna inercja człowieka przy wprowadzaniu kolejnych rekordów (wprowadzanie rekordów przeplata się z ich sortowaniem). Metoda taka pozwala na uniknięcie dłuższego oczekiwania na posortowanie zbioru, jakie występuje przy tradycyjnym układzie: wprowadzenie wszystkich rekordów — sortowanie całości.

Janusz MORBITZER

Program: Wyszukiwanie binarne

```
 20 REM   wyszukiwanie   binarne
 30 REM p,k - początek i koniec
    żywej części zbioru
 40 REM m-nr rekordu środkowego
 50 REM a$(n,q)- tablica z upo-
    rzadkowanymi rekordami
 60 REM w$ - klucz wyszukiwania
 70 REM d=LEN w$ (0<d<=q)
 80 REM gdzie=0- użycie podpro-
    gramu do wyszukiwania (poz-
    nr znalezionego rekordu)
 90 REM gdzie=1 -użycie podpro-
    programu do sortowania (poz
    -pozycja, na której ma zna-
    leźć się rek. o kluczu w$)
100 REM ************************
110 LET p=1: LET k=n
120 IF (k-p)>=0 THEN GO TO 150
130 IF gdzie=1 THEN LET poz=x:
    RETURN
140 PRINT "Brak rekordu!":
    PAUSE 100: RETURN
150 LET m=1+INT ((p+k-1)/2)
160 IF a$(m, TO d)=w$ THEN LET
    poz=m: PRINT "Znaleziono
    rekord": RETURN
170 IF a$(m, TO d)<w$ THEN LET
    p=m+1: GO TO 120
180 LET k=m-1: GO TO 120
190 REM   koniec   podprogramu
```

Zmienne systemowe ROM Spectrum

ATRYBUTY

BORDCR	23624
ATTR—P	23693
ATTR—T	23695
MASK—P	23694
MASK—T	23696
P—FLAG	23697
K—DATA	23565
TVDATA	23566

Atrybuty ekranu to przede wszystkim kolory, ale niektórzy rozszerzają to pojęcie i wyróżniają:

INK — kolor atramentu, czyli liter i punktów;
PAPER — kolor tła;
BRIGHT — tryb rozjaśniający kolory;
FLASH — tryb cyklicznej wymiany kolorów atramentu i tła powodujący efekt migania.
INVERSE — tryb zamiany kolorów atramentu i tła;
OVER — tryb nakładania punktu na punkt powodujący jego zniknięcie (1 XOR 1 = 0)

Kolory podstawowe (INK, PAPER) skojarzone są z polem tekstowym, co oznacza, że znak na ekranie może posiadać tylko jedną wartość atramentu i tła. Rodzi to wiele kłopotów głównie w trybie graficznym.

Obszar atrybutów ekranu zajmuje tylko 768 bajtów (adresy 22528... 23295). Każdy bajt składa się z ośmiu bitów:

bit	7	6	5	4	3	2	1	0
stan	0	0	1	1	1	0	0	0
wartość	128	64	32	16	8	4	2	1

Bity 0, 1, 2 określają INK, bity 3, 4, 5, PAPER, bit 6 odpowiada za BRIGHT, a bit 7 za FLASH w danym polu tekstowym. W przykładzie powyższym zawartość bajtu wynosi 56, co oznacza PAPER 7, INK 0.

W Spectrum występują dwa pojęcia atrybutu — główny i tymczasowy. Jeśli podamy instrukcję np. PAPER 4: INK 7. CLS ustalamy atrybut główny, jeśli zaś: PLOT PAPER 1, INK 6; 10, 10 ustawiamy tymczasowy dla jednej instrukcji.

Atrybut główny trzyma zmienna ATTR P, tymczasowy — ATTR—T. Pozostałe tryby (INVERSE, OVER) przechowywane są w P—FLAG w następującej konfiguracji:

```
bit     !  znaczenie
--------------------------------
0       !  OVER tymczasowy
1       !  OVER główny
2       !  INVERSE tymczasowy
3       !  INVERSE główny
4       !  INK 9 tymczasowy
5       !  INK 9 główny
6       !  PAPER 9 tymczasowy
7       !  PAPER 9 główny
--------------------------------
```

Tryb 9 atramentu lub tła określa wydruk w kontrastowym kolorze niezależnie od stanu atrybutu głównego.
Stan atrybutów podaje następujący program:

```
9000 LET a=PEEK 23693
9002 LET b=PEEK 23697
9004 PRINT "FLASH ";a>127
9006 LET a=a-128*(a>127)
9008 PRINT "BRIGHT ";a>63
9010 LET a=a-64*(a>63)
9012 PRINT "PAPER ";INT (a/8),CHR$ (32+25*(b>127))
9014 LET b=b-128*(b>127)
9016 PRINT "INK ";a-8*INT (a/8),CHR$ (32+25*(b>31))
9018 LET b=b-32*(b>31)
9020 PRINT "INVERSE ";b>7
9022 LET b=b-8*(b>7)
9024 PRINT. "OVER ";b>1
```

Jeśli **PAPER** lub **INK** zostały wymuszone przez kontrast pojawia się przy nich cyfra 9

Tak się akurat składa, że BASIC zawiera wszystkie potrzebne rozkazy dotyczące atrybutów, przez co wyszczególnione na początku zmienne stosowane są głównie w języku wewnętrznym. Mało wykorzystywaną właściwością Spectrum są tzw. atrybuty maskujące (lub przezroczyste). Ustawia się je komendami **PAPER** 8, **INK** 8, **BRIGHT** 8 i **FLASH** 8. Powodują zamaskowanie bitów odpowiedzialnych za wyróżniony atrybut w pamięci ekranu, więc po wydruku pozostaje on bez zmiany. „Ósemki" przechowują zmienne **MASK—P** i **MASK—T** w podobny sposób, jak w **ATTR—P**. Sprawdźmy te atrybuty

```
 10 LET a$="Każda litera jest widoczna !"
 20 PAPER 7: INK 7: CLS: REM mgła
 30 FOR a=22528 TO 22559
 40 POKE a,INT (RND*128)
 50 NEXT a: PAUSE 0
 80 FOR a=1 TO LEN a$
 90 PRINT a$(a);
110 PAUSE 10
120 NEXT a
```

Wydruk nastąpił, ale nic nie widać, gdyż atrament jest taki sam, jak tło. Zamaskujmy więc przypadkowe atrybuty, które wprowadziliśmy bezpośrednio do pamięci ekranu
60 **POKE** 23694,127: **REM PAPER** 8, **INK** 8, **BRIGHT** 8

Teraz wydruk jest widoczny, choć gdzieniegdzie słabo. Zależy to od stanu atrybutu w danym polu. Dodajmy kontrastowy **INK**
70**POKE** 23697, 32: **REM INK** 9

Obecnie w każdym polu atrament musi kontrastować z tłem, przez co każda litera staje się dobrze widoczna. Jeszcze tylko kropka nad z..
100 **IF** a= 3 **THEN PRINT CHR$** 8, "' ;
70**POKE** 23697, 34: **REM INK** 9, **OVER** 1

Zmienna TVDATA przechowuje pod 23566 ostatnio używany kod kontrolny atrybutu ekranu (16... 21), kod AT (22) lub TAB (23), natomiast pod 23567 pierwszy parametr funkcji AT (czyli numer linii) lub TAB (numer kolumny). Możemy więc zlokalizować linię, w której nastąpił ostatni wydruk:

```
10 PRINT AT INT (RND*22),0;" linia jest przypadkowa ";
20 PRINT FLASH 1;PEEK 23567
30 PAUSE 0: CLS: GOTO 10
```

Atrybuty można również wprowadzać w trybie EXTENDED bezpośrednio z klawiatury. Bajt ostatnio wykorzystywanego w ten sposób koloru lub stanu innych atrybutów (0 lub 1) przechowuje K—DATA.

Poza tym na ekranie widać jeszcze ramkę (BORDER), której kolor umieszczony jest w BORDCR. Wartość tej zmiennej jest jednocześnie atrybutem linii edycyjnych. Oto sposób na zgranie kolorów ramki i edytora.
10 **BORDER 4**
20 **INPUT**;: **REM** czyszczenie dwu linii edycji oraz tła ekra-

nu i edytora:
10 **PAPER** 4: **CLS**
20 **POKE** 23624, 32: **REM PAPER** 4, **INK** 0
30**INPUT**.

Ciekawym efektem jest podzielenie ramki na szereg pasków, a nawet próba jej zgrania z kolorami głównego ekranu

```
10 BORDER 4: INPUT ;
20 FOR a=1 TO 44
30 PRINT PAPER 1+3*(a>22),: NEXT a
40 PAUSE 1: BORDER 1: BORDER 1: BORDER 1: BORDER 1:
   BORDER 1: BORDER 4: BORDER 4: BORDER 4: BORDER 4:
   GOTO 40
```

Maksymalna liczba pasków osiągana w BASIC'u wynosi 10 (w języku wewn. 625). Nieznaczne ruchy ramki przy naciskaniu klawiszy są spowodowane długością czasu odszukiwania klawisza (kolor ramki i stan klawiatury są identyfikowane w tym samym porcie).

W kolejnym odcinku trochę szczegółów o interpretacji i błędach.

Krzysztof MAMCARZ

„W szponach ATARI" dokończenie ze str. 6

Podaje również informację o posiadanych figurach (Mtr1) oraz ocenę aktualnej pozycji (Psnl). Dodatnie wartości świadczą o przewadze komputera

Informacje o naszym ruchu możemy przekazać dwojako:

1) podając współrzędne pola początkowego i końcowego bierki szachowej; po podaniu współrzędnych każdego pola naciskamy RETURN,

2) za pomocą strzałek sterujących przesuwamy kursor na pole, z którego chcemy wykonać ruch, naciskamy RETURN, przesuwamy kursor na pole, na które chcemy wykonać posunięcie i naciskamy RETURN.

Po wykonaniu nieprawidłowego ruchu komputer wyświetla komunikat „Illegal move". Współrzędne wprowadzone pomyłkowo możemy kasować za pomocą klawiszy ESC i DELETE

Komputer sygnalizuje zamiar wykonania ruchu komunikatem „Let me think". Naciśnięcie klawisza ESC, w czasie gdy komputer „myśli" nad ruchem, spowoduje przerwanie analizy możliwych posunięć, wyświetlenie komunikatu „Escape" i wykonanie najlepszego ruchu, jaki został znaleziony do momentu przerwania analizy. Jeśli komputer przewiduje szybką wygraną, uprzedza nas o tym komunikatem „Mate in N" — gdzie N jest liczbą ruchów dzielących nas od mata

Po zakończeniu gry komputer informuje o wyniku: Dawn — remis, checkmate — mat, Stalemat — pat, Time up — przekroczenie limitu czasu. Po tych komunikatach pyta: „What now" (co teraz?) i czeka na wprowadzenie nowych instrukcji

Naciśnięcie CTRL R pozwala na odtworzenie rozegranej partii. Program zatrzymuje się po każdym ruchu na czas od 0 do 20 sekund (podawany przez nas). Odtwarzanie może być w każdej chwili przerwane klawiszem ESC.

Po naciśnięciu CTRL N — nowa gra. Możliwe jest określenie stopnia trudności (CTRL Q) oraz wybranie typu gry (CTRL T). Po CTRL Q komputer pyta, czy ma korzystać z posiadanej książki debiutów (Book?) — odpowiadamy 1 lub 0 (tak/nie). Możemy również określić, czy komputer ma dążyć do zwycięstwa, czy do remisu. Podajemy wartości dla mtrl z przedziału —9,+9 oraz dla psnl z przedziału —60,60; im niższe wartości, tym silniejsze dążenie komputera do zwycięstwa.

Program zawiera 6 typów partii rozgrywanych w różnym tempie i stylu. Po naciśnięciu CTRL T jesteśmy proszeni o wybranie jednego z sześciu wariantów:

1) partia turniejowa — należy podać liczbę ruchów do pierwszej i drugiej kontroli czasu oraz terminy kontroli,

2) partia, dla której określamy średni czas ruchu,

3) partia o zadanym czasie trwania — przekroczenie czasu kończy się porażką,

4) partia równoważna — komputer „myśli" tak długo jak przeciwnik,

5) partia o nieograniczonym czasie — stosowana przy poszukiwaniu najlepszego wariantu ruchu (czas poszukiwania może trwać nawet dobę);

6) rozwiązywanie zadań — komputer pyta o liczbę ruchów do mata

Możliwe są także dodatkowe utrudnienia gry, których celem jest ćwiczenie pamięci. Po naciśnięciu **CTRL V** możemy wybrać jedną z następujących wersji: 1 — niewidoczne białe bierki; 2— niewidoczne czarne bierki; 3 — niewidoczne białe i czarne. Dodatkowo za pomocą **CTRL Z**, możemy wyłączyć dźwięk.

Program jest bardzo elastyczny pod względem wyboru partnerów. Dostępne są następujące warianty gry: komputer — komputer (**CTRL P**), człowiek — człowiek (**CTRL S**), bez podawania czegokolwiek człowiek (**Opponent**) gra z komputerem (**Colossus**). Podczas gry z komputerem możemy zmusić go do wykonania ruchu za nas (**CTRL G**), następuje wtedy odwrócenie szachownicy i kontynuujemy grę bierkami komputera. Zmianę koloru bierek realizujemy po **CTRL O**, a zmianę koloru ekranu — **CTRL C**

W czasie gry możliwe jest cofnięcie ruchu (**CTRL B**), zarówno naszego, jak i komputera, nawet kilkakrotnie. Możliwe jest wtedy wykonanie jednego lub więcej ruchów — **CTRL F**.

Można również „poprosić" komputer o wykonanie nieco gorszego ruchu (następny najlepszy) — **CTRL U**.

W wypadku rozwiązywania różnorodnych zadań szachowych pożądana jest możliwość swobodnego ustawiania figur na szachownicy. Po naciśnięciu **CTRL A** pojawia się komunikat: *„Alter position: White. Command?"*, który sygnalizuje gotowość komputera do ustawiania białych bierek. W tym celu należy ustawić kursor w wybranym polu i nacisnąć jedną z następujących liter: C (czyszczenie pola), P (pionek), N (skoczek), B (goniec), R (wieża), Q (hetman) lub K (król). Naciśnięcie klawisza W usuwa wszystkie bierki. Aby ustawić figury drugiej strony, należy nacisnąć S Wyjście z trybu ustawiania poprzez naciśnięcie E

Jeśli partia szachów przeciąga się i nie mamy czasu na jej dokończenie, możemy przechować na taśmie magnetycznej aktualną sytuację i przełożyć partię na później. Po naciśnięciu CTRL D pojawia się komunikat „Load or Save". W zależności od potrzeby naciskamy L lub S (naciśnięcie innego klawisza powoduje wyjście z tego trybu pracy) Komputer poleca przygotować magnetofon i nacisnąć dowolny klawisz.

Colossus chess 3.0 jest najbardziej interesującym programem szachowym, w wersji taśmowej, jaki udało nam się znaleźć dla komputerów Atari 800 XL i 130 XE.

Tomasz MROWIEC
Ludwik PIELA

Oprogramowanie znacznika świetlnego

Procedury graficzne w systemach komputerowych stosowane są do wprowadzania, przetwarzania i zobrazowania danych w postaci graficznej. Wprowadzanie danych w postaci graficznej jest ułatwione dzięki użyciu urządzeń operatorskich takich jak manipulator kulowy lub pióro świetlne [1]. Użytkownik wskazuje określone punkty na ekranie monitora za pomocą znacznika świetlnego, sterowanego odpowiednio oprogramowanym urządzeniem operatorskim.

Znacznik świetlny może być również sterowany uniwersalną klawiaturą alfanumeryczną. Rozwiązanie takie przedstawione zostanie poniżej.

Wybrany podzbiór znaków klawiatury alfanumerycznej został oprogramowany dla potrzeb sterowania znacznikiem. Oprogramowanie to jest częścią szkoleniowego systemu graficznego (SSG), który został opracowany w języku BASIC i uruchomiony w komputerze MERITUM I.

[1] *Chmurzyński J.: Grafika komputerowa. „IKS", nr 2/1986*

Struktura SSG

Struktura SSG przedstawiona została na rys. 1. W skład SSG wchodzą tablice współrzędnych — rys. 2 oraz procedury, zestawione w tabeli 1. Początkowy fragment programu SSG przedstawiono poniżej:

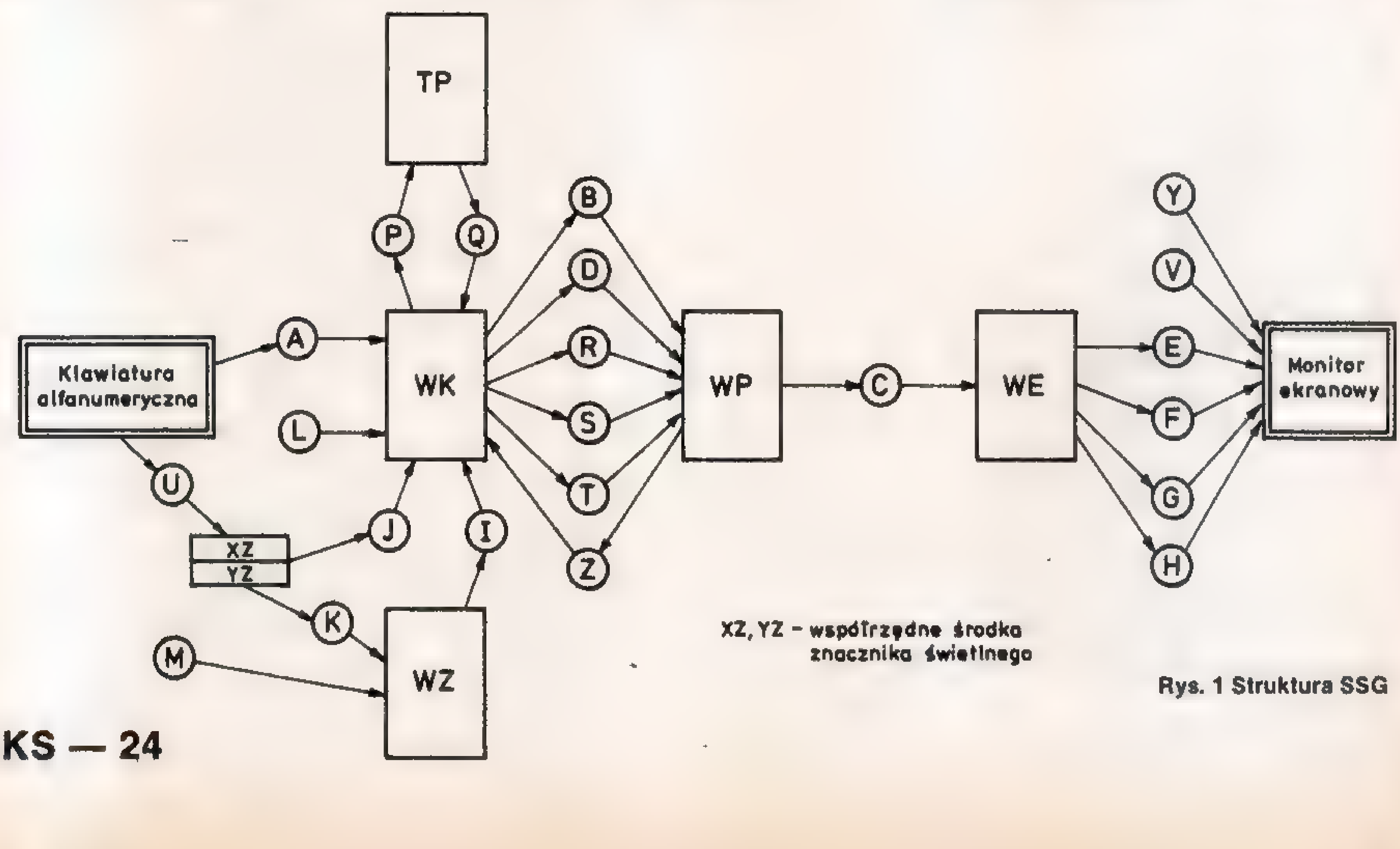
```
  1 REM * * * * * SZKOLENIOWY SYSTEM GRAFICZNY
  2 REM * * * * * OPRACOWAŁ J. CHMURZYŃSKI 1985
  3 REM * * * * * MERITUM 1 W. 1. 0
 10 DIM WK (21,2)
 11 DIM WZ (21,2)
 12 DIM WP (21,2)
 13 DIM WE (21,2)
 14 DIM TP (21,2)
 20 DIM P1 (2)
 21 DIM P2 (2)
 22 DIM P3 (2)
 23 DIM T1 (2,2)
 24 DIM T2 (2,2)
 25 DIM T3 (2,2)
 99 CLS
100 INPUT „NR PROCEDURY"; NP
101 IF NP =  1 GOTO 250
102 IF NP =  2 GOTO 150
103 IF NP =  3 GOTO 153
104 IF NP =  4 GOTO 155
105 IF NP =  5 GOTO 157
106 IF NP =  6 GOTO 170
107 IF NP =  7 GOTO 172
108 IF NP =  8 GOTO  99
109 IF NP =  9 GOTO 290
110 IF NP = 10 GOTO 174
111 IF NP = 11 GOTO 164
112 IF NP = 12 GOTO 168
113 IF NP = 13 GOTO 160
114 IF NP = 14 GOTO 162
115 IF NP = 15 GOTO 400
116 IF NP = 16 GOTO 555
117 IF NP = 17 GOTO 560
118 IF NP = 18 GOTO 166
148 PRINT „ZLY NR PROCEDURY"
149 GOTO 100
```

Poszczególne tablice przechowują następujące dane:

- tablice WK i TP — współrzędne kartezjańskie punktów charakterystycznych obrazu,
- tablica WP — przekształcone współrzędne kartezjańskie punktów charakterystycznych obrazu,
- tablica WE — współrzędne ekranowe punktów charakterystycznych obrazu,
- tablica WZ — współrzędne kartezjańskie punktów charakterystycznych obrazu wskazanych znacznikiem świetlnym

Obieg danych w SSG

Dane w SSG przyjmować mogą postać pojedynczych współrzędnych X, Y punktu lub ciągów współrzędnych X, Y punktów. Punkty te mogą być elementami składowymi obrazu lub też punktami początkowymi i końcowymi odcinków linii prostej.



Rys. 1 Struktura SSG

Tabela 1

Procedury SSG

Nazwa	FUNKCJA
A	Wprowadzenie ciągu N par współrzędnych kartezjańskich z klawiatury alfanumerycznej do tablicy WK
B	Przesyłanie zawartości tablicy WK do tablicy WP bez przekształceń współrzędnych
C	Zamiana współrzędnych kartezjańskich punktów z tablicy WP na współrzędne ekranowe i przesyłanie ich do tablicy WE
D	Zmiana skali współrzędnych punktów z tablicy WK i przesyłanie ich do tablicy WP
E	Zobrazowanie na monitorze ekranowym linii według punktów charakterystycznych z tablicy WE
F	Wygaszenie linii według punktów charakterystycznych z tablicy WE
G	Zobrazowanie na monitorze ekranowym punktów charakterystycznych z tablicy WE
H	Wygaszanie punktów charakterystycznych z tablicy WE
I	Przesyłanie zawartości tablicy WZ do tablicy WK
J	Przesyłanie współrzędnych środka znacznika świetlnego na pierwsze miejsce tablicy WK
K	Przesyłanie współrzędnych środka znacznika świetlnego na kolejne miejsce tablicy WZ
L	Zerowanie tablicy WK
M	Zerowanie tablicy WZ
P	Przesłanie zawartości tablicy WK do tablicy TP
Q	Przesłanie zawartości tablicy TP do tablicy WK
R	Przekształcenie obrazu opisanego punktami charakterystycznymi w tablicy WK: przesunięcie o dany wektor
S	Przekształcenie obrazu opisanego punktami charakterystycznymi w tablicy WK: obrót wokół środka układu współrzędnych
T	Przekształcenie obrazu opisanego punktami charakterystycznymi w tablicy WK: obrót wokół dowolnego punktu
U	Sterowanie znacznikiem świetlnym (patrz rys. 4; tabela 3)
V	Wygaszenie wszystkich punktów na ekranie monitora
Y	Zobrazowanie na ekranie monitora osi X — Y układu współrzędnych
Z	Przesłanie zawartości tablicy WP do tablicy WK

Współrzędne punktów wprowadzane są z klawiatury alfanumerycznej do tablicy WK (procedura A). Drugim sposobem wprowadzania współrzędnych punktów jest użycie znacznika świetlnego, sterowanego klawiaturą alfanumeryczną (procedura U). Współrzędne środka znacznika świetlnego XZ, YZ kierowane są do tablicy WZ (procedura K) oraz zobrazowane są na ekranie monitora (procedury J, B, C, G). Po skompletowaniu ciągu współrzędnych punktów, wskazanych za pomocą znacznika świetlnego, zawartość tablicy WZ może być przeniesiona do tablicy WK (procedura I).

Obraz opisany punktami charakterystycznymi, których współrzędne znajdują się w tablicy WK, może być przekształcony i przesłany do tablicy WP (procedury D, R, S, T). Współrzędne kartezjańskie punktów charakterystycznych z tablicy WP przekształcane są do postaci współrzędnych ekranowych (procedura C). Postać ekranowa współrzędnych służy do wyświetlania punktów lub odcinków linii (procedury G, E) albo też do wygaszania punktów lub odcinków (procedury H, F).

Tablica TP służy do chwilowego przechowywania obrazu.

0	N	—
1	X1	Y1
2	X2	Y2
3	X3	Y3
	⋮	⋮
21	X21	Y21

N — ilość par współrzędnych X, Y znajdujących się w tablicy

Rys. 2 Struktura tablicy współrzędnych

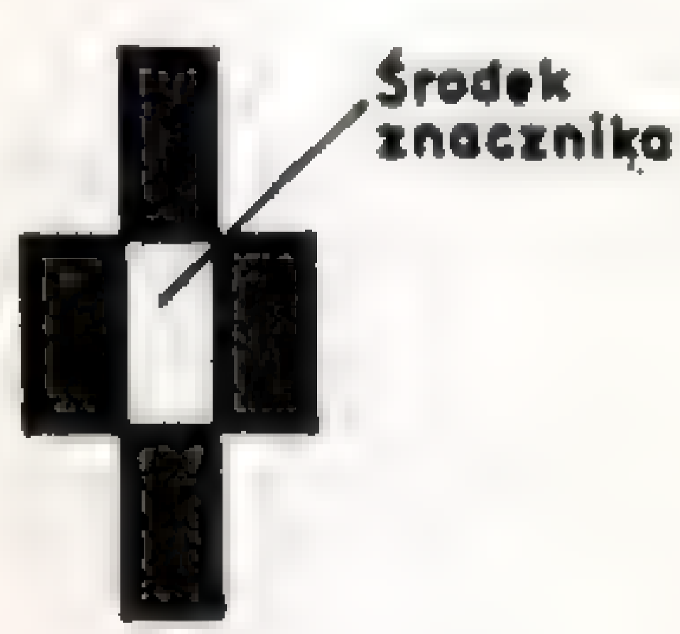
Podczas złożonych transformacji obrazu stosowane jest przesyłanie zawartości tablicy WP do tablicy WK (procedura Z).

Dyrektywy operatorskie SSG

Kolejnością wykonywania procedur SSG sterują dyrektywy operatorskie. Dyrektywy złożone są z identyfikatorów i parametrów. Dyrektywy operatorskie wprowadzane są do SSG przez klawiaturę alfanumeryczną. Identyfikatory dyrektyw przyjmują wartość liczbową z przedziału od 1 do 18. W tabeli 2 zestawiono dyrektywy operatorskie SSG wraz z operacjami wykonywanymi po wprowadzeniu dyrektyw do SSG.

Sterowanie znacznikiem świetlnym

Znacznik świetlny w SSG złożon jest z 4 punktów ułożonych w kształci krzyża (rys. 3). Współrzędne środk



Rys. 3 Znacznik świetlny

znacznika przechowywane są w zmiennych XZ, YZ. Procedura U ustawia początkowo znacznik na środku ekranu. W celu łatwego odróżnienia znacznika od innych elementów obrazu, znacznik migocze. Migotanie znacznika zapewnia program cyklicznego wyświetlania i wygaszania znacznika na ekranie monitora.

Do sterowania znacznikiem świetlnym wydzielono dodatkowe dyrektywy, przedstawione w tabeli 3. Znacznik może być przesuwany ze zmienną szybkością, zależną od skoku znaczni-

ka SZ. Wielkość skoku ustala się dyrektywami sterowania znacznikiem świetlnym 1, 3, 5 i 7.

Algorytm sterowania znacznikiem świetlnym przedstawia rys. 4. Poszczególne bloki algorytmu oznaczono dodatkowo numerami początkowych wierszy procedury sterowania znacznikiem świetlnym (procedura U).

Procedura sterowania znacznikiem świetlnym

Poniżej przedstawiono procedurę sterowania znacznikiem świetlnym w SSG. Procedura ta inicjowana jest dyrektywą operatorską nr 15 i zapewnia sterowanie znacznikiem zgodnie z algorytmem przedstawionym na rys. 4. Procedura ta umożliwia:

- wprowadzenie do tablicy WZ opisu obrazu w postaci ciągu par współrzędnych charakterystycznych,
- wyświetlanie punktów na ekranie monitora w miejscach położenia punktów charakterystycznych (dyrektywa Q),
- wyświetlanie linii łamanej, łączącej punkty charakterystyczne (dyrektywa L).

Po wprowadzeniu wymaganej liczby punktów charakterystycznych dane te mogą być przesłane z tablicy WZ do tablicy WK (procedura 1) i poddane przekształceniom za pomocą procedury SSG.

Tabela 3

Dyrektywy sterowania znacznikiem świetlnym

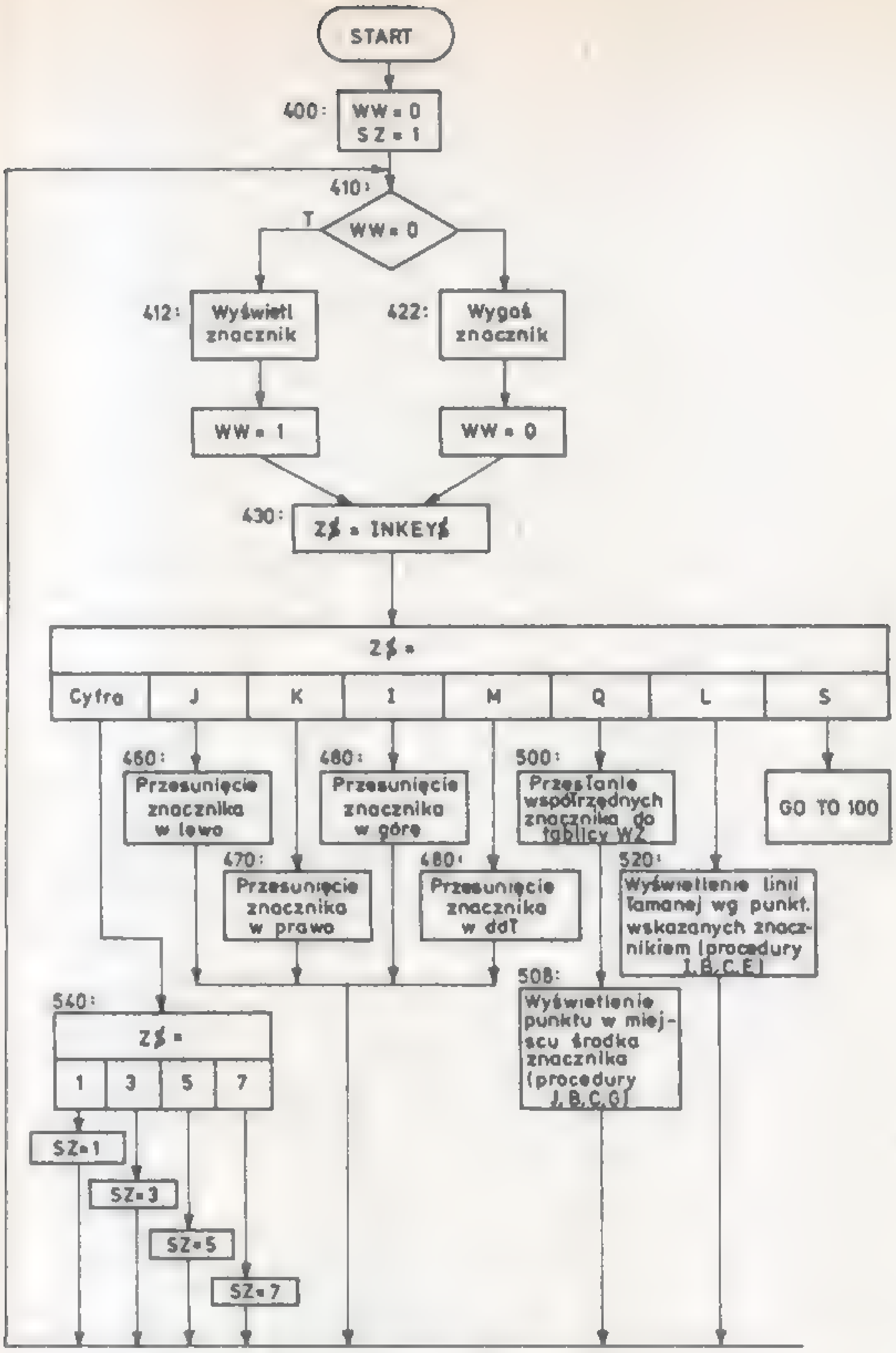
IDENTYFIKATOR DYREKTYWY	OPERACJE DYREKTYWY
J	Przesunięcie znacznika w lewo
K	Przesunięcie znacznika w prawo
I	Przesunięcie znacznika w górę
M	Przesunięcie znacznika w dół
Q	Wykonanie operacji K oraz ciągu operacji J, B, C, G
L	Wykonanie ciągu operacji I, B, C, E
1	Ustalenie skoku znacznika SZ = 1 j. r.
3	Ustalenie skoku znacznika SZ = 3 j. r.
5	Ustalenie skoku znacznika SZ = 5 j. r.
7	Ustalenie skoku znacznika SZ = 7 j. r.
S	Koniec sterowania znacznikiem świetlnym GOTO 100

```
400 REM * * * * * PROCEDURA U
401 WW = 0
402 XZ = 64
403 YZ = 24
405 SZ = 1
405 XP = 64
406 YP = 24
410 IF WW = 0 GOTO 412
411 IF WW = 1 GOTO 422
412 REM * * * * * * WYSWIETL ZNACZNIK
413 XZ = XP
414 YZ = YP
415 WW = 1
417 SET (XZ + 1, YZ)
418 SET (XZ — 1, YZ)
419 SET (XZ, YZ + 1)
420 SET XZ, YZ — 1)
421 GOTO 430
422 REM * * * * * WYGAS ZNACZNIK
423 WW = 0
425 RESET (XZ + 1, YZ)
426 RESET (XZ — 1, YZ)
427 RESET (XZ, YZ + 1)
428 RESET (XZ, YZ — 1)
429 GOTO 430
430 REM * * * * * * SLEDZENIE ZNACZNIKA
431 Z$ = INKEY$
432 IF Z$ = "J" GOTO 460
433 IF Z$ = "K" GOTO 470
434 IF Z$ = "I" GOTO 480
435 IF Z$ = "M" GOTO 490
436 IF Z$ = "Q" GOTO 500
437 IF Z$ = "L" GOTO 520
438 IF Z$ = "S" GOTO 100
449 GOTO 540
450 GOTO 410
460 REM * * * * * * KLAWISZ J
461 XP = XZ — 1 * SZ
462 IF XP — 1 < GOTO 465
464 GOTO 410
465 XP = XZ
466 GOTO 410
470 REM * * * * * * KLAWISZ K
471 XP = XZ + 1 * SZ
472 IF XP + 1 > 127 GOTO 475
474 GOTO 410
475 XP = XZ
476 GOTO 410
480 REM * * * * * * KLAWISZ I
481 YP = YZ — 1 * SZ
482 IF YP — 1 < 0 GOTO 485
484 GOTO 410
485 YP = YZ
486 GOTO 410
490 REM * * * * * * KLAWISZ M
491 YP = YZ + 1 * SZ
492 IF YP + 1 > 47 GOTO 495
494 GOTO 410
495 YP = YZ
```

Dyrektywy operatorskie SSG

Tabela 2

NUMER DYREKTYWY (IDENTYFIKATOR)	OPERACJE DYREKTYWY	PARAMETRY DYREKTYWY
1	A	— LICZBA PUNKTÓW — WSPÓŁRZĘDNE X, Y PUNKTÓW
2	B + C	—
3	E	—
4	F	—
5	D + C	—
6	P	—
7	Q	—
8	V	—
9	Y	—
10	R + C	— DX, DY WEKTOR PRZESUNIĘCIA OBRAZU
11	S + C	— FI — KĄT OBROTU
12	T + C	— DX DY WSPÓŁRZĘDNE PUNKTU OBROTU — FI — KĄT OBROTU
13	G	—
14	H	—
15	U	patrz tabela 3, rys. 4
16	M	—
17	L	—
18	Z	—



Rys. 4 Algorytm sterowania znacznikiem swietlnym

```
496 GOTO 410
500 REM * * * * * * * WYŚWIETL PUNKT
WG ZNACZNIKA
501 WZ (0, 0) = WZ (0, 0) + 1
502 N = WZ (0, 0)
503 WZ (N, 0) = XZ — 64
504 WZ (N, 1) = (24 — YZ) * 1.845
505 WK (0, 0) = 1
506 WK (1, 0) = XZ — 64
507 WK (1, 1) = (24 — YZ) * 1.845
508 GOSUB 260
509 GOSUB 280
510 GOSUB 380
511 GOTO 410
520 REM * * * * * * * KRESLENIE LINII
ZA ZNACZNIKIEM
521 FOR I = 1 TO WZ (0, 0)
522 WK (I, 0) = WZ (i, 0)
523 WK (I, 1) = WZ (I, 1)
524 NEXT I
525 WK (0, 0) = WZ (0, 0)
526 GOSUB 260
527 GOSUB 280
528 GOSUB 300
529 GOTO 410
540 REM * * * * * * * ZMIANA SZE-
ROKOŚCI PRZESUWU ZNACZNIKA
541 IF Z$ = "1" GOTO 546
542 IF Z$ = "3" GOTO 548
543 IF Z$ = "5" GOTO 550
544 IF Z$ = "7" GOTO 552
545 GOTO 410
546 SZ = 1
547 GOTO 410
548 SZ = 3
549 GOTO 410
550 SZ = 5
551 GOTO 410
552 SZ = 7
553 GOTO 410
```

Programy , pozostałych procedur SSG przedstawione zostaną w drugiej części artykułu.

Jerzy CHMURZYŃSKI

SAMI PISZEMY GRY

Cezary SOBCZAK

Znając podstawowe instrukcje języka BASIC, spróbujmy sami napisać grę Zaczynamy od wymyślenia scenariusza. A oto przykład: *podróżując rakietą po wszechświecie w poszukiwaniu przygód, w pewnym momencie nasz czujnik pokładowy zasygnalizował, że kończy się paliwo. Na nasze szczęście w pobliżu znalazła się planeta, o której nasz komputer dał nam informację, iż znajdują się na niej składniki niezbędne do wytworzenia potrzebnego nam paliwa. Lecz cóż to? Nasze czujniki wykryły obecność na planecie zbiorników ze skumulowaną antymaterią i spotkanie z takim zbiornikiem grozi wybu*chem. Wprawdzie komputer pokładowy wyświetla nam położenie niebezpiecznych zbiorników, lecz pojawiające się zakłócenia powodują to, że ich rozmieszczenie po pewnym czasie staje się niewidoczne. Wybierając się w drogę musimy zatem zapamiętać ich rozmieszczenie i tak się poruszać, aby się z nimi nie zderzyć.

Przystępujemy teraz do budowy gry Komputer osobisty AMSTRAD CPC 6128 może wyświetlać znaki w trzech trybach (wielkościach): tryb 0, tryb 1 i tryb 2.

W trybie 0 (MODE 0) może zapisać 20 znaków w linii,
w trybie 1 (MODE 1) może zapisać 40 znaków w linii,
w trybie 2 (MODE 2) może zapisać 80 znaków w linii,

w każdym trybie można zapisać 25 linii.

Aby zrozumieć różnice pomiędzy poszczególnymi trybami, po włączeniu komputera (automatycznie ustawiany jest tryb 1) wyświetl kilka dowolnych znaków. Następnie zmień tryb pisząc np. MODE 0 i wyświetl znów kilka znaków. Zobaczysz, że w trybie 2 znaki są najmniejsze, w trybie 0 — największe. Do naszej gry wybieramy tryb 0 (MODE 0), a więc mamy do dyspozycji pole ekranu o wymiarach 20 na 25 znaków Przystępujemy do precyzowania założeń gry, ustalania zmiennych i tablic oraz rysujemy schemat gry.

Załóżmy, że:

— nasza załoga składa się z trzech osób, które pojedynczo wychodzą na

poszukiwanie składników paliwa,

— rozlokowano 10 niebezpiecznych zbiorników,

— na planecie znajduje się 5 składników naszego paliwa, z czego do jego wytworzenia potrzeba nam przynajmniej trzech.

Lista zmiennych i tablice:

zal — określa aktualną liczbę członków załogi,

n — ilość rozlokowanych zbiorników,

m — ogólna ilość rozmieszczonych na planecie składników paliwa,

k — niezbędna ilość składników potrzebnych do wykonania paliwa (k ≤ m), a więc umożliwiających start rakiety i opuszczenie planety,

l — ilość aktualnie zebranych składników paliwa, l ≥ k,

minx (n) — tablica współrzędnych x zbiorników

miny (n) — tablica współrzędnych y zbiorników

palx (m) — tablica współrzędnych x składników paliwa

paly (m) — tablica współrzędnych y składników paliwa

Rozpoczynając grę, kolejny członek załogi znajduje się w lewym górnym rogu ekranu (współrzędne x = 1, y = 2; wiersz 1 rezerwujemy na wyświetlanie informacji np. o pozostałych przy życiu członkach załogi itp.), natomiast rakieta w prawym dolnym rogu ekranu (x = 20, y = 25). Rys. 1.

Przystępujemy do kodowania naszego programu. Ustalmy jeszcze, że podprogramy będą rozpoczynały się od linii 1000.

10 **rem** nadanie wartości początkowych
20 **mode** 0: zal=3:n=10:m=5:k=3:l=0
30 **dim** minx (n) : **dim** miny (n) : **dim** palx (m) : **dim** paly (m)

Generujemy losowe współrzędne rozmieszczenia niebezpiecznych zbiorników oraz składników paliwa. Wykorzystamy instrukcję **RND** obliczającą kolejną liczbę losową z przedziału (0—1). Ponieważ współrzędne muszą być liczbami całkowitymi i z zakresu odpowiednio x od 1 do 20 i y od 2 do 25, wygenerowane liczby mnożymy przez 100, zaokrąglamy do liczb całkowitych i badamy przynależność do określonego przedziału. Może się zdarzyć, że wygenerowane współrzędne zbiornika będą identyczne ze współrzędnymi składnika paliwa. Można tego uniknąć wprowadzając odpowiednie badanie i zmieniając współrzędne jednego z elementów, ale pozbawiłoby to dodatkowego elementu „dreszczyka" naszej gry. Może się również zdarzyć, że wygenerowane współrzędne zbiornika będą identyczne z początkowymi współrzędnymi naszego ludzika (1,2) oraz położenia rakiety (20,25). W tym wypadku należałoby jednak zmienić położenie niebezpiecznego zbiornika, gdyż nie dałoby to nam szansy rozpoczęcia bądź zakończenia gry.

Napiszemy teraz podprogram generujący współrzędne,

1000 **rem** podprogram wyznaczania współrzędnych,
1010 x = **cint** (**rnd*** 100) : **if** x < — 1 **or** x > — 20 **then** 1010
1020 y = **cint** (**rnd*** 100) : **if** y < — 2 **or** y > — 25 **then** 1020
1025 **if** (x = 1 **and** y = 2) **or** (x = 20 **and** y = 25) **then** 1010
1030 return

który wykorzystamy do zobrazowania ustawienia zbiorników. Jako znak niebezpiecznego zbiornika przyjmujemy jeden ze znaków naszego komputera oznaczony numerem 231: 40 **rem** generowanie i wyświetlanie zbiorników

50 **for** i = 1 to **n**
60 **gosub** 1000
70 minx (i) = x : miny (i) = y
80 **locate** x,y : **print chr$** (231); : **next**

Dopisz teraz wiersz

85 **for** a = 1 **to** 1000 : **next**:**cls**:**goto** 40

i uruchom program. Zobaczysz, jak wyglądają kolejne warianty rozmieszczenia niebezpiecznych zbiorników

Przerwij program naciskając dwukrotnie klawisz [ESC], **wykasuj wiersz 85**, napisz mode 1 lub mode 2 i piszemy nasz program dalej.

Do zakrycia („utajnienia") rozmieszczenia zbiorników wykorzystamy rozkazy grafiki komputera. W tym wypadku ekran podzielony jest na 640 punktów (tzw. pixeli) w poziomie i 400 punktów w pionie. Początkowe współrzędne kursora grafiki (x = 0, y = 0) umieszczone są w lewym dolnym rogu ekranu. Pole ze zbiornikami będziemy zasłaniać „kurtyną" przemieszczającą się z lewej strony ekranu na prawą. Piszemy:

90 **for** i = 0 **to** 640 **step** 3
100 **move** i, 1 : **drawr** 1,380
120 **next**
125 **goto** 125

Uruchom program i zobacz, jak zasłaniane są niebezpieczne zbiorniki. Naciśnij dwa razy klawisz [ESC], **wykasuj wiersz 125**, napisz mode 1 lub mode 2. Aby nieco utrudnić, zakrywajmy pole jednocześnie z dwóch stron:

110 **move** 640—i, 1: **drawr** 1,380 : **if** i > 320 **then** 130

Spróbuj sam dopisać instrukcje powodujące zasłanianie pola z czterech stron naraz.

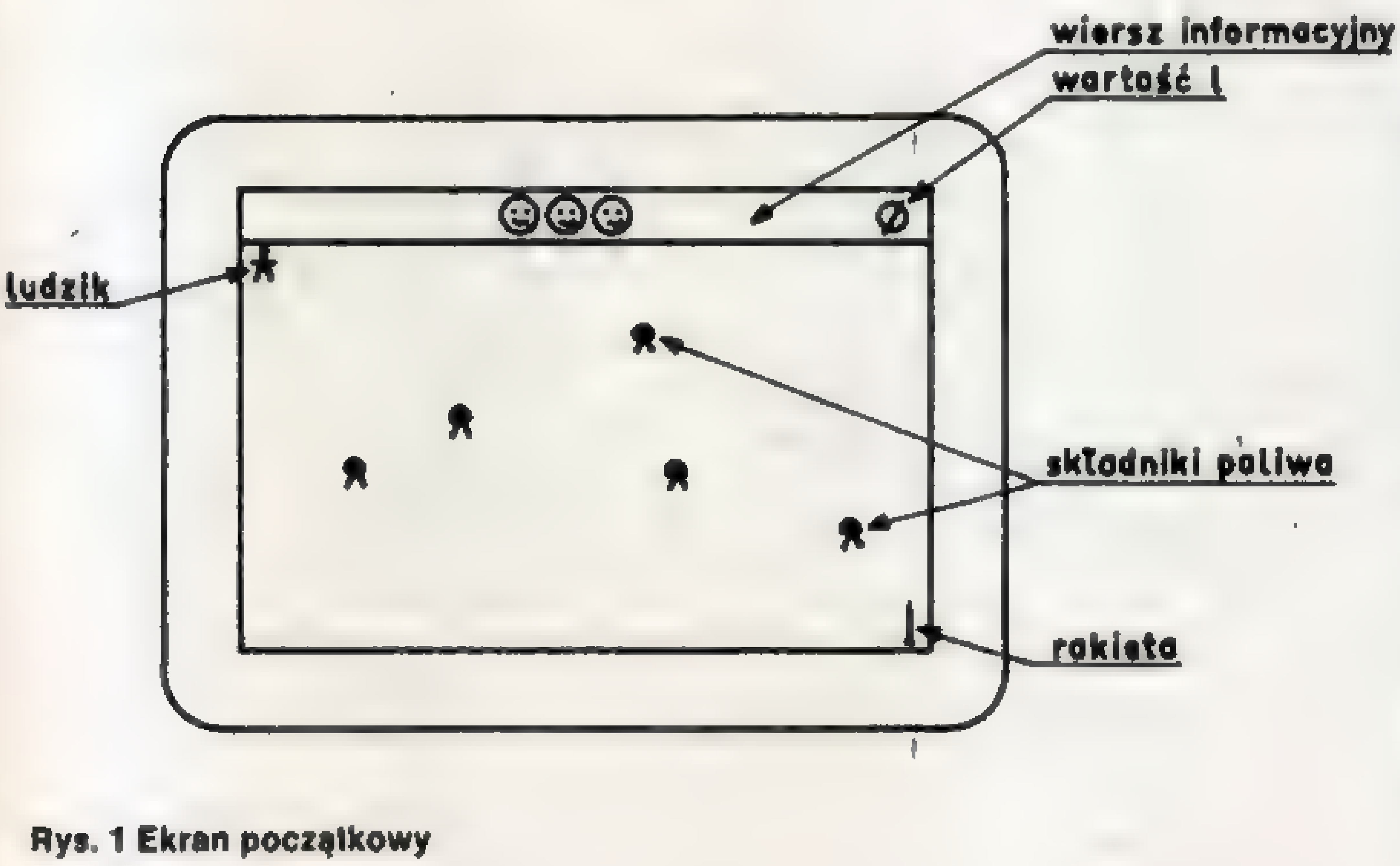
Skoro niebezpieczne zbiorniki są już niewidoczne, umieśćmy na naszej planecie komponenty paliwa, które musimy zebrać. Wykorzystamy znak o numerze 229.

130 **rem** generowanie i wyświetlanie składn. paliwa
140 **for** i = 1 **to** m
150 **gosub** 1000
160 **palx** (i) = x : **paly** (i) = y
170 **locate** x,y : **print chr$** (229); : **next**
175 **goto** 175

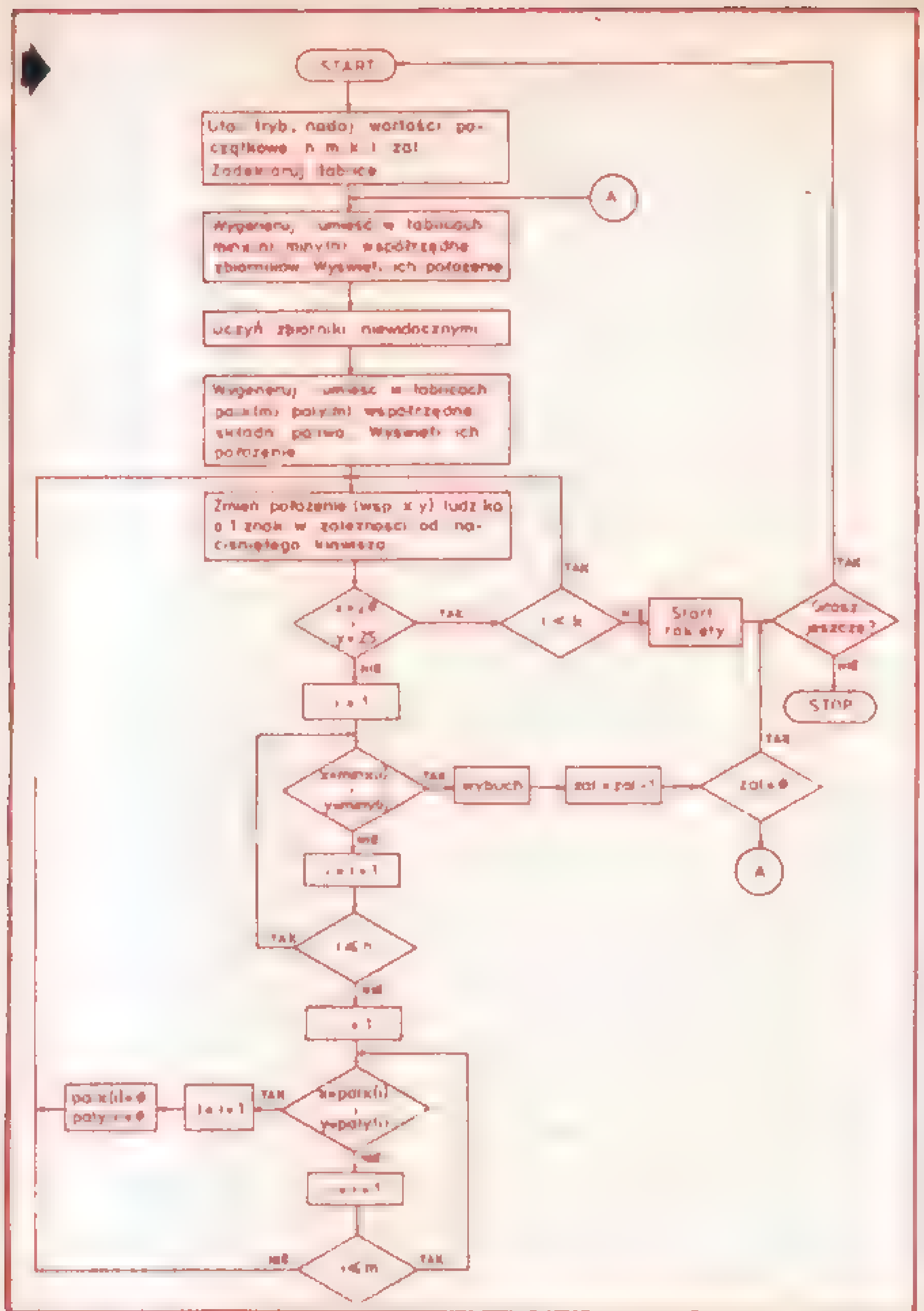
Uruchom program i zobacz jeden z możliwych wariantów rozmieszczenia składników paliwa. Naciśnij dwa razy klawisz [ESC], **skasuj wiersz 175**, napisz mode 1 lub mode 2. Umieścimy teraz na odpowiednich miejscach naszego ludzika (znak numer 249) i rakietę (znak numer 239) oraz napiszemy podprogram wyświetlający informację o pozostałych przy życiu członkach załogi:

180 x = 1 : y = 2 **locate** x,y : **print chr$** (249); : **locate** 20, 25 : **print chr$** (239);
190 **gosub** 1130
1130 **rem** podaje ilość pozostałej przy życiu załogi
1140 **locate** 18,1 ; **print** 1;
1150 **for** i = 1 **to** zal
1160 **locate** 8 + i, 1 : **print chr$** (224); : **next** : **return**

Przystępujemy do sterowania ruchem ludzika w zależności od naciśniętego klawisza. Wszystkim klawiszom komputera



Rys. 1 Ekran początkowy



Schemat gry

dokończenie na str. 30

W naszym komputerlandzie

W naszym komputerlandzie trwał właśnie niczym nie zmącony świąteczny nastrój. Z różnych stron napływały na ręce pracowitego Spektrusia urodzinowe życzenia, głównie by kolejne lata były wesołe, zdrowe i pogodne, wypełnione nieskończonym pasmem sukcesów.

Atmosferę tę zakłócił jednak telegram oznaczony symbolem „PILNY". Jego treść początkowo obraźliwa, następnie życzliwa i radosna, w miarę dalszego czytania stawała się wulgarna, a na końcu nawet tragiczna — ogólnie jednak mało zrozumiała. Brzmiała ona:

> stop OCH NIEROBY stop SERDECZNE ŻYCIE NA stop WIELE CZĘŚCI DO MASZYNKI stop MI DUPA SIEKIERA DOŚĆ NAKOPAŁ stop PIOTRUŚ NIE ŻYJE stop PRZYJEDŹCIE.

Telegram wywołał w zespole robotów ogromny niepokój i poruszenie. Podczas gdy gorączkowo zaczęto się zastanawiać co począć, jeden z robotów nie przejmując się poddawał pod wątpliwość prawdziwość treści, w związku z czym przestawił się na kontestację. Rozmyślając nad tekstem wykonał on we współpracy z komputerem operację zwaną tłumaczeniem telegramu. Okazało się, że właściwa treść telegramu jest następująca:

> stop KOCHANE ROBOTY stop SERDECZNE ŻYCZENIA stop WIELE SZCZĘŚCIA I W DOMU SZYNKI stop MIODU PASIEKĘ RADOŚCI NA KOPY stop SPEKTRUŚ NIECH ŻYJE stop PRZYJACIEL

No! Roboty odetchnęły z ulgą

Podglądał: Eugeniusz MLECZAK





OLIVETTI M19

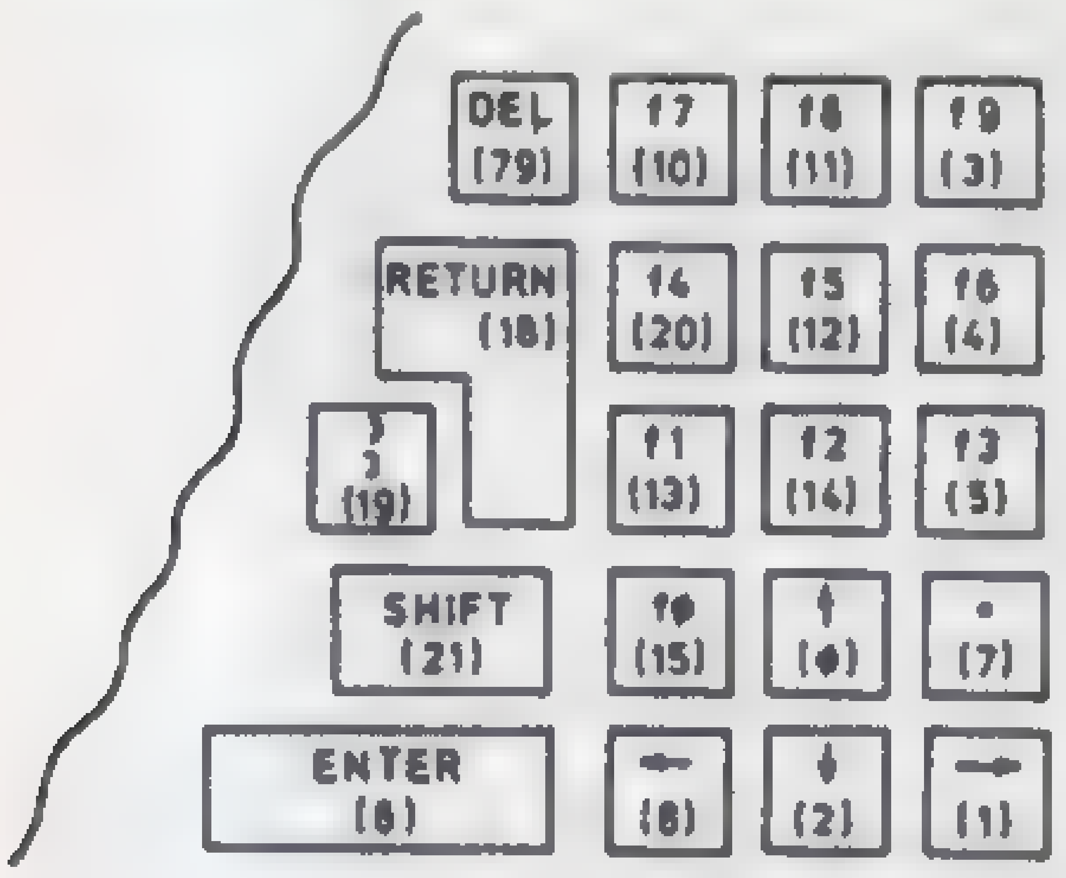
Komputer klasy XT przeznaczony do użytku jako komputer edukacyjny. Pracuje pod systemem MS-DOS 2.11 lub 3.1. M19, ma RAM 256 Kb natomiast pamięć zewnętrzną stanowić mogą floppy dyski po 360 Kb lub floppy dysk 360 Kb plus dysk twardy 20 Mb. Jeśli cena tego sprzętu będzie przyzwoita to ma szansę na szerokie zastosowanie nie tylko w ośrodkach naukowych, ale również w szkołach. Bogate oprogramowanie i możliwość współpracy z wieloma urządzeniami zewnętrznymi są najważniejszym atutem M19.

OLIVETTI M28

Nowy model opracowany w firmie Olivetti, kompatybilny z IBM. Może pracować pod MS-DOS i XENIX V.286 SCo — po raz pierwszy oferowanym w Polsce zgodnym ze standardem światowym. Bogate oprogramowanie narzędziowe jest atutem tego urządzenia. Pozostałe parametry: RAM 512 Kb MOS ROM z możliwością rozbudowy do 1 Mb; dwa tryby znakowe na monitorze monochromatycznym lub kolorowym — 80×25 lub 40×25 znaków; rozdzielczość obrazu 640×400, 640×200 lub 320×200 punktów.

dokończenie ze str. 29

przyporządkowane¹ są określone numery (rys. 2). Instrukcja **INKEY** sprawdza, które klawisze zostały naciśnięte i generuje odpowiednią wartość odpowiedzi w zależności od naciśniętego klawisza. W naszej grze wykorzystamy klawisze sterujące ruchem kursora znakowego (strzałki w lewo, w prawo, w górę, w dół) mające numery odpowiednio: 8, 1, 0, 2



Zasada sterowania ruchem ludzika jest następująca: w zależności od naciśniętego klawisza wyznacza się nowe współrzędne położenia ludzika (zwiększone lub zmniejszone o jeden znak), drukowany jest jego symbol graficzny, natomiast w miejscu określonym poprzednimi (starymi) współrzędnymi drukowana jest spacja. Aby ludzik „nie uciekł" nam z ekranu wprowadzimy odpowiednie ograniczenia na jego współrzędne.

Ujmiemy to w następujące podprogramy.

1040 **rem** w lewo
1050 **if inkey** (8) = 0 **and** x > 1 **then gosub** 1100 : x = x — 1
1055 **rem** w prawo
1060 **if inkey** (1) = 0 **and** x < 20 **then gosub** 1100 : x = x +1
1065 **rem** do góry
1070 **if inkey** (0) = 0 **and** y > 2 **then gosub** 1100 : y = y — 1
1075 **rem** w dół
1080 **if inkey** (2) = 0 **and** y < 25 **then gosub** 1100 : y = y + 1
1085 **locate** x,y : **print chr$** (249);
1090 **return**
1100 **locate** x,y : **print** " ";
1110 **sound** 3,500,1,,,13
1120 **return**

Napisz jeszcze:
200 **gosub** 1040
205 **goto** 200

Uruchom program i naciskając klawisze ze strzałkami zobaczysz jak ludzik porusza się po ekranie.

Naciśnij dwa razy klawisz [ESC] **skasuj wiersz 205,** powróć do poprzedniego trybu.

Piszemy nasz program dalej zgodnie ze schematem gry:

210 **rem** dojście do rakiety
220 **if** x = 20 **and** y = 25 **then** 420
230 **rem** wejście na niebezpieczny zbiornik
240 **for** i = 1 **to** n
250 **if** x = minx (i) **and** y = miny (i) **then** 340
260 **next**
270 **rem** trafienie na składnik paliwa
280 **for** i = 1 **to** m
290 **if** x = palx (i) **and** y = paly (i) **then** palx (i) = 0 : paly (i) = 0 : l = l + 1 **locate** 18,1 : **print** l ; **goto** 200
300 **next**
310 **goto** 200
420 **rem** po dojściu do rakiety
430 **if** l < k **then** 580
440 **rem** start rakiety
450 **for** w = 1 **to** 150
460 **sound** 1,500—w, 2, 15—cint (0,1 * w)
470 **if** w > 19 **then** 510
480 **locate** 10,25 — w : **print chr$** (239); : **cls**
490 **ink** 0, 9, 24 : **locate** 5, 12 : **print** „KONIEC"
500 **ink** 0, 1 : **ink** 1, 26
510 **next**
520 **mode** 1 : **print** „GRASZ JESZCZE? (T/N)"
530 **d$** = **inkey$**
540 **if** d$ = "" **then** 530
550 **if** d$ = "T" **or d$** = "t" **then run**
560 **if** d$ = "N" **or** d$ = "n" **then** 640
570 **gotto** 530
340 **rem** wybuch
350 **sound** 3,500,120,,,,10
360 **for** i = 1 **to** 7
370 **locate** x,y : **print chr$** (144); : **cls locate** x,y : **print chr$** (239); : **ink** 0,9,24 : **next** : **ink** 0,1 : cls
380 zal = zal — 1
390 **if** zal = 0 **then** 520
400 **gosub** 1130 : **goto** 40
580 **rem** za mało składników
590 **locate** 2,1 : **print** "ZA MALO SKLADNIKOW";
600 **for** a = 1 **to** 1000 : **next**
610 **locate** 2,1 : **print space$** (15);
620 **locate** 20,25 : **print chr$** (239);
630 x = x —.1 : **gosub** 1130 : **goto** 200
640 **mode** 2

Uporządkujmy numerację linii naszego programu pisząc **RENUM** i ostatecznie program nasz wygląda następująco

```
10 REM
20 MODE 0:zal=3:n=10:m=5:k=3:l=0
30 DIM minx(n):DIM miny(n)
40 DIM palx(m):DIM paly(m)
50 REM
60 FOR i=1 TO n
70 GOSUB 670
80 minx(i)=x:miny(i)=y
90 LOCATE x,y:PRINT CHR$(231);:NEXT
100 FOR i=0 TO 640 STEP 3
110 MOVE i,1:DRAWR 1,380
120 MOVE 640-i,1:DRAWR 1,380:IF i>320 TH
EN 140
130 NEXT
140 REM
150 FOR i=1 TO m
160 GOSUB 670
170 palx(i)=x:paly(i)=y
190 LOCATE x,y:PRINT CHR$(229);:NEXT
190 x=1:y=2:LOCATE x,y:PRINT CHR$(249);
200 LOCATE 20,25:PRINT CHR$(239);
210 GOSUB 840
220 GOSUB 710
230 REM
240 IF x=20 AND y=25 THEN 450
250 REM
260 FOR i=1 TO n
270 IF minx(i)=x AND miny(i)=y THEN 36
280 NEXT
290 REM
300 FOR i=1 TO m
310 IF palx(i)=x AND paly(i)=y THEN 340
320 NEXT
330 GOTO 220
340 palx(i)=0:paly(i)=0:l=l+1:LOCATE 18,
1
350 PRINT l:GOTO 220
360 REM
370 SOUND 3,500,120,,,,10
380 FOR i=1 TO 7
390 LOCATE x,y:PRINT CHR$(144);:CLS:LOCA
TE x,y
400 PRINT CHR$(238);:INK 0,9,24:NEXT
410 INK 0,1:CLS
420 zal=zal-1:l=0
430 IF zal=0 THEN 550
440 GOSUB 840:GOTO 50
450 REM
460 IF l<k THEN 610
470 REM
480 FOR w=1 TO 150
490 SOUND 1,500-w,2,15-CINT(0.1*w)
500 IF w>19 THEN 540
510 LOCATE 10,25-w:PRINT CHR$(239);:CLS
520 INK 0,9,24:LOCATE 5,12:PRINT "K O N
I E C";
530 INK 0,1:INK 1,26
540 NEXT
550 MODE 1:PRINT "GRASZ JESZCZE ? (T/N)"
;
560 d$=INKEY$
570 IF d$="" THEN 560
580 IF d$="t" OR d$="T" THEN RUN
590 IF d$="n" OR d$="N" THEN 880
600 GOTO 560
610 REM za malo skladnikow
620 LOCATE 2,1:PRINT "ZA MALO SKLADNIKOW
";
630 FOR A=1 TO 1000:NEXT
640 LOCATE 2,1:PRINT STRING$(18," ");
650 LOCATE 20,25:PRINT CHR$(239);
660 x=x-1:GOSUB 840:GOTO 220
670 REM
680 x=CINT(RND*100):IF x<=1 OR x>=20 THE
N 680
690 y=CINT(RND*100):IF y<=2 OR y>=25 THE
N 690
700 RETURN
710 REM w lewo
720 IF INKEY(8)=0 AND x>1 THEN GOSUB 810
:x=x-1
730 REM w prawo
740 IF INKEY(1)=0 AND x<20 THEN GOSUB 81
0:x=x+1
750 REM do gory
760 IF INKEY(0)=0 AND y>2 THEN GOSUB 810
:y=y-1
770 REM na dol
780 IF INKEY(2)=0 AND y<25 THEN GOSUB 81
0:y=y+1
790 LOCATE x,y:PRINT CHR$(249);
800 RETURN
810 LOCATE x,y:PRINT " ";
820 SOUND 3,500,1,,,13
830 RETURN
840 REM
850 LOCATE 18,1:PRINT l;
860 FOR i=1 TO zal
870 LOCATE 8+i,1:PRINT CHR$(224);:NEXT:R
ETURN
880 MODE 2
```

Program możemy sobie uatrakcyjnić zmieniając np. ilości zbiorników (wartość n), ilości składników paliwa (wartość m), i ilość niezbędnych składników do wytworzenia paliwa (k) — zmiany w wierszu 20. Trzeba jednak zaznaczyć, że zbyt duże wartości n i m spowodują, że ludzik będzie się poruszał wolniej (przeglądanie coraz większych tablic). Może ktoś z Was zaproponuje inny, bardziej efektywny sposób poruszania się ludzika, bądź szybszego przeglądania tablic.

Wprowadzenie ograniczenia czasowego (np. w ciągu 30 sek.) na dojście do rakiety i wystartowanie na pewno zwiększy atrakcyjność gry. Życzę dobrej zabawy

Cezary SOBCZAK

Liga Myślących

A oto wylosowane nazwiska Czytelników, którzy za prawidłowe rozwiązanie wszystkich zadań LIGI MYŚLĄCYCH z 5 nr. „IKS-a" otrzymują nagrody książkowe: Andrzej BARCEWICZ z Łomży, Sławomir RYDEL z Kolna, Wojciech WÓJCIK z Ostrowca Świętokrzyskiego, Paweł URBAN z Rzeszowa, Janina SZMYDT z Lubonia.

Oto wylosowane nazwiska Czytelników, którzy za prawidłowe rozwiązanie wszystkich zadań z 6 nr. „IKS-a" otrzymują nagrody książkowe: Marek ZDEB z Pszczyny, Marek KRUŻEL z Tarnowa, Barbara GOŁKOWSKA z Grudziądza, Ryszard BANIEWICZ z Bydgoszczy, Tadeusz FRYSKA z Poznania.

Za poprawne rozwiązanie wszystkich zadań z 7 nr. „IKS-a" książki otrzymują: Andrzej ERD z Radomia, Janusz GWÓŹDŹ z Zawiercia, Marian RABCZAK z Gdańska, Krzysztof TOPOLSKI ze Zbąszynia, Witold WRÓBEL z Katowic.

W siódmym numerze „IKS-a" skończyliśmy pierwszą edycję LIGI MYŚLĄCYCH. W siedmiu numerach „IKS-a" przedstawiono 27 zadań o różnym stopniu trudności. Wszystkie zadania rozwiązał prawidłowo i otrzymał maksymalną liczbę 37 punktów HUBERT NOWACZYK z Grudziądza. Został on zwycięzcą pierwszej edycji LIGI MYŚLĄCYCH i otrzymuje magnetofon MK 232 p.

Kolejne miejsca, z 36 punktami, zajęli:
Andrzej ERD z Radomia, Antoni GORYL z Krakowa, Janusz MORBITZER z Krakowa, Andrzej WRZEŚNIOWSKI z Krakowa i otrzymują radioodbiorniki ANETA.

Za miesiąc rozpoczynamy kolejną edycję LIGI MYŚLĄCYCH. Czekamy na propozycję zadań — najciekawsze trafią na łamy LIGI. Myślący do pracy!

Notowania komputerowe

GIEŁDA

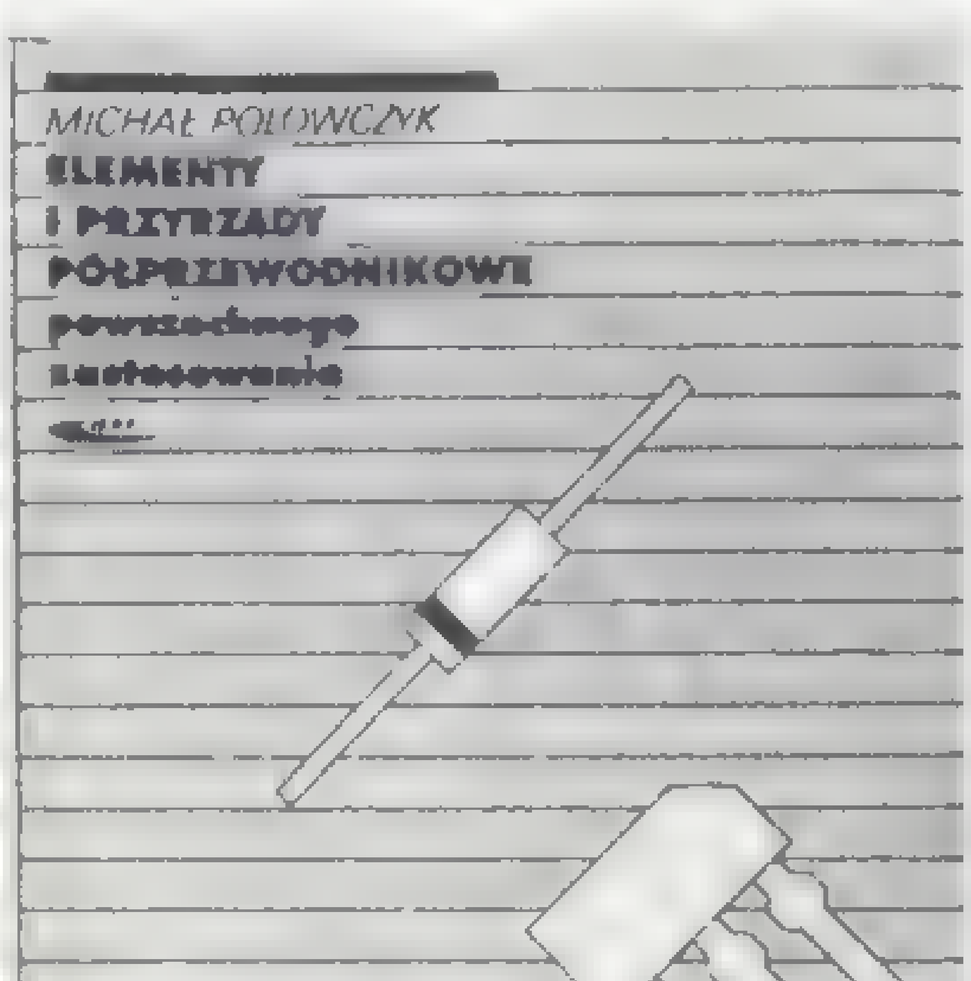
ZX Spectrum Plus	125 tys. zł
ZX Spectrum 128 KB	195 tys. zł
Commodore 16 + magnet. 1531	100 tys. zł
Commodore C-64	170 tys. zł
Atari 800 XL	125 tys. zł
Atari 1050 + joystick 5 + 10 dyskietek	185 tys. zł
Magnetofon 1050	50 tys. zł
Stacja dyskietek 1050	170 tys. zł
Atari — programy: gry	300 — 500 zł
programy użytkowe	200 — 400 zł
programy kopiujące	500 — 700 zł
Amstrad 464 z monit. kolor.	320 tys. zł
Drukarka Seikosha GP 50S	100 tys. zł

Studio SYSTEM — Warszawa-Wola

Commodore C-16	92 tys. zł
Sharp M-701 z plotterem	150 tys. zł
Schneider (Amstrad) CPC 6128	885 tys. zł
Schneider (Amstrad) PCW 8256	1 200 tys. zł
Schneider (Amstrad) PCW 8512	1 540 tys. zł
Interface'y KEMPSON	10 200 zł
Interface'y SINCLAIR	10 620 zł
Dyskietki 3"	6,2 — 8,4 tys. zł
Dyskietki 5,25" 2D	1 071 zł
Dyskietki 5,25" 2D firmowe	3 500 zł
Zestaw programów SPECTRUM (min. 8)	1 600 zł
Bogata biblioteka ATARI	400 zł/szt.

dokończenie ze str. 12

3) **„Elementy i przyrządy półprzewodnikowe powszechnego zastosowania"** (cena 790 zł). Poziom conajmniej studencki. Michał Polowczyk zaczynając od fizyki półprzewodników, poprzez rezystory, halotrony i diody doprowadza nas do tranzystorów oraz elementów fotooptycznych.



Jest to bazą do trzech głównych części: obwody scalone analogowe, obwody scalone cyfrowe TTL oraz obwody scalone cyfrowe MOS. Wytrwały czytelnik pozna tajemnice „cegiełek" komputera

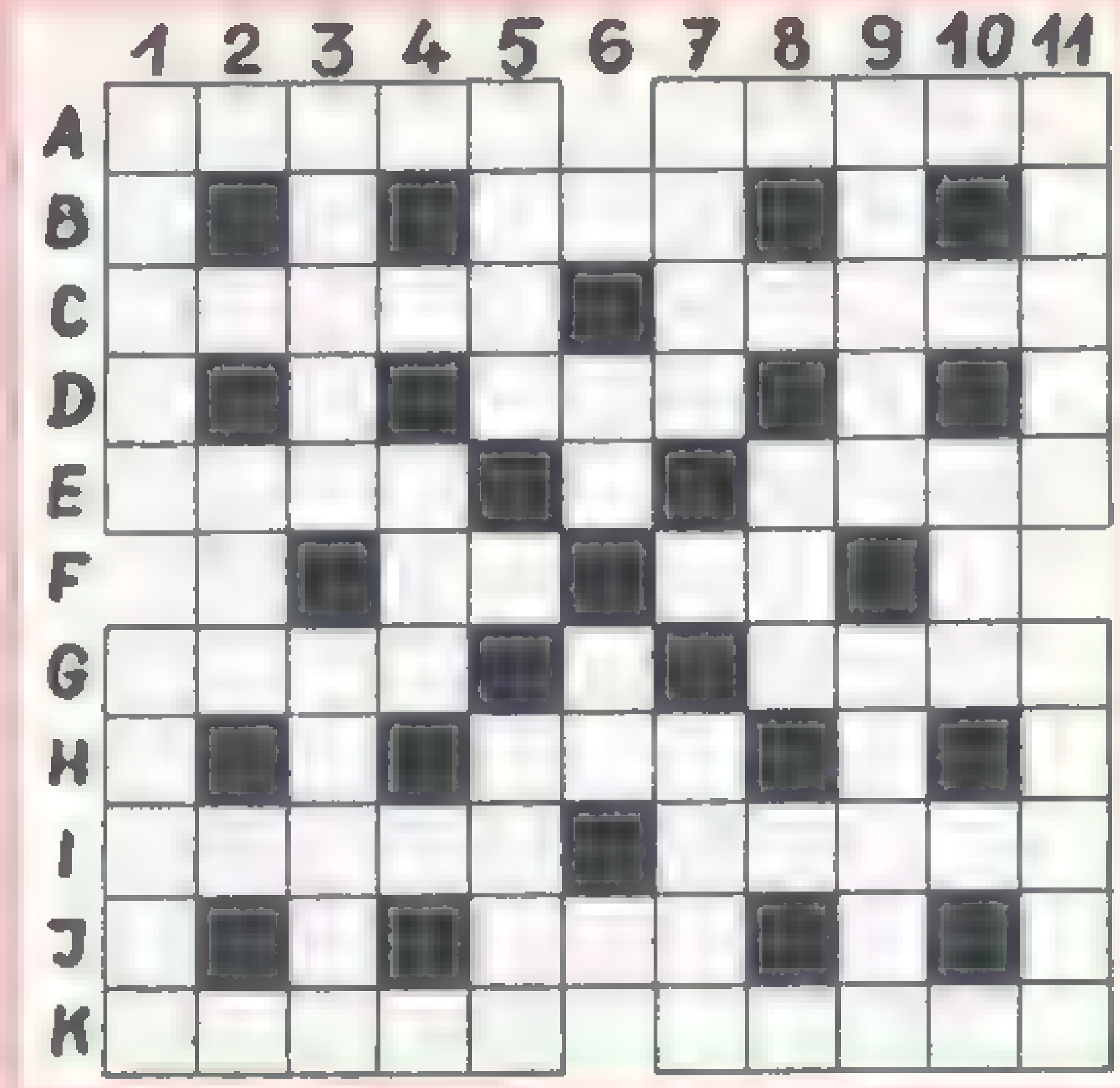
* * *

4) Użytkownicy ZX Spectrum mają wreszcie dobry podręcznik: **„Przewodnik po ZX Spectrum"** (cena 480 zł) K. Kuryłowicz, D. Madej i K. Marasek. Starali się wnikliwie omówić wszystkie zakamarki i możliwości tego najpopularniejszego w Polsce mikrokomputera. Może to być z pewnością książka dla początkujących, zaś zaawansowani znajdą w niej wskazówki, gdzie poszukiwać dalszych informacji Uzupełnieniem wiadomości o zmiennych systemowych ROM jest też „Iksowski" cykl „Sztuki i sztuczki". A rozszerzeniem informacji o Basicu nasz kurs tego języka

„Przewodnik" przyda się każdemu fanowi mikrokomputerów, często bowiem wykracza poza ramy Spectrum — ta wiedza z pewnością będzie pożyteczna. Szkoda jedynie, że na pierwsze tego typu wydawnictwo na tak dobrym poziomie — tak długo trzeba było oczekiwać. Uwaga nakład 150 000 tys

C.M.

Krzyżówka nr 3



POZIOMO: A1) Może być na przykład aperturowa, dualna, indeksowa itp., A7) Sekwencja rozkazów, która została celowo określona przez programistę i jest wykonywana przez procesor w sposób powtarzalny, aż do momentu spełnienia założonego warunku, B5) Jednostka gęstości zapisu na magnetycznym nośniku danych, C1) Określona porcja np. promieniowania, C7) Uniwersalny język programowy stosowany w zagadnieniach ekonomicznych i zarządzania, D5) Jedno z wcieleń Janusza Gajosa, E1) Robocza nazwa komputerów „Jednolitego systemu", E8) Grecki Mars, F4) Jednostka oporności elektrycznej, F7) Stała część nazwy systemów operacyjnych dla komputerów rodziny RIAD lub firmy IBM, G1) Skrótowa nazwa angielskiego specjalistycznego systemu przetwarzania danych przeznaczonych do informowania kierownictwa, G8) Część ciała ludzkiego, H5) Potrawa na ratunek, I1) Osoba żyjąca przy klasztorze nie składająca ślubów, I7) Odwrotność różniczki, J5) Imię żeńskie, K1) Metropolita Jordanii, K7) W starożytnym Rzymie pieśń żałobna.

PIONOWO: A1) Urządzenie zmieniające symbol pewnego alfabetu na symbole innego alfabetu, G1) Obraz religijny w sztuce wsch.-chrześcijańskiej, E2) Skrótowa nazwa firmy będącej największym producentem sprzętu komputerowego na świecie, A3) Pokaz, parada, G3) Mahometanizm, E4) Ogólna nazwa systemów operacyjnych dla komputerów rodziny RIAD, lub firmy IBM zapewniających mniejsze możliwości aniżeli system OS, A5) Starożytne liczydła, H5) Pistolet maszynowy skonstruowany przez Shepparda i Turpina, D6) Przyjęta z języka angielskiego nazwa spójnika — lub, G6) Klasyczny japoński balet dramatyczny, A7) Nazwa systemu informatycznego przeznaczonego do kierowania produkcją, H7) Nazwa pakietu programów przeznaczonych dla komputerów serii Odra-1300 oraz ICL-1900 i stosowanych do kontroli zapasów magazynowych, E8) Stosowana w języku rosyjskim skrótowa nazwa zautomatyzowanego systemu zarządzania, A9) Środki transportowe, G9) Może być operacyjny (mnożenia, sumujący itp.), E10) Skrót oznaczający komputer hybrydowy, A11) Najpopularniejszy instrument muzyczny w starożytnej Grecji, atrybut Euterpe, G11) Lęk, strach.

Hasło: (D3, H1, H3) (B6, K10, F8, F5, B1) (E4, I3, E8) (I7, B7, B3, I2, E2, E10)

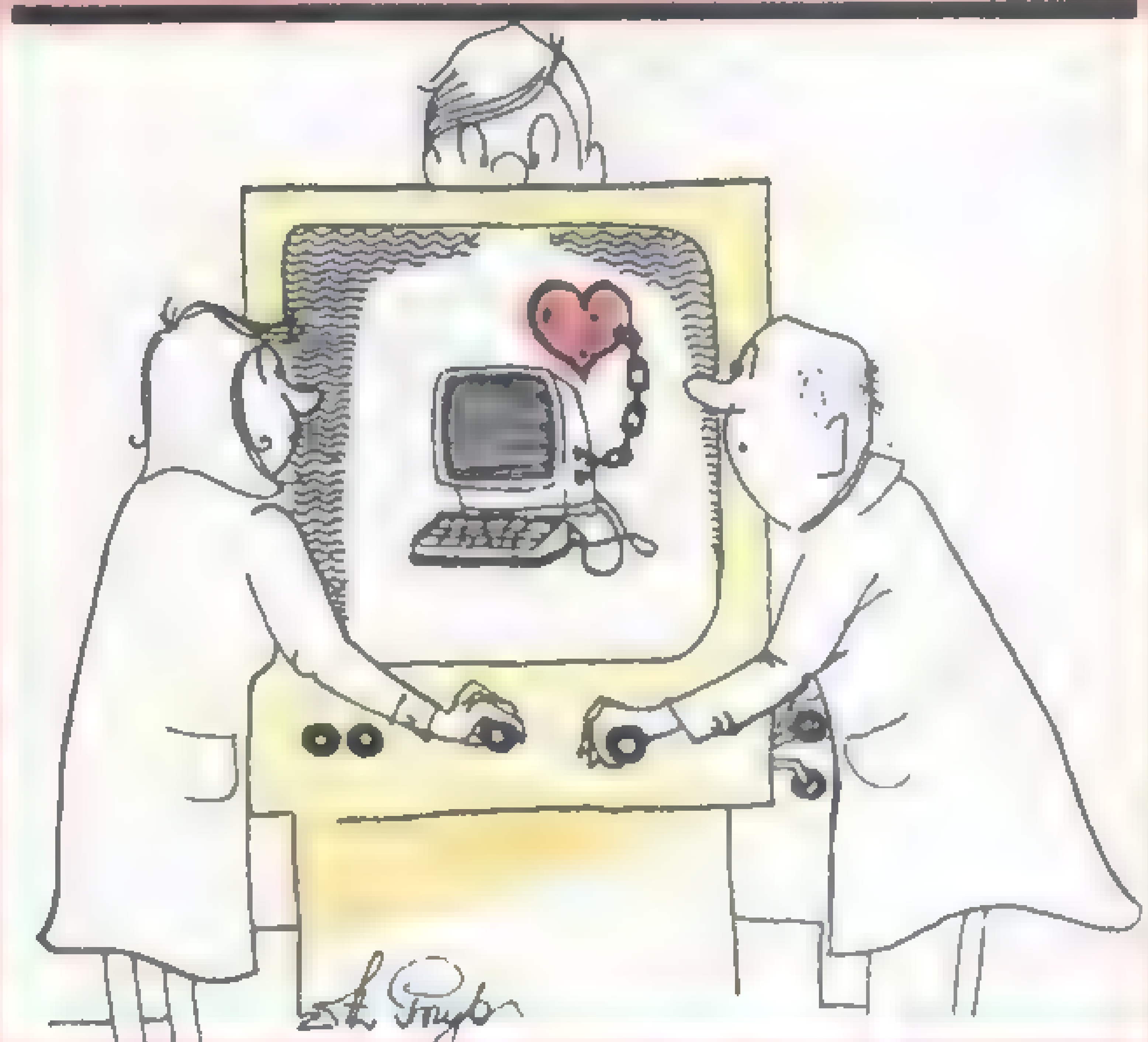
Rozwiązanie (tylko hasło) należy przesłać pod adresem redakcji do końca kwietnia br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI



SUN — GRAFIK — SYSTEM

Szczególną właściwością tego urządzenia jest rozdzielczość jego ekranu, 1152×900 punktów. Komputer dysponuje 256 barwami, z których różne jasności tworzą paletę zawierającą 16 700 000 kolorów!



Rys. Michał Przybyłowski

IKS — Dodatek Żołnierza Wolności. Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrotów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8587. Nr. ind. 361682. K-71

4

INFORMATYKA
KOMPUTERY
SYSTEMY



Ankieta str. 30

W numerze:

- Komputer i stylistyczne linie papilarne — czyli komputer w roli detektywa — str. 4
- Gra w wilki i owcę (Spectrum) — str. 5
- W Szponach Atari (3) — str. 7
- Kreska CPC 464 (3) — str. 14
- Radiokomputer zawsze we wtorki... — fotoreportaż — str. 16 — 17
- Sztuki i sztuczki (6) czyli zmienne systemowe ROM (Spectrum) — str. 18
- Odkrywcy gwiazd str. 20
- Pascal (4) — str. 21
- Przesunięcie zawartości ekranu — str. 27
- Pogoń (Sharp) — str. 27
- Liga Myślących — str. 30
- Ankieta — str. 30

SPECTRUM

```
 100 REM [illegible]
 110 BORDER 6: BEEP .2,25: BEEP
.2,14: BEEP .1,2
 120 INPUT "Jak sie nazuwasz ? "
; LINE i$
 130 IF LEN i$<3 OR LEN i$>20 TH
EN PRINT #0;AT 1,0;"OD 3 DO 20 L
ITER !": PAUSE 200: GO TO 110
 140 FOR n=1 TO LEN i$: IF NOT (
CODE i$(n)>64 AND CODE i$(n)<91
OR CODE i$(n)>96 AND CODE i$(n)<
123) THEN PRINT #0;AT 1,0;"TYLKO
 LITERY !": PAUSE 200: GO TO 110
 150 NEXT n
 160 IF CODE i$(1)>91 THEN LET i
$(1)=CHR$ (CODE i$(1)-32)
 170 FOR n=2 TO LEN i$: IF CODE
i$(n)<91 THEN LET i$(n)=CHR$ (CO
DE i$(n)+32)
 180 NEXT n: LET l=LEN i$
 190 IF i$(l)="n" THEN LET i$=i$
+"ie"
 200 IF i$(l)="s" THEN LET i$=i$
+"iu"
 210 IF i$(l-2 TO )="iek" THEN L
ET i$(l-2 TO )="ku"
 220 IF i$(l-1 TO )="ek" THEN LE
T i$(l-1 TO )="ku"
 230 IF i$(l-1 TO )="ik" THEN LE
T i$=i$+"u"
 240 IF i$(l-1 TO )="io" THEN LE
T i$(l-1 TO )="iu"
 250 IF i$(l-1 TO )="la" THEN LE
T i$(l-1 TO )="lu"
 260 IF i$(l-1 TO )="sa" THEN LE
T i$(l-1 TO )="so"
 270 IF i$(l-1 TO )="ka" THEN LE
T i$(l-1 TO )="ko"
 280 IF i$(l-1 TO )="ta" THEN LE
T i$(l-1 TO )="to"
 290 IF i$(l-1 TO )="ja" THEN LE
T i$(l-1 TO )="jo"
 300 IF i$(l-1 TO )="na" THEN LE
T i$(l-1 TO )="no"
 303 IF i$(l-1 TO )="ra" THEN LE
T i$(l-1 TO )="ro"
 305 IF i$(l-1 TO )="ia" THEN LE
T i$(l-1 TO )="iu"
 310 IF i$(l-1 TO )="sz" THEN LE
T i$=i$+"u"
 315 LET a$="Witaj "+i$+" !"
 320 LET z=31-LEN a$
 330 CLS : FOR n=1 TO 10: BEEP .
05,2*n: PRINT AT (n-1)*2+3,RND*z
; BRIGHT 1; PAPER RND*7; INK 9;a
$: NEXT n
 340 PAUSE 100: GO TO 110
 999 CLS : SAVE "Iwiona" LINE 10
0: BEEP .1,10: PRINT '"Verify
Iwiona": VERIFY "": GO TO 100
```

KOMUNIKAT

Po szczegółowym zapoznaniu się ze wszystkimi pracami nadesłanymi na konkurs miesięcznika „IKS" na utwór fantastyczny pt. „Przed nami XXI wiek" jury konkursowe w składzie:

Ireneusz CZYŻEWSKI	—	przewodniczący
Elżbieta JEZIERSKA	—	sekretarz
Włodzimierz GOGOŁEK	—	członek
Andrzej MONASTYRSKI	—	członek
Tadeusz PIEKŁO	—	członek

na posiedzeniu w dniu 11 marca 1987 r. postanowiło:

1. Nie przyznawać I i II nagrody.
2. Przyznać trzy III nagrody po 5000 zł nw. autorom:
   Robertowi JASTRZĘBSKIEMU z Łodzi (20 lat) za pracę pt. „Ocalony"
   Michałowi NIEDŹWIEDZIOWI z Krakowa (17 lat) za pracę pt. „Gdy wyłączą prąd"
   Andrzejowi DYRSZCE ze Świętochłowic za pracę pt. „Finał"
3. Wyróżnić nagrodami po 3000 zł nw. autorów:
   Romualda PAWLAKA z Sosnowca za pracę pt. „Los człowieka — los maszyny"
   Wiesławę FRYMARK z Płocka (15 lat) za pracę pt. „Przedszkolaki — rocznik 1995"
   Marka RUDNICKIEGO z Wrocławia (19 lat) za pracę pt. „Świadomość"
   Tomasza ROLĘ z Krakowa (16 lat) za pracę pt. „Ślepe marzenia"
   Autora, który zastrzegł sobie nazwisko (do wiadomości red.) — za pracę pt. „Pojedynek"
4. Wszystkie prace nagrodzone i wyróżnione zakwalifikować do druku w miesięczniku „IKS".
5. Ponadto zakwalifikować do druku prace n.w. autorów: Jana GRUSZCZYŃSKIEGO ze Szczecina (30 lat) pt. „System lojalności", P. KRÓLAKA pt. „Stan gotowości"
6. Przyznać dyplom uznania dla Ewy SIEROCKIEJ z Kalisza (13 l.) za opowiadanie pt. „Dziecko dr. Hakiosa".

Jednocześnie jury wyraża podziękowanie wszystkim pozostałym 96 autorom nadesłanych prac za udział w naszym konkursie.

Uwagi o konkursie na str. 31

Kolejną rewię komputerów mamy już za sobą. Wrocławski „INFOSYSTEM'87" udowodnił, że zainteresowanie informatyką nie słabnie. Targowa oferta była tak obfita, że polski rynek z pewnością jej nie potrafi skonsumować. Jednocześnie obok komputerowych potęg pojawiły się wśród wystawców nikomu dotąd nie znane firmy. Wiele z nich to efemerydy, których na następnych targach nikt pamiętać nie będzie.

Tymczasem, kiedy wrocławskie oferty szokują nowoczesnością, my będziemy wreszcie mieli Juniora. Zatem za parę lat, jeśli na drodze do jego masowej produkcji nic nie stanie, będziemy wreszcie mieli sprzęt, nie wiadomo natomiast czy będziemy mogli jednocześnie zaproponować odpowiednie jego oprogramowanie. A brakuje przede wszystkim oprogramowania edukacyjnego, brakuje również programów użytkowych dla sprzętu tej klasy. Wypada wierzyć, że nasi Czytelnicy poradzą sobie z tym problemem.

Pod naszym adresem nadchodzi wiele programów napisanych przez Czytelników. Długie wydruki (jeśli program otrzymaliśmy tylko w takiej postaci) już tracą na starcie i mają małe szanse trafić do druku. Dlatego też proponujemy

GIEŁDĘ POMYSŁÓW,

w której oceniać będziemy najlepsze pomysły w najkrótszych programach. Może być każdy komputer, może być dowolna postać programu — liczy się tylko pomysł. I tylko takie będą każdego miesiąca trafiały na nasze łamy. Na autorów najlepszych pomysłów czekają co miesiąc laury triumfatorów.

REDAKCJA

Microsoft C v.4.0

Firma MICROSOFT rozpoczęła sprzedaż czwartej wersji kompilatora języka C dla mikrokomputerów IBM PC, który cechuje :



- zaawansowana optymalizacja kodu wykonywana automatycznie w trakcie kompilacji ;
- kompatybilność z systemem operacyjnym 286 XENIX oraz zgodność z pracami normalizacyjnymi ANSI ;
- możliwość wykorzystania programu CodeView ułatwiającego lokalizację błędów ;
- wykorzystanie całej dostępnej pamięci zapisywalnej ;
- implementacja zmiennych rejestrowych;
- pięć modeli pamięci wykorzystujących zalety segmentowej architektury procesora INTEL 8086 ;
- możliwość łączenia własności różnych modeli pamięci w pojedynczym programie ;
- bogata biblioteka funckji (200);
- możliwość łączenia programów napisanych w MS C z programami napisanymi w MS FORTRANIE , MS Pascalu i MS Macro Assemblerze ;
- duży zestaw komunikatów umożliwiających zlokalizowanie błędów .

smARTWORK i HiWIRE



Trzy lata temu WINTEK zaprojektował i wykonał program o nazwie smARTWORK , który ułatwia projektowanie płytek drukowanych . Od tego czasu tysiące inżynierów uczyniło go popularnym programem CAD w elektronice. W lutym b.r. ukazała się nowa wersja programu (w.1.2 r6) , jednocześnie firma WINTEK wprowadziła program towarzyszący o nazwie HiWIRE ,reklamując go , iż rozpoczyna on prace ,gdy smARTWORK ją zakończył . HiWIRE jest programem wspomagającym rysowanie schematów elektrycznych . Istotną cechą programów jest możliwość korzystania z biblioteki gotowych elementów ,każdy z programów kosztuje 895 $.

▫▫ ▪ ▫▫ M I K R O D E P E S Z E ▫▫ ▪ ▫▫

Paul Allen,który wraz z Billem Gaty utworzył korporację Microsoft, założył nową firmę A S Y M E T R I X .Głównym jej zadaniem ma być pisanie oprogramowania przeznaczonego dla mikrokomputerów wyposażonych w mikroprocesory 80286 i 80386 ▪ ▫▫ ▪ ▫▫ ▪

opracowali: Mariusz Jarzębowski i Jacek Wojtala

KOMPUTER I STYLISTYCZNE LINIE PAPILARNE

Każdy z nas może spróbować swych sił jako detektyw o specjalności ... literatura. „Literacki detektyw" stara się zidentyfikować autora przez odpowiednią analizę próbki jego twórczości. Często prowadzi to do wyróżnienia rzeczywistego autora spośród grona kandydatów do miana twórcy analizowanego dzieła.

Dotychczas w badaniach tego typu wykorzystywano szereg różnych metod identyfikacji. T. C. Mendenhall stosował częstotliwość długości słów jako podstawę do oceny autorstwa pewnych dzieł Szekspira. Badacz ten policzył długość 400 000 słów z utworów Szekspira i 200 000 słów Bacona. Następnie narysował krzywą częstotliwości każdego autora. Krzywe z Szekspira i Bacona różniły się między sobą. Natomiast krzywa Ch. Marlowe'go z szekspirowską były niestety niemal identyczne.

G. Udny Yule użył do badania autorstwa częstotliwości występowania pewnych rzeczowników, jakimi posługiwali się potencjalni twórcy analizowanego dzieła. A. Ellegard użył podobnej metody opierającej się na częstotliwości występowania pewnych słów, do [illegible] listów które zostały opublikowane w Londynie (od stycznia 1769 do stycznia 1772). Ellegard wyróżnił trzech [illegible] autorów ze wskazaniem na Sir P. Francis'a.

F. Mosteller i D. Wallace liczyli częstotliwość występowania drugorzędnych słów, by określić, którzy z autorów: A. Hamilton czy J. Madison napisali 12 artykułów w znanym wówczas piśmie THE FEDERALIST (lata 1787 — 1788)[1]

Mosteller i Wallace uznali, iż drugorzędne słowa, takie jak „a", „an", „by", „to" i „that" są doskonałymi „wyróżnikami" autorów.

Inną metodę przyjął W. Bennett, dowiódł, że częstotliwość używania pojedynczych liter jest wyróżnikiem między próbkami tekstu, jednak tylko na poziomie określonego języka (na przykład angielski i hiszpański). Natomiast częstotliwość par liter zazwyczaj wystarcza do określenia różnic między autorami. Bennett przeprowadził badania, używając częstotliwość par liter w celu przyporządkowania autorów: Hemingway, Poe, Baldwin, Joyce, Szekspir, Washington i Lincoln pracom będącym dotychczas anonimowymi.

Analogia z detektywem pozwala uznać, że autor pozostawia znaczące ślady w swych pracach, które można uznać jako stylistyczne odciski palców. Istotne jest, że „ślady te" nie są związane ze stylem, ale tworzą wiele parametrów, które mogą wyróżniać autora.

J. Tankard (autor prezentowanego opracowania) napisał kilka programów na Apple II, które umożliwiały wykorzystać analizę jedno- i dwuliterową. Następnie programy zostały przetestowane na wspomnianych artykułach pisma „The Federalist".

Programy te, zgodnie z założeniem, czytały teksty ze zbiorów, liczyły częstotliwość występowania liter samodzielnych i par liter, normalizowały obliczenia do standardowych próbek np.: 1000 liter lub 10 000 liter oraz rachowały poszukiwane różnice częstotliwości.

Napisanie programów okazało się bardzo proste. Podstawowym był do analizy pojedynczych liter, jego prosta modyfikacja liczyła częstotliwość par liter. Prostota tego zabiegu stanowiła o „sile komputera". Programy zostały napisane w Basicu — liczenie częstotliwości samodzielnych liter, par liter i analizy statystycznej.

Punkt wyjścia opisywanego zastosowania mikrokomputera stanowiło odnalezienie tekstów, których autor był znany. Teksty Madisona zawierały 18 087 znaków, natomiast Hamiltona 9399 znaków. Nie miało to istotnego wpływu na wyniki, bowiem wcześniej dokonano normalizacji częstotliwości próbki do 1000 dla samodzielnych liter i 10 000 dla par liter.

Analiza częstotliwości wskazała, że tylko 9 z 12 przypisywanych dotychczas Madisonowi artykułów było jego [illegible]

Wytłumaczenie tej różnicy, wobec wyników Mosteller'a i Wallace'a leży prawdopodobnie w rozmiarze próbki tekstów, które były traktowane jako znane teksty. Tworzyło ją aż 70 000 słów wobec 6000 u Tankarda (1/3 Hamiltona i 2/3 Madisona). Inną przyczyną jest zapewne formalny styl, jaki narzucało pismo „The Federalist".

Próbki tekstu przeznaczone do komputerowej analizy powinny być umieszczone w samodzielnych zbiorach. Wykonano to za pomocą standardowego programu Apple Writer, który ponadto umożliwił łatwą kontrolę i korektę wprowadzonych tekstów.

Liczenie częstotliwości dla pojedynczych liter, za pomocą programów Tankarda trwało od 10 do 15 minut dla tekstów o długości 2000 — 3000 słów. Programy te zawierały szereg dodatkowych funkcji, na przykład drukowanie tabel.

Następnym logicznym krokiem w tego typu badaniach powinno być podniesienie poziomu analizy do „triad" — trzech liter. Jednak rozszerzenie analizy do trzech liter wnosi sobą problem trudny do rozwiązania w mikrokomputerze APPLE II — pojemność pamięci RAM.

Analiza triad powinna podnieść wiarygodność rozpoznawania autorstwa, bowiem teoretycznie aż 17 576 zmiennych może wyróżniać autorów (dla języka angielskiego: 26 × 26 × 26 liter). Zabieg ten umożliwił by uwzględnienie 3-literowych słów — niemal pełną analizę „drugorzędnych słów", które w badaniach Mostellera i Wallace'a okazały się dobrymi wyróżnikami — stylistycznymi odciskami palców.

Opracował:
WŁODZIMIERZ GOGOŁEK

[1] *Hamilton Alexander 1755—1804, polityk amerykański, przywódca federalistów. Madison James, 1751—1836, prezydent Stanów Zjednoczonych (1809—1817) jeden z przywódców walk o niepodległość.*



PROGRAM 7 Amstrad

Kopiowanie ekranu

Program pozwala na przeniesienie obrazu z ekranu na papier za pomocą dowolnej drukarki. Kopiowany obraz otrzymujemy w formacie 7×13 cm, a czas kopiowania wynosi około 1 minuty bez względu na złożoność kopiowanego obrazu.

W celu wykonania kopii, przed wprowadzeniem właściwego programu należy wczytać niniejszy program.

Wciśnięcie CTRL i COPY spowoduje przeniesienie obrazu na papier. Proces ten możesz zatrzymać w każdej chwili przytrzymując dłużej klawisz SHIFT.

Uwaga!!!

Program wykorzystuje komputer do wykonywania kopii, a następnie zostaje automatycznie skasowany, dlatego należy pamiętać, aby po przepisaniu z zamieszczonego wydruku najpierw zapisać go na dysku lub taśmie, a dopiero później sprawdzić poprawność działania.

```
1       ******************************
8       *     Kopiowanie ekranu      *
10      *    autor: Cliff Lawson     *
20      *    copyright (c) AMSOFT    *
21      *     opracowal dla IKS      *
30      *     Cezary Dobrowolski     *
31      ******************************
40 ZONE 3: MODE 2: LOCATE 12.10: PRINT "
   prosze czekac.";
50 MEMORY HIMEM 353
60 addr=HIMEM+1
70 lin=180: REM pierwsza linia danych
80 ON ERROR GOTO 160
90 ps=1: sum=0
100 READ a$
110 n=VAL("&"+MID$(a$.ps.2))
120 ps=ps+3
130 IF ps<26 THEN POKE addr,n:addr=addr+
1:sum=(sum+n)MOD 256 ELSE IF sum<>n THEN
 PRINT:PRINT:PRINT"blad danych w lini".l
in:PRINT CHR
$(7):MEMORY HIMEM +353:END
140 IF ps 27 THEN GOTO 110
150 lin=lin+10:PRINT ".";:GOTO 90
160 IF ERR=4 AND ERL=100 THEN MODE 2:LOC
ATE 22,10:PRINT"Nacisnij [CTRL] i [COPY]
 dla rozpoczecia kopiowania":LOCATE 25.1
2:PRINT "Trz
ymaj [SHIFT] dla przerwania":CALL HIMEM+
1:NEW ELSE PRINT "blad :".ERR,"w",ERL
170 END
180 DATA 21 E1 E9 22 30 00 F7 EB 1F
190 DATA 21 49 01 19 4E 23 46 79 B4
200 DATA 80 28 15 E5 60 69 19 E5 99
210 DATA 4E 23 46 60 69 19 44 4D 2A
220 DATA E1 71 23 70 E1 23 18 E4 E5
230 DATA 21 49 00 06 81 0E 00 11 10
240 DATA 2F 00 CD D7 BC C9 C5 D5 F2
250 DATA E5 F5 3E 09 CD 1E BB 28 EF
260 DATA CA 3E 17 CD 1E BB 28 03 30
270 DATA CD 59 00 F1 E1 D1 C1 C9 53
280 DATA 00 00 00 00 00 00 00 00 00
290 DATA 00 00 00 00 00 00 00 00 00
300 DATA DD 21 52 00 DD 36 01 00 64
310 DATA DD 36 02 00 DD 36 03 90 BB
320 DATA DD 36 04 01 DD 36 05 00 30
330 DATA DD 36 00 00 3E 1B CD CC C5
340 DATA EB C5 E5 CD 2B BD 3E 41 A9
350 DATA CD 2B BD 3E 06 CD 2B BD AE
360 DATA CD 2E BD 38 FB 3E 1B CD 11
370 DATA 2B BD 3E 4B CD 2B BD 3E 64
380 DATA 40 C0 2B BD 3E 01 CD 2B 2C
390 DATA ED DD 36 00 00 DD 36 06 E9
400 DATA [illegible] CD CB [illegible] 36 DD 5E 01 [illegible]
410 DATA DD 56 02 DD 6E 03 DD 66 C6
420 DATA C4 CD F0 BB B7 28 04 DD 3C
430 DATA CB 00 C6 DE 34 06 DD 7E 03
440 DATA [illegible] FE [illegible] 28 12 DD 6E 03 [illegible]
450 DATA DD 66 04 2B 2B DD 75 03 F2
460 DATA DD 74 04 18 CC 18 A9 CD C7
470 DATA 2E BD 38 FB DD 7E 00 CD 46
480 DATA 2B BD DD 34 01 DD 34 01 0C
490 DATA [illegible] 03 DD 34 02 DD 7E 01 [illegible]
500 DATA [illegible] [illegible] [illegible] [illegible] 0D 7E [illegible] FE DA
510 DATA [illegible] 28 12 DD 6E [illegible] DD 66 CD
520 DATA 04 11 0C 00 19 DD 75 03 8F
530 DATA DD 74 04 18 8C DD 34 05 0F
540 DATA 3E [illegible] [illegible] 28 6D 3E [illegible] CD 15
550 DATA 2B DD 3E 15 CD 1E B6 [illegible] 01
560 DATA 0F DD 36 01 00 DD 36 [illegible] 3D
570 DATA 00 DD 7E 05 FE 22 20 A5 45
580 DATA 7E 18 CD 2B BD 3E 40 CD [illegible]
590 DATA 2E DD E1 E1 C1 C9 EB C9 B4
600 DATA [illegible] [illegible] [illegible] [illegible] 42 [illegible] 5F [illegible] E8
610 DATA 00 00 00 00 00 00 00 00 00
```

Gra w wilki i owcę

Jesteś władcą czterech wilków, których zadaniem jest zamknięcie owcy w śmiertelnym pierścieniu. Gra ma prostą szatę graficzną, gdyż przy minimalnych zmianach może być *„wpalcowana"* nie tylko na SPECTRUM.

Wprowadzenie współrzędnych ruchu jest w pełni kontrolowane i nie ma obawy, by wykonać ruch niepoprawny.

Teraz trochę o owcy — włada nią komputer. Owca porusza się (losowo?) do przodu, do momentu, gdy odkryje przejście między wilkami. Wtedy dąży do przejścia najkrótszą drogą. Odkrycie najkrótszej drogi umożliwia jej algorytm LEE: wokół pola owcy (suma = 0, patrz zmienna „SUMA"). Każde pole, na które w jednym ruchu może owca stanąć, przyjmuje wartość SUMA + 1. Teraz każde nowe pole jest analizowane analogicznie i wstawiana jest tam znowu wartość o 1 większa od pola startowego. Tak tworzony jest graf, który zostanie skończony, gdy wszystkie dostępne pola będą zajęte. Jeśli do interesującego nas pola jest dojście, to pole to ma jakąś wartość, jeśli nie, to wartość tego pola nie zmieniała się podczas budowy grafu.

▶

PROGRAM 8 SPECTRUM

Przykład:

```
  A B C D E F G H
8 2 # 2 # 2 # 4 # 8
7 # 1 # 1 # 3 # 5 7
6 2 # 0 # 2 # 4 # 6
5 # 1 # 1 # 3 # 5 5
4 2 # 2 # 2 # W # 4
3 # W # W # 3 # W 3
2 8 # 6 # 4 # 4 # 2
1 # 7 # 5 # 5 # 5 1
  A B C D E F G H
```

W ten sposób do pola F1 z pola C6 możemy dojść pięcioma ruchami (przy nieruchomych wilkach, które nie odetną nam drogi). Drogę wyznaczamy, cofając się z pola F1 (po zmniejszających się cyfrach) do pola C6.

Krzysztof PÓŹNIAK

```
   1 REM gra w wilki i owce
Krzysztof Pozniak © 1986 HOBBIT
   2 PRINT "          GRA W WILKI I
 OWCE" PRINT   PRINT : PRINT "P
oruszasz sie wilkami podajac  
spolrzedne poczatku i konca ru-
hu wilkiem duzymi literami:
      np : ...H9,..G8": PRINT : P
RINT : PRINT "     PRZYJEMNEJ ZA
BAWY"
  10 PRINT : PRINT : PRINT : PRI
NT "NACISNIJ DOWOLNY KLAWISZ"
  20 DIM p$(10,10)
  30 DIM p(10,10)
  35 DIM m(2,5)
  36 DIM a$(2): DIM b$(2)
  37 DIM c(13,4)
  40 FOR k=1 TO 10. READ p$(k).
NEXT k
  50 FOR k=1 TO 5: FOR l=1 TO 2
 READ m(l,k): NEXT l. NEXT k
  55 REM PAUSE 0 - program czek:
 az nacisniesz dowolny klawisz
  60 PAUSE 0
  85 REM AT 0,0; - druk od leweg
o gornego rogu ekranu
  90 PRINT AT 0,0;: FOR k=1 TO 1
2: PRINT "
             ": NEXT k
 100 LET droga=0: LET gra=0: LET
 zwyc=0
 105 PRINT AT 0,0.: GO SUB 1000
 107 GO SUB 2000
 108 FOR k=1 TO 4: PRINT "
                          ": NEXT
 k
 109 IF gra<=2 THEN GO SUB 6000
 110 IF gra>=3 THEN GO SUB 4000
 125 PRINT AT 0,0;: GO SUB 1000
 127 GO SUB 2000
 128 IF zwyc>0 THEN GO TO 200
 130 GO SUB 3000
 135 LET gra=gra+1
 140 IF zwyc=0 THEN GO TO 105
 200 PRINT . PRINT
 210 IF zwyc=1 THEN PRINT "ZWYCI
EZYLA OWCA"
 220 IF zwyc=2 THEN PRINT "ZWYCI
EZYLY WILKI"
 230 FOR k=1 TO 200. NEXT k
 235 REM RESTORE - program czyta
 dane z DATA od poczatku
 240 RESTORE : GO TO 40
 999 STOP
1000 REM organizacja pola gry
1010 FOR m=0 TO 1: FOR k=2 TO 8
STEP 2: FOR l=2 TO 8 STEP 2: LET
 p(k+m,l+m)=1: NEXT l: NEXT k: N
EXT m
1020 LET p(m(1,1),m(2,1))=10: FO
R k=2 TO 5: LET p(m(1,k),m(2,k))
=0 NEXT k
1030 RETURN
2000 REM druk pola gry
2010 LET p$(m(1,1),m(2,1))="O":
FOR k=2 TO 5: LET p$(m(1,k),m(2,
k))="W": NEXT k
2020 FOR k=1 TO 10: FOR l=1 TO 1
0: PRINT p$(k,l);  NEXT l: PRINT
 "
                         ": NEXT k
2030 RETURN
3000 REM ruch wilka
3005 INPUT "wilk z pola :",a$,"
na pole:";b$
3010 FOR k=2 TO 9: IF a$(1)=p$(1
,k) THEN LET xs=k: GO TO 3030
3020 NEXT k: LET blad=1: GO TO 3
500
3030 FOR k=2 TO 9: IF a$(2)=p$(k
,1) THEN LET ys=k: GO TO 3050
3040 NEXT k: LET blad=1. GO TO 3
500
3050 FOR k=2 TO 9: IF b$(1)=p$(1
,k) THEN LET xk=k: GO TO 3070
3060 NEXT k: LET blad=1: GO TO 3
500
3090 FOR k=2 TO 9: IF b$(2)=p$(k
,1) THEN LET yk=k: GO TO 3110
3100 NEXT k: LET blad=1: GO TO 3
500
3110 FOR k=2 TO 5: IF m(1,k)=ys
AND m(2,k)=xs THEN GO TO 3130
3120 NEXT k: LET blad=2: GO TO 3
500
3130 IF p(yk,xk)<>1 OR yk<>ys-1
OR xk<>xs+1 AND xk<>xs-1 THEN LE
T blad=3: GO TO 3500
3140 LET m(1,k)=yk: LET m(2,k)=x
k: LET p$(ys,xs)=" "
3150 RETURN
3500 REM blad propozycji ruchu
3505 PRINT AT 0,0.: GO SUB 2000
3510 IF blad=1 THEN PRINT "niezr
ozumiale wspolrzedne ruchu"
3520 IF blad=2 THEN PRINT "zle o
kreslenie polozenia wilka"
3530 IF blad=3 THEN PRINT "bledn
a propozycja ruchu wilkiem"
3540 GO SUB 3600 PRINT "powtorz
 propozycje ruchu"
3545 FOR k=1 TO 6: PRINT : NEXT
k
3550 GO TO 3000
3600 REM wydruk ruchu
3610 PRINT " z pozycji : ";a$;"
"
3620 PRINT "na pozycje : ";b$;"
"
3630 RETURN
4000 REM algorytm LEE
4005 LET suma=10
4006 LET nh=2  LET pc=0  LET c=0
 : LET c(1,1)=m(1,1): LET c(1,2)=
m(2,1)
4007 LET inf=0
4010 FOR c=2 TO 0 STEP -2: LET c
(13,1+c)=nh-1  LET nh=1. FOR h=1
 TO c(13,1+c)  LET k=c(h,1+pc)
LET l=c(h,2+pc)
4020 FOR m=-1 TO 1 STEP 2. FOR n
=-1 TO 1 STEP 2
4025 LET wy=k-n  LET wx=l-m
4030 IF p(wy,wx)>suma+1 OR p(wy,
wx)=1 THEN LET inf=1. LET p(wy,w
x)=suma+1: LET c(nh,1+c)=wy: LET
 c(nh,2+c)=wx  LET nh=nh+1
4040 NEXT n: NEXT m
4045 NEXT h
4050 IF inf=1 THEN LET suma=suma
+1: LET pc=c. NEXT c  GO TO 4007
4060 IF suma=10 THEN LET zwyc=2
 RETURN
4070 IF droga=0 THEN GO SUB 5000
 : LET pzmlos=zmlos: GO SUB 5000:
 LET kzmlos=zmlos
4071 LET min=100. FOR k=6-pzmlos
+3 TO 6+pzmlos*3 STEP pzmlos
4072 IF kzmlos=-1 AND p(9,k)>1 T
HEN LET min=p(9,k): LET px=k  GO
 TO 4090
4075 IF kzmlos=1 AND p(9,k)>1 AN
D p(9,k)<min THEN LET min=p(9,k)
 : LET px=k
4080 NEXT k: IF min=100 THEN LET
 droga=0: GO TO 4100
4090 LET droga=1. LET py=9  LET
suma=min: GO SUB 4500
4095 FOR k=2 TO 4  IF m(1,1)<m(1
,k) THEN RETURN
4097 NEXT k: LET zwyc=1: RETURN
4100 FOR g=gra TO gra+2  LET py=
8-INT (g/4)  LET px=8-2*(g-4*INT
 (g/4)): IF py=3 OR py=5 OR py=7
 OR py=9 THEN LET px=11-px
4110 IF p(py,px)>1 AND p(py,px)<
>10 THEN LET suma=p(py,px)  GO S
UB 4500  RETURN
4120 NEXT g
4130 FOR k=2 TO 9: FOR l=2 TO 9.
 IF p(k,l)=11 THEN LET py=k. LET
 px=l: LET suma=11: GO SUB 4500:
 RETURN
4140 NEXT l: NEXT k
4150 STOP
4500 REM szukanie drogi
4505 GO SUB 5000  FOR k=-1 TO 1
STEP 2: FOR l=-1 TO 1 STEP 2: IF
 p(py-l,px+zmlos*k)<>suma-1 THEN
 GO TO 4530
4510 IF p(py-l,px+zmlos*k)=10 TH
EN LET m(1,1)=py  LET m(2,1)=px.
 LET p$(py-l,px+zmlos*k)=" "  RE
TURN
4520 LET suma=suma-1: LET py=py-
l: LET px=px+zmlos*k
4525 GO TO 4505
4530 NEXT l. NEXT k
4540 STOP
5000 REM funkcja losowa
5010 LET zmlos=1
5020 LET f=RND
5030 IF (f>=.1 AND f<.3) OR (f>.
5 AND f<.505) OR (f>.54 AND f<6)
 OR f>0.85 THEN LET zmlos=-1
5040 RETURN
6000 REM ruch owcy losowy
6010 GO SUB 5000  LET p$(m(1,1),
m(2,1))=" ": LET m(1,1)=m(1,1)+1
  LET m(2,1)=m(2,1)+zmlos
6020 RETURN
9000 DATA "  ABCDEFGH  ",
          "8 # # # #8",
          "7# # # # 7",
          "6 # # # #6",
          "5# # # # 5",
          "4 # # # #4",
          "3# # # # 3",
          "2 # # # #2",
          "1# # # # 1",
          "  ABCDEFGH  "
9005 REM pozycje startowe pionow
9010 DATA 2,6,9,3,9,5,9,7,9,9
```

Wielojęzyczne Atari

Ludwik PIELA
Tomasz MROWIEC

Wielu ludzi traktuje ośmiobitowe komputery Atari jako maszyny do gier. Wynika to zarówno z wcześniejszej renomy tej firmy, (o czym już pisaliśmy), jak i mnogości oferowanych gier, które często przesłaniają pozostałe programy. Jednak Atari 400, 800, XL i XE mają normalny mikroprocesor i są zdolne do działania w ponad dziesięciu językach programowania, od BASICa do LISPa.

Poniżej przedstawiamy przegląd języków programowania, dostępnych obecnie dla 8-bitowych komputerów Atari.

Komputery mogą wykonywać tylko takie programy, które mają postać liczb dwójkowych, będących kodami rozkazów maszyny. Mówimy, że są to programy w **języku** lub **kodzie maszynowym**

Wczesne mikrokomputery były wyposażone w zestaw przełączników na płycie czołowej, służących do wprowadzania 0 i 1 do komórek pamięci. Diody świecące wskazywały zawartość poszczególnych bajtów. Jednak ręczne ustawianie przełączników w celu wprowadzenia rozkazów do pamięci było źródłem znacznych błędów. Na szczęście, komputery Atari nie są programowane w ten sposób.

Język asembler

Programy w języku maszynowym są prawie nieczytelne dla ludzi. Dla ułatwienia projektowania takich programów i ich późniejszego zrozumienia używane są inne sposoby ich przedstawiania. Jeżeli kody rozkazów zastąpimy symbolami mnemotechnicznymi, a adresy i wartości ich nazwami, to otrzymamy język asemblera. Program, który tłumaczy symbole rozkazów na odpowiadające im kody maszynowe i zamienia nazwy na odpowiednie adresy i wartości nazywa się **asemblerem**.

Pewne asemblery, zawierające różne udogodnienia dla programistów, a zwłaszcza możliwość tworzenia i używania zestawów procedur, zwanych makroinstrukcjami, noszą nazwę **makroasemblerów**

Chociaż nie tak trudny do używania jak kod maszynowy, w dalszym ciągu język asemblera jest wygodniejszy dla komputera, niż dla użytkownika.

Dla komputerów Atari istnieje sześć różnych języków asemblera: **Atari Assembler-Editor**, **Atari Macro Assembler-Editor**, **MAC/65**, **MAE (Macro Assembler-Editor)**, **Synassembler**, **Datasm/65**. Różnią się one złożonością, elastycznością, wielkością i łatwością użycia

BASIC

BASIC jest najbardziej popularnym z języków programowania mikrokomputerów. W rzeczywistości nie jest to jeden język, lecz cała rodzina języków o wspólnym rdzeniu

BASIC stał się popularny głównie z powodu jego *przyjazności* dla użytkownika. Inne języki komputerowe są skomplikowane i używają nieznanych słów, symboli, i składni. BASIC „rozmawia" w bardzo prostym języku angielskim, używając stosunkowo małej liczby słów, które mogą być z początku niezbyt zrozumiałe

BASIC ma pewne wady spowodowane przez wrodzony brak struktury. Często mówi się, że w tym języku programiści mają zbyt dużo swobody w wykonywaniu skoków. Jeśli skomplikowany program w BASICu nie został dobrze udokumentowany za pomocą komentarzy, to jest trudny do zrozumienia nawet dla autora

Dla komputerów Atari istnieją trzy wersje języka BASIC. Najbardziej popularny jest **Atari BASIC**. Początkowo istniał na 8KB pakiecie ROM (cartridge), a następnie, po modyfikacji, został wbudowany do komputerów serii XL. Został on opracowany przez firmę Optimized Systems Softwarte (OSS).

Po rozszerzeniu oryginału o ponad czterdzieści dodatkowych funkcji, powstał **BASIC A+**. Dostarczany jest na dyskietce i zajmuje około 16KB pamięci RAM. Jest łatwiejszy w użyciu niż Atari BASIC i umożliwia programowanie strukturalne. Służą do tego instrukcje IF... THEN... ENDIF i WHILE... ENDWHILE. Inne usprawnienia obejmują funkcję TAB, instrukcję INPUT. Dostarczone są dodatkowe funkcje manipulowania ciągami znaków, konkatenacja i możliwość wyszukiwania podciągów za pomocą funkcji FIND. Włączono również kilka funkcji ułatwiających uruchamianie programów. TRACE umożliwia wyświetlanie komunikatów błędów podczas testowania programu. Funkcja IF ERR służy do testowania warunków błędu i sterowania przebiegiem programu. Możliwe jest także kasowanie grupy linii za pomocą jednej instrukcji DELETE

Jedną z najciekawszych własności języka BASIC A+ jest obszerny zestaw rozkazów graficznych gracz (pocisk/-player/missile-P/M). Czynią one grafikę P/M tak łatwą w użyciu jak PLOT i DRAWTO.

Firma OSS opracowała również dwa inne dialekty tego języka: BASIC XL i BASIC XE. Pierwszy z nich jest nieznacznym rozwinięciem BASIC A+ umieszczonym w pamięci ROM.

BASIC XE wprowadzony został dla ATARI 130XE wkrótce po pojawieniu się tego komputera w 1985 roku. Wykorzystuje on dodatkowe 64KB pamięci RAM, dlatego może działać tylko na ATARI 130XE. Podobnie jak BASIC XL dostarczany jest w pakiecie ROM ze zbiorem „Extensions" na dyskietce Umożliwia wykorzystanie 35KB na pamiętanie zmiennych i 65KB na program (po podaniu komendy „Extend"). Odznacza się pewnymi udoskonaleniami w stosunku do innych wersji. Dotyczą one głównie funkcji działających na ciągach znaków. Konkatenację trzech ciągów znakowych zapisuje się następująco: A$=A$, B$, C$. Lepiej rozwiązane są również tablice tekstowe. Ich wymiary podaje się w taki sposób jak dla tablic numerycznych. Na przykład, DIM Y $ (3, 12) określa wymiary tablicy zawierającej trzy elementy o długości 12 znaków każdy

BASIC XE ma łatwą w użyciu instrukcję sortowania działającą na tablicach numerycznych i tekstowych. Ma, również, podobnie jak BASIC XL, wbudowane instrukcje DOS-u, można zatem kasować, wyprowadzać, zmieniać nazwy, zabezpieczać i odbezpieczać zbiory z wnętrza programów.

Atari Microsoft BASIC (ASMB) dostarczany jest w dwóch wersjach: dyskowej lub „cartridge" (dysk /ASMB II) Mają one te same możliwości, lecz polecany jest ASMB II, ponieważ pakiet ROM jest trwalszym nośnikiem niż dysk chroniony przed kopiowaniem.

ASMB jest nadzbiornikiem Microsoft BASIC, wykorzystującym specyficzne możliwości Atari w zakresie dźwięku i

grafiki. Dużą jego wadą jest brak kontroli składni podczas wprowadzania linii programu; błędy wykrywane są dopiero podczas działania programu. Pozwala za to wybrać typ zmiennej numerycznej, całkowitą lub rzeczywistą oraz jej dokładność — pojedynczą lub podwójną. Funkcje matematyczne realizowane są przez pakiet zmiennoprzecinkowy systemu operacyjnego, dzięki temu wzrasta szybkość obliczeń.

Jedną z cennych zalet ASMB jest możliwość akceptowania funkcji zdefiniowanych przez użytkownika. Pozwala to modyfikować instrukcje języka.

Być może najbardziej znacząca różnica między tą a innymi wersjami BASICa polega na sposobie manipulowania ciągami znaków. ASMB nie wymaga deklarowania wielkości dla ciągów jednowymiarowych, dopuszcza maksymalną długość ciągu 255 znaków i pozwala stosować prawdziwe tablice tekstowe. Zawiera również inne użyteczne funkcje: konkatenację, podciągi lewy, środkowy i prawy oraz wyszukiwanie podciągu.

ASMB zaopatrzony jest w instrukcje umożliwiające automatyczne numerowanie, zmianę numeracji i kasowanie linii. Umożliwia śledzenie realizacji programów i przekazywanie wartości zmiennych z jednego programu do drugiego. Dodatkowe lub zmodyfikowane instrukcje obejmują **WAIT** (zatrzymanie realizacji programu na określony czas), **IF... THEN... ELSE** oraz połączenie **PLOT** i **DRAWTO** w jedną instrukcję — **PLOTTO**.

PASCAL

PASCAL został opracowany przez Kathleen Jensen i Niklausa Wirtha dla zademonstrowania zasad języka strukturalnego. Jest łatwy do nauki i dostosowany do definiowania struktur danych, niezbędnych do rozwiązywania problemów. Wprowadza pewne typy danych (takie jak integer, real i Boolean), lecz daje programiście swobodę w definiowaniu nowych. Zdefiniowane mogą być także nowe funkcje i procedury, dopuszczalne jest manipulowanie danymi znakowymi i ciągami znaków.

Jest to język kompilowany, lecz często nie bezpośrednio na kod maszynowy, lecz na pośredni pseudokod zwany *p-code*. Jest on przechowywany i w czasie działania programu interpretowany na kod maszynowy komputera.

Istnieją trzy implementacje języka Pascal dla komputerów Atari. Pierwsza z nich, zwana **Atari Pascal Language System**, oparta na języku standardowym, jest bardzo podobna do UCSD Pascal. Drobne różnice dotyczą możliwości graficznych i dźwiękowych. Niestety użycie tej wersji jest ograniczone; wymaga dwóch stacji dyskowych, nig-

PROGRAM ATARI 9

Edytor — wydruk 1

```
32600 IF FL THEN 32616
32602 DIM L$(115),SV$(115),C2$
(2),B$(115),M$(119),S$(98),E$(
69),A$(1):FL=1:STMTAB=PEEK(136
)+PEEK(137)*256
32604 GRAPHICS 0:POKE 710,0:P=
0:ABR=0:? "CZY CHCESZ UZYWAC S
KROTOW INSTRUKCJI";
32605 INPUT A$:IF A$="T" OR A$
="t" THEN ABR=1
32606 B$(1)=" ":B$(115)=" ":B$
(2)=B$
32616 OPEN #17,4,0,"E:":L$=" "
:GOSUB 32662:START=0
32618 POKE 766,1:POKE 83,39:PO
SITION 1,3:IF LEN(L$)<39 THEN
? L$:GOTO 32624
32620 IF LEN(L$)<77 THEN ? L$(
1,38):? L$(39,LEN(L$)):GOTO 32
624
32622 ? L$(1,38):? L$(39,76):?
 L$(77,LEN(L$))
32624 POKE 752,0:POKE 766,0:PO
KE 559,34:POKE 82,1:POKE 83,38
:POSITION 0,10:? " ";:INPUT #1
7;L$:POKE 766,1
32626 IF (L$="P" OR L$="p") AN
D START=0 THEN P=1:L$=""
32628 IF L$="E" OR L$="e" THEN
 E=1:POSITION 1,10:? SV$:GOTO
32624
32630 IF L$="L" OR L$="l" THEN
 GOSUB 32704
32632 IF L$="" AND P=1 THEN 32
685
32634 IF L$="" THEN 32624
32636 IF L$="B" OR L$="b" THEN
 GRAPHICS 0:? "WPROWADZ 'GOTO
32600' W CELU KONTYNUACJI":END
32638 IF L$(1,1)="E" OR L$(1,1
)="e" THEN E=1:TRAP 32624:EL=V
AL(L$(2)):POSITION 1,9:LIST EL
:GOTO 32624
32640 SV$=L$:TRAP 32624:X=VAL(
L$)
32642 START=1:IF P AND  NOT E
THEN 32652
32644 GOSUB 32674:IF  NOT ABR
OR P THEN 32652
32646 POKE 766,0:? CHR$(125):P
OSITION 0,3:L=VAL(L$):LIST L:?
 :? :? "CONT":L$=B$
32648 POSITION 0,0:POKE 842,13
:STOP
32650 POKE 842,12:A=USR(ADR(S$
),ADR(L$),4):L$=L$(1,A)
32652 CHKSUM=USR(ADR(M$),ADR(L
$),LEN(L$)):CHKSUM=CHKSUM+PEEK
(1542)*65536
32654 CHK=CHKSUM-(INT(CHKSUM/6
76)*676):HI=INT(CHK/26):LO=CHK
-(HI*26):C2$(1)=CHR$(HI+65):C2
$(2)=CHR$(LO+65)
32656 IF  NOT P OR E THEN E=0:
GOSUB 32662:IF  NOT P THEN 326
60
32658 POKE 83,39:POKE 752,1:FO
R X=3 TO 5:POSITION 1,X:? B$(1
,38):POSITION 1,X+7:? B$(1,38)
:NEXT X:POKE 83,38
32660 POKE 766,1:POKE 83,38:PO
SITION 7,7:? C2$:POKE 752,0:GO
TO 32618
32662 GOSUB 32702:POKE 766,0:P
OKE 752,1:? "}":POKE 82,1:DL=P
EEK(560)+256*PEEK(561)+4
32664 POKE DL-1,70:POKE DL+2,6
:POKE DL+3,112:POKE DL+4,112:P
OKE DL+5,112:POKE DL+13,112:PO
KE DL+14,112
32666 POKE DL+22,112:POKE DL+2
3,112:POKE DL+24,65:POKE DL+25
,PEEK(560):POKE DL+26,PEEK(561
):POKE 83,39
32668 POSITION 20,0:? "     edw
for basic       ":POSITION 0,7:?
 "             OKNO WPROWADZANIA
              "
32670 POSITION 0,1:? "
     OKNO KOMUNIKATU
  ":POSITION 1,7:? "KOD :";
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:PO
KE 766,1:POKE 82,0:POKE 83,39:
POSITION 0,3:? L$:? :? :? "
CONT":POSITION 0,0
32676 POKE 842,13:STOP
32678 POKE 842,12:TRAP 32682:A
=USR(ADR(E$),VAL(L$)):IF A=4 T
HEN POP :GOTO 32682
32680 RETURN
32682 GOSUB 32662:SOUND 0,75,1
0,8:FOR X=1 TO 20:NEXT X:SOUND
 0,0,0,0:POSITION 1,3:? "BLAD
SKLADNI!":POKE 766,1
32684 POKE 83,38:POSITION 1,10
:? SV$:GOTO 32624
32686 LINE=PEEK(STMTAB)+PEEK(S
TMTAB+1)*256:IF LINE>32599 THE
N 32690
32688 OFS=PEEK(STMTAB+2):STMTA
B=STMTAB+OFS:POSITION 1,9:LIST
 LINE:GOTO 32624
32690 POSITION 1,10:? "Numer l
inii nie moze byc wiekszy od
 32599 !"
32692 FOR I=1 TO 1000:NEXT I:G
OTO 32624
32702 POKE 16,112:POKE 53774,1
12:RETURN
32704 POSITION 5,12:? "KASETA
CZY DYSKIETKA";:INPUT L$:IF L$
<>"K" AND L$<>"D" THEN 32704
32706 IF L$="D" THEN 32714
32708 POSITION 1,12:? "Przygot
uj kasete i nacisnij RETURN":G
OSUB 32720
32710 LIST "C:",1,32559:GOTO 3
2718
32714 POSITION 0,12:? "NAZWA Z
BIORU (D:FILENAME.EXT)    ";CHR
$(30);CHR$(30);CHR$(30);:INPUT
 L$
32715 TRAP 32716:OPEN #1,4,0,L
$:CLOSE #1:GOTO 32722
32716 LIST L$,1,32559
32718 POSITION 0,12:? "

:RETURN
32720 GET #17,A:RETURN
32722 ? CHR$(125);"ZBIOR JUZ I
STNIEJE!":? "CZY WYKASOWAC GO"
;:INPUT A$:TRAP 32722:IF A$="T
" THEN 32716
32724 FOR X=12 TO 15:POSITION
0,X:? B$:NEXT X:GOTO 32714
```

dy nie był oficjalnie rozprowadzany przez Atari, jest szczególnie wolny w kompilacji i nie ma wbudowanego edytora

Dwie pozostałe wersje to: **Draper Pascal** i **Kyan Pascal**. Pierwszy z nich nie odpowiada oficjalnemu standardowi. Nie jest pełną implementacją języka (standardu ISO lu UCCD) i zawiera skrócony zakres typów danych, brak formatowania liczb i ograniczenia w przekazywaniu parametrów. Draper Pascal jest wystarczający do nauki języka lub prostych zastosowań. Jest łatwy w użyciu i nietrudny do nauki, lecz nie jest zalecany do poważniejszych prac.

Kyan Pascal jest jednym ze standardów języka. Umożliwia pisanie nowych programów lub używanie wcześniej napisanych, zgodnych ze standardami.

Wersja ta ma kilka uzupełnień w stosunku do standardowego języka. Należą do nich: możliwość wykorzystania grafiki o dużej rozdzielczości, łączenie zbiorów zawierających programy wynikowe, rozkazy manipulowania ciągami znaków i zbiory o dostępie bezpośrednim. Kyan Pascal nie wytwarza kodu pośredniego, lecz znacznie szybszy kod asemblera, który nie wymaga interpretera

JĘZYK C

Język C opracowany został w B[illegible] Laboratories dla systemu operacyjne go UNIX. Jako język wysokopoziomo wy dostarcza programiście tyle możl wości, ile asembler. Umożliwia równie przenoszenie programów między róż nymi maszynami prawie bez żadnyc modyfikacji. Jest to język strukturalny pozwala pracować praktycznie z do wolnymi strukturami danych. Łączy w sobie możliwości asemblera i Pascala Przeznaczony jest głównie dla fachowców. Z powodu wyjątkowej zwięzłości programy są mało czytelne. Struktura programu tworzona jest za pomocą funkcji zdefiniowanych przez użytkownika i standardowych. Program pisany jest przy użyciu edytora i może składać się z kilku grup oddzielnie kompilowanych funkcji, zwanych modułami, które później zostaną połączone w jeden program. Tworzenie programów ułatwiają biblioteki funkcji standardowych ora wcześniej zdefiniowanych przez użyt kównika

Język C dostępny jest dla Atari w dwóch wersjach: **Tiny C** (C/65) **Deep Blue C.**

LISP (LISt Processing language)

LISP jest językiem przeznaczonym do obliczeń nienumerycznych. Był on pierwszym językiem używanym w badaniach nad sztuczną inteligencją Nadaje się głównie do przetwarzania tak zwanych struktur listowych. Zawiera funkcje, zmienne i operatory arytmetyczne, lecz wygląda obco dla programistów używających BASICa

W języku LISP istnieją dwa typy danych: atomy i listy. Atomem jest dowolna kombinacja znaków alfanumerycznych zaczynająca się od litery. Lista składa się z atomów i innych list ujętych w nawiasy. Programy w języku LISP same są listami operatorów i danych. Dlatego mogą same siebie modyfikować w czasie wykonywania. Wszystkie funkcje mogą by używane jako argumenty. Język dysponuje wieloma wbudowanymi funkcjami i umożliwia tworzenie własnych. Teksty programów są mało czytelne z powodu olbrzymiej liczby nawiasów.

Aktualnie dostępna jest tylko jedna wersja LISPu dla Atari, opracowana przez firmę Datasoft

FORTH

FORTH jest trudny do nauczenia się z powodu rozbudowanego zestawu niełatwych do zapamiętania instrukcji oraz odmienności od innych używanych języków. Ponadto jego wyrażenia zapisywane są zawsze w odwrotnej notacji polskiej (RPN), dzięki czemu nie stosuje się nawiasów. Nazywany jest czasem językiem „nieskończonym" z powodu prawie nieograniczonej swobody w tworzeniu nowych słów lub funkcji, które uzupełniają zestaw podstawowy i są traktowane do indywidualnych potrzeb

FORTH jest językiem strukturalnym tak dalece, że nie ma GO TO ani etykiet instrukcji. Programy kompilowane są na bardzo oszczędny kod wynikowy. Dzięki temu pełny FORTH może zmieścić się w 16KB pamięci... i wciąż pozostaje miejsce na program.

Dla Atari dostępne są dwie wersje tego języka: **FIG-FORTH** — rozprowadzany przez FORTH Interest Group (FIG), oraz **Val-FORTH** — rozpowszechniany przez Valpar International. Val-FORTH zawiera oddzielne moduły, kupowane w miarę potrzeb, wykorzystujące specyficzne własności Atari, takie jak grafika gracz (pocisk, przerwania programu wyświetlania /DLI) oraz możliwości dźwiękowe

PILOT

PILOT (Program Inquiery for Learning Or Teaching) był pierwszym językiem przeznaczonym do nauczania wspomaganego przez komputer i został zaimplementowany w wielu systemach.

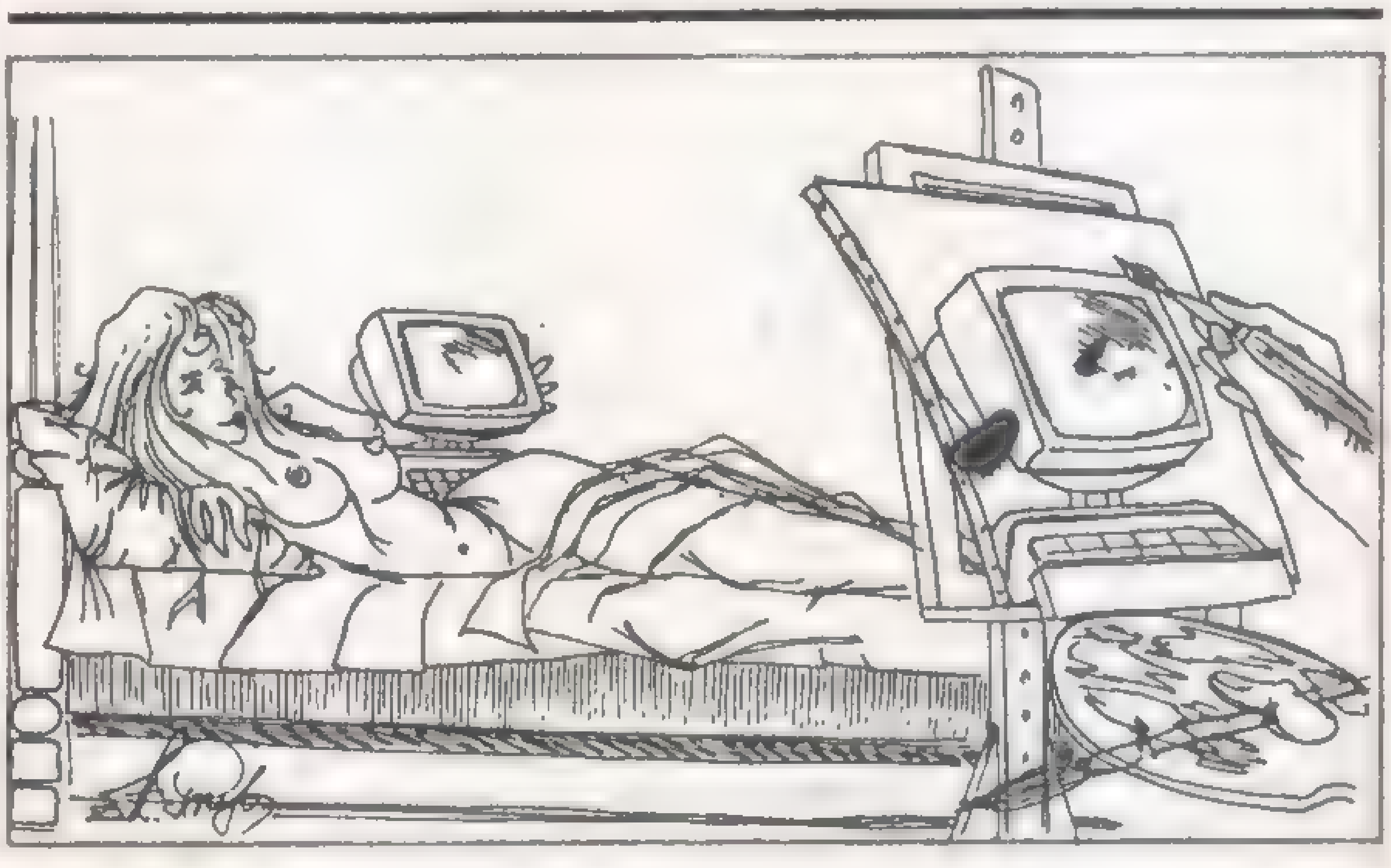
Instrukcje języka dzielą się na cztery kategorie.

1) instrukcje podstawowe jednoliterowe, standardowe dla wszystkich wersji języka. Dzięki temu programy można przenosić z jednego komputera na drugi.
2) instrukcje ustawiające różnego rodzaju parametry związane z komputerem,
3) instrukcje systemu zbiorów związane z przechowywaniem i wyszukiwaniem programów i danych, na taśmie lub dysku,
4) instrukcje *grafiki żółwia* — specyficzne dla wersji Atari języka

PILOT jest prawdopodobnie najłatwiejszym językiem do nauki dla nowych użytkowników komputera

LOGO

Logo jest poddialektem języka LISP, łatwym w użyciu, z *przyjaznym* dla użytkownika interfejsem graficznym. Jest językiem specjalizowanym, przeznaczonym głównie do ułatwienia zapoznania się z komputerem. Umożliwia natychmiastowe obserwowanie wyników działania i korygowanie błędów Dysponuje dobrze rozwiązaną grafiką, chociaż wbrew powszechnej opinii nie jest językiem graficznym. Programy Logo są przejrzyste i czytelne oraz tworzone w sposób strukturalny. Dzięki swoim własnościom jest bardzo dobrym językiem edukacyjnym do wspomagania nauki programowania. Nie nadaje się do rozwiązywania skompli-



kowanych problemów numerycznych lub przetwarzania dużych ilości danych.

Atari Logo jest dobrą implementacją, która oprócz innych pożytecznych własności umożliwia manipulowanie kilkoma (do 4) żółwiami.

ACTION!

ACTION! jest czymś pośrednim między językiem Pascal a C. Jednak z powodu podobieństwa do BASICa nawet doświadczeni programiści napotykają wiele problemów.

System ACTION! składa się z edytora, samego języka, kompilatora i monitora. Edytor, pozwalający tworzyć i modyfikować programy, zgłasza się bezpośrednio po załadowaniu systemu. Zawiera dwa okna tekstowe, które mogą być przesuwane przez program. Dodatkowo wyszukuje i podmienia, kasuje linie oraz przesuwa bloki linii.

Po napisaniu programu musi być przetworzony na kod maszynowy za pomocą kompilatora, który umożliwia również łączenie kilku oddzielnych modułów w jeden program. Przydatna jest możliwość włączania wcześniej napisanych podprogramów uniwersalnych do wnętrza określonego programu.

Monitor jest sekcją sterującą systemu. Umożliwia on wywołanie edytora lub kompilatora, uruchomienie programu lub zakończenie sesji. Zapewnia również dostęp do DOS.

Sam język jest strukturalny. Podobnie do C jego procedury lub moduły mogą być pisane oddzielnie, a następnie łączone razem. Wyrażenia mogą zawierać operatory: arytmetyczne, działające na bitach lub relacji. Istnieje wiele przydatnych instrukcji, takich jak **IF... THEN... ELSE — IF... ELSE** oraz pętli **WHILE, UNTIL** i **DO**.

System ACTION! ma także bibliotekę procedur użytkowych obejmujących operacje wejścia/wyjścia, wspomagające wejście/wyjście i funkcje systemowe. Procedury wspomagające wejście/wyjście są szczególnie użyteczne dla programistów Atari, należą do nich następujące funkcje: **Open**, **Close**, **XIO**, **Note**, **Point**, **Color**, **Sound**, **Stick** i **Paddle**.

Doświadczony programista, działający w niestrukturalnym świecie BASICa, może mieć pewne kłopoty w dostosowaniu się do ACTION! Jednak po kilku godzinach wszystko stanie się naturalne. Trudniejszy może być powrót do BASICa, szczególnie po zapoznaniu się z szybkością i możliwościami.

Omówiliśmy krótko dziesięć języków, pewne z nich z kilkoma dialektami — razem osiemnaście różnych sposobów „rozmawiania" z ATARI. Czyni to 8-bitowy komputer ATARI porównywalnym z innymi, droższymi maszynami. Zatem ich właściciele rozporządzają poważnymi narzędziami, które mogą być przydatne w programowaniu użytkowym.

Każdy język ma swoje mocniejsze i słabsze strony. Pewne z nich są lepsze do zastosowań naukowych, podczas gdy inne lepiej służą początkującym użytkownikom. Widoczne są także inne różnice: szybkość, możliwości graficzne i łatwość użycia.

Edytor — wydruk 2

```
10 DIM L$(120),ML$(119);A$(1)
20 GRAPHICS 0:POKE 710,0:? "Dy
sk czy Kaseta";:INPUT A$:IF A$
<>"K" AND A$<>"D" THEN 20
30 IF A$="K" THEN 50
40 ? "Umiesc sformatowana dysk
ietke w stacji dyskow, ":? "na
stepnie nacisnij RETURN"
45 INPUT L$:OPEN #1,8,0,"D:ML.
DAT"
50 ? :? "Przysotuj kasete, nac
isnij RETURN.";:INPUT L$:OPEN
#1,8,0,"C:"
60 L$="32608 M$(1)=":L$(13)=CH
R$(34)
70 N=119:GOSUB 130:L$(14)=ML$(
1,58):L$(LEN(L$)+1)=CHR$(34):?
 #1;L$
80 L$(1)="32610 M$(59)=":L$(14
)=CHR$(34):L$(15)=ML$(59):L$(L
EN(L$)+1)=CHR$(34):? #1;L$
90 L$(1)="32612 S$=":L$(10)=CH
R$(34)
100 ML$="":N=98:GOSUB 130:L$(1
1)=ML$:L$(LEN(L$)+1)=CHR$(34)
? #1;L$
110 L$(1)="32614 E$=":L$(10)=C
HR$(34)
120 ML$="":N=69:GOSUB 130:L$(1
1)=ML$:L$(LEN(L$)+1)=CHR$(34):
? #1;L$:END
130 FOR X=1 TO N:READ A:ML$(X)
=CHR$(A):NEXT X:RETURN
140 DATA 104,104,133,204,104,1
33,203,104,104,133,205,169,0,1
41,3,6,141,2,6,141,4,6,141,5,6
150 DATA 141,6,6,238,3,6,32,68
,218,172,2,6,177,203,133,212,3
2,170,217,32,182,221,32,68,218
160 DATA 173,3,6,133,212,32,17
0,217,32,219,218,32,210,217,16
5,212,141,0,6,165,213,141,1,6,
24
170 DATA 173,0,6,109,4,6,141,4
,6,173,1,6,109,5,6,141,5,6,144
,3,238,6,6,238,2
180 DATA 6,172,2,6,196,205,208
,176,173,4,6,133,212,173,5,6,1
33,213,96
190 DATA 104,104,133,204,104,1
33,203,104,104,141,255,6,169,0
,133,213,216,165,88,133,205,16
5,89,133,206
200 DATA 174,255,6,24,165,205,
105,40,133,205,144,2,230,206,2
02,208,242,160,0,177,205,201,6
4,144,18
210 DATA 201,96,144,19,201,128
,144,18,201,192,144,6,201,224,
144,7,176,8,24,105,32,144,3,56
,233
220 DATA 64,145,203,200,192,11
4,240,2,208,215,177,203,201,32
,208,3,136,208,247,200,132,212
,96
230 DATA 104,104,141,254,6,104
,141,253,6,169,0,133,213,216,1
65,136,133,205,165,137,133,206
,160,0,177
240 DATA 205,205,253,6,208,8,2
00,177,205,205,254,6,240,15,16
0,2,177,205,24,101,205,133,205
,144,228
250 DATA 230,206,176,224,160,4
,177,205,201,55,240,4,160,0,24
0,0,132,212,96
```

Edytor BASICa

Z pewnością wielu z nas zdarzyło się popełnić błędy podczas wprowadzania do komputera programów zamieszczonych w postaci wydruków w różnorodnych czasopismach. Można mówić o szczęściu, gdy błąd wykrywany był przez interpreter języka podczas wprowadzania lub realizacji programu. A co zrobić, gdy żaden błąd nie został wykryty, a program nie chce działać? Jedynym sposobem jest wtedy porównanie wydruku z prowadzonym programem, znak po znaku, linia po linii. Najczęściej okazuje się, że błąd wyniknął z nieznacznego przekłamania w nazwie zmiennej lub zmianie operatora i jest trudny do wykrycia.

Poniższy program pomoże nam we wprowadzaniu zamieszczanych programów. W celu ułatwienia identyfikacji błędów każda linia programu badana jest bezpośrednio po wprowadzeniu. W określony sposób obliczana jest suma kontrolna, która zamieszczona jest również w wydruku programu. W razie zgodności mamy pewność, że linia programu wprowadzona została bezbłędnie. Jeśli zakończymy wprowadzanie programu za pomocą Edytora BASICa, możemy być pewni, że nie zawiera on żadnych błędów.

Program zachowuje wszystkie własności edycyjne Atari, dopuszcza także stosowanie skrótów instrukcji. Umożliwia również sprawdzenie całego programu, jeśli został on wprowadzony w sposób konwencjonalny, przy wykorzystaniu wbudowanego edytora.

Wprowadzenia Edytora

W celu utworzenia własnej kopii Edytora BASICa należy dokładnie wykonać jedną z poniższych instrukcji.

Wersja dyskowa.

1) Wprowadzić Wydruk nr 1, starannie i bez błędów.
2) Przechować program na dy-

ATARI

Sumy kontrolne

```
XS 90 DEG
DA 100 DIM NM$(20),SYN$(3),H1$
   (3),H2$(3),NMPAD$(100):NM$=
   "":H1$="7":H2$="7":DOT=0:FI
   RST=0:NUMALL=0
FP 105 NMPAD$="789/ SIN COS TA
   N 456* L10 +/- ATN 123- LOG
    EXP SQR 0.=+ X^2 1/X Y^X
       CLR"
QK 110 GRAPHICS 0:POKE 752,1:S
   ETCOLOR 2,0,0:REM CZYSZCZEN
   IE EKRANU I CZARNY KOLOR TL
   A
QM 120 GOSUB 1000:REM KRESLENI
   E KALKULATORA
HI 130 TRAP 1100:REM ODCZYT PR
   ZYCISKU
CG 140 POSITION X,Y:PRINT H1$
JV 150 XINC=(STICK(0)=7)*(X<24
   )-(STICK(0)=11)*(X>11)
GA 160 YINC=2*((STICK(0)=13)*(
   Y<12)-(STICK(0)=14)*(Y>4))
NC 170 IF XINC=1 AND X=14 THEN
    XINC=2
WE 180 IF XINC=-1 AND X=16 THE
   N XINC=-2
XU 190 IF XINC=1 AND X>=16 THE
   N XINC=4
EB 200 IF XINC=-1 AND X>=20 TH
   EN XINC=-4
WE 210 X=X+XINC:Y=Y+YINC:IF Y=
   12 THEN X=16
CR 220 INDEX=(Y-4)/2*17+X-10:X
   LEN=(X>=16)*2
GR 230 H1$=NMPAD$(INDEX,INDEX+
   XLEN)
TL 240 H2$="":FOR LOOP=1 TO LE
   N(H1$):H2$(LOOP,LOOP)=CHR$(
   ASC(H1$(LOOP,LOOP))+128):NE
   XT LOOP
SF 250 POSITION X,Y:PRINT H2$:
   IF STRIG(0)=1 THEN FOR WAIT
   =1 TO 20:NEXT WAIT:GOTO 130
ZV 252 IF STRIG(0)=0 THEN 252
JN 253 IF H1$="CLR" THEN HOLD=
   0:FIRST=0:DOT=0:NM$="":SYN$
   ="":POSITION 11,1
MW 254 IF H1$="CLR" THEN PRINT
   "                       ":NUMALL
   =0:GOTO 130
BY 255 IF LEN(NM$)>16 OR NUMAL
   L=1 THEN 280
PO 260 IF (X<14 AND Y<10) OR (
   X=11 AND Y=10) THEN NM$(LEN
   (NM$)+1)=H1$:REM SZUKANIE L
   ICZBY
KR 270 IF (X=12 AND Y=10) AND
   DOT=0 THEN DOT=1:NM$(LEN(NM
   $)+1)=H1$:REM KROPKA DZIESI
   ETNA DODANA DO LICZBY
MG 280 POSITION 11,1:PRINT "
                  ":POSITION 2
   8-LEN(NM$),1:PRINT NM$
AD 282 IF LEN(NM$)=0 AND FIRST
   =0 THEN NM$=STR$(HOLD)
NZ 285 IF LEN(NM$)=0 THEN 130
VU 290 IF X<>14 THEN GOTO 350:
   REM OPERATORY ARYTMETYCZNE
PQ 300 IF FIRST=0 THEN FIRST=1
   :HOLD=VAL(NM$):NM$="":SYN$=
   H1$:DOT=0:NUMALL=0:GOTO 130
ST 310 GOSUB 600
NH 345 NM$=STR$(HOLD):POSITION
    28-LEN(NM$),1:PRINT NM$:NM
   $="":SYN$=H1$:DOT=0:NUMALL=
   0:GOTO 130
PO 350 IF X=13 AND Y=10 THEN G
   OSUB 600:FIRST=0:POSITION 1
   1,1:PRINT "
    ":GOTO 345:REM =
PI 360 IF X<16 THEN 130
OZ 370 IF H1$="SIN" THEN HOLD2
   =SIN(VAL(NM$))
IF 380 IF H1$="COS" THEN HOLD2
   =COS(VAL(NM$))
IB 390 IF H1$="TAN" THEN HOLD2
   =SIN(VAL(NM$))/COS(VAL(NM$)
   )
ZG 400 IF H1$="L10" THEN HOLD2
   =CLOG(VAL(NM$))
JH 410 IF H1$="+/-" THEN HOLD2
   =-(VAL(NM$))
EE 420 IF H1$="ATN" THEN HOLD2
   =ATN(VAL(NM$))
BE 430 IF H1$="LOG" THEN HOLD2
   =LOG(VAL(NM$))
VC 440 IF H1$="EXP" THEN HOLD2
   =EXP(VAL(NM$))
JM 450 IF H1$="SQR" THEN HOLD2
   =SQR(VAL(NM$))
KI 460 IF H1$="X^2" THEN HOLD2
   =VAL(NM$)*VAL(NM$)
OP 470 IF H1$="1/X" THEN HOLD2
   =1/VAL(NM$)
QP 480 IF H1$="Y^X" THEN POSIT
   ION 11,1:PRINT "
        ":POSITION 28-LEN(NM$
   ),1:PRINT NM$:GOTO 300
FB 500 NM$=STR$(HOLD2):POSITIO
   N 11,1:PRINT "
      ":POSITION 28-LEN(NM$),
   1:PRINT NM$:NUMALL=1:GOTO 1
   30
PE 600 IF SYN$="/" THEN HOLD=H
   OLD/VAL(NM$)
GP 610 IF SYN$="*" THEN HOLD=H
   OLD*VAL(NM$)
LW 620 IF SYN$="-" THEN HOLD=H
   OLD-VAL(NM$)
IM 630 IF SYN$="+" THEN HOLD=H
   OLD+VAL(NM$)
RO 635 IF SYN$="Y^X" THEN HOLD
   =HOLD^VAL(NM$)
ZK 640 RETURN
PR 999 END
DP 1000 REM KRESLENIE KALKULAT
   ORA
VB 1010 POSITION 10,0:PRINT CH
   R$(17):POSITION 28,0:PRINT
   CHR$(5)
EV 1020 POSITION 10,13:PRINT C
   HR$(26):POSITION 28,13:PRIN
   T CHR$(3)
GE 1030 FOR LOOP=11 TO 27:POSI
   TION LOOP,0:PRINT CHR$(18):
   POSITION LOOP,13:PRINT CHR$
   (18)
WH 1035 POSITION LOOP,2:PRINT
   CHR$(18):NEXT LOOP
XW 1040 FOR LOOP=1 TO 12:POSIT
   ION 10,LOOP:PRINT CHR$(124)
   :POSITION 28,LOOP:PRINT CHR
   $(124):NEXT LOOP
QN 1045 POSITION 10,2:PRINT CH
   R$(1):POSITION 28,2:PRINT C
   HR$(4)
DV 1050 POSITION 11,4:PRINT "7
   89/ SIN COS TAN"
LZ 1060 POSITION 11,6:PRINT "4
   56* L10 +/- ATN"
QI 1070 POSITION 11,8:PRINT "1
   23- LOG EXP SQR"
CR 1080 POSITION 11,10:PRINT "
   0.=+ X^2 1/X Y^X"
YD 1085 POSITION 16,12:PRINT "
   CLR":X=11:Y=4:POSITION X,Y:
   PRINT "7"
BA 1090 RETURN
MH 1100 REM PROCEDURA BLEDU
TD 1110 POSITION 11,1:PRINT "
                  ":POSITION
   17,1:PRINT "BLAD"
IR 1120 HOLD=0:FIRST=0:DOT=0:N
   M$="":SYN$="":NUMALL=0
PO 1130 FOR LOOP=1 TO 250:NEXT
    LOOP
OQ 1140 GOTO 130
```

sku za pomocą rozkazu SAVE „D: EDYTORW1. BAS".

3) Oczyścić pamięć komputera rozkazem NEW
4) Wprowadzić Wydruk nr 2
5) Dla bezpieczeństwa przechować program na dysku za pomocą rozkazu SAVE „D: EDYTORW2 BAS".
6) Uruchomić program instrukcją RUN i wykonać wyświetlone na ekranie polecenia. Zbiór danych o nazwie ML. DAT zapisany będzie na dysku.
7) Załadować do pamięci pierwszy program rozkazem LOAD „D EDYTORW1. BAS".
8) Dołączyć zbiór utworzony przez program nr 2 rozkazem ENTER „D: ML. DAT".
9) Przechować program wynikowy rozkazem LIST „D: EDYTOR LST".

Wersja taśmowa

1) Wprowadzić Wydruk nr 1.
2) Przechować program na kasecie za pomocą rozkazu CSAVE (nie przewijać kasety).
3) Oczyścić pamięć komputera rozkazem NEW
4) Wprowadzić Wydruk nr 2.
5) Dla bezpieczeństwa przechować program na taśmie.
6) Uruchomić program rozkazem RUN i wykonać wyświetlone na ekranie polecenia. Zbiór danych zostanie zapisany na kasecie
7) Przewinąć taśmę i załadować pierwszy program rozkazem CLOAD
8) Dołączyć zbiór utworzony przez program nr 2 rozkazem ENTER C
9) Na innej kasecie przechować program wynikowy rozkazem LIST „C:".

Używanie Edytora

Spójrzmy na wydruk programu zamieszczony za Edytorem. Zauważmy, że każda linia programu poprzedzona jest przez kod dwuliterowy. Jest to suma kontrolna tej linii, a nie część programu

W celu wprowadzenia programu z zamieszczonego wydruku załaduj Edytor BASICa rozkazem **ENTER** i uruchom. Zostaniesz zapytany, czy chcesz używać skrótów instrukcji. Jeśli tak, to wprowadź **T** i naciśnij **RETURN** W przeciwnym razie podaj **N**. Ekran telewizora (monitora) zostanie podzielony na dwa „okna". W górnym wyświetlana będzie każda linia po przetworzeniu przez Edytor oraz suma kontrolna obliczona dla tej linii. W dolnym oknie odbywa się wprowadzanie i poprawianie linii

Gdy program oczekuje na wprowadzenie, kursor pojawia się przy lewym marginesie okna wprowadzania. Wprowadź linię programu i naciśnij RETURN. Nastąpi sprawdzenie jej poprawności oraz obliczenie sumy kontrolnej. Linia programu pojawi się w oknie komunikatu wraz z wygenerowaną sumą kontrolną. Jeśli wyświetlony kod zgadza się z zamieszczonym w wydruku, można przejść do następnej linii programu. W przeciwnym wypadku należy podać rozkaz **E** (Edit) i nacisnąć **RETURN**. Ostatnio wprowadzona linia pojawi się w dolnym oknie, gdzie może zostać poprawiona. Po wprowadzeniu poprawek naciśnij **RETURN**

Można wywołać dolną linię programu, wcześniej wprowadzoną, którą chcemy poprawić, za pomocą rozkazu E z numerem linii. Na przykład, E230 wyświetli linię 230 w oknie wprowadzania.

Jeśli używamy skrótów BASICa dwie wersje rozkazu E pracują różnie. Rozkaz E bez numeru linii wyświetli ją w takiej postaci, w jakiej została wprowadzona. Po dodaniu numeru linii będzie ona wyświetlona w postaci rozwiniętej (bez skrótów).

W dowolnej chwili można wyjść z Edytora BASICa przez naciśnięcie B (BASIC). Można teraz przejrzeć swoją pracę. Zauważmy, że linie 32600 i wyższe należą do Edytora. Nasz program ma niższe numery. Aby powrócić do Edytora, należy podać GOTO 32600

W razie wprowadzenia dłuższych programów możemy mieć ochotę na robienie przerw, oczywiście po uprzednim zabezpieczeniu naszej pracy. W tym celu wprowadzamy **L** i **RETURN**, a następnie przechowujemy program w wybranej pamięci zewnętrznej w postaci umożliwiającej późniejsze łączenie

Jeśli ktoś nie chce używać Edytora BASICa, preferując wbudowany edytor Atari, także może sprawdzić i ewentualnie poprawić wprowadzony już i przechowany w pamięci zewnętrznej program. W tym celu należy załadować go do pamięci, następnie dołączyć Edytor BASICa za pomocą ENTER i uruchomić go rozkazem **GOTO 32600** Na propozycję używania skrótów musimy odpowiedzieć **N**. Gdy edytor pojawi się na ekranie, należy podać rozkaz **P** (Post — przetwarza) i pierwsza linia programu pojawi się w oknie wprowadzania. W celu przetworzenia jej przez Edytor należy nacisnąć RETURN. Linia programu, wraz z sumą kontrolną, pojawi się w górnym oknie. Jeżeli suma kontrolna zgadza się z podaną w wydruku, należy dwa razy nacisnąć **RETURN** i przetwarzana będzie następna linia programu.

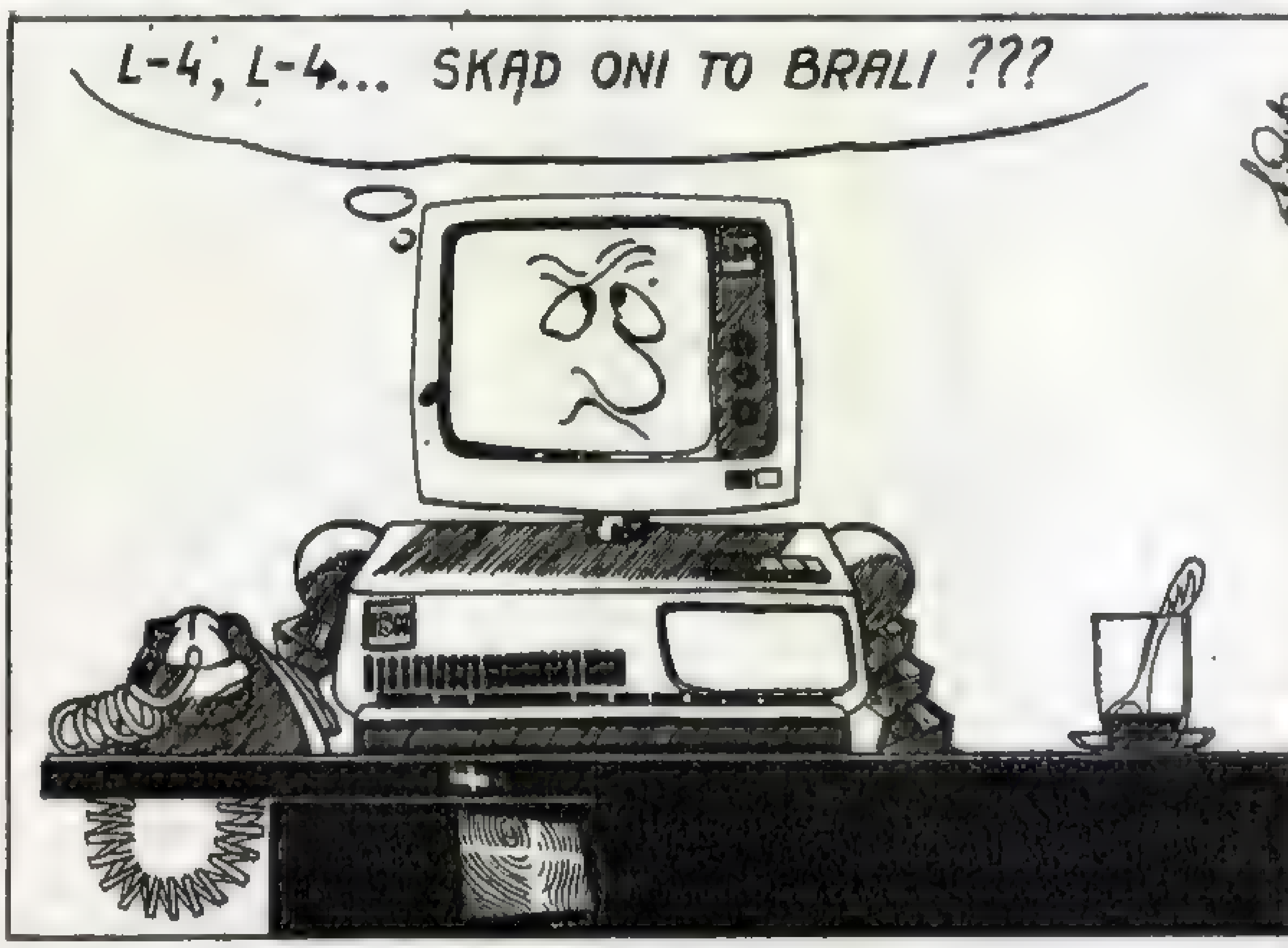
Po stwierdzeniu, że musimy poprawić aktualną linię, podajemy rozkaz E i zostanie ona przesunięta do okna wprowadzania.

Może się zdarzyć, że po wprowadzeniu programu za pomocą Edytora BASICa nie działa on właściwie. Możliwe są dwie przyczyny takiej sytuacji Po pierwsze, mogło się zdarzyć, że nie wykonaliśmy dokładnie instrukcji programu. Dlatego zawsze należy wcześniej przeczytać artykuł dotyczący programu. Po drugie, chociaż możemy zaufać Edytorowi BASICa, że unikniemy błędów wprowadzania, nie może nam on powiedzieć, czy nie przeskoczyliśmy pewnych linii. Jeśli program nie chce działać, należy upewnić się, czy wprowadziliśmy wszystkie linie.

Edytor BASICa zgodny jest z podobnymi programami używanymi aktualnie przez „ANALOG Computing" i „ANTIC". Opracowany został na podstawie następujących programów: BASIC Editor, BASIC Editor II i Typo II. Dzięki temu można go wykorzystać do wprowadzania programów zamieszczonych w tych czasopismach.

W celu bliższego zapoznania się z działaniem Edytora BASICa zamieszczamy krótki program, którego działanie poznają Czytelnicy, którzy wprowadzą go bezbłędnie.

Ludwik PIELA
Tomasz MROWIEC



SPECTRUM

```
  10 DIM a(50): LET n=0: LET max
=0: BORDER 4: CLS
  20 LET n=n+1: INPUT "wartosc (
";(n);") ? (0 = koniec) ";a(n):
IF a(n)<0 THEN LET a(n)=ABS a(n)
  30 IF a(n)>max THEN LET max=a(
n)
  40 IF a(n)=0 THEN GO TO 60
  50 GO TO 20
  60 LET s=INT (255/(n-1))-1: LE
T wsp=175/max: FOR m=1 TO n-1: I
F n<10 THEN PRINT #0;AT 1,(m-1)*
(32/(n-1));a(m);
  70 PLOT 1+(m-1)*(s+1),0: DRAW
s-2,0: DRAW 0,a(m)*wsp: DRAW -(s
-2),0: DRAW 0,-a(m)*wsp. NEXT m
  80 BEEP .2,26: PAUSE 0: GO TO
10
  90 SAVE "SLUPKI" LINE 10
```

Słupki
(J. Rubinowicz)

Spełnienie marzeń...



Joanna Worońska, studentka I roku na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego sumiennie przygotowywała się do pierwszego i jedynego egzaminu sesji zimowej — Wstępu do informatyki. Po emocjach egzaminacyjnych, gdy „napięcie" ją na tyle opuściło, że była w stanie zebrać rozproszone myśli, poprosiliśmy Joasię o rozmowę:

— Ciężko było?

— Wyobraźnia podpowiadała mi bardziej koszmarne przeżycia. Najgorsza jest trema.

— Joasiu, dlaczego zdecydowałaś się zdawać na informatykę?

— Na pewno duży wpływ mieli moi rodzice, którzy są informatykami. Pracują w Instytucie Podstaw Informatyki w PAN. Ja do trzeciej klasy licealnej myślałam o biologii. Jednak w pewnym momencie bardziej przekonałam się do matematyki. Uważam, że informatyka ma przed sobą ogromną przyszłość. Rozwija się. Daje możliwość intelektualnego wyżycia się.

— Czy do egzaminu przygotowywałaś się, ćwicząc m.in. na komputerze?

— Na wydziale, owszem, mamy komputer z 8 końcówkami, w tym niestety 4 popsute. Musimy więc szukać innych „dojść".

— Obecnie, jak sama się orientujesz, jak grzyby po deszczu powstają najróżniejsze kluby mikrokomputerowe...

— I bardzo dobrze, tylko jedno mam pragnienie. By spotkanie z komputerem nie kończyło się na grach. Żeby młodzi zapaleńcy wzbogacali swoją wiedzę, docierali do fachowych pism m.in. waszego „IKS-a", szperali w podręcznikach. Po prostu coś więcej wnosili i wynosili.

— Zdałaś egzamin, zrobiłaś pierwszy krok w przyszłość, czego ci życzyć już teraz.



— Oczywiście samych sukcesów i podtrzymania tradycji rodzinnej — chcę być informatykiem — to moje marzenie.

— Chyba bliskie spełnienia.

— Oby.

— Dziękuję za rozmowę.

ANKA

SPECTRUM

Bajty

(J. Rubinowicz)

```
  1 REM zapis na tasme RUN 9999
 100 FOR n=32 TO 127
 105 POKE 23692,255
 110 PRINT "CHR$ "; BRIGHT 1;CHR
$ n; BRIGHT 0;TAB 9;"ADRES "; BR
IGHT 1;15616+(8*(n-32)); BRIGHT
0,TAB 24;"CODE "; BRIGHT 1;n
 120 FOR w=0 TO 7
 130 PRINT TAB (4*w);PEEK (15360
+8*n+w);
 140 NEXT w
 150 PRINT : PRINT
 160 BEEP .01.9. BEEP .02,14: PA
USE 40
 170 IF INKEY$<>"" THEN PAUSE 0
 180 NEXT n
 190 GO TO 100
 200 CLS : BORDER 4. PRINT ''''"
 Program pokazuje bajty okre-
 slajace budowe cyfr, liter i
 znakow oznaczonych kodami od
 32 do 127 oraz ich adresy.

 Wyswietlanie programu mozna
 zatrzymac na przez nacisnie-
 cie dowolnego klawisza."
 210 PAUSE 500: CLS : GO TO 80
9999 CLS : SAVE "CHR$32-127" LIN
E 200: PRINT AT 4,12; PAPER 4; B
RIGHT 1;" VERIFY ": BEEP .2,30::
 VERIFY "": GO TO 200
```

KRESKA
(BASIC CPC 464) (3)
Modułowy program graficzny

W tym odcinku rozwiniemy nasz program o następujące funkcje:
— kreślenie prostopadłościanu i ostrosłupa,
— zapełnianie kolorem figur zamkniętych,
— aerograf.

Kreślenie prostopadłościanu i ostrosłupa opiera się na znanej już nam z poprzedniego odcinka zasadzie, zaznaczania punktów charakterystycznych przez naciśnięcie klawisza **COPY** bądź przycisku **FIRE**. W obu wypadkach określamy trzy punkty wierzchołkowe podstawy prostokątnej. Współrzędne czwartego punktu wyliczane są programowo. W związku z tym podobieństwem algorytmów kreślenia powyższych brył, są one rysowane na ekranie monitora za pomocą tej samej procedury. W celu wyróżnienia tych funkcji wprowadziłem parametr *bs*, który przyjmuje następujące wartości:

P — *kreślenie prostopadłościanu,*
O — *kreślenie ostrosłupa.*

W wypadku realizacji opcji **„O"** z menu **BRYŁY**, w dalszej kolejności wyliczane są współrzędne środka podstawy (*xs%, ys%*), a następnie z tego punktu wyprowadzana jest wysokość. Kolejne naciśnięcie klawisza **COPY** lub przycisku **FIRE** powoduje ustalenie współrzędnych punktu wierzchołkowego ostrosłupa (*x%, y%*), a następnie przez wykorzystanie instrukcji BASICa — **DRAW** wykreślenie bryły. Wybór opcji **„P"** determinuje wyprowadzenie wysokości prostopadłościanu z punktu o współrzędnych (*x3%, y3%*), tj. ostatniego wyznaczanego przez nas wierzchołka podstawy. Kolejne naciśnięcie przycisku **FIRE** lub klawisza **COPY** uruchamia wyliczanie wysokości prostopadłościanu, a w konsekwencji wszystkich współrzędnych drugiej podstawy. Po dokonaniu tych obliczeń następuje wykreślenie bryły.

Kolejną funkcją, którą chcę przedstawić, jest wypełnianie kolorem figur zamkniętych. Przyjęty przeze mnie algorytm jest prosty i dlatego niedoskonały. W razie przypadku zapełniania kolorem figur o bardziej skomplikowanych kształtach, zamierzony efekt często uzyskujemy przez kilkakrotne uaktywnienie procedury. Proponuję udoskonalenie tego rozwiązania we własnym zakresie. Procedura działa wg następującego algorytmu:

1. *Ruch kursora graficznego po ekranie.*
2. *Czy nastąpiło przyciśnięcie* **FIRE/COPY**?
   a. *tak — przejdź do punktu 3.*
   b. *nie — przejdź do punktu 1.*
3. *Testowanie koloru punktu startowego wypełniania oraz zapamiętanie jego współrzędnych (xpz, ypz).*
4. *Wykreślenie punktu bieżącego kolorem określonym przez zmienną kolor oraz badanie koloru punktu o współrzędnych (x+2, y), leżącego na prawo wzdłuż osi* **X**.
5. *Czy kolory są zgodne?*
   a. *tak — przejdź do punktu 4,*
   b. *nie — przejdź do punktu 6.*
6. *Postępowanie jak w punkcie 4, z tą różnicą, że badane kolejne punkty mają współrzędne (x—2, y).*
7. *Czy kolory są zgodne?*
   a. *tak — przejdź do punktu 6,*
   b. *nie — przejdź do punktu 8.*
8. *Analiza koloru punktu leżącego powyżej punktu startowego — (xpz, y+2). Czy kolory są zgodne?*
   a. *tak — przejdź do punktu 4,*
   b. *nie — przejdź do punktu 9.*
9. *Analiza koloru punktu leżącego poniżej punktu startowego (xpz, y—2). W wypadku zgodności przechodzimy do punktu 4, niezgodności 10.*
10. *Wyłączenie zapełnienia i przejście do punktu 1.*

Ostatnią funkcją, którą dziś przedstawię, jest **AEROGRAF**. Jest ona uruchamiana przez naciśnięcie klawisza funkcyjnego **FIRE/COPY**. Rezultatem jej działania jest uzyskanie na ekranie monitora efektu malowania rozpylaczem aerozolowym. Sama procedura kreśli w jednym cyklu po pięć punktów za pomocą instrukcji BASICa **PLOTR**. Współrzędne kolejnych punktów ustalane są przy wykorzystaniu wbudowanego generatora liczb pseudolosowych (instrukcja BASICa RND). Aerograf wyłączamy przez ponowne wciśnięcie klawisza funkcyjnego.

Powyższe uzupełnienia do programu **KRESKA** proponuję wprowadzić tak samo, jak przedstawiłem to w II części cyklu.

W kolejnym odcinku zostaną przedstawione funkcje:
— *wprowadzenie tekstów na rysunek,*
— *kreślenie wielokątów wpisanych w okrąg,*
— *kreślenie różnych trójkątów,*
— *elipsa,*
— *walec.*

Janek

Modułowy program graficzny

```
650 PRINT #2,"(A)EROGRAF (B)RYLY (Z)APEL
NIANIE(G)UMKA (W)OLNE KRESLENIE LUB("+CH
R$(240)+")";
685 IF INKEY(69)=0 THEN GOTO 4000
686 IF INKEY(54)=0 THEN GOTO 4500
687 IF INKEY(71)=0 THEN GOTO 5500
2315 qq%=0:GOSUB 1700:kol1=kol:xs%=xp%:y
s%=yp%
2355 IF qq%=0 AND pr%=1 THEN PLOT xs%,ys
%,kol1:qq=1:MOVE x%,y%
3997 REM ********************************
3998 REM *             AEROGRAF         *
3999 REM ********************************
4000 CLS #2:PRINT #2,"AEROGRAF - WYLACZA
NY  P - POWROT"
4010 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
4020 rb%=0
4030 GOSUB 1030:GOSUB 1170:GOSUB 1080:GO
SUB 970:GOSUB 1030
4040 IF INKEY(27)=0 THEN GOTO 560
4050 IF INKEY(F%)=0 AND rb%=0 THEN rb%=1
:FOR i=1 TO 200:NEXT i:LOCATE #2,12,1:PR
INT #2,"WLACZONY ":GOTO 4060 ELSE IF INK
EY(F%)=0 AND rb%=1 THEN rb%=0:LOCATE #2,
12,1:PRINT #2,"WYLACZONY":FOR i=1 TO 200
:NEXT i:GOTO 4030
4060 IF rb%=1 THEN GOTO 4070 ELSE GOSUB
970:GOTO 4030
4070 FOR i=1 TO 5
4080 x1%=INT((RND*10)-5)
4090 x2%=INT((RND*10)-5):x3%=366-YPOS:IF
 x3%<x2% THEN x2%=x3%
4100 PLOTR x1%,x2%,kolor
4110 NEXT
4120 GOTO 4030
4500 CLS #2:PRINT #2,"(P)ROSTOPADLOSCIAN
 (O)STROSLUP (M)ENU KRESLENIE"
4510 IF INKEY(38)=0 THEN GOTO 650
4520 IF INKEY(27)=0 THEN b$="P":GOTO 460
0
4530 IF INKEY(34)=0 THEN b$="O":GOTO 460
0
4540 GOTO 4510
4600 REM ********************************

4610 REM *       PROSTOPADLOSCIAN I     *
4615 REM *           OSTROSLUP          *
4620 REM ********************************
4630 CLS #2:PRINT #2,"WYZNACZ PIERWSZY P
UNKT PODSTAWY"
4640 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
4650 o%=0
4660 GOSUB 1030:GOSUB 1170:GOSUB 970:GOS
UB 1030:GOSUB 1080:GOSUB 970
4670 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
4680 IF INKEY(F%)=0 THEN 4700
4690 GOTO 4660
4700 PLOT x%,y%,1
4710 x1%=x%:y1%=y%
4720 LOCATE #2,1,1:PRINT #2," WYZNACZ DR
UGI PUNKT"
4730 GOSUB 1030:GOSUB 1170:GOSUB 970:GOS
UB 1030:GOSUB 1080:GOSUB 970
4735 IF o%=0 AND (x1%<>x OR y1%<>y) THEN
 PLOT x1%,y1%,1:MOVE x,y:o%=1
4740 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
4750 IF INKEY(F%)=0 THEN 4770
4760 GOTO 4730
4770 PLOT x%,y%,1:x2%=x%:y2%=y%:o%=0
4780 LOCATE #2,1,1:PRINT #2,"WYZNACZ RZU
T TRZECIEGO PUNKTU  "
4790 GOSUB 1030:GOSUB 1170:GOSUB 970:GOS
UB 1030:GOSUB 1080:GOSUB 970
4800 IF o%=0 AND (x2%<>x OR y2%<>y) THEN
 PLOT x2%,y2%,1:MOVE x,y:o%=1
4810 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
4820 IF INKEY(F%)=0 THEN 4835
4830 GOTO 4790
4835 PLOT x%,y%,1:x3%=x%:y3%=y%:o%=0
4840 LOCATE #2,1,1:PRINT #2,"WYZNACZ WYS
OKOSC              "
4845 x4%=x1%+(x3%-x2%):y4%=y3%-(y2%-y1%)
:IF b$="O" THEN xs%=x1%+(x3%-x1%)/2:ys%=
y1%+(y3%-y1%)/2:MOVE xs%,ys%:x%=xs%:y%=y
s%
4850 GOSUB 1030:GOSUB 1170:GOSUB 970:GOS
UB 1030:GOSUB 1080:GOSUB 970
4860 IF o%=0 AND (x3%<>x OR y3%<>y) THEN
 PLOT x3%,y3%,1:MOVE x,y:o%=1
4870 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
4880 IF INKEY(F%)=0 THEN 4900
4890 GOTO 4850
4900 IF b$="O" THEN 4970
4910 h%=y-y3%
4920 x5%=x1%:x6%=x2%:x7%=x3%:x8%=x4%
4930 y5%=y1%+h%:y6%=y2%+h%:y7%=y3%+h%:y8
%=y4%+h%
4940 MOVE x1%,y1%:DRAW x2%,y2%,kolor:DRA
W x3%,y3%,kolor:DRAW x4%,y4%,kolor:DRAW
x1%,y1%,kolor
4950 MOVE x5%,y5%:DRAW x6%,y6%,kolor:DRA
W x7%,y7%,kolor:DRAW x8%,y8%,kolor:DRAW
x5%,y5%,kolor
4960 DRAW x1%,y1%,kolor:MOVE x8%,y8%:DRA
W x4%,y4%,kolor:MOVE x6%,y6%:DRAW x2%,y2
%,kolor:MOVE x7%,y7%:DRAW x3%,y3%,kolor:
MOVER 1,0:x%=XPOS:y%=YPOS:GOTO 5000
4970 MOVE x1%,y1%:DRAW x2%,y2%,kolor:DRA
W x3%,y3%,kolor:DRAW x4%,y4%,kolor:DRAW
x1%,y1%,kolor
4980 DRAW x%,y%,kolor:DRAW x2%,y2%,kolor
:MOVE x3%,y3%:DRAW x%,y%,kolor:DRAW x4%,
y4%,kolor
4990 MOVE x%,y%
5000 CLS #2
5010 PRINT #2,"(M)ENU KRESLENIE     (B)R
YLY    (J)ESZCZE RAZ"
5020 IF INKEY(38)=0 THEN 560
5030 IF INKEY(54)=0 THEN 4500
5040 IF INKEY(45)=0 THEN 4630
5050 GOTO 5020
5500 REM ********************************
5510 REM *     ZAPELNIANIE KOLOREM      *
5520 REM ********************************
5530 CLS #2:PRINT #2,"ZAPELNIANIE - WYLA
CZANE (P)OWROT"
5540 LOCATE #2,1,2:PRINT #2,"UZYWAJ : "+
CHR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+
"  COPY LUB JOYSTICK";
5545 rb%=0
5550 GOSUB 1700
5560 GOSUB 1030:GOSUB 1170:GOSUB 970:GOS
UB 1030:GOSUB 1080:GOSUB 970
5570 IF sk%=0 AND pr%=1 THEN PLOT xp%,yp
%,kol:MOVE x%,y%:sk%=1
5600 IF INKEY(27)=0 THEN GOTO 560
5610 IF INKEY(F%)=0 AND rb%=0 THEN rb%=1
:FOR i=1 TO 200:NEXT i:LOCATE #2,15,1:PR
INT #2,"WLACZONE ":GOTO 5620 ELSE IF INK
EY(F%)=0 AND rb%=1 THEN rb%=0:LOCATE #2,
15,1:PRINT #2,"WYLACZONE":FOR i=1 TO 200
:NEXT i:GOTO 5560
5615 GOTO 5560
5620 pz=TEST(x+2,y):xz=x:yz=y:wsk=0:xpz=
xz:ypz=yz
5630 PLOT x,y,kolor:IF x>640 THEN 5700 E
LSE MOVER 1,0:x=x+2:ppz=TEST(x,y):IF pz=
ppz THEN GOTO 5630
5640 MOVE xz,yz:x=xz:y=yz
5650 PLOT x,y,kolor:IF x<0 THEN 5700 ELS
E MOVER -1,0:x=x-2:ppz=TEST(x,y):IF pz=p
pz THEN GOTO 5650
5655 IF wsk=1 THEN GOTO 5670
5660 yz=yz+2:pzz=TEST(xz,yz):x=xz:y=yz:I
F pzz=pz THEN MOVE xz,yz:GOTO 5630
5665 wsk=1:MOVE xpz,ypz:xz=xpz:yz=ypz
5670 yz=yz-2:pzz=TEST(xz,yz):x=xz:y=yz:I
F pzz=pz THEN MOVE xz,yz:GOTO 5630
5680 MOVER 0,-2:x%=XPOS:y%=YPOS:GOTO 600
0
5700 CLS #2:PRINT #2,"  UWAGA! FIGURA NI
EDOMKNIETA.   (M)ENU KRESLENIE (Z)APELNI
ANIE";
5705 MOVER 0,2:x%=XPOS:y%=YPOS
5710 IF INKEY(38)=0 THEN 560
5720 IF INKEY(71)=0 THEN 5530
5730 GOTO 5710
6000 GOTO 5530
```

Godzina 22.50. Tu „Radiokomputer" program Rozgłośni Harcerskiej Polskiego Radia. Zawsze we wtorki o 22.50 i piątki o 16.00. Program IV.



▲ Za stołem realizatorskim Marek Dalba



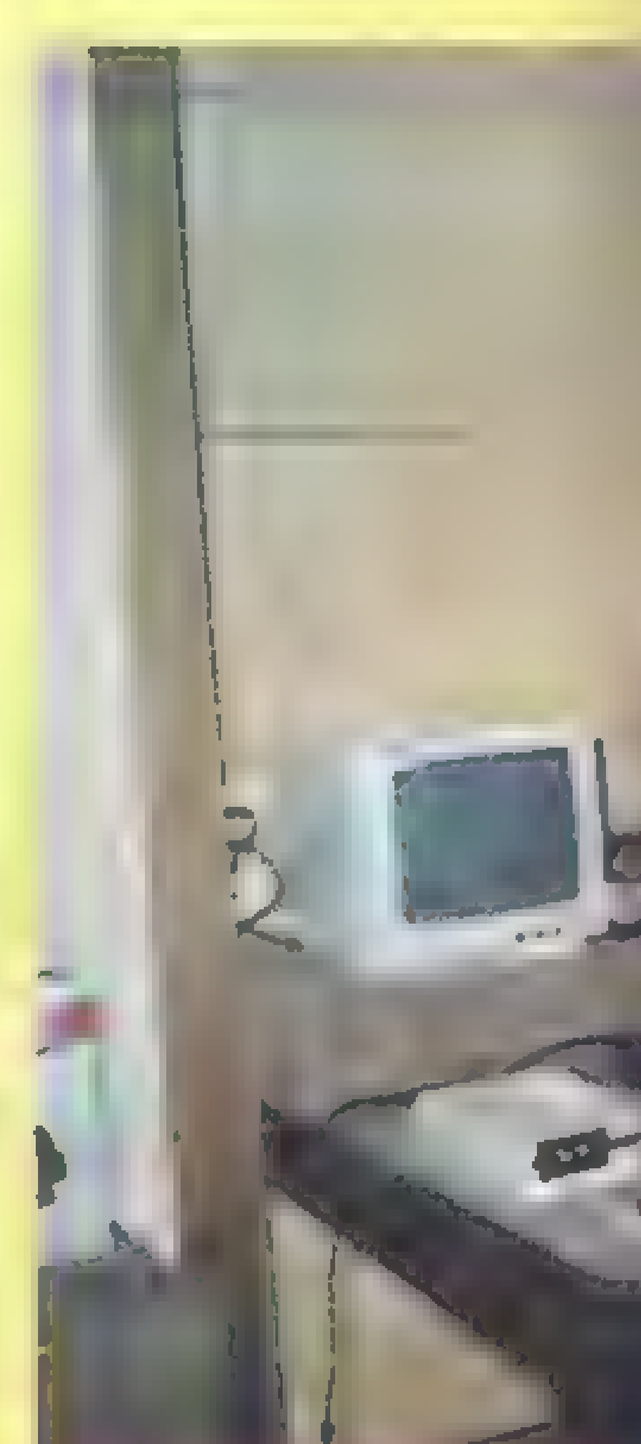
W studiu goście programu Roman Dawidson i Maciej Kasperski z „Przeglądu Technicznego" — dziś rozmawiają z Piotrem Tymochowiczem na temat wystawy komputerów.



Muzyczny przerywnik — głos ma Adam Halber. ▶



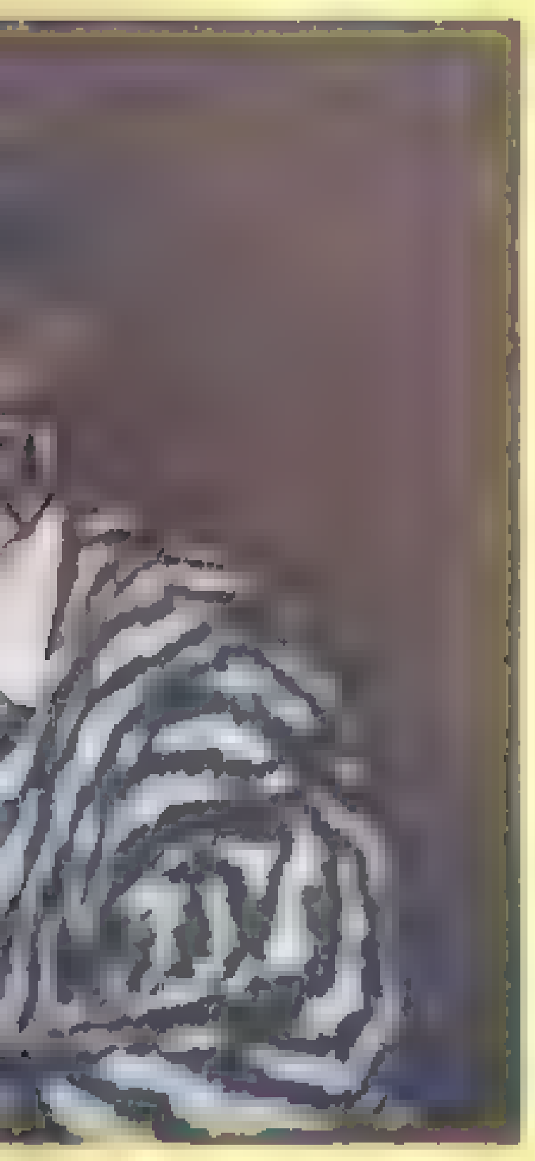
Na Mokotows „ciepłe" komp transakcje zała









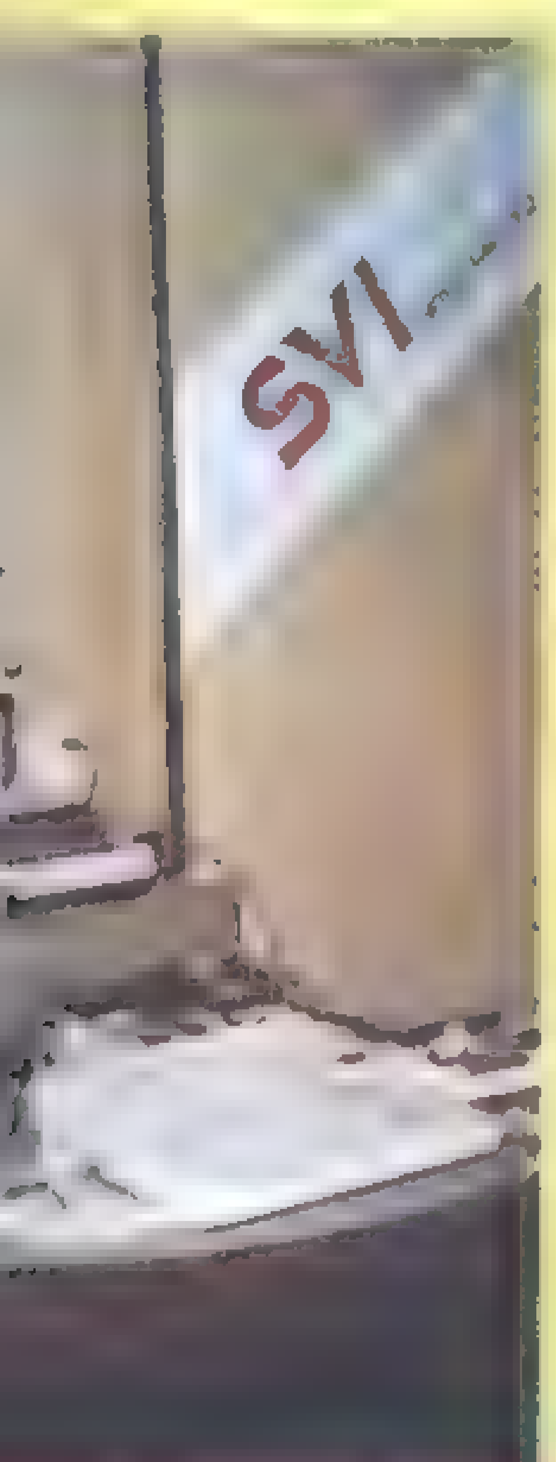


Na tę chwilę czekają użytkownicy Atari — Tomasz Jordan przygotowuje program do emisji radiowej. Tuż za nim Janusz Sikorski — sekretarz Rozgłośni Harcerskiej PR.



Foto: Jan Zelman

skiej tłok. Do sklepu Centralnej Składnicy Harcerskiej przywieźli nputery. Najważniejsze to dostać się do środka. Przy ladzie ratwia się błyskawicznie.







ZMIENNE SYSTEMOWE ROM SPECTRUM

INTERPRETACJA I BŁĘDY

T-ADDR	23668/9
CH-ADD	23645/6
NEWPPC	23618/9
NSPPC	23620
PPC	23621/2
SUBPPC	23623
OLDPPC	23662/3
OSPPC	23664
NXTLIN	23637/8
X-PTR	23647/8
RASP	23608
ERR-NR	23610/1
NMIADD	23728/9

Interpretacja instrukcji, którą absorbujemy komputer, składa się z dwóch etapów: analizy poprawności zapisu (syntaktyki) oraz wykonania (egzekucji). Odpowiada za to w ROM-ie blok zwany **interpreterem**. Analiza syntaktyczna polega na wykryciu komendy i skojarzonych z nią elementów, które zawarte są w tablicy parametrycznej. Komputer sprawdza, czy użyto odpowiednich separatorów, typów zmiennych lub innych słów kluczowych (np. THEN musi być w wyrażeniu z IF). Adres kolejnego elementu odczytywanego z tablicy znajduje się w zmiennej **T-ADDR**. Część kontrolna interpretacji zachodzi już po naciśnięciu ENTER przy wyjściu z edytora. Adresy kolejno czytanych bajtów umieszczane są w CH-ADD przy wykorzystaniu restartów 24 i 32. Błąd składni sygnalizowany jest migającym znakiem zapytania, drukowanym pod adresem wziętym z X-PTR.

Jeśli instrukcja została wczytana w trybie interakcyjnym (bez numeru linii), to zostanie zinterpretowana natychmiast, jeśli zaś podany był numer linii, to będzie umieszczona w obszarze programu. Przejście do fazy wykonania nastąpi wtedy po rozkazach RUN, GOTO lub GOSUB.

Warunkiem koniecznym rozpoczęcia egzekucji jest umieszczenie w zmiennych **NEWPPC** i **NSPPC** numeru linii i wyrażenia w niej, od których należy zacząć. Normalnie wykonuje to interpreter, wrzucając tam parametr rozkazów RUN, GOTO, czy linię autostartu zakodowaną przez LINE.

Nic nie stoi na przeszkodzie, aby wykorzystać NEWPPC i PPC. Poniższy przykład jest ulepszeniem instrukcji GOTO o numer wyrażenia w linii:

```
1000 POKE 23618,linia-256*INT (linia/256)
1010 POKE 23619,INT (linia/256)
1020 POKE 23620,wyr
```

W momencie pojawienia się potrzeby dokładnego skoku, np. do trzeciego wyrażenia w linii 8000, piszemy:

LET linia = 8000 : **LET** wyr = 3 : **GOTO** 1000

Zwróćmy uwagę, że skok jest reakcją na zmianę NSPPC. Dopóki będzie tam wartość 255, skok nie nastąpi. Można więc wyjaśnić tzw. autostart programu. Przygotujmy 3 bajty. W pierwszym młodszy bajt numeru linii, w drugim starszy, a w trzecim numer wyrażenia w linii. Np.

POKE 60000, 70
POKE 60001, 0
POKE 60002, 5

Zapiszmy to: **SAVE** „start" **CODE** 6E4, 3.
Jeśli teraz po wczytaniu programu wykonamy:

LOAD „start" **CODE** 23618, 3

to program po prostu się uruchomi od 5 wyrażenia w linii 70.

W **PPC** i **SUBPPC** umieszczana jest w trakcie egzekucji linia i wyrażenie aktualnie wykonywane. Właśnie te wartości uwzględnia komunikat o błędzie systemowym.

Zmienne **OLDPPC** i **OSPPC** stają się aktywne w razie przerwania programu klawiszem BREAK lub w wyniku rozkazu STOP. Zostają załadowane numerem linii i wyrażenia, od których wznowi się egzekucja po rozkazie **CONTINUE**. Interpreter zna te dane, gdyż w NXTLIN przechowuje sobie adres kolejnej linii programu już w momencie zajęcia się aktualną. Wpiszmy w dowolne miejsce programu linię:

xx PRINT PEEK 23637 + 256 * PEEK 23638-50

a otrzymamy adres początkowy linii po niej następującej. Rozkaz CONTINUE może zastępować w czasie egzekucji komendę GOTO. Np.

```
100 PRINT "wciśnij klawisz": IF NOT LEN INKEY$ THEN POKE 23662,100: POKE 23664,2: CONTINUE
```

Jeśli w programie jest dużo skoków do określonej linii, wtedy można jej numer schować do OLDPPC/OSPPC i używać CONTINUE zamiast GOTO.

Przykładowa konstrukcja menu wygląda tak:

```
 90 POKE 23662,100: POKE 23663,0: PO
KE 23664,1
 100 PAUSE 0: IF INKEY$<"1" OR INKEY$
>"4" THEN CONTINUE
 110 IF INKEY$="1" THEN PRINT "PODPRO
GRAM 1": CONTINUE
 120 IF INKEY$="2" THEN PRINT "PODPRO
GRAM 2": CONTINUE
 130 IF INKEY$="3" THEN PRINT "PODPRO
GRAM 3": CONTINUE
 140 IF INKEY$="4" THEN PRINT "PODPRO
GRAM 4": CONTINUE
```

Inny przykład rozwiązania menu symuluje użycie instrukcji On n GOTO... nieznanej w Spectrum — Basic. Linia 150 wyszczególnia skoki, które odpowiadają poszczególnym opcjom.

```
 100 INPUT "(1..3) ",x:
IF x<1 OR x>3 THEN GOTO 100
 110 POKE 23618,200: POKE 23619,0: POK
E 23620,x
 200 GOTO 600: GOTO 1100: GOTO 7020
 600 REM opcja 1
1100 REM opcja 2
7020 REM opcja 3
```

A teraz trochę o błędach...

Oprócz omówionego już błędu składni Spectrum może sygnalizować przekroczenie dopuszczalnych sposobów pracy już na etapie edycji. Buczy wtedy zniechęcająco, a długość tego dźwięku „trzyma" zmienna **RASP**. Normalnie jest tam wartość 64, co daje nieco ponad 1 sek. Błędy wykryte podczas egzekucji programu powodują przerwanie interpretacji i wykonania restartu 8. Pobierany jest adres procedury drukującej raport (4867), który przechowuje zmienna ERR-SP. Bajt ERR-NR dostarcza kod błędu.

W programie można wymusić dowolny raport przez:

```
10 POKE 23610, kod:
```

lub:

```
10 POKE 23728, 207: REM RST 8
20 POKE 23729, kod
30 RANDOMIZE USR 23728
```

W ostatnim przykładzie umieściłem program w obszarze zmiennych systemowych. W podręczniku oba adresy figurują jako wolne, choć w istocie miały służyć przechowywaniu adresu procedury przerwania niemaskowalnego (NMI). W standardowym Spectrum nie znalazło to zastosowania, ale oba wolne bajty „zagospodarował system dyskowy z nim współpracujący, mianowicie TOS (Timex Operating System). Znaczenie bajtów jest wtedy następujące:

23728 **SYSERR** — kod błędu lub 0 jeśli nie ma błędu

23729 **TRAP** — jeśli wartość inna niż 0, to system nie drukuje raportów, umieszczając tylko kod błędu w SYSERR:

Program poniższy powoduje, że jeśli nie powiedzie się odczyt tablicy zmiennych f, to egzekucja nie zostanie wstrzymana, a po naciśnięciu klawisza „r" można odczyt powtórzyć np. po zmianie niewłaściwej dyskietki:

```
  10 POKE 23729,255
  20 LOAD *"F$"DATA f$()
  30 IF PEEK 23728<>0 THEN GOTO 1000
  40 PRINT f$(1)
  50 STOP
1000 PRINT #1;AT 0,0;"błąd dyskowy "
;PEEK 23728',"r powtarza"
1010 PAUSE 1: PAUSE 0: INPUT ()
1020 GOTO 20+20*(PEEK 23556<>82)
```

W następnym numerze o wskaźnikach obszaru programu i zmiennych Basic.

K. MAMCARZ

SPECTRUM (test monitora)

```
 100 REM Program-MONITEST
 105 FOR n=USR "a" TO USR "a"+6
STEP 2: POKE n,170: POKE n+1,255
: NEXT n
 110 BORDER 4: CLS : BORDER 4
 115 FOR n=0 TO 31: PRINT #0; PA
PER 7;AT 0,n;"A";AT 1,n;"A": NEX
T n
 120 FOR n=0 TO 174 STEP 2: PLOT
 0,n: DRAW 255,0: NEXT n
 125 FOR n=0 TO 255 STEP 2: PLOT
 n,0: DRAW 0,175: NEXT n
 130 BEEP .2,26: PAUSE 0: PRINT
AT 6,0;""
 135 FOR m=1 TO 6: FOR n=0 TO 7:
 PRINT PAPER n;"  ";: NEXT n: NE
XT m
 140 PRINT AT 11,0;""
 145 FOR m=1 TO 6: FOR n=0 TO 7:
 PRINT PAPER n; BRIGHT 1;"  ";:
NEXT n: NEXT m
 150 BEEP .2,14: PAUSE 0
 155 PLOT 41,175: DRAW 175,-175:
 PLOT 40,0: DRAW 175,175: PLOT 4
0,1: DRAW 175,0: DRAW 0,174: DRA
W -174,0: DRAW 0,-174
 160 BEEP .2,2: PAUSE 0
 165 FOR n=-60 TO 69: BEEP .05,n
: PRINT AT 1,1;" ";n;" ": PAUSE
3: NEXT n: PAUSE 100: GO TO 0100
9000 CLS : SAVE "MONITEST" LINE
100: PRINT AT 3,11; FLASH 1; BRI
GHT 1;" VERIFY ": VERIFY "MONI
TEST": GO TO 100
```

(J. Rubinowicz)

Odkrywcy gwiazd



„Nasz klub jest narzędziem pracy — komputerem posługujemy się w astronomii" — mówi **Mieczysław Borkowski**, prezes łódzkiego Klubu Komputerowego Młodych Mistrzów Techniki „Andromeda". Jest to jeden z nielicznych, jeśli nie jedyny polski klub komputerowy, który zajmuje się wyłącznie astronomią. „Andromeda" jest jednym z kół zainteresowań istniejących przy Specjalistycznym Ognisku Pracy Pozaszkolnej Planetarium i Obserwatorium Astronomicznego miasta Łodzi

Dlaczego właśnie astronomia? Krótka rozmowa z prezesem daje odpowiedź na to pytanie. Planetarium prowadzi dla szkół, przedszkoli i zorganizowanych grup zajęcia popularyzujące wiedzę astronomiczną. Pomaga w tym: model planetarium, filmy video, gromadzone od 10 lat (niestety czarno-białe, nagrywane i wyświetlane na polskim magnetowidzie MTV-10) oraz programy komputerowe. Dzięki nim, obserwujemy niebo, można odbywać podróż w czasie i przestrzeni. Na monitorze wyświetla się obraz nieba widzianego z dowolnego punktu kuli ziemskiej, w dowolnym momencie, sprzed roku lub na przykład za kilka tysięcy lat. Na ekranie można zatem obejrzeć zaćmienie słońca opisane przez Prusa w *„Faraonie"* lub niebo, jakie w czasie swojego życia oglądał Mikołaj Kopernik. Program ten wypisuje na żądanie podstawowe informacje encyklopedyczne o każdym obiekcie wyświetlonym na ekranie. Człowiek, oglądając wygwieżdzone niebo, stoi nieporadny wobec ogromnej liczby „migających" ciał niebieskich. Nawet jeśli korzysta się z pomocy astronoma lub atlasu, trudno zidentyfikować oglądane gwiazdy. Na niebie po prostu nie ma punktów odniesienia, a gwiazdy są do siebie podobne. Natomiast po monitorze porusza się krzyżyk, którym precyzyjnie wybiera się gwiazdę lub grupy gwiazd i otrzymuje wyczerpujące informacje — astronoma zastąpi program edukacyjny

Inny program, traktowany trochę z przymrużeniem oka, sporządza horoskopy. Trzeba znać tylko dokładne współrzędne swojego miejsca urodzenia (odczytuje się je z precyzyjnych map geodezyjnych) i czas z dokładnością co do sekundy, gwarantuje to większą precyzję w określaniu indywidualnej drogi życia. W „Andromedzie" tworzone są również własne programy jak mówi Mieczysław Borkowski: „wykorzystuję naturalną ochotę zabawy do pisania programów dla astronautyki" Programy zagraniczne z reguły są uniwersalne, ale pociąga to za sobą mniejszą dokładność. Natomiast programy własne klubu napisane zostały z myślą o konkretnym problemie, w związku z czym wykorzystują dokładniejsze wzory i przez to są bardziej precyzyjne. Jednym z nich jest program „Refleks", służący do oceny i trenowania refleksu niezbędnego przy obserwacji zjawisk astronomicznych. Mierzy się czas zakrycia gwiazd przez tarczę Księżyca, dla obserwacji nadawany jest w paśmie 78,5 kHz czas astronomiczny z dokładnością do jednej setnej sekundy. Wyniki obserwacji: czas, współrzędne punktu przesyłane są do obserwatorium w Tokio. Napływające tam informacje o zjawiskach astronomicznych z całego świata służą profesjonalistom od obliczeń.

„Andromeda", podobnie jak wiele innych klubów, wyposażony jest w zbieraninę sprzętu — ma ZX Spectrum (6), Commodore C128 (2), Amstrada CPC 6128 i Meritum. Zmusza to członków klubu do przeróbek programów, utrudnia to również obsługę techniczną Choć z naprawą sprzętu klub ma bardzo mało kłopotów, a to dzięki Mikołajowi Przychodzie — właścicielowi prywatnego zakładu usługowego. Po klubie krąży anegdotka: po pół roku leżenia w państwowym zakładzie komputer, którego „absolutnie" nie można było naprawić, ożył po dotknięciu przez „złotą rękę" pana Mikołaja. Po dwóch latach działalności Mieczysław Borkowski spokojnie może powiedzieć *„Andromeda ma szansę działania — gdyż ma bazę techniczną, zapewniającą obsługę sprzętu, po prostu takie są możliwości działania Planetarium"*

Komputery ożywiły zainteresowania dzieci astronomią. Nie znaczy to bynajmniej, że wszystkie marzą o karierze astronomów. Traktują zajęcia w klubie jako hobby, jako sposób poznawania naszego świata. Motto ich działalności określa cytat umieszczony w sali klubu. *„Pożytki astronomii płyną stąd, że uświadamia ona nam, jak małe jest ciało człowieka, a jak wielki jest jego umysł"*. Podczas codziennej gonitwy zapominamy o ogromie wszechświata, o tym, że zaobserwowano już tysiące galaktyk, z których każda składa się z miliardów gwiazd podobnych do Słońca. Ziemia jest jedną z planet krążących po orbicie wokół Słońca, planetą zamieszkaną, której mieszkańcy jak na razie wykazują zbyt dużo energii, by się nawzajem zabijać. Zgodnie ze słowami prezesa „Andromedy", astronomia uczy nas, że jesteśmy niewielką cząstką kosmosu. Ziemia podlega jako fragment kosmosu ruchom astronomicznym. Dlatego powinniśmy przestać myśleć egoistycznie, jak panowie życia i śmierci organizmów ziemskich, a zacząć myśleć jak elementy kosmosu.

AMSTRAD — układ

```
10 MODE 1:BORDER 0:PAPER 0:PEN 1
20 INK 0,0:INK 1,26:INK 2,13,26:DEG
30 FOR x=1 TO 360:ORIGIN 320,200
40 DRAW 20*COS(x),20*SIN(x),1
50 PLOT 45*COS(x),15*SIN(x)
60 ORIGIN 275,200
70 DRAW 3*COS(x),3*SIN(x),1
80 ORIGIN 320,200
90 PLOT 70*COS(x),21*SIN(x)
100 ORIGIN 250,200
110 DRAW 3*COS(x),3*SIN(x)
120 ORIGIN 320,200
130 PLOT 95*COS(x),30*SIN(x)
140 PLOT 120*COS(x),44*SIN(x)
150 PLOT 157*COS(x),56*SIN(x)
160 PLOT 197*COS(x),67*SIN(x)
170 PLOT 233*COS(x),78*SIN(x)
180 PLOT 270*COS(x),89*SIN(x)
190 PLOT 300*COS(x),99*SIN(x)
200 ORIGIN 20,200
210 DRAW 4*COS(x),4*SIN(x)
220 ORIGIN 50,200
230 DRAW 4*COS(x),4*SIN(x)
240 ORIGIN 87,200
250 DRAW 8*COS(x),8*SIN(x)
260 ORIGIN 163,200
270 DRAW 12*COS(x),12*SIN(x)
280 ORIGIN 123,200
290 DRAW 9*COS(x),9*SIN(x)
300 ORIGIN 225,200
310 DRAW 4*COS(x),4*SIN(x)
320 ORIGIN 200,200
330 DRAW 4*COS(x),4*SIN(x):NEXT
340 ORIGIN 0,0:t=TIME+700:WHILE TIME<t
350 PLOT RND*640,RND*400:WEND
360 PLOT RND*640,RND*400,2
370 GOTO 370
```

Dariusz OTTO

PASCAL (4)

Spośród dotychczas przedstawionych programów dwa ostatnie (por. cz. 3, przykłady 9 i 10) należą do „najdłuższych". Czytelnik zauważył zapewne, że istotny wpływ na ich długość mają powtarzające się podobne sekwencje instrukcji. I tak, w obu programach:

— dla wczytania dwóch macierzy (a i b) zapisano dwukrotnie podobne sekwencje instrukcji,

— dla wyprowadzenia na ekran monitora trzech macierzy (a, b i c) zapisano trzykrotnie podobne sekwencje instrukcji.

Sytuacja taka występuje w wielu opracowywanych programach. Często bowiem musimy w różnych miejscach wykonywać niemal identyczne (z wyjątkiem argumentów) sekwencje instrukcji. Dla programisty wygodne jest więc wyodrębnienie takich sekwencji i traktowanie ich jako swojego rodzaju wzorów (podobnie jak w matematyce), zwanych w językach programowania procedurami, z ustaloną listą zmiennych — zwanych parametrami formalnymi. W chwili korzystania ze wzoru (procedury) podawana byłaby nazwa wzoru (procedury) wraz z listą argumentów, zwanych parametrami aktualnymi. Byłoby to równoznaczne z zastąpieniem parametrów formalnych parametrami aktualnymi. Procedury wyznaczające wartość i występujące w wyrażeniach nazywane są funkcjami. Procedury i funkcje są tematem niniejszej części.

1. PROCEDURY

Procedura występująca w bloku programu musi być opisana w jego części definiującej, w postaci, tzw. deklaracji procedury. W przykładzie 11 przedstawiono zmodyfikowaną wersję programu mnożenia dwóch macierzy (por. cz. 3), w których powtarzające się podobne sekwencje instrukcji, czytania i wyprowadzania macierzy, zadeklarowano jako procedury „czytać", „pisać".

Są one wykorzystywane w części operacyjnej programu w postaci tzw. wywołań procedury lub inaczej, instrukcji procedury.

Przykład 11

```
PROGRAM prog11;
 (mnozenie macierzy - wersja z procedurami)
 CONST
  dw=1; (dolna wartosc indeksu wierszy)
  gw=7; (gorna wartosc indeksu wierszy)
  dk=1; (dolna wartosc indeksu kolumn)
  gk=7; (gorna wartosc indeksu kolumn)
 TYPE
  tablica=ARRAY[dw..gw,dk..gk] OF real;
 VAR
  p,           (liczba wierszy macierzy a)
  q1,          (liczba kolumn macierzy a )
  q2,          (liczba wierszy macierzy b)
  r:integer;   (liczba kolumn macierzy b )
  a,b,         (macierze wejsciowe)
  c:tablica;   (macierz wynikowa  )
  i,j,k:integer; (zmienne robocze)
  s:real;
 PROCEDURE czytmac(pdw,ilw,pdk,ilk:integer;
                   VAR mac:tablica);
   (parametry procedury:
     pdw - dolna wartosc indeksu wierszy,
     ilw - liczba wierszy,
     pdk - dolna wartosc indeksu kolumn,
     ilk - liczba kolumn,
     mac - adres macierzy)
  VAR
    i,j:integer; (zmienne robocze)
  BEGIN
   FOR i:=pdw TO ilw DO
    BEGIN
     FOR j:=pdk TO ilk DO
      READ(mac[i,j]);
     READLN
    END
  END;
 PROCEDURE piszmac(pdw,ilw,pdk,ilk:integer;
                   VAR mac:tablica);
   (parametry procedury - jak
        w procedurze czytmac)
  VAR
    i,j:integer; (zmienne robocze)
  BEGIN
   FOR i:=pdw TO ilw DO
    BEGIN
     FOR j:=pdk TO ilk DO
      WRITE(' ':5,mac[i,j]:10);
     WRITELN
    END
  END;
 BEGIN
  READLN(p,q1);(wczytanie wymiarow)
               { macierzy a       }
  IF (p>gw) OR (q1>gk) THEN
   BEGIN
    WRITELN;
    WRITELN('niepoprawne wymiary macierzy a')
   END
  ELSE
   BEGIN
    czytmac(dw,p,dk,q1,a);
    READLN(q2,r);(wczytanie wymiarow }
                 (macierzy b
    IF (q2>gw) OR (r>gk) THEN
     BEGIN
      WRITELN;
      WRITELN('niepoprawne wymiary macierzy b')
     END
    ELSE
     BEGIN
      czytmac(dw,q2,dk,r,b);
      WRITELN;(wypisanie wczytanych macierzy }
      WRITELN(' ':5,'macierz a');
      piszmac(dw,p,dk,q1,a);
      WRITELN(' ':5,'macierz b');
      piszmac(dw,q2,dk,r,b);
      IF q1=q2 THEN (sprawdzenie zgodnosci  )
       BEGIN        (wymiarow macierzy a i b)
        FOR i:=dw TO p DO (mnozenie macierzy)
         FOR j:=dk TO r DO
          BEGIN
           s:=0;
           FOR k:=dk TO q1 DO
            s:=s+a[i,k]*b[k,j];
           c[i,j]:=s
          END;
        WRITELN;
        WRITELN(' ':5,'macierz c=a*b');
        piszmac(dw,p,dk,r,c)
       END
      ELSE
       WRITELN('niezgodne wymiary macierzy',
               ' a i  b')
     END
   END
END.
```

Deklaracja procedury składa się z nagłówka procedury i bloku procedury. W ogólnym przypadku nagłówek procedury ma postać:

PROCEDURE
nazwa (lpf);

gdzie:

PROCEDURE — słowo kluczowe, oznaczające deklarację procedury,
nazwa — nazwa procedury,
lpf — lista parametrów formalnych określająca liczbę, rodzaj, kolejność i typ parametrów formalnych.

Należy zauważyć, że w listach parametrów formalnych występują zapisy o postaci odpowiadającej zapisom deklaracji zmiennych (por. cz. 1), przy czym część zapisów może być poprzedzona słowem kluczowym **VAR**. Te ostatnie to parametry wejściowo-wyjściowe. Drugi rodzaj parametrów formalnych to parametry wejściowe — nie są one poprzedzone żadnym słowem kluczowym. Parametry wejściowe służą do przekazywania wartości parametrów aktualnych (tzn. parametrów, którymi zostaną zastąpione parametry formalne w wywołaniu procedury) do procedury. Ten rodzaj parametrów nie może być wykorzystywany do przekazywania

wyników działania procedury na zewnątrz procedury. Parametry wejściowo-wyjściowe służą do przekazywania wartości parametrów aktualnych do procedury oraz umożliwiają przekazywanie wyników jej działania na zewnątrz. Po wywołaniu procedury działania na parametrach wejściowo-wyjściowych wykonywane są faktycznie na odpowiadających im parametrach aktualnych.

Blok procedury jest zbudowany analogicznie, jak blok programu. Składa się z części definiującej i części operacyjnej. W części definiującej mogą występować:

- definicje stałych,
- definicje typów,
- deklaracje zmiennych,
- deklaracje procedur.

Część operacyjna to ciąg instrukcji zawarty między parą słów kluczowych **BEGIN—END**. Separatorem, kończącym blok procedury, jest średnik (w wypadku programu jest to kropka).

Wywołanie procedury powoduje zastąpienie parametrów formalnych parametrami aktualnymi oraz wykonanie sekwencji instrukcji opisanej w części operacyjnej bloku procedury. Ogólna postać wywołania procedury jest następująca:

nazwa (lpa);

gdzie:

nazwa — nazwa zadeklarowanej procedury,

lpa — lista parametrów aktualnych; należy podkreślić, że musi być ona zgodna (co do ilości, kolejności rodzaju i typu parametrów) z listą parametrów formalnych.

Program „prog 11" wykonano dla takich samych danych, jak w programie „prog 10" (por. cz. 3), uzyskując identyczny rezultat.

Zajmijmy się teraz bliżej obiektami, które mogą wystąpić w części operacyjnej procedur. W tym celu rozpatrzmy program „prog 12" z przykładu 12 (jest to jedno z możliwych rozwiązań zadania 8 z cz. 3).

Przykład 12

```
PROGRAM prog12;
  CONST
    n=100;
  TYPE
    ciag=ARRAY[1..n] OF real;
  VAR liczby:ciag;
      ilosc:integer;
  PROCEDURE prostewyb(ilosc:integer;VAR
                           liczby:ciag);
  {porzadkowanie ciagu liczb rzeczywistych
   metoda przez proste wybieranie}
    VAR i,j,k:integer;
        x:real;
    BEGIN
     FOR i:=1 TO ilosc-1 DO
      BEGIN
       k:=i;
       x:=liczby[i];
       FOR j:=i+1 TO ilosc DO
        IF liczby[j]<x THEN
         BEGIN
          k:=j;
          x:=liczby[j]
         END;
       liczby[k]:=liczby[i];
       liczby[i]:=x
      END
    END;
  PROCEDURE czytajliczby;
    {czytanie ciagu liczb}
    VAR i:integer;
    BEGIN
      FOR i:=1 TO ilosc DO
       READ(liczby[i])
    END;
  PROCEDURE drukujliczby;
    {wyprowadzanie ciagu liczb}
    VAR
      ilpw,
      reszta,
      i,j,k:integer;
    BEGIN
      ilpw:=ilosc DIV 5;
      reszta:=ilosc MOD 5;
      WRITELN;
      k:=0;
      FOR i:=1 TO ilpw DO
       BEGIN
        FOR j:=1 TO 5 DO
          WRITE(' ',liczby[k+j]:10:3);
        WRITELN;
        k:=k+5
       END;
       FOR i:=1 TO reszta DO
         WRITE(' ',liczby[k+i]:10:3)

    END;
 BEGIN
  REPEAT
   WRITELN;
   WRITELN('podaj ilosc wyrazow ciagu <=',
      n);
   READLN(ilosc)
  UNTIL (ilosc>0) AND (ilosc<=n);
  WRITELN('wprowadz wyrazy ciagu');
  czytajliczby;
  WRITELN;
  WRITELN('wprowadzone wyrazy ciagu:');
  drukujliczby;
  prostewyb(ilosc,liczby);
  WRITELN;
  WRITELN('wyrazy ciagu po uporzadkowaniu:');
  drukujliczby
 END.
```

W części definiującej programu zadeklarowano trzy procedury, przy czym dwie z nich są procedurami bezparametrowymi, tzn. lista parametrów formalnych jest pusta. W każdej z procedur występują tzw. zmienne lokalne. Są to zmienne zadeklarowane w części definiującej procedury (nie są „widoczne" poza procedurą). Dlatego też w różnych procedurach mogą być deklarowane zmienne o takich samych nazwach.

W procedurach „czytajliczby" i „drukujliczby" wykorzystywane są zmienne „liczby" i „ilość", zadeklarowane poza tymi procedurami (w części definiującej programy). Są to tzw. zmienne globalne, deklarowane w bloku nadrzędnym, tj. otaczającym daną procedurę. Należy zaznaczyć, że w bloku deklarowanej procedury są dostępne wszystkie zmienne zadeklarowane w dowolnym z bloków nadrzędnych. Dotyczy to również stałych, typów i procedur.

W procedurze „prostewyb" występują parametry formalne, których nazwy są identyczne z nazwami zmiennych zadeklarowanych w części definiującej program. „Konflikt nazw" jest pozorny, gdyż nazwy parametrów formalnych są lokalne dla danej procedury.

Program „prog 12" wykonano dla następującego ciągu liczb: 44, 33, — 2,51, 26, 17, 15, 14, — 35, 8, 6, 5, 4, 2,2, 3 i — 2 uzyskując następujące wyniki:

```
                 wprowadzone wyrazy ciagu:

    44.000     33.000     -2.000     51.000     26.000
    17.000     15.000     14.000    -35.000      8.000
     6.000      5.000      4.000      2.000      2.000
     3.000     -2.000
 wyrazy ciagu po uporzadkowaniu:

   -35.000     -2.000     -2.000      2.000      2.000
     3.000      4.000      5.000      6.000      8.000
    14.000     15.000     17.000     26.000     33.000
    44.000     51.000
```

Widać wyraźnie, że działanie procedury „prostewyb" odbywa się na zmiennej „liczby" zadeklarowanej w części definiującej programu, przekazanej do procedury jako parametr wejściowo-wyjściowy.

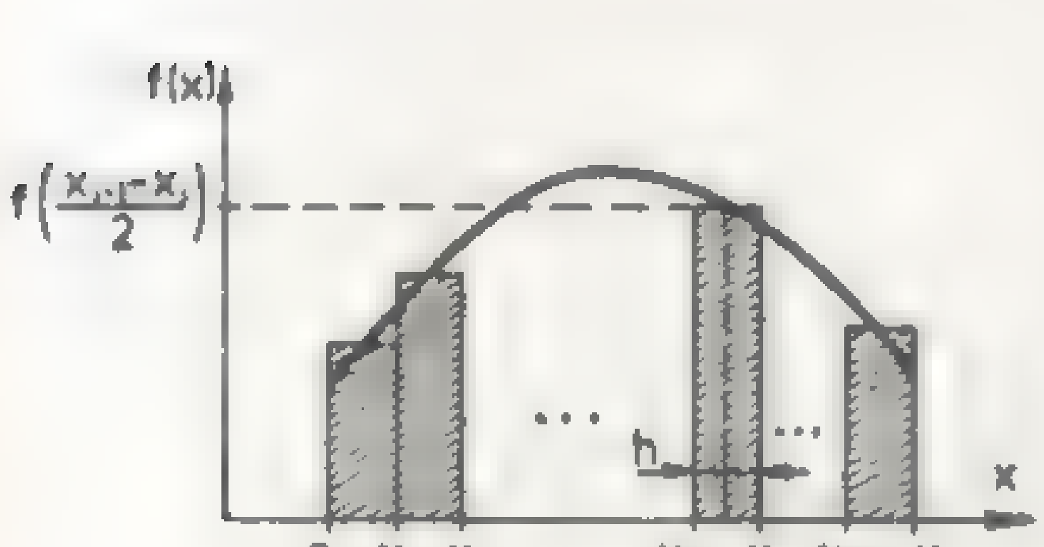
Kończąc omawianie procedur, warto podkreślić, że umiejętność posługiwania się nimi ma zasadniczy wpływ na jakość i styl pracy programisty. Procedury służą bowiem nie tylko do skracania tekstu programu, lecz również pozwalają na jego strukturalizację, tj. podzielenie na logicznie spójne części. Prawidłowa struktura programu przyczynia się do zwiększenia jego czytelności i ułatwia weryfikację programu. Warto zatem często posługiwać się procedurami nawet wtedy, gdy dany ciąg instrukcji występuje w programie tylko raz i argument skrócenia tekstu nie wchodzi w grę.

2. Funkcje

W kilku wcześniej zaprezentowanych programach wykorzystywane były standardowe funkcje **ABS** i **SQRT**. Język PASCAL pozwala na definiowanie własnych funkcji tzw. funkcji niestandardowych. Mogą one być używane do strukturalizacji wyrażeń arytmetycznych. Sposób definiowania i użycia funkcji niestandardowych (jest podobny, jak w procedurach) omówimy na przykładzie funkcji wyznaczającej wartość całki

$$\int_a^b f(x)\,dx$$

metodą prostokątów. W metodzie tej przedział całkowania od $a = x_0$ do $b = x_n$ dzieli się na *n* równych części punktami $x_1, \ldots x_i, \ldots x_n$. Następnie oblicza się wartości funkcji podcałkowej w punktach: dla $i \in \langle 1,n \rangle$ $\frac{x_i - x_{i-1}}{2}$



Wartość całki jest sumą pól prostokątów utworzonych jak na rysunku. Można ją zatem wyrazić wzorem:

$$\int_a^b f(x)\,dx \approx h \sum_{i=1}^{n} f(x_i)$$

gdzie:

n — liczba podprzedziałów,

$h = \frac{b-a}{n}$

szerokość podprzedziału,

$x_i = a + (i - \frac{1}{2})\,h$

W przykładzie 13 przedstawiono funkcję wyznaczającą wartość całki

$$\int_a^b (\ln x + 5x)\,dx$$

Przyjęto tu, że kolejne przybliżenia całki mają być obliczone (przy podwojonej każdorazowo liczbie podprzedziałów) dotąd, aż bezwzględna wartość różnicy między dwoma kolejnymi przybliżeniami będzie większa od zadanej dokładności obliczeń.

Przykład 13

```
PROGRAM prog13;
  VAR
   dp,gp,dokl,wartcalki:real;
          .(zmienne robocze)
  FUNCTION calka(a,b,eps:real):real;
   (wyznaczanie wartosci calki dla
    funkcji f(x)=ln(x)+5*x; parametry:
     a-dolna granica przedzilu calkowania,
     b-gorna granica przedzilu calkowania,
     eps-dokladnosc obliczen.)
   VAR
    n:integer; (liczba podprzedzialow  )
    h,         (szerokosc podprzedzialu)
    w1,        (wartosc calki z kroku  )
               (poprzedniego           )
    w2,        (aktualna wartosc calki )
    x:real:    (zmienne robocze        )
    i:integer;
   BEGIN
    n:=1;             (ustawienie wartosci)
    h:=b-a;           (poczatkowych       )
    x:=a+h/2:
    w2:=(LN(x)+5*x);
    REPEAT
     n:=2*n;
     h:=h/2;
     w1:=w2;
     w2:=0;
     x:=a+h/2;
     FOR i:=1 TO n DO
      BEGIN
       w2:=w2+(LN(x)+5*x);
       x:=x+h
      END;
     w2:=w2*h
    UNTIL (ABS(w2-w1) <= eps);
    calka:=w2
   END;
BEGIN
 WRITELN;
 WRITELN('podaj:-dolna granice przedzi',
        'alu calkowania');
 WRITELN(' ':6,'-gorna granice przedzi',
        'alu calkowania');
 WRITELN(' ':6,'-dokladnosc obliczen');
 READ(dp,gp,dokl);
 wartcalki:=calka(dp,gp,dokl);
 WRITELN;
 WRITELN('wartosc calki dla funkcji',
        '  f(x)=ln(x)+5*x');
 WRITELN('w przedziale ',dp,' ':2,gp);
 WRITELN('wyliczona z dokladnoscia ',dokl);
 WRITELN('wynosi: ',wartcalki)
END.
```

Deklaracja funkcji zapisywana podobnie, jak deklaracja procedury, przy czym nazwa funkcji odgrywa podwójną rolę: jako nazwa sekwencji instrukcji i jako nazwa zmiennej uzyskującej określoną wartość na skutek wykonania funkcji.

Różnice między deklaracją funkcji a deklaracją procedury są następujące:

a) inna postać nagłówka:
   - — słowo kluczowe FUNCTION zamiast PROCEDURE,
   - — lista parametrów formalnych nie może być pusta,
   - — musi być podany typ wyniku funkcji (dopuszczalny jest tylko typ prosty),

b) w części operacyjnej bloku funkcji musi wystąpić co najmniej jedna instrukcja przypisania, której lewą stroną jest nazwa tej funkcji, i co najmniej jedna z tych instrukcji musi być wykonana podczas wywołania funkcji, wartością funkcji jest ostatnia wartość nadana nazwie funkcji (jeśli żadne przypisanie nie nastąpi, to wartość funkcji jest nieokreślona),

c) wywołanie funkcji ma postać identyczną jak wywołanie procedury, przy czym musi wystąpić jako element wyrażenia

Zasady przekazywania parametrów i dostępności obiektów są identyczne jak dla procedur.

Program „prog 13" wykonano dla a = h, b = 11 i eps = 10, uzyskując następujący wynik:

```
wartosc calki dla funkcji  f(x)=ln(x)+5$x
w przedziale   4.0000000000E+00    1.1000000000E+01
wyliczona z dokladnoscia   1.0000000000E-04
wynosi   2.7633169037E+02
```

Proponujemy rozwiązanie dotychczas przedstawionych zadań z wykorzystaniem procedur i funkcji.

Stefan ROZMUS

SKOCZEK SPECTRUM

```
 100 REM dane rozruchowe
 110 LET pamiec=60000: LET rekor
d=60300: POKE rekord,1. POKE rek
ord+1,1
 120 GO TO 0600
 130 REM poczatek cyklu
 140 CLS : LET skok=1: LET powt=
0: LET w1=10: LET w2=16: LET stw
1=10: LET stw2=16
 150 PRINT #0,AT 0,1;"Z = zapis"
;AT 0,21;"SKOK";AT 1,1;"C = nowy
 cykl",AT 1,21;"REKORD ";rek-1
 160 PRINT AT 21,1;"R = dem. rek
ordu";AT 21.21;"CYKL   ";cykl
 170 INK 2. PLOT 12,20: DRAW 231
,0: DRAW 0,152: DRAW -231,0: DRA
W 0,-152: INK 0
 180 REM poczatek skoku
 190 IF INKEY$="r" THEN GO SUB 0
500
 200 IF INKEY$="z" THEN GO TO 05
40
 210 IF INKEY$="c" THEN GO TO 03
60
 220 IF w2<2 OR w2>28 OR w1<2 OR
 w1>18 OR ATTR (w1,w2)=56 THEN B
EEP .01,26  LET w1=stw1. LET w2=
stw2: LET powt=powt+1
 230 IF powt>36 THEN BEEP .5.2:
BEEP .25,-10  GO TO 0370
 240 IF ATTR (w1,w2)=48 THEN LET
 powt=0
 PRINT INK 0; PAPER 7; INVERSE (
(stw1+stw2)<>4*INT ((stw1+stw2)/
4));AT stw1-1,stw2;a$;AT stw1,st
w2,b$
 PRINT INK 2, PAPER 7, BRIGHT 1;
 FLASH 1;AT w1-1,w2;a$;AT w1,w2;
b$: POKE pamiec+2*skok,w1: POKE
pamiec+2*skok+1,w2
 PRINT #0,AT 0.28. BRIGHT (skok>
(rek-1)), PAPER 6+(skok>(rek-1))
,skok  LET stw1=w1. LET stw2=w2.
 LET skok=skok+1
 250 REM losowanie kierunku skoku
 260 LET st=1+INT (RND*8)
 270 IF st=1 OR st=8 THEN LET w1
=w1-4
 280 IF st=2 OR st=7 THEN LET w1
=w1-2
 290 IF st=3 OR st=6 THEN LET w1
=w1+2
 300 IF st=4 OR st=5 THEN LET w1
=w1+4
 310 IF st=6 OR st=7 THEN LET w2
=w2-4
 320 IF st=5 OR st=8 THEN LET w2
=w2-2
 330 IF st=1 OR st=4 THEN LET w2
=w2+2
 340 IF st=2 OR st=3 THEN LET w2
=w2+4
 350 GO TO 0180
 360 REM nowy rekord
 370 IF skok>rek THEN FOR n=38 T
O -2 STEP -3: BEEP .02,n: NEXT n
: PRINT AT 20,1, FLASH 1; BRIGHT
 1, PAPER 7,"           NOWY REKOR
D!         " LET rek=skok  GO S
UB 0490  GO SUB 0390
 380 LET cykl=cykl+1: GO TO 0130
 390 REM wzorzec rekordowy
 400 CLS : PLOT 12,20  DRAW 231,
0: DRAW 0,152: DRAW -231,0: DRAW
 0,-152
 410 PRINT AT 20,1, FLASH 1; BRI
GHT 1; PAPER 7;"          NOWY RE
KORD!   "
 420 PLOT 136,94: FOR n=1 TO rek
-1: PRINT #0;AT 1,15;n
 430 LET p2=8+8*PEEK (rekord+2*n
+1): LET p1=175-8*PEEK (rekord+2
*n)
 440 DRAW p2-PEEK 23677,p1-PEEK
23678  DRAW 1,1  DRAW -2,0: DRAW
 0,-2. DRAW 2,0: DRAW 0,1  DRAW
-1,0: NEXT n
 450 PAUSE 150. PRINT #0,AT 1,15
;"   ": FOR n=1 TO rek-1: PRINT
#0,AT 1,15,n
 460 LET p1=PEEK (rekord+2*n): L
ET p2=PEEK (rekord+2*n+1)
 470 PRINT INK 0; PAPER 7. BRIGH
T 1;AT p1-1,p2;a$;AT p1,p2;b$. B
EEP .02,26: PAUSE 2: BEEP .02,21
:      PRINT INK 2, PAPER 7, BRIG
HT 1; INVERSE ((p1+p2<>4*INT ((p
1+p2)/4))),AT p1-1,p2,a$,AT p1,p
2;b$: NEXT n: PAUSE 150: RETURN
 480 REM przeniesienie rekordu
 490 FOR n=1 TO 2*rek  POKE (rek
ord+n),PEEK (pamiec+n)  POKE (re
kord+n+1),PEEK (pamiec+n+1): NEX
T n: RETURN
 500 REM demonstracja rekordu
 510 GO SUB 0400
 520 GO TO 0130
 530 REM zapis na tasme
 540 POKE rekord,rek: POKE rekor
d+1,cykl
 550 CLS : PRINT '"Save:  SKOC
ZEK S"  BEEP .1,1  SAVE "SKOCZEK
 5" LINE 0570. BEEP .1,5: SAVE "
 SKOCZEK"CODE rekord,2*rek: BEE
P .1,8: PRINT '"Verify: SKOCZEK
 5": BEEP .1,13: VERIFY "": VERI
FY ""CODE : RESTORE : CLS : GO T
O 0130
 560 REM start
 570 PAPER 6: INK 0: BORDER 6: C
LS : PAPER 6.     PRINT ''''"
 Program ""SKOCZEK SAMOUCZEK""
    sam koryguje wlasne bledy
       i zachowuje w pamieci
       najlepsze osiagniecia."''''"
   Zapis programu i wynikow na
        tasme - klawisz Z."''''"
 Demonstracja rekordowego
     cyklu - klawisz R."
 580 PRINT PAPER 7, BRIGHT 1, FL
ASH 1;''''"Czekaj - trwa ladowan
ie danych !"
 590 INK 6: LOAD " SKOCZEK"CODE
: BEEP 1,-10: INK 0: CLS
 600 FOR n=USR "a" TO USR "a"+31
: READ bt: POKE n,bt  NEXT n  RE
STORE
 610 DATA 0,1,3,7,15,31,23,0,64,
224,112,248,240,248,240,240,1,3,
1,7,3,15,31,0,224,224,128,224,19
2,240,248,0
 620 LET rek=PEEK 60300:: LET cy
kl=PEEK 60301: LET skok=1: LET s
t=1: LET powt=0: LET w1=11: LET
stw1=11: LET w2=16: LET stw2=16
 630 BORDER 6: PAPER 6: CLS : BO
RDER 6: PAPER 6
 640 LET a$=CHR$ 144+CHR$ 145: L
ET b$=CHR$ 146+CHR$ 147
 650 GO TO 0140
```

(J. Rubinowicz)

PROGRAM 10 AMSTRAD

```
10 '      ****************************
20 ''   *      Wielkanocne Jajka       *
30 '    *            autor             *
40 '    *        R.A.Waddilove         *
50 '    *         Na podstawie         *
60 '    * Computing With The Amstrad   *
70 '    *          opracowal           *
80 '    *     Cezary Dobrowolski       *
90        ****************************
100 MODE 1:MEMORY &8223
110 GOSUB 1170:  inicjalizacja
120 GOSUB 530:' instrukcja
130 MODE 0
140 WHILE 1
150 GOSUB 750:REM set up
160 CALL &8364:REM main loop
170 IF (PEEK(&8320) AND 128) THEN GOSUB
970:GOSUB 200 ELSE GOSUB 420
180 WEND
190 END
200 '_________Zakonczenie gry_________
210 FOR i%=0 TO 7000:NEXT
220 MODE 1
230 CALL &BC02:CALL &BB4E:REM reset vdu
240 INK 0,0:BORDER 0
250 PAPER 3:PEN 1
260 LOCATE 10,1
270 PRINT " Jajka Wielkanocne"
280 PAPER 0
290 score=10*(PEEK(start) AND &F0)/&10+(
PEEK(start) AND &F)
300 IF best<score THEN best=score
310 LOCATE 3,5
320 PRINT "Skauci zdolali zlapalc ";scor
e;"jajek"
330 PEN 2:LOCATE 8,14
340 IF score=best THEN PRINT "Jestes mis
trzem!" ELSE PRINT "Rekordzista zbobyl
best
350 PEN 3:LOCATE 6,15
360 PRINT "Nacisnij klawisz :"
370 WHILE INKEY$<>"":WEND
380 WHILE INKEY$="":WEND
390 POKE start,0:level=9
400 MODE 0
410 RETURN
420 '_________Ladowanie_____________
430 PEN 10:LOCATE 4,10
440 PRINT "B R A W O !"
450 SOUND 129,239,160,4
460 FOR i%=0 TO 1000:NEXT
470 SOUND 130,95,105,4
480 FOR i%=0 TO 1000:NEXT
490 SOUND 132,40,50,4
500 FOR i%=0 TO 2000:NEXT
510 CLS:level=level+2
520 RETURN
530  __________Instrukcja_____________
540 CLS
550 INK 0,6:INK 1,6:INK 3,25
560 BORDER 3
570 PEN 1:LOCATE 1,25
580 PRINT "Wielkanoc"
590 FOR i%=0 TO 144 STEP 2
600 FOR j%=0 TO 16 STEP 2
610 IF TEST(i%,j%) THEN PLOT 155+i%*2,35
5+j%*2,2:PLOT 155+i%*2,357+j%*2:PLOT 157
+i%*2,355+j%*2:PLOT 157+i%*2,357+j%*2
620 NEXT
630 NEXT
640 LOCATE 1,25:PRINT SPACE$(15);
650 INK 1,21
660 PEN 3:LOCATE 1,6:PRINT "  Jestes naj
lepszym pilotem w klubie,":PRINT"    prz
ygotowujesz Swiateczny pokaz"
670 PRINT:PRINT "     Twoim zadaniem jes
t rzucanie":PRINT"Swiatecznych jajek na
piramide Skautow,","        utworzona na
pasie starto
wym"
680 PRINT:PRINT TAB(8);"Jesli Skaut zlap
ie jajko",TAB(11);"natychmiast ucieka,";
PRINT"  Kiedy juz wszytstcy beda mieli ja
jka,":PRINT
TAB(9);"pas bedzie pusty.":PRINT TAB(8);
"Bedziesz mogl wyladowac"
690 PEN 1:PRINT:PRINT TAB(3)"Rzucasz jaj
ka naciskajac klawisz spacji.":LOCATE 7,
25:PRINT "     Programy I K S"
700 PEN 2:LOCATE 7,22:PRINT CHR$(24);" N
acisnij jakikolwiek klawisz."
710 MOVE 0,0:DRAW 0,398,1:DRAW 638,398:D
RAW 638,0:DRAW 0,0:MOVE 0,345:DRAW 640,3
45:MOVE 0,18:DRAW 640,18
720 WHILE INKEY$<>"":WEND
730 WHILE INKEY$="":WEND
740 RETURN
750 '_________Poczatek_______________
760 BORDER 1
770 DATA 2,6,1,3,16,3,24,15,12,26,25,0,0
,0,0,0
780 RESTORE 770
790 FOR i%=0 TO 15
800 READ j%:INK i%,j%
810 NEXT
820 PAPER 8:LOCATE 1,25
830 PRINT SPACE$(20);
840 PAPER 0
850 PEN 9:LOCATE 7,1
860 PRINT "Trafien:"
870 CALL &834F,&C050,pdata,&D0C:'samolot
880 address=&C6F8
890 FOR y%=level TO 1 STEP -2
900 FOR x%=9-level TO 8-level+y%
910 CALL &834F,address+x%*4,mdata,&1004:
'rysunek skauta
920 NEXT
930 address=address-&A0+4
940 NEXT
```

```
950 FOR i%=0 TO 1000:NEXT
960 RETURN
970 '____________Zderzenie______________
980 SOUND 130,0,0,0
990 FOR i%=1 TO 10
1000 LOCATE 1,1:PRINT CHR$(11)
1010 INK 1,INT(RND*27)
1020 SOUND 4,RND*1000,5,6,0,0,1
1030 FOR j%=1 TO 100:NEXT
1040 LOCATE 1,25:PRINT CHR$(10)
1050 SOUND 4,RND*1000,5,6,0,0,1
1060 FOR j%=1 TO 100:NEXT
1070 NEXT
1080 INK 1,6
1090 PAPER 8:LOCATE 1,25
1100 PRINT SPACE$(20);
1110 PAPER 0
1120 SOUND 1,50,300,5,0,1
1130 FOR i%=0 TO 3000:NEXT
1140 PEN 10:LOCATE 7,3
1150 PRINT "Oh nie!":PRINT:PRINT TAB(4)"
Uderzyles w":PRINT:PRINT TAB(6)"skauta '
"
1160 RETURN
1170 '__________inicjalizacja___________
1180 PRINT "Mysle..."
1190 pdata=HIMEM+1:mdata=pdata+156
1200 edata=HIMEM++221:start=HIMEM+250
1210 ENT 1,150,1,2
1220 ENT -2,10,5,1,10,-5,1
1230 ENT 3,200,1,5
1240 ENT 4,50,-4,1
1250 RESTORE 1430
1260 checksum=0
1270 FOR i%=1 TO 236
1280 READ j%:POKE (HIMEM+i%),j%
1290 checksum=checksum+j%
1300 NEXT
1310 IF checksum<>21768 THEN PRINT "Erro
r in character data, please check.":END
1320 RESTORE 1510
1330 checksum=0
1340 FOR i=&831D TO &8508
1350 READ a$:POKE i,VAL("&"+a$)
1360 checksum=checksum+VAL("&"+a$)
1370 NEXT
1380 IF checksum<>42197 THEN PRINT "Blad
 w kodzie maszynowym":PRINT "Sprawdz blo
k danych.":END
1390 POKE start,0
1400 level=9:best=0
1410 RETURN
1420 '__________Samolot_________________
1430 DATA 192,0,0,0,0,0,240,160,0,0,0,0,
192,128,0,0,0,0,240,8,0,0,0,0,192,128,0,
0,0,0,176,48,0,0,0,0,192,192,0,0,0,0,48,
32,0,0,0,0,1
92
1440 DATA 192,0,0,0,0,16,0,64,0,0,0,192,
192,128,0,0,64,204,136,64,128,0,4,132,12
,72,192,192,192,204,204,192,192,128,4,19
2,192,192,19
2,12,12,12,12
1450 DATA 12,192,192,4,192,192,192,192,1
2,12,12,12,12,72,192,12,192,192,192,192,
192,192,192,192,192,192,192,4,192,192,19
2,192,192,19
2,192,192,192,192,128,4,8,0,0,0,0,0,0,4,
0,0,0,4,12,0,0,0,0,0,0,12,8,0,0,0
1460  ____________Skaut_________________
1470 DATA 32,0,0,32,136,240,160,136,216,
24,88,136,152,48,48,136,136,36,32,136,13
6,16,0,136,204,196,196,156,204,200,204,1
36,68,200,20
4,0,68,200,204,0,68,200,204,0,68,136,204
,0,68,136,204,0,68,136,204,0,68,136,204,
0,192,128,192,128
1480  ____________Jajko_________________
1490 DATA 0,188,168,20,188,188,252,252,2
52,20,188,188,0,188,168,136
1500 '__________kod maszynowy___________
1510 DATA 00,00,00,00,00,00,00,00
1520 DATA 00,00,00,00,00,00,01,00
1530 DATA 02,E8,03,00,04,32,00,02
1540 DATA 00,03,0A,00,00,04,E8,03
1550 DATA 82,00,04,2C,01,00,05,32
1560 DATA 00,82,00,00,D0,07,00,05
1570 DATA 0A,00,DD,4E,00,DD,46,01
1580 DATA DD,5E,02,DD,56,03,DD,6E
1590 DATA 04,DD,66,05,C3,E6,84,CD
1600 DATA 06,B9,F5,21,50,C0,22,1E
1610 DATA 83,21,00,00,22,21,83,3E
1620 DATA 01,32,20,83,CD,63,84,11
1630 DATA 00,00,21,00,00,CD,10,BD
1640 DATA CD,A6,83,CD,0D,BD,01,1B
1650 DATA 00,AF,ED,42,38,F5,3A,20
1660 DATA 83,E6,C0,20,07,3E,42,CD
1670 DATA 1E,BB,28,D8,F1,CD,0C,B9
1680 DATA C9,2A,21,83,7C,B5,20,28
1690 DATA 3A,20,83,EE,01,32,20,83
1700 DATA C8,3E,2F,CD,1E,BB,C8,2A
1710 DATA 1E,83,01,54,38,09,22,21
1720 DATA 83,11,00,83,01,03,05,CD
1730 DATA E6,84,21,34,83,C3,AA,BC
1740 DATA 22,23,83,01,03,38,09,30
1750 DATA 04,01,50,C0,09,3A,C0,82
1760 DATA BE,20,06,2B,2B,2B,BE,28
1770 DATA 2A,3E,03,BE,28,1D,2A,23
1780 DATA 83,01,00,10,09,30,04,01
1790 DATA 50,C0,09,22,25,83,22,21
1800 DATA 83,11,00,83,01,05,03,CD
1810 DATA A3,84,C9,21,46,83,CD,AA
1820 DATA BC,18,38,E5,21,3D,83,CD
1830 DATA AA,BC,21,01,0C,CD,75,BB
1840 DATA 3A,1D,83,C6,01,27,32,1D
1850 DATA 83,F5,E6,F0,CB,3F,CB,3F
1860 DATA CB,3F,CB,3F,C6,30,CD,5D
1870 DATA BB,F1,E6,0F,C6,30,CD,5D
1880 DATA BB,E1,11,C0,82,01,04,10
1890 DATA CD,E6,84,11,00,83,01,03
1900 DATA 05,2A,21,83,CD,E6,84,21
1910 DATA 00,00,22,21,83,C9,2A,1E
1920 DATA 83,01,F0,C6,AF,ED,42,38
1930 DATA 08,3A,20,83,F6,40,32,20
1940 DATA 83,21,28,83,CD,AA,BC,2A
1950 DATA 1E,83,22,23,83,23,22,25
1960 DATA 83,22,1E,83,11,24,82,01
1970 DATA 0D,0C,CD,A3,84,2A,1E,83
1980 DATA 01,5B,20,09,3E,00,BE,C8
1990 DATA 21,20,83,CB,FE,C9,ED,43
2000 DATA 29,83,ED,53,27,83,0E,02
2010 DATA CD,19,BD,F3,ED,5B,27,83
2020 DATA 3A,29,83,47,C5,3A,2A,83
2030 DATA 47,2A,23,83,1A,AE,77,23
2040 DATA 13,10,F9,2A,23,83,01,00
2050 DATA 08,09,30,04,01,50,C0,09
2060 DATA 22,23,83,C1,10,DE,2A,25
2070 DATA 83,22,23,83,0D,20,CD,FB
2080 DATA C9,22,23,83,C5,41,2A,23
2090 DATA 83,1A,AE,77,23,13,10,F9
2100 DATA 2A,23,83,01,00,08,09,30
2110 DATA 04,01,50,C0,09,22,23,83
2120 DATA C1,10,E1,C9,00,00,00,00
```

Podprogramy przesuwające zawartość ekranu

(MERITUM)

Działają niezależnie, a parametrami dla każdego z nich jest numer wiersza lub kolumny, od której ma być przesunięty tekst lub grafika (adres komórki odpowiadającej temu parametrowi jest o 4 większy od adresu startowego) oraz liczba wierszy lub kolumn do przesunięcia (adres o 6 większy od adresu startowego). Adresy startowe przedstawione są na listingu podprogramów. Instrukcje z linii 8175 modyfikują sposób użycia funkcji USR (0). Po ich wykonaniu podprogramy możemy uruchamiać przez bezpośrednie deklarowanie adresu (np. U = USR (30000) uruchamia podprogram od adresu 30000).

Piotr BIEDRZYCKI

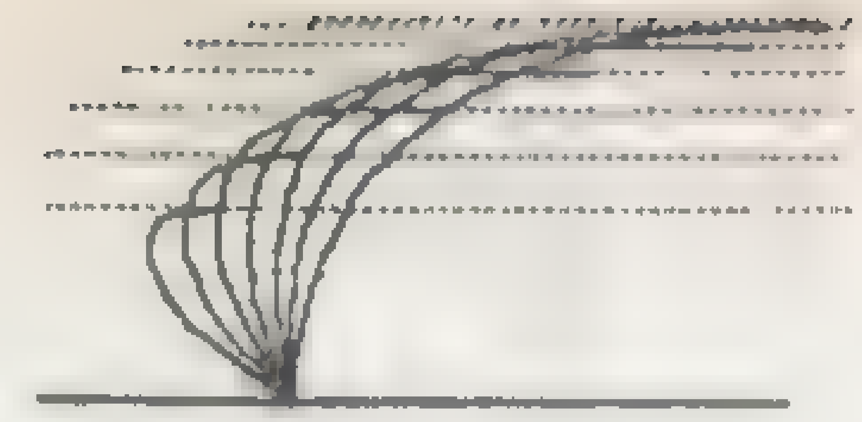
```
8175 A=16448:POKEA,205:POKEA+1,127:
     POKE A+2,10:POKE A+3,233:
     POKE 16526,64 :POKE 16527,64
8200 'skroll do gory U=USR(30000)
8205  FOR A=30000 TO 30062
8210    READ K:POKE A,K
8215 NEXT    ' 30004-O   30006-I
8220 DATA  229,213,197,30,32,62,32,
     254,0,40,48,30,84,117,131,254,
    65,242,107,117,22,60,62,64,131,
    95,198,64,111,98,6,0,62,14,245,
    14,32,237,176,58,54,117,254,64,
     40,9,79,62,64,145,71,35,19,16,
  252,241,61,32,231,193,209,225,201
8225 'skroll do dolu U=USR(30070)
8230  FOR A=30070 TO 30131
8235     READ K:POKE A,K
8240  NEXT  ' 30074-O  30076-I
8245 DATA   229,213,197,30,2,62,32,
    254,0,40,47,30,153,117,131,254,
65,242,176,117,22,63,198,127,95,98,
214,64,111,6,0,62,14,245,14,32,237,
     184,58,124,117,254,64,40,9,79,
    62,64,145,71,43,27,16,252,241,
          61,32,231,193,209,225,201
8250 'skroll w prawo U=USR(30140)
8255  FOR A=30140 TO 30189
8260    READ K:POKE A,K
8265  NEXT  ' 30144-O  30146-I
8270 DATA   229,213,197,30,0,62,15,
     254,0,40,35,87,131,254,17,242,
    234,117,33,0,60,1,64,0,9,61,32,
   252,122,43,43,229,209,43,1,62,0,
     237,184,43,43,27,27,61,32,244,
                    193,209,225,201
8275 'skroll w lewo U=USR(30190)
8280  FOR A=30190 TO 30240
8285    READ K:POKE A,K
8290  NEXT  ' 30194-O  30196-I
8295 DATA   229,213,197,30,0,62,15,
     254,0,40,36,87,131,254,17,242,
    29,118,123,60,33,192,59,1,64,0,
   9,61,32,252,122,35,229,209,35,1,
    62,0,237,176,35,35,19,19,61,32,
                244,193,209,225,201

9000 CLS:PRINT"   PROBA DZIALANIA
A-ADRES
O-OD KTOREJ KOLUMNY/LINI
I-ILE KOLUMN/LINI"
9045 CLEAR 70 :FOR S=0 TO 13:
     PRINT@ S*64+32,"X";:NEXT S
9050 PRINT @512,STRING$(64,"X");
9055 PRINT@832,"":
     INPUT"A,O,I";A,O,I
9060 IF A<>30000 AND A<>30070
     AND A<>30140 AND A<> 30190
     THEN GOTO 9055
9065 POKE A+4,O:POKE A+6,I:
     U=USR(A)
9070 IF INKEY$="" THEN 9070
9999 GOTO 9000
```

SPECTRUM

```
   10 DIM x(60): DIM y(60)
   20 FOR n=1 TO 11: LET x(n)=0:
LET y(n)=16*(n-1): NEXT n
   30 FOR n=12 TO 27: LET x(n)=16
*(n-12): LET y(n)=175: NEXT n
   40 FOR n=28 TO 38: LET x(n)=25
5: LET y(n)=16*(39-n)-1: NEXT n
   50 FOR n=39 TO 54: LET x(n)=16
*(55-n)-1: LET y(n)=0: NEXT n
   55 FOR n=55 TO 60: LET x(n)=x(
n-54): LET y(n)=y(n-54): NEXT n
   59 BORDER 6: PRINT "";: CLS : B
ORDER 6
   60 FOR a=1 TO 5: FOR n=1 TO 55
: FOR m=n TO n+a: PLOT x(n),y(n)
: DRAW (x(m)-x(n)),(y(m)-y(n)),
2.5: NEXT m: BEEP .02,n/2: NEXT
n: PAUSE 100: CLS : NEXT a
   70 CLS : GO TO 60
   90 SAVE "5 RAMEK" LINE 1: VERI
FY "": GO TO 1
```

Grafika (J. Rubinowicz)



POGOŃ

Zjawisko *pogoni* jest tak stare, jak życie na Ziemi. Ma ono swoją teorię (cynicy twierdzą, iż ścigająca samolot rakieta manewruje dokładnie tak samo, jak... wiejski kundel goniący rowerzystę).

Załóżmy, że cel został zlokalizowany i w chwili „zero" w jego kierunku odpalono rakietę. Rysunek przedstawia kolejne „sekundy" *pogoni*.

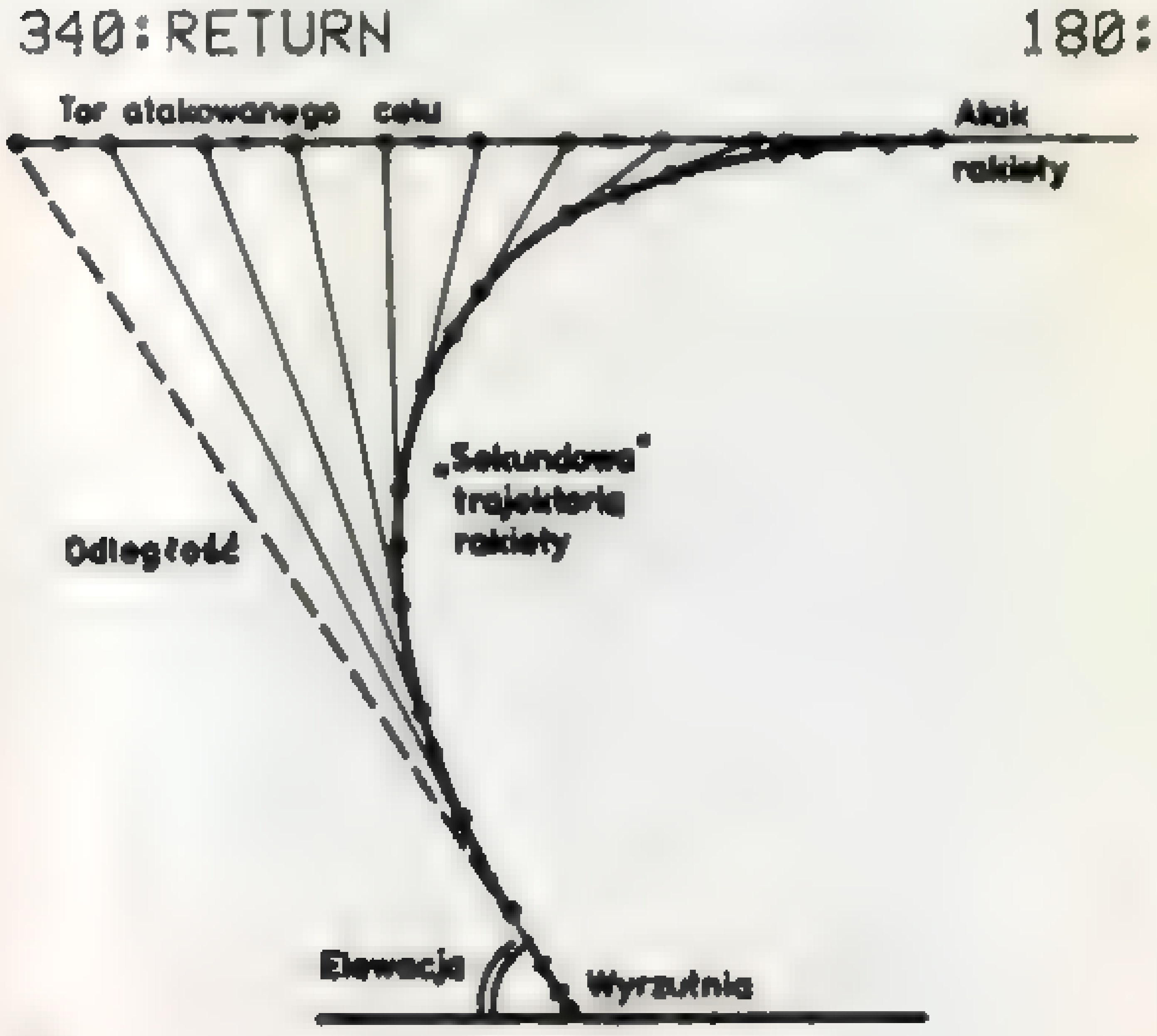
Rakieta, wyposażona w mechanizm samonaprowadzania tak koryguje swój lot, aby cel był „na osi" rakiety (... kundel tak biegnie, aby rowerzysta był „przed nosem"). W konsekwencji otrzymujemy misterną *trajektorię pogoni*. Jeśli rakieta jest szybsza i jeśli ma dosyć paliwa, to nieuchronnie doścignie cel.

Zagadnienie pogoni ma bogatą literaturę. W anglojęzycznych podręcznikach matematyki ma ono swoją (komiczną) nazwę BANG-BANG PROBLEM.

Program wywołuje etykieta „A" (RUN 70). Zmodelowany „radar namierzający" ma zasięg (odległość) około 20 km, zaś elewacja mieści się w granicach od 30 do 90 stopni.

Wyraz „mnożnik" ma znaczenie pomocnicze: dla SHARP 1500-PC wystarcza M = 10.

```
190:X1=X2:Y1=Y2:
    GOSUB 10
200:X2=M*X:Y2=M*Y:
    LINE -(X1,Y1)-
    (X2,Y2),0,3
210:IF ABS X2>140
    LET Z=1:PRINT
    "UCIECZKA":
    GOTO 230
220:IF Q>UGOTO 190
230:LINE -(X2,Y2)-
    (0,0),9:TEXT :
    IF Z=1THEN END
240:PRINT "PUNKT T
    RAFIENIA"
250:X=INT (X2*1E3)
    /(M*1E3):PRINT
    "X=";STR$ X;"
    KM"
260:Y=INT (Y2*1E3)
    /(M*1E3):PRINT
    "Y=";STR$ Y;"
    KM"
270:PRINT "CZAS PO
    GONI T=";STR$
    T;" SEK":LF 1:
    END
280:PRINT "DANE WY
    JSCIOWE"
290:PRINT "ODLEGLO
    SC";R;" KM"
300:PRINT "ELEWACJ
    A";E;" DEG"
310:PRINT "P R E D
     K O S C"
320:PRINT "CELU";P
    ;" KM/S"
330:PRINT "RAKIETY
    ";U;" KM/S"
340:RETURN
```




```
 10:T=T+1:U=U+P
 20:Q=(U-X)*(U-X)+
    (H-Y)*(H-Y):Q=
    √Q
 30:IF Q>0LET F=
    ASN ((U-X)/Q)
 40:X=X+(U*SIN F):
    Y=Y+(U*COS F)
 50:RETURN
 60:BEEP 3,100,300
 70:"A":CLEAR :
    DEGREE
 80:INPUT "ODLEGLO
    SC (KM) ";R:R=
    ABS R:IF R=0
    GOTO 60
 90:INPUT "ELEWACJ
    A (DEG) ";E:IF
    E<30OR E>90
    GOTO 60
100:L=R*COS E:H=R*
    SIN E:IF L>10
    OR H>10GOTO 60
110:INPUT "U(KM/S)
     CELU ";P:P=
    ABS P:IF P=0
    GOTO 60
120:INPUT "U(KM/S)
     RAKIETY ";U:U
    =ABS U:IF U<=P
    GOTO 60
130:U=-L:INPUT "MN
    OZNIK M=";M:M=
    ABS M:GOSUB 28
    0
140:GRAPH :LINE -(
    0,0)-(200,0),0
    ,2
150:LINE -(200,0)-
    (70,0),9:SORGN
160:HM=H*M:LM=-L*M
    :LINE -(0,0)-(
    LM,HM),9
170:LINE -(LM,HM)-
    (150,HM),1,1
180:LINE -(100,HM)
    -(0,0),9
```



Przykłady

```
DANE WYJSCIOWE
ODLEGLOSC 10 KM
ELEWACJA 30 DEG
P R E D K O S C
CELU 0.4 KM/S
RAKIETY 0.5 KM/S
PUNKT TRAFIENIA
X=-2.1817 KM
Y=5.0004 KM
CZAS POGONI T=16 S
EK

DANE WYJSCIOWE
ODLEGLOSC 10 KM
ELEWACJA 45 DEG
P R E D K O S C
CELU 0.4 KM/S
RAKIETY 0.5 KM/S
PUNKT TRAFIENIA
X=2.1966 KM
Y=7.071 KM
CZAS POGONI T=23 S
EK

DANE WYJSCIOWE
ODLEGLOSC 10 KM
ELEWACJA 90 DEG
P R E D K O S C
CELU 0.4 KM/S
RAKIETY 0.5 KM/S
UCIECZKA
```

JANUSZ MILLER
SHARP PC-1500
P O G O N

KOMPUTER W SZKOLE

Jak uczyć programowania? Jak dotąd brak jednoznacznej koncepcji. Powstał co prawda *„Program powszechnej edukacji w zakresie wiedzy informatycznej i zastosowania techniki komputerowej w procesach kształcenia w latach 1986—1990"* (tak brzmi pełna jego nazwa), ale jest to dopiero początek drogi. Cóż, mamy dopiero drugi rok okresu, który program obejmuje i choć jednoznacznie formułuje on cele informatycznej edukacji w szkołach wszystkich typów, określa szczegółowe założenia i przedstawia propozycje konkretnych działań — pozostaje jedynie suchym zapisem.

Nie ma bowiem armat; nie ma nadal komputerów, które można by wykorzystywać powszechnie. Program powstał na fali szybko rosnącej liczby sprzętu mikro w kraju. Sprzętu, owszem, przybywa i to coraz lepszego, natomiast w szkołach jest go niewiele — w Polsce może kilka tysięcy. Dopiero przyszłość przynieść ma elwrowskiego *Juniora*, tymczasem uczenie pływania na sucho nie zawsze ma sens. Trudno liczyć na to, że nauka programowania i ta cała informatyczna teoria (sięgająca zastosowań informatyki) dadzą jakikolwiek efekt.

W szkolnej edukacji informatycznej czekają nas dwa problemy. Po pierwsze komputer jako urządzenie, które trzeba nauczyć się wykorzystywać i poznać. Po drugie komputer jako zwykła szkolna pomoc, która ma służyć nauczycielowi, podobnie jak dziś magnetofon czy rzutnik. A na to trzeba poczekać. Powtórzmy jeszcze raz: brakuje sprzętu, nie ma programów edukacyjnych, nie ma jeszcze licznej dobrej kadry i to zarówno tej, która ma uczyć informatyki, jak i tej, która ma się nią posługiwać.

Stan ten już wkrótce doprowadzić może do sytuacji, w której komputerowcy staną się elitą wśród swoich rówieśników, wyróżniających się nie tylko posiadaniem komputera, ale i wiedzą, którą dzięki niemu zdobyli. A przecież nie o to chodzi.

Sieć ZX Spectrum przeznaczona dla nauczania programowania — to propozycja rzemiosła (Zetatronik — Warszawa) wykonana jest na podstawie zmodyfikowanej wersji systemu dyskowego ZEDOS-2 i składa się ze stanowiska nauczyciela oraz ze stanowisk uczniów (do 16). Każde ze stanowisk wyposażone jest oczywiście w mikrokomputer oraz monitor. Każdy z mikrokomputerów pracujących w sieci dysponuje pełną przestrzenią adresową pamięci (RAM) co oznacza, że oprogramowanie systemowe sieci nie zajmuje miejsca w pamięci RAM dostępnej dla użytkownika.

Stanowiska uczniowskie (równouprawnione względem siebie) są wyposażone dodatkowo w interfejsy służące do obsługi systemu dyskowego i do komunikacji ze stanowiskiem nauczycielskim. Każde z nich pozwala na korzystanie ze wszystkich komend systemu dyskowego ZEDOS-2 i ma dostęp do dysku umieszczonego w stacji nr 1 stanowiska nauczyciela. Uczniowie mają dostęp tylko do zbiorów przez siebie zapisanych. Jedynie nauczyciel może sięgać do wszystkich zbiorów.







LIGA MYŚLĄCYCH

Rozpoczynamy kolejną rundę Ligi Myślących. Jak zwykle w każdym numerze czeka na Was kilka zadań. Redakcja czeka na odpowiedzi, a laury na najlepszych.

Zadanie 1

Spośród 80 uczniów klas ósmych pewnej szkoły, 34 bierze udział w konkursie matematycznym, 24 uczniów w konkursie fizycznym, 16 uczniów w konkursie chemicznym, w tym 8 uczniów w konkursie matematycznym i konkursie fizycznym, 7 uczniów w konkursie matematycznym i konkursie chemicznym, 6 uczniów w konkursie fizycznym i konkursie chemicznym, 5 uczniów w konkursach: matematycznym, chemicznym i fizycznym.

Proszę podać, ilu uczniów bierze udział tylko w jednym, a ilu nie bierze udziału w żadnym z podanych konkursów?

Zadanie 2

Niech A będzie zbiorem dzielników naturalnych liczby 18, B zbiorem liczb pierwszych mniejszych od 18, C zbiorem liczb złożonych mniejszych od 15.

Proszę podać wszystkie elementy każdego z tych zbiorów oraz utworzyć sumę i część wspólną A, B i C.

Zadanie 3

W liczbie trzycyfrowej suma cyfr setek i jedności równa się 15. Jeżeli cyfry liczby poszukiwanej napiszemy w odwrotnej kolejności, to otrzymamy liczbę o 297 większą od poszukiwanej. Gdybyśmy zaś przed cyfrą setek liczby poszukiwanej dopisali cyfrę 7, wówczas nowo utworzona liczba czterocyfrowa byłaby o 1151 większa od liczby czterocyfrowej, jaką otrzymalibyśmy, dopisując na końcu liczby poszukiwanej cyfrę 8.

Proszę wyznaczyć poszukiwaną liczbę

Zadanie 4

Z miasta A w kierunku miasta B, odległego od miasta A o 300 km, wyjechał samochód osobowy, a w tym samym czasie z miasta B w kierunku A wyjechał samochód ciężarowy. Gdyby samochód osobowy wyjechał o 48 minut wcześniej niż ciężarowy, to samochody minęłyby się po upływie 2 godzin jazdy samochodu ciężarowego. Gdyby zaś samochód ciężarowy wyjechał o 1 godzinę i 20 minut wcześniej niż osobowy, to samochody minęłyby się po upływie 2 godzin jazdy samochodu osobowego. Z jaką prędkością jechał samochód osobowy, a z jaką ciężarowy?

Ile czasu upłynęło od chwili mijania się obu samochodów? Jak daleko od miasta A minęły się samochody?

Zadanie 5

Pewną liczbę kości do gry w domino rozdzielono na n stosów, po czym z pierwszego stosu przełożono na drugi $\frac{1}{n}$-tą część kości znajdujących się w pierwszym stosie, a następnie z otrzymanego wtedy stosu przełożono $\frac{1}{n}$-tą część kości na trzeci stos itd. Na koniec z n-tego stosu przełożono $\frac{1}{n}$-tą część na pierwszy stos i okazało się, że w każdym stosie było po A kości. Ile kości było początkowo w każdym stosie?

Ankieta

Minął już rok od chwili ukazania się pierwszego numeru naszego pisma. W każdym kolejnym staramy się zaspokoić informacyjne potrzeby naszych Czytelników. Znamy je z listów nadchodzących do redakcji, ze spotkań. Tym razem, podobnie jak przed rokiem, prosimy o odpowiedź na poniższą ankietę. Z pewnością Wasze odpowiedzi pomogą nam w redagowaniu „IKSa". Wśród tych, którzy nadeślą pod naszym adresem swoje uwagi rozlosujemy nagrody.

A teraz odpowiedzcie

Kto nas czyta?

1) Mam lat

2) Mieszkam w mieście, które ma
   a) więcej niż 150 tys. mieszkańców
   b) 50—150 tys. mieszkańców
   c) mniej niż 50 tys. mieszkańców
   d) mieszkam na wsi

3) Jestem
   a) uczniem szkoły podstawowej
   b) uczniem szkoły średniej
   c) studentem uczelni cywilnej
   d) studentem uczelni wojskowej
   e) pracuję

4) Posiadam lub mam dostęp do
   a) ZX Spectrum
   b) Atari
   c) Commodore (jaki?)
   d) Amstrad (jaki?)
   e) inny — jaki? ....

5) Korzystam z komputera
   a) w domu (własny)
   b) w szkole
   c) w pracy
   d) w klubie
   e) w klubie garnizonowym
   f) u kolegi

6) Czytam „IKSa"
   a) systematycznie
   b) czasami
   c) bardzo rzadko

7) Za pożyteczne uważam:		8) za niepotrzebne uważam
a)	wykłady Basica	..
b)	wykłady Pascala	....
c)	wykłady Logo	....
d)	publikacje dot. systemów mikrokomputerowych	.....
e)	programy gier	.....
f)	programy użytkowe	....
g)	programy graficzne	....
h)	informacje o nowościach tech.	....
i)	zadania Ligi Myślących	.
j)	porady techniczne	....
k)	Twoje propozycje	.....
....		.....
....		....

Nasz adres:
Redakcja „IKS"
W-wa 00-950
ul. Grzybowska 77

Pełne rozwiązania zadań prosimy przesyłać pod adresem redakcji do końca czerwca br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Pocztowa giełda

Rozwiązanie krzyżówki
z nr. 1/87 „IKS-a"

Hasło brzmi:
„IKS to litera i pismo o komputerach"

Bony pieniężne (1000 zł) wylosowali: Magdalena Rucka — Wrocław, Leszek Szydłowski — Sochaczew, Marek Tararako — Gorzów Wlkp., Krystyna Feluś — Regulice, Ryszard Ratajczyk — Wrocław.

Nagrody książkowe otrzymują: Joanna Marciniak — Warszawa, Katarzyna Kosińska — Konin, Adam Berg — Gdynia, Maria Paluchowska — Poznań, Bogusław Oleksy — Jaworzno, Stanisław Wrzos — Polkowice, Tomasz Wojtczak — Warszawa, Juliusz Kobylanka — Żory, Jolanta Guzik — Bielsko-Biała, Arkadiusz Soroka — Toruń.

W naszym komputer-landzie

Zgodnie z duchem czasu pracowity robot Spektruś postanowił przeprowadzić atestację swojego stanowiska pracy. W tym celu ustawił na stole komputer i połączył go z monitorem. Obok umieścił drukarkę. Całość podłączył do sieci. Sprawdził jeszcze raz połączenia

Przy stole postawił krzesło i usiadł na nim skupiony, przybierając właściwą pozycję, zgodną z zasadami higieny pracy umysłowej. Włączył komputer i położył palce na jego klawiszach. Poprawił się na krześle. Uniósłszy lekko głowę do góry — odetchnął głęboko. Skontrolował, czy papieru w drukarce wystarczy, a następnie, czy jego kończyny górne zginają się prawidłowo. Przeprowadził krótką gimnastykę palców, po czym próbnie przekonał się, czy trafiają one w klawisze.

Ocenił, że siedzi za daleko od komputera — przybliżył się nieco. Teraz objął wzrokiem całą klawiaturę i uznał, że oświetlenie jest niewystarczające. Zapalił dodatkową żarówkę.

Zaczął myśleć intensywnie pozostając dłuższą chwilę w stanie koncentracji. Jeszcze raz wykonał wymienione czynności. Obserwował kątem oka zegarek i śledził mijający czas.

Zbliżał się już „fajrant" gdy Spektruś mógł podsumować wyniki atestacji.

Okazało się, że największym mankamentem na jego stanowisku pracy był jaskrawo występujący brak pomysłu...

Podglądał:
Eugeniusz MLECZAK

„Nasz XXI wiek"

Pokłosie konkursu

Każdy konkurs literacki — jest czymś w rodzaju sondy, pozwalającej zorientować się, jaki jest stan świadomości autorów, uczestniczących w rywalizacji. Jest tak, bowiem nadesłane prace nie tylko odzwierciedlają wyobraźnię, talent, wiedzę i społeczne samopoczucie autorów, ale wiele mówią także o ich stosunku do życia, o łączących się z tym przeświadczeniach i oczekiwaniach.

Nasz konkurs, adresowany głównie do młodych, przyniósł ilościowo bardzo obfity plon. Sto osiem prac, nadesłanych z całej Polski — z dużych i małych miast, osiedli i wsi — obrazuje geografię naszej poczytności. Cieszy nas rozległy zasięg oddziaływania pisma, potwierdzający coraz szersze upowszechnianie się „myślącej" elektroniki i rosnące zainteresowanie tą dziedziną wśród dzieci i młodzieży.

Między pracami, nadesłanymi na konkurs — ogromną większość stanowią utwory kilkunastolatków, ale nie brakuje też opowiadań pisanych przez dorosłych. Najstarszy z autorów liczy sobie ponad 60 lat...

Proporcje wiekowe pozwalają się domyślić, iż głównym nurtem fascynacji tematycznych są źródła, płynące z oglądania filmów telewizyjnych i bajek oraz lektur łatwych w odbiorze, nie poruszających głębszych problemów moralnych i psychologicznych. Stąd też dominująca w pracach tendencja do tworzenia atrakcyjnej i dynamicznej, przygodowej fabuły. Próby takie cechują znakomitą większość nadesłanych na konkurs prac. Zaiste, przemożny jest wpływ telewizji na umysły najmłodszych!

Dlatego też lektura pokłosia konkursu — a czytaliśmy wszystkie prace z jednakową uwagą — trochę nuży. Pomijamy tu świadomie niedostatki warsztatu — a więc klarowność i logikę wywodu, umiejętność konstruowania wątków i intrygi — jako zrozumiałe, debiutanckie grzechy. Często musieliśmy też przymykać oko na kłopoty ze stylem i ortografią, choć to już nie te czasy, by porywać się na literacką ekspresję bez znajomości reguł językowych. Przecież świat komputerów też ma swoją gramatykę i ortografię!

Wyobrażenia o przyszłości — są niemal lustrzanym odbiciem stereotypowych lęków, fobii i realnie istniejących zagrożeń. Skrajny katastrofizm artykułowany jest przez opisy zagłady cywilizacji, wojen gwiezdnych, buntów komputerów itd. Przynosi to dość schematyczne i banalne wizje XXI wieku. Czarnowidztwo i ekstremalny pesymizm — tak najkrócej można by scharakteryzować wyobrażenia znakomitej większości młodych autorów o tym, co ich czeka w przyszłości. Człowiek osaczony, zniewolony przez technikę — tak widzą perspektywę nadchodzących lat ci autorzy, którzy nie wieszczą atomowej zagłady.

Takie też w większości są (bo nie mieliśmy wszak innego wyboru) i prace nagrodzone oraz wyróżnione. Przy podejmowaniu decyzji — kierowaliśmy się przede wszystkim świeżością i oryginalnością pomysłu, wartościami literackimi, przenikliwością psychologiczną i intelektualną dojrzałością prac.

Ile warte są wyróżnione w konkursie opowiadania? Najlepiej ocenią to sami Czytelnicy. Przyjemnej lektury!

Tadeusz PIEKŁO

KRZYŻÓWKA nr 4



Litery odczytane w kolejności utworzą nazwy czterech urządzeń. Rozwiązaniem krzyżówki jest odpowiedź — jakie są to urządzenia. Rozwiązania prosimy nadsyłać na kartkach pocztowych pod adresem redakcji do 31 czerwca br.

Mikrociekawostki



Standard Spectrum został utrzymany. Firma co prawda miała spore kłopoty i w konsekwencji zmieniła właściciela. Amstrad jednak zadbał o tych, którzy przyzwyczaili się do Spectrum — powstał nowy mikrokomputer ZX Spectrum + 2 o pamięci 128 kB. W porównaniu z poprzednimi wersjami nowe urządzenie ma oczywiście lepsze parametry; zrezygnowano z wielofunkcyjnej klawiatury, magnetofon zaś jest integralną częścią ZX Spectrum + 2.

Trudno wróżyć różową przyszłość temu komputerowi. Ostrą konkurencję stanowią dla niego Amstrady sprowadzane do Polski przez Polanglię Ltd. Co prawda ceny tych ostatnich wzrosły od 1 stycznia o ok. 12,5 procent i za Amstrada PCW 8512 z podwójną stacją dysków, monitorem, drukarką i edytorem tekstu płaci się 500 funtów, zaś Spectrum + 2 kosztuje jedynie 200 funtów. Warto dodać, że pełny serwis tych komputerów prowadzi Sp. z oo Refleks.

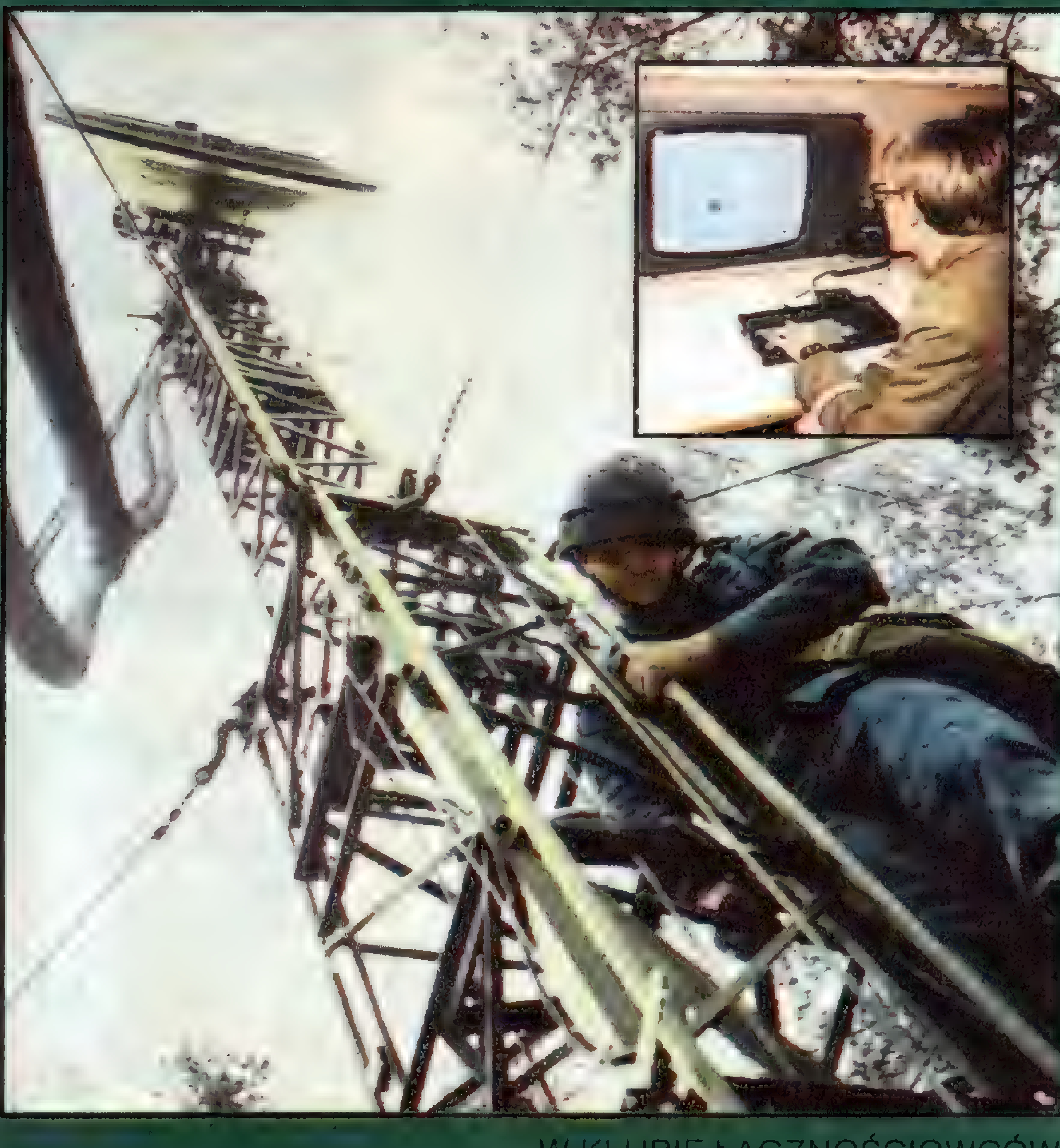


„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8709. Nr. ind. 361682. K-80

5 1987 (13)

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek Żołnierza Wolno. ISSN 0860 2794 Cena 50 zł



W KLUBIE ŁĄCZNOŚCIOWCÓW

str. 15—16

W numerze:

Czy komputer może uczyć? — str. 4

W szponach Atari:

Kreślenie okręgów — str. 5
— Supertekst — str. 7
— Reversi — str. 9

Sharp:
Kto winien? — str. 10

Mikrokomputery IBM (2) — str. 12

Infosystem'87 — str. 13

CPC 464 — Kreska (3) — str. 14

W klubie, który będzie — str. 15

Spectrum:
Sztuki i sztuczki (7) — str. 18
Pascal (5) — str. 21
Basic (7) — str. 24

Myślenie po godzinach, czyli z czego żyją rzemieślnicy — str. 29

Liga Myślących — str. 30

UWAGA! PRZYPOMINAMY

Już wkrótce do kiosków Ruchu dotrze drugi numer specjalnego numeru „IKS-a" poświęconego wyłącznie programom. Wierzymy, że wśród kilkunastu propozycji nasi Czytelnicy znajdą coś ciekawego dla siebie. Zatem uwaga użytkownicy Atari, Commodorów, Spectrum i Amstradów, pytajcie o programy „IKS-a" w kioskach.

Szyfr już złamany

Znamy już szczęśliwego laureata naszego konkursu. Spośród nadesłanych do redakcji kilkunastu tysięcy prawidłowych rozwiązań pięcioletnia Monika wylosowała zwycięzcę.



Tak poznaliśmy zwycięzcę

Uwaga. Niebieska koperta: Posiadaczem komputera **ZX Spectrum** +, magnetofonu, joysticka i zestawu programów ufundowanego przez firmę ICL został

BOGDAN OLSZEWSKI z SOSNOWCA

I jeszcze dwie koperty. Radioodbiorniki „Maria" otrzymują: Bożena Silezin-Budek z Warszawy i Michał Stajszczak również z Warszawy.

A teraz jeszcze kilka słów o konkursie. Naszym zdaniem odczytanie kryptogramu nie było zadaniem prostym. Najważniejszą sprawą było oczywiście odnalezienie klucza, a potem krok po kroku należało odczytać niezrozumiały ciąg znaków, w którym znalazły się dodatkowe utrudnienia.



A to już wszyscy szczęśliwcy

Zatem najważniejszy był klucz. W kolejnych numerach staraliśmy się podać o nim bliższe informacje. Ale ten trud okazał się niepotrzebny. Już wkrótce po ukazaniu się pierwszego numeru zaczęły napływać rozwiązania. Prawidłowe?

Klucz nie sprawił większych kłopotów — pisano. I słusznie, klucz miał postać Data:1945.05.09. — jest to szczególny dzień w historii naszego narodu i świata. Kiedy znamy klucz, reszta jest formalnością. Odczytany kryptogram miał postać:

WSZYSTKIE BOJOWE OPERACJE MIAŁY JEDEN AZYMUT: *WOLNA POLSKA*

Okazało się, że tęgich głów nam nie brakuje, a liczba rozwiązań wskazuje, jak wielu doskonale daje sobie radę z matematyką.

Zwycięzcom gratulujemy.

****

************** N O W I N K I **************

****

***** STC POBIŁA REKORD ŚWIATA *****

STC - Standard Telephones and Cables wygrała kontrakt wartości 3 milionów funtów na położenie kabla światłowodowego o długości 56 mil łączącego wyspę Man (Wielka Brytania) z lądem stałym. Kabel ten pozwoli na wprowadzenie cyfrowej telekomunikacji na wyspie. Po raz pierwszy kabel światłowodowy tej długości nie będzie potrzebował po drodze wzmacniaczy co jest swoistym rekordem świata.

***** TANI SUPERKOMPUTER *****

Ceny superkomputerów rozpoczynają się od kwoty 7.5 miliona dolarów, gdy tymczasem Uniwersytet w Southampton obiecuje już niedługo taki komputer za.....750 tysięcy dolarów. Superkomputer składać się będzie z 320-stu transputerów czyli kompletnych komputerów zawartych na "kości" wielkości ...znaczka pocztowego.

***** ICL I SUN *****

Jeszcze pod koniec ubiegłego roku zostało podpisane porozumienie pomiędzy ICL - International Computers Ltd. i firmą SUN Microsystems Ltd. specjalizującą się w systemach graficznych. W ramach porozumienia nastąpi sprzedaż systemów komputerowych ICL i jako stacji graficznych systemów SUN - wymagających maszyn o dużej mocy przetwarzania. Drugą płaszczyzną porozumienia jest wzajemna współpraca w zakresie projektowania i konstruowania urządzeń graficznych nowej generacji.



Na zdjęciu pozbawiony migotania 19 calowy monitor kolorowy o rozdzielczości 1152 * 900 punktów. Może wyświetlać jednocześnie 256 kolorów lub odcieni szarości z palety ......16.7! milionów. Stacje graficzne SUN wyposażone są w 32 - bitowy mikroprocesor MOTOROLA 68020, a ich moc przetwarzania wynosi od 1.5 do 4 MIPS (milionów operacji wykonywanych w ciągu jednej sekundy). Stacje graficzne firmy SUN są obecnie uważane za najlepsze na świecie.

***** DWA NOWE PC APPLA *****

Firma Apple wylansowała w USA dwa nowe modele komputera osobistego McIntosh. Nazwane zostały MacIntosh II i MacIntosh SE. Posiadają znacznie większą moc obliczeniową niż ich poprzednicy i mogą być rozbudowywane. Oba mogą wiernie naśladować PC IBM-owskie i ich kopie-przy czym MacIntosh II proponuje bardziej zaawansowaną technologię od orginalnego IBM PC przy znacznie niższej cenie.

****

(sch)

CZY KOMPUTER MOŻE UCZYĆ?

Kilka dni temu miałem okazję porozmawiać z Robertem — absolwentem — „skomputeryzowanej uczelni". Zważywszy na bardzo ciekawą (i pouczającą) treść tej rozmowy przedstawiam ją w niewielkim skrócie.

Komputery w uczelni Roberta wykorzystywane są niemal we wszystkich dziedzinach życia szkoły: zarządzanie, administracja, prace naukowo-badawcze, wszelkiego typu banki informacji — a w szczególności informacja biblioteczna i naukowa. Jednak najciekawszą, moim zdaniem, sferą zastosowań jest nauczanie wspomagane przez komputer, tym bardziej że dotyczy ono przedmiotów humanistycznych.

Podczas nauki wspomaganej przez maszynę, uczący się ma do dyspozycji indywidualne stanowisko — miejsce odizolowane od pozostałej części sali o „wyciszających" ścianach (w kolorze pastelowym). Mieszczący się tam stolik wyposażony jest w mikrokomputer, który stanowi element sieci komputerowej uczelni. Dzięki połączeniom (sieci) z kilkoma liniami telefonicznymi — każde stanowisko ma łączność z wieloma innymi uczelniami, bibliotekami, a także z domem.

Rozmiar mikrokomputera, często nazywanego nanokomputerem, jest odwrotnie proporcjonalny do jego możliwości. Przede wszystkim, nanokomputer wyposażono w kolorowy monitor „z dostępem" do każdego z 1024 x 1024 pikseli. Oznacza to, iż każdy umowny punkt (piksel) ekranu może mieć inny kolor — zmieniany zależnie od potrzeb. W efekcie pozwala to uzyskać dobrej jakości postać tekstów oraz bardziej złożonych rycin na ekranie. Istotny udział w komforcie odbioru (czytelności) tej formy informacji ma fakt, iż obraz na ekranie „odświeżany" jest 100 razy w czasie jednej sekundy — dwukrotnie częściej niż ma to miejsce w standardach TV (50—60 Hz). Rezultatem tego jest „papierowy" obraz, którego oglądanie, nawet przez kilka godzin, nie męczy i jest zdecydowanie mniej szkodliwe dla zdrowia oczu.

Mówiąc o ekranie, nie sposób nie wspomnieć o tym, że „wskazanie komputerowi" dowolnego punktu ekranu, polega na dotknięciu tego miejsca, na przykład po prostu palcem. Czynność tę można także wykonać „tradycyjną myszką", piórem świetlnym lub tak zwanym digityzerem. Urządzenia te umożliwiają przemieszczanie na ekranie kursora (migającą plamkę).

Zatem podczas nauki geografii, a w zasadzie kontroli zdobywania wiadomości z tego przedmiotu — komputer wyświetla na ekranie mapę określonego kraju i żąda wskazania na niej miejsc usytuowania niektórych bogactw naturalnych. Jednocześnie na lewej części ekranu ukazują się małe okienka z obrazkami symbolizującymi węgiel, siarkę, miedź itp. Są to tak zwane ikony. Na przykład ikona siarki to po prostu rycina zapałek z symbolem Si. Uczący się „wybiera" dowolną ikonę przez jej dotknięcie (lub wskazanie). Potwierdzeniem, że komputer „zrozumiał" nasz wybór jest ukazanie się tej ikony w lewym górnym rogu ekranu. Wystarczy teraz wskazywać odpowiednie miejsca na mapie. Gdy znajomość geografii Polski jest przedmiotem kontroli — wskazanie wybrzeża jako miejsca bogatych pokładów węgla napotyka na zrozumiałe „zdziwienie" komputera, natomiast w poprawnie wskazanych punktach pojawiają się miniaturki ikon. Gdy węgiel został „rozdysponowany", uczący się dotyka ikonę siarki, następnie miedzi itp.

W trakcie tej pracy można oczywiście popełnić błąd, który został zauważony po pewnej chwili — jego poprawie i innym typowym czynnościom służy zestaw stałych ikon (niezależnie od przedmiotu) wyświetlanych w dolnej części ekranu. Symbolizują one takie czynności jak na przykład: kasowanie (mazanie, ścieranie) — najczęściej w postaci obrazka gumki, prośba o pomoc (ikona uniesionej do góry dłoni), polecenie zapisywania wyróżnionych informacji (związanych z treściami nauczania) do własnego (prywatnego) zbioru — dostępnego także w domu, powiększanie fragmentu ekranu (obrazek lupy, szkła powiększającego) itp.

Wydaje się, że bardzo atrakcyjną zdolnością komputera podczas zajęć z geografii jest powiększanie fragmentów obrazu aktualnie wyświetlanego na ekranie monitora. Na przykład oglądając mapę Polski prosimy o powiększenie jej centralnej części — w odpowiedzi komputer wyświetla fizyczną mapę województwa warszawskiego i części sąsiednich województw. Można żądać kolejnych powiększeń — ostatnim krokiem jest np. plan miasta Łowicza.

Regulacji jasności i kontrastu obrazu zazwyczaj dokonuje sam komputer, bowiem uprzedza chwilę, w której użytkownik sam zdecydowałby się na zmianę tych parametrów ekranu. Wykonywane jest to w bardzo przemyślny sposób — komputer mierzy dokładnie czas naszych odpowiedzi na obrazy (treści) ukazywane na ekranie. Korzystając ze znanej (maszynie) zależności między czasem reakcji na informacje (komunikaty) wyświetlane na monitorze, a skutkami wspomnianych regulacji (im większy kontrast tym szybciej rozpoznawany jest obraz — niestety kosztem zdrowia oczu i pogorszenia samopoczucia), komputer dobiera odpowiednie parametry wyświetlanego obrazu. W praktyce ustawiona jest taka jasność i kontrast ekranu by pogodzić potrzeby (szybkość oraz poprawność) i możliwości (nasze zdrowie) odczytywanych z niego treści.

Mając na uwadze indywidualne wymagania uczących się, w podobny sposób jak jasność i kontrast, komputer dobiera wielkość i proporcje stosowanych znaków i symboli. Uwzględnia to między innymi stan zdrowia naszych oczu. Na początku współpracy z użytkownikiem maszyna stopniowo zwiększa wielkość, aż do chwili, gdy dalsze powiększenie nie daje widocznego skutku (szybkość i poprawność odczytywania informacji z ekranu nie zmienia się) lub po prostu nie jest fizycznie to już możliwe. Analogicznie przebiega proces odwrotny — kontrolowane zmniejszanie znaków. Procedura ta, łącznie z podobnie wykonywaną weryfikacją proporcji znaków, realizowana jest bez przerwy. Ma to na celu uwzględnianie postępującego zmęczenia uczących się.

Wracając do meritum sprawy — nauki wspomaganej przez komputer — rozpoczyna się ona od prezentacji — uczący wymienia swoje nazwisko, pisząc je na ekranie (za pośrednictwem klawiatury) lub ostatnio po prostu (wyraźnie) na papierze, który stanowi jeden ze śledzonych przez kamerę obiektów. Jest ona sprzężona z komputerem, który odpowiednio analizuje przekazywane mu obrazy. Również informacje można przekazać do maszyny za pośrednictwem głosu — rozpoznaje ona ponad siedem tysięcy słów. Czasami niektóre z nich trzeba powtarzać, szczególnie podczas pierwszych godzin współpracy z komputerem, bowiem z czasem maszyna się „uczy" rozpoznawać nasz głos. Zatem przekazanie informacji do komputera jest czasem o wiele prostsze niż wykonanie tej samej czynności w rozmowie z kolegą — maszyna jest bardziej cierpliwa i dostarcza dodatkowe środki przekazywania informacji — pismo oraz ikony. W ogólności stanowi to efekt prac nad interfejsem użytkownika, często nazywanym „życzliwością komputera" lub „łatwy w użyciu".

Komputer wiedząc z kim ma do czynienia, sięga do odpowiednich informacji, które dotyczą użytkownika. Wśród nich jest życiorys uczącego się, dane o jego stanie zdrowia (w tym oczu), o cechach osobowości, znaczących doświadczeniach. Uzupełnieniem tych informacji są dane aktualne: ocena dotychczas zdobytej przez uczącego się wiedzy na temat będący przedmiotem, zajęć, a także aktualny stan zdrowia, który stanowi kolejny temat „rozmowy" po przedstawieniu się studenta. Mówiąc lub pisząc, można poinformować maszynę, że nasza obecność wynika tylko z obowiązku — stanowi to cenną wytyczną do doboru stosowanej przez komputer strategii nauczania — rzadziej zwraca uwagę, najczęściej chwali i nie wymusza dużej aktywności uczenia się. Jak ON to czyni — jest bardzo przebiegły, bo poza znajomością naszego życiorysu, samopoczucia, kontroluje naszą aktualną aktywność emocjonalną. Czyni to bardzo dyskretnie i bez przerwy — poprzez wspomniany precyzyjny pomiar czasów reakcji (odzwierciedlają one w dużej mierze aktywność emocjonalną człowieka) oraz gdy wyrazimy zgodę! — kontrolując szybkość przepływu krwi oraz EKG i GSR (galwaniczną reakcję skóry) — wymaga to założenia na lewy przegub całkiem eleganckiej „bransoletki". Podłączenie to, początkowo budzi mieszane uczucia studentów i przez kilka pierwszych zajęć „bransoletka" wisi „na gwoździu". Jednak przykład bardziej zaawansowanych w „bojach z komputerem" przekonuje, tym bardziej że to nie boli, a nauka przebiega o ponad 30% sprawniej (szybciej i „trwalej").

Jak to się dzieje? Komputer dysponując pomiarami z „bransoletki" szybko określa naszą optymalną aktywność emocjonalną, to znaczy taką, przy której najszybciej uczymy się. Istnieje bowiem ścisła zależność naszej aktywności i efektów uczenia się. Gdy materiał jest zbyt trudny, aktywność nasza wzrasta ponad granice poprawnego funkcjonowania tych części mózgu, które odpowiadają za przyswajanie nowej wiedzy, często zachodzi proces odwrotny — nasza aktywność maleje (po prostu nudzimy się) — materiał jest zbyt trudny. Spadek aktywności emocjonalnej towarzyszy także prezentacji zbyt łatwego materiału — mechanizm i skutek jest ten sam — nudzenie się. Maszyna „zdając sobie z tego sprawę" dobiera odpowiednią trudność materiału, by efekty nauki były jak największe. Jeśli i to nie pomaga, komputer dysponuje setkami komend o doskonale znanych mu efektach działania na zmianę stanu aktywności emocjonalnej uczących się. Są to zazwyczaj pochwały, nagany, propozycje przerwy lub jeszcze bardziej przekonujące komentarze.

Temat ten stanowił początek naszej kolejnej rozmowy z Robertem — absolwentem „skomputeryzowanej uczelni".

W. G.

Tomasz MROWIEC
Ludwik PIELA

Sposoby kreślenia okręgów

Podczas pisania programów wykorzystujących lub tworzących grafikę od czasu do czasu musimy wykreślić okrąg. Wtedy z zazdrością patrzymy na właścicieli ZX Spectrum, którzy mogą to zrobić za pomocą jednej instrukcji BASIC'a. Mamy nadzieję, że po przeczytaniu tego artykułu i wprowadzeniu zamieszczonych programów, również właściciele Atari będą mogli kreślić różnorodne okręgi różnymi metodami. Zaprezentujemy kilka procedur od najprostszej, w języku BASIC, która potrzebuje na kreślenie około 60 sekund, do procedury w języku maszynowym, która kreśli okrąg w ułamku sekundy.

Pierwszy z prezentowanych programów „punktuje" okrąg w ciągu około 60 sekund, bez względu na wielkość. Trwa do dosyć długo. Gdy musieliśmy wykreślić kilka różnych okręgów na ekranie i czekaliśmy na efekt końcowy, nasunęły się nam pewne refleksje. Okrąg jest symetryczny, dlaczego zatem nie wykorzystać tej własności? Jeżeli znamy współrzędne jednego punktu, możemy zrobić jego „odbicie" względem osi x lub osi y. To znaczy, jeśli wiemy, że punkt (x,y) leży na okręgu, to punkt (x,—y) również leży na okręgu. To samo jest prawdą dla (—x, y) i (—x,—y). Musimy zatem wykonać tylko jedną czwartą obliczeń. Ale dlaczego nie iść dalej? Okręgi są także symetryczne względem linii y=x oraz y=—x. Jeśli wiemy, że (x,y) leży na okręgu, wtedy to samo dotyczy (x,y). Teraz musimy wyznaczyć tylko jedną ósmą punktów. Opisana metoda wykorzystana jest w drugim programie.

Niestety, nawet wykonując tylko jedną ósmą obliczeń, wciąż potrzebujemy ponad 10 sekund na wykreślenie okręgu. Być może istnieje lepszy sposób? Zamiast używania funkcji trygonometrycznych sinus i cosinus wykorzystajmy następujące równanie:

$$x \cdot x + y \cdot y = r \cdot r$$

Nie jest ono zbyt przydatne, lecz po przekształceniu otrzymamy:

$$y = SQRT\ (r \cdot r - x \cdot x)$$

Zatem wszystko co musimy zrobić, to wyznaczyć y dla wartości x, zmieniających się od —r do r. Ponieważ funkcja pierwiastka kwadratowego daje tylko wartości dodatnie, musimy pamiętać o wykreśleniu ujemnych wartości pierwiastka. Program 3 jest przykładem, jak to zrobić. Metoda ta jest szybsza od wykorzystującej funkcje trygonometryczne, lecz wciąż wymaga ponad 16 sekund. Użyjmy zatem sposobu wypróbowanego w programie 2. W efekcie otrzymujemy program 4, który zużywa tylko pięć sekund na kreślenie dużych okręgów i nieco mniej dla mniejszych.

I to jest minimalny czas kreślenia okręgu, możliwy do uzyskania w języku BASIC za pomocą znanych wzorów. Dalsze sukcesy możemy uzyskać, zmieniając albo sposób podejścia, albo język programowania, albo jedno i drugie.

Zacznijmy od pierwszego. Wyobraźmy sobie ekran jako siatkę, której oczka reprezentowane są przez elementarne punkty graficzne (pixele). Każdy punkt adresowany jest za pomocą dwóch współrzędnych (x,y), które są zawsze liczbami całkowitymi. Wprawdzie możemy napisać PLOT 0.5,0.5, lecz adresy są zaokrąglane do wartości całkowitych. Dzięki temu, jeśli jesteśmy w pewnym punkcie na okręgu i próbujemy określić, gdzie będzie następny punkt, możemy przenieść się w jednym z ośmiu kierunków. Jeśli podzielimy okrąg na ćwiartki, wtedy poprawne są tylko trzy z tych kierunków. Jeżeli jednak podzielimy okrąg na osiem części, wtedy możemy wybrać tylko jeden z dwóch kierunków. Na przykład, jeśli jesteśmy na okręgu w punkcie (r,O), następnym punktem może być albo (r,1), albo (r—1,1). Ponieważ ekran nie może kreślić punktów innych niż o adresach całkowitych, istnieje mały błąd, który nie zawsze równy jest zero. Zastosujemy sposób sprawdzony w programie 2. Zajmuje to tylko trzy sekundy i nigdy nie kreślimy linii o dowolnej długości. W ten sposób działa program 5.

Zauważmy także, że uzyskujemy żądany efekt, wykorzystując tylko dodawanie i odejmowanie. Takie programy mogą być łatwo przetłumaczone na język maszynowy. Program 7 jest przykładem procedury kreślenia okręgów w języku maszynowym. Program 6 służy do wywołania tego ostatniego i zużywa mniej niż 0,2 sekundy na kreślenie okręgu. Wywołanie odbywa się za pomocą funkcji USR. Przekazywanymi parametrami są kolejno: adres kodu procedury, współrzędne środka okręgu x i y, promień okręgu i tryb kreślenia.

Poszczególne wartości trybu kreślenia oznaczają: 0 — zgaszenie punktu; 1 — zapalenie punktu; 2 — odwrócenie punktu (tj. zapalenie lub zgaszenie, w zależności od stanu, w jakim się znajduje).

Jedyną trudnością w używaniu programu w języku maszynowym jest niekontrolowanie czy okrąg nie wyszedł poza ekran.

```
KZ 100 REM KRESLENIE OKREGU
QO 110 REM
KV 120 REM PROGRAM #1
QS 130 REM
MW 140 REM CZAS KRESLENIA OKOLO 6
   0 SEK.
MF 200 DEG
ZT 210 GRAPHICS 8:COLOR 1
PD 230 SETCOLOR 2,0,0
BF 240 A=160:B=80:R=50
QZ 300 FOR ALFA=0 TO 360
JA 310 X1=INT(R*COS(ALFA)+0.5)
LJ 320 Y1=INT(R*SIN(ALFA)+0.5)
GD 330 PLOT A+X1,B+Y1
VY 340 NEXT ALFA
```

```
KZ 100 REM KRESLENIE OKREGU
QO 110 REM
LN 120 REM PROGRAM #2
QS 130 REM
HC 140 REM CZAS KRESLENIA OKOLO 1
   0 SEK.
MF 200 DEG
ZT 210 GRAPHICS 8:COLOR 1
EI 230 SETCOLOR 2,0,0 A=160
IH 250 B=80:R=50 PLOT A+R,B
CE 300 FOR ALFA=0 TO 45
JA 310 X1=INT(R*COS(ALFA)+0.5)
LJ 320 Y1=INT(R*SIN(ALFA)+0.5)
GD 330 PLOT A+X1,B+Y1
HB 340 PLOT A-X1,B+Y1
HN 350 PLOT A+X1,B-Y1
IL 360 PLOT A-X1,B-Y1
GG 370 PLOT A+Y1,B+X1
HE 380 PLOT A-Y1,B+X1
HQ 390 PLOT A+Y1,B-X1
HV 400 PLOT A-Y1,B-X1
VT 410 NEXT ALFA
```

```
KZ 100 REM KRESLENIE OKREGU
QO 110 REM
MF 120 REM PROGRAM #3
QS 130 REM
PL 140 REM CZAS KRESLENIA OKOLO 1
   7 SEK.
QW 150 REM
ZT 210 GRAPHICS 8 COLOR 1
EI 230 SETCOLOR 2,0,0 A=160
QZ 250 B=80 R=50
HT 270 X0=-R:Y0=0
SQ 300 FOR X1=-R TO R
WA 310 Y1=INT(0.5+SQR(R*R-X1*X1))
EJ 320 PLOT A+X0,B+Y0:DRAWTO A+X1
   ,B+Y1
IF 330 PLOT A+X0,B-Y0:DRAWTO A+X1
   ,B-Y1
HS 340 X0=X1:Y0=Y1
GQ 350 NEXT X1
```

```
KZ 100 REM KRESLENIE OKREGU
QO 110 REM
MX 120 REM PROGRAM #4
QS 130 REM
UT 140 REM CZAS KRESLENIA OKOLO 5
    SEK.
QW 150 REM
ZT 210 GRAPHICS 8:COLOR 1
EI 230 SETCOLOR 2,0,0:A=160
QZ 250 B=80:R=50
FP 270 X0=-R:Y0=0:X1=-R
WP 290 Y1=INT(0.5+SQR(R*R-X1*X1))
EF 300 PLOT A+X0,B+Y0:DRAWTO A+X1
   ,B+Y1
HH 310 PLOT A-X0,B+Y0:DRAWTO A-X1
   ,B+Y1
ID 320 PLOT A+X0,B-Y0:DRAWTO A+X1
   ,B-Y1
LF 330 PLOT A-X0,B-Y0:DRAWTO A-X1
   ,B-Y1
ED 340 PLOT A+Y0,B+X0:DRAWTO A+Y1
   ,B+X1
HF 350 PLOT A-Y0,B+X0:DRAWTO A-Y1
   ,B+X1
IB 360 PLOT A+Y0,B-X0 DRAWTO A+Y1
   ,B-X1
LD 370 PLOT A-Y0,B-X0:DRAWTO A-Y1
   ,B-X1
IA 380 X0=X1 Y0=Y1
YU 390 IF -X1>=Y1 THEN X1=X1+1:GO
   TO 290
```

```
KZ 100 REM KRESLENIE OKREGU
QO 110 REM
NP 120 REM PROGRAM #5
QS 130 REM
SL 140 REM CZAS KRESLENIA OKOLO 3
    SEK.
QW 150 REM
ZT 210 GRAPHICS 8:COLOR 1
PD 230 SETCOLOR 2,0,0
BF 240 A=160:B=80:R=50
YD 273 FI=0:Y1=0:X1=R
BY 300 FIY=FI+Y1+Y1+1
KA 310 FIXY=FIY-X1-X1+1
GB 320 PLOT A+X1,B+Y1
GZ 330 PLOT A-X1,B+Y1
HL 340 PLOT A+X1,B-Y1
IJ 350 PLOT A-X1,B-Y1
GE 360 PLOT A+Y1,B+X1
HC 370 PLOT A-Y1,B+X1
HO 380 PLOT A+Y1,B-X1
IM 390 PLOT A-Y1,B-X1
UL 400 FI=FIY:Y1=Y1+1
LX 420 IF ABS(FIXY)<ABS(FIY) THEN
    FI=FIXY:X1=X1-1
TP 430 IF X1>=Y1 THEN 300
```

```
KZ 100 REM KRESLENIE OKREGU
QO 110 REM
OH 120 REM PROGRAM #6
QS 130 REM
EA 140 REM CZAS KRESLENIA OKOLO
WT 150 REM 0.1833 SEK.
ZT 210 GRAPHICS 8:COLOR 1
PD 230 SETCOLOR 2,0,0
BF 240 A=160:B=80:R=50
DB 270 P=7*16*16*16
VJ 300 I=USR(P,A,B,R,1)
KA 310 FIXY=FIY-X1-X1+1
GB 320 PLOT A+X1,B+Y1
GZ 330 PLOT A-X1,B+Y1
HL 340 PLOT A+X1,B-Y1
IJ 350 PLOT A-X1,B-Y1
GE 360 PLOT A+Y1,B+X1
HC 370 PLOT A-Y1,B+X1
HO 380 PLOT A+Y1,B-X1
IM 390 PLOT A-Y1,B-X1
UL 400 FI=FIY:Y1=Y1+1
LX 420 IF ABS(FIXY)<ABS(FIY) THEN
    FI=FIXY:X1=X1-1
TP 430 IF X1>=Y1 THEN 300
```

```
EW 10 REM PODPROGRAM KRESLENIA KO
   LA-28000
PU 20 GOSUB 28000
YR 30 END
BJ 28000 FOR I=0 TO 758:READ A:PO
   KE 28672+I,A:NEXT I
OU 28004 RESTORE 29500
MT 28005 FOR I=1577 TO 1584:READ
   A:POKE I,A:NEXT I
DS 28010 RETURN
GW 28672 DATA 104,104,141,5,6,104
EG 28678 DATA 141,4,6,104,141,7
CY 28684 DATA 6,104,141,6,6,104
DA 28690 DATA 141,9,6,141,12,6
DT 28696 DATA 104,141,8,6,141,11
YR 28702 DATA 6,104,104,141,10,6
UB 28708 DATA 201,3,144,1,96,169
YD 28714 DATA 0,141,13,6,141,14
DX 28720 DATA 6,141,15,6,141,16
KB 28726 DATA 6,24,173,4,6,109
HL 28732 DATA 11,6,141,25,6,173
EH 28738 DATA 5,6,109,12,6,141
JG 28744 DATA 26,6,24,173,4,6
JS 28750 DATA 109,13,6,141,29,6
JG 28756 DATA 173,5,6,109,14,6
IZ 28762 DATA 141,30,6,56,173,4
IX 28768 DATA 6,237,11,6,141,27
IX 28774 DATA 6,173,5,6,237,12
QH 28780 DATA 6,141,28,6,56,173
GF 28786 DATA 4,6,237,13,6,141
FX 28792 DATA 31,6,173,5,6,141
EG 28798 DATA 14,6,141,32,6,24
FY 28804 DATA 173,6,6,109,11,6
IM 28810 DATA 141,33,6,173,7,6
FR 28816 DATA 109,12,6,141,34,6
IL 28822 DATA 24,173,6,6,109,13
NE 28828 DATA 6,141,37,6,173,7
LB 28834 DATA 6,109,14,6,141,38
PP 28840 DATA 6,56,173,6,6,237
JH 28846 DATA 11,6,141,35,6,173
FR 28852 DATA 7,6,237,12,6,141
PY 28858 DATA 36,6,56,173,6,6
MP 28864 DATA 237,13,6,141,39,6
LG 28870 DATA 173,7,6,237,14,6
JE 28876 DATA 141,40,6,173,25,6
GV 28882 DATA 141,0,6,173,26,6
KM 28868 DATA 141,1,6,173,37,6
LL 28894 DATA 141,2,6,173,38,6
CX 28900 DATA 141,3,6,32,106,114
DD 28906 DATA 173,27,6,141,0,6
ED 28912 DATA 173,28,6,141,1,6
QN 28918 DATA 32,106,114,173,25,6
GK 28924 DATA 141,0,6,173,26,6
KF 28930 DATA 141,1,6,173,39,6
EG 28936 DATA 141,2,6,173,40,6
DX 28942 DATA 141,3,6,32,106,114
ED 28948 DATA 173,27,6,141,0,6
FD 28954 DATA 173,28,6,141,1,6
UB 28960 DATA 32,106,114,173,29,6
CO 28966 DATA 141,0,6,173,30,6
FL 28972 DATA 141,1,6,173,33,6
IE 28978 DATA 141,2,6,173,34,6
EX 28984 DATA 141,3,6,32,106,114
AB 28990 DATA 173,31,6,141,0,6
CV 28996 DATA 173,32,6,141,1,6
SO 29002 DATA 32,106,114,173,29,6
BB 29008 DATA 141,0,6,173,30,6
FW 29014 DATA 141,1,6,173,35,6
GV 29020 DATA 141,2,6,173,36,6
DK 29026 DATA 141,3,6,32,106,114
YO 29032 DATA 173,31,6,141,0,6
BI 29038 DATA 173,32,6,141,1,6
ND 29044 DATA 32,106,114,173,14,6
DO 29050 DATA 205,12,6,240,3,144
LQ 29056 DATA 10,96,173,13,6,205
ML 29062 DATA 11,6,144,1,96,173
CW 29068 DATA 11,6,133,4,173,12
FX 29074 DATA 6,133,5,173,13,6
UE 29080 DATA 133,205,173,14,6,13
   3
MZ 29086 DATA 206,6,4,38,5,6
NT 29092 DATA 205,38,206,56,165,2
   05
JM 29098 DATA 109,15,6,141,17,6
XF 29104 DATA 165,206,109,16,6,14
   1
JV 29110 DATA 18,6,24,173,17,6
QS 29116 DATA 229,4,141,19,6,173
EB 29122 DATA 18,6,229,5,141,20
NC 29128 DATA 6,173,18,6,16,27
SG 29134 DATA 73,255,141,22,6,173
MW 29140 DATA 17,6,73,255,24,105
BQ 29146 DATA 1,141,21,6,173,22
XF 29152 DATA 6,105,0,141,22,6
GR 29158 DATA 24,144,9,141,22,6
EX 29164 DATA 173,17,6,141,21,6
LD 29170 DATA 173,20,6,16,27,73
WY 29176 DATA 255,141,24,6,173,19
ER 29182 DATA 6,73,255,24,105,1
IS 29188 DATA 141,23,6,173,24,6
AV 29194 DATA 105,0,141,24,6,24
MW 29200 DATA 144,9,141,24,6,173
JO 29206 DATA 19,6,141,23,6,173
IV 29212 DATA 17,6,141,15,6,173
GH 29218 DATA 18,6,141,16,6,24
BR 29224 DATA 173,13,6,105,1,141
EI 29230 DATA 13,6,173,14,6,105
CQ 29236 DATA 0,141,14,6,173,22
MY 29242 DATA 6,205,24,6,144,39
LV 29248 DATA 208,8,173,21,6,205
UQ 29254 DATA 23,6,144,29,173,19
DH 29260 DATA 6,141,15,6,173,20
OM 29266 DATA 6,141,16,6,56,173
UO 29272 DATA 11,6,233,1,141,11
BE 29278 DATA 6,173,12,6,233,0
LJ 29284 DATA 141,12,6,76,55,112
WE 29290 DATA 173,2,6,133,205,169
ML 29296 DATA 0,133,206,6,205,38
ND 29302 DATA 206,6,205,38,206,6
TA 29308 DATA 205,38,206,165,205,
   133
HO 29314 DATA 4,165,206,133,5,6
UP 29320 DATA 205,38,206,6,205,38
PG 29326 DATA 206,24,165,205,101,
   4
YO 29332 DATA 133,205,165,206,101
   ,5
LU 29338 DATA 133,206,173,0,6,133
DX 29344 DATA 4,173,1,6,133,5
AU 29350 DATA 70,5,102,4,70,5
WN 29356 DATA 102,4,70,5,102,4
OU 29362 DATA 24,165,205,101,4,13
   3
AC 29368 DATA 205,165,206,101,5,1
   33
```

```
LI 29374 DATA 206,24,165,205,101,
   88
WZ 29380 DATA 133,205,165,206,101
   ,89
GI 29386 DATA 133,206,173,0,6,41
ČF 29392 DATA 7,170,160,0,173,10
NN 29398 DATA 6,208,10,189,41,6
MW 29404 DATA 73,255,49,205,145,2
   05
VS 29410 DATA 96,201,1,208,8,189
PS 29416 DATA 41,6,17,205,145,205
TU 29422 DATA 96,189,41,6,81,205
XK 29428 DATA 145,205,96,0,0,0
MN 29500 DATA 128,64,32,16,8,4,2,
   1
```

Supertekst

Każdy z właścicieli komputera Atari wcześniej czy później zetknie się z dosyć dużą trudnością, jaką jest umieszczenie na ekranie jednocześnie rysunku i tekstu. Część problemów rozwiązuje okno tekstowe, lecz nie wszystkie. Chciałoby się mieć znacznie większe możliwości, np. literatury o różnej wysokości, w różnych kolorach, w dowolnym miejscu, itp. Podprogram opisany w 5 numerze „IKS-a" z ubiegłego roku umożliwiał umieszczanie tekstów na ekranie w trybie graficznym 8. Miał jednak duże ograniczenia: był bardzo wolny (napisany w języku BASIC), pozwalał wykorzystać tylko wielkie litery i cyfry oraz tylko w jednym trybie graficznym.

Poniżej prezentujemy procedurę w języku maszynowym, która umożliwia kreślenie obrazów znaków w dowolnym trybie tekstowym lub graficznym. Ma ona następujące możliwości:

1) Kreśli cały zestaw znaków ATASCII, włączając wielkie i małe litery, znaki graficzne, symbole specjalne i negatywy każdego z tych znaków w dowolnym trybie graficznym lub tekstowym. Mogą być również kreślone opracowane przez nas znaki. W tym celu należy zmienić zawartość zmiennej systemowej CHBAS (o adresie 756) dla wskazania położenia własnego zestawu znaków.
2) Umożliwia podanie ciągu kreślonych znaków. Długość ciągu ograniczona jest wyłącznie ilością miejsca przeznaczonego na jego wyświetlenie.
3) Umożliwia określenie pozycji początkowej kreślonego ciągu, mogą to być dowolne współrzędne (x,y) na ekranie.
4) Pozwala na zakrywanie istniejących już rysunków lub łączenie kreślonych znaków z grafiką.
5) Wybieranie rejestrów kolorów używanych dla tła i kreślonych znaków.
6) Skalowanie każdego ciągu znaków niezależnie, zarówno w poziomie, jak i w pionie, przez określenie liczby wierszy i kolumn dla każdego znaku. Aktualna wielkość każdego znaku zależy od rozmiaru elementarnego punktu graficznego wybranego trybu.
7) Ciągi znaków mogą być kreślone w jednym z czterech kierunków.

Opisane możliwości stają się dostępne przez wywołanie procedury w języku maszynowym za pomocą instrukcji USR. Jej składnia jest następująca:

A = USR(ADR(ASM$), ADR(S$), LEN (S$), XS, YS, ORIENT, IWIER, IKOL, RKZ, RKT, PRIOR)

Używane parametry mają następujące znaczenie:

ADR(ASM$) — Początkowy adres procedury „Supertekst".

ADR(S$) — Adres kreślonego ciągu, a dokładniej adres pierwszego znaku tego ciągu. Poprawny będzie dowolny adres, może on również wskazywać podciąg zawarty w długim ciągu.

LEN(S$) — Parametr ten określa liczbę kreślonych znaków. Funkcja LEN dostarcza właściwej wartości w wypadku kreślenia całego ciągu.

XS, YS — Współrzędne (x,y) początku kreślonego tekstu (lewy dolny róg pierwszego znaku). Wskazany punkt używany jest również jako oś obrotu podczas obracania ciągu. (xs, ys) muszą określać punkt wewnątrz granic aktualnego trybu graficznego.

ORIENT — Parametr ten określa kierunek kreślonego ciągu znaków. Tekst obracany jest w kierunku przeciwnym do ruchu wskazówek zegara od poziomej półosi +x wokół punktu (xs, ys). Wartość parametru powinna być całkowita i interpretowana jest następująco:
ORIENT = 0 — obrót o 0 stopni.
ORIENT = 1 — obrót o 90 stopni.
ORIENT = 2 — 180 stopni.
ORIENT = 3 — obrót o 270 stopni.
Wartość ORIENT interpretowana jest modulo 3, dlatego ORIENT = 4 oznacza to samo, co ORIENT = 0.

IWIER — Określa, ile wierszy na znak będzie kreślonych, definiuje zatem wysokość każdego znaku. Zwykle IWIER jest większe lub równe osiem, poprawne są także wartości dodatnie mniejsze niż osiem, lecz w efekcie kreślone są znaki z „opuszczonymi" wierszami. Maksymalną wartością jest IWIER = = 255.

IKOL — Liczba kolumn na znak, definiuje zatem szerokość każdego znaku.

RKZ — Określa rejestr koloru, który będzie używany do kreślenia znaków. Pośrednio określa kolory znaków. W trybach tekstowych (0—2) powinien być określony jako kod ATASCII.

RKT — Określa rejestr koloru tła dla każdego ciągu znaków. Jeśli wartość parametru PRIOR jest zerowa, to RKT nie ma wpływu na kreślone znaki.

PRIOR — Określa priorytet tła kreślonego ciągu znaków. Jeśli PRIOR= 0, tło znaków nie jest kreślone i istniejąca na ekranie grafika nie będzie naruszona. Jeśli PRIOR ma wartość dodatnią, kolor pośrednio określony przez RKT jest kreślony dla tła.

Podczas określania wartości parametrów należy uwzględnić następujące uwagi:

1. Procedura „Supertekst" nie sprawdza, czy kreślone znaki mieszczą się w obszarze wyświetlania.
2. Negatywy znaków mogą być tworzone dwoma sposobami:

a) podanie negatywów znaków w ciągu, który będzie kreślony,

b) podanie normalnych znaków w ciągu i zamiana wartości RKZ i RKT.

3. Jeśli parametrom RKZ i RKT przydzielona jest ta sama wartość (i prior jest dodatnie), ciąg znaków zostanie wykreślony lecz pojawi się jako ciągłe kolorowe bloki.

4. Jeśli podana jest niewłaściwa ilość parametrów w instrukcji USR, „Supertekst" nie wykona żadnej akcji.

5. Wartość A zwracana przez funkcję USR nie ma żadnego znaczenia.

Przedstawiony poniżej program zawiera zarówno opisaną procedurę, jak i programy demonstracyjne pokazujące sposoby jej używania.

„Supertekst" rozpoczyna się od linii o numerze 8000. Początkowe instrukcje służą do wprowadzania właściwej procedury w języku wewnętrznym komputera. Proszę zwrócić uwagę na nietypowe instrukcje DATA oraz rozbudowane instrukcje ładowania kodu maszynowego. Spowodowane to jest specyficzną postacią procedury, która umożliwia umieszczenie jej w dowolnym obszarze pamięci, a w szczególności w zmiennej tekstowej. Taką postać nazywamy przemieszczalną. Wszystkie adresy w instrukcjach są względne, wyznaczone względem początku procedury. Właśnie te adresy są oznaczone jako liczby ujemne w instrukcjach DATA.

SUPER TEKST *ATARI*

```
HW 1000 REM
ZC 1010 REM ----INICJACJA-----
GT 1030 DIM S$(40):DEG :? "PROSZE
    CZEKAC 45 SEK.":GOSUB 8000
II 1040 REM
CR 1050 REM -------DEMO #1--------
IO 1060 REM
XM 1070 GRAPHICS 7+16
VY 1080 S$="INFORMATYKA":XS=24:YS
   =24:ORIENT=0:NROWS=24:NCOLS=8:
   FCR=3:BCR=0:PRIOR=0:GOSUB 8000
HL 1090 S$="KOMPUTERY":XS=24:YS=4
   4:ORIENT=0:NROWS=8:NCOLS=8:FCR
   =1:BCR=0:PRIOR=0:GOSUB 8000
UT 1100 S$="SYSTEMY":XS=24:YS=95:
   ORIENT=0:NROWS=32:NCOLS=16:FCR
   =2:BCR=0:PRIOR=0:GOSUB 8000
ZP 1110 S$=" ATARI  ":NROWS=8:NCO
   LS=8:FCR=3:BCR=1:PRIOR=1
YI 1120 XS=7:YS=64:ORIENT=1:PRIOR
   =1:GOSUB 8000
PB 1130 XS=151:YS=2:ORIENT=3:GOSU
   B 8000
ZR 1140 GOSUB 7000
IN 1150 REM
BM 1800 REM -----DEMO #2----------
   -
IP 1810 REM
EX 1820 GRAPHICS 7+16:COLOR 3
WH 1840 PLOT 34,2:DRAWTO 126,2:DR
   AWTO 126,94:DRAWTO 34,94:DRAWT
   O 34,2
QP 1850 PLOT 51,19:DRAWTO 109,19:
   DRAWTO 109,77:DRAWTO 51,77:DRA
   WTO 51,19
TS 1860 PLOT 60,28:DRAWTO 100,28:
   DRAWTO 100,68:DRAWTO 60,68:DRA
   WTO 60,28
CT 1870 PLOT 34,2:DRAWTO 60,28
HX 1880 PLOT 126,2:DRAWTO 100,28
DF 1890 PLOT 126,94:DRAWTO 100,68
SZ 1900 PLOT 34,94:DRAWTO 60,68
DL 1910 S$="ATARI":FCR=2:BCR=0:NR
   OWS=8:NCOLS=8:PRIOR=0
MM 1920 XS=60:YS=27:ORIENT=0:GOSU
   B 8000
WS 1930 XS=101:YS=28:ORIENT=3:GOS
   UB 8000
WH 1940 XS=98:YS=69:ORIENT=2:GOSU
   B 8000
SV 1950 XS=58:YS=67:ORIENT=1:GOSU
   B 8000
CS 1960 NROWS=16:NCOLS=11:FCR=1
NX 1970 XS=53:YS=18:ORIENT=0:GOSU
   B 8000
SR 1980 XS=110:YS=21:ORIENT=3:GOS
   UB 8000
CA 1990 XS=107:YS=78:ORIENT=2:GOS
   UB 8000
NW 2000 XS=50:YS=75:ORIENT=1:GOSU
   B 8000
VP 2010 XS=61:YS=67:ORIENT=0:FCR=
   3:BCR=2:NROWS=38:NCOLS=39:PRIO
   R=1
CY 2020 FOR I=1 TO 10
MP 2030 S$="IKSIKSIKSX":S$=S$(I,I
   ):GOSUB 8000
FC 2040 NEXT I
ZT 2050 GOSUB 7000
QU 2060 GOTO 1070
IC 7000 REM
VS 7010 REM ----ZMIANA KOLORU-----
   -
II 7020 REM
CT 7030 FOR I=1 TO 50:FOR J=0 TO
   2:SETCOLOR J,RND(0)*16,RND(0)*
   16:FOR W=1 TO 5:NEXT W:NEXT J:
   NEXT I:RETURN
NG 8000 IF ASMLD=1 THEN A=USR(ADR
   (ASM$),ADR(S$),LEN(S$),XS,YS,O
   RIENT,NROWS,NCOLS,FCR,BCR,PRIO
   R):RETURN
EZ 8010 ASMLD=1
SA 8020 DIM ASM$(725)
DO 8030 FOR I=ADR(ASM$) TO ADR(AS
   M$)+724
QG 8040 READ A
BJ 8050 ON (SGN(A)+2) GOSUB 8080,
   8160,8220
FO 8060 NEXT I
QP 8070 GOTO 8000
RD 8080 READ B
QX 8090 ADDR=ABS(A)+256*ABS(B)+AD
   R(ASM$)
KS 8100 ADDRHI=INT(ADDR/256)
EE 8110 ADDRLO=ADDR-256*ADDRHI
WQ 8120 POKE I,ADDRLO
SQ 8130 POKE I+1,ADDRHI
KY 8140 I=I+1
AX 8150 RETURN
QZ 8160 READ B
EI 8170 IF B<0 THEN 8090
AQ 8180 POKE I,A
CB 8190 POKE I+1,B
KO 8200 I=I+1
AN 8210 RETURN
AA 8220 POKE I,A
AT 8230 RETURN
FS 8240 DATA 104,141,-255,0,10,17
   0,240,8
OD 8250 DATA 104,157,-255,0,202,7
   6,-6,0
QO 8260 DATA 173,-255,0,201,10,24
   0,1,96
BK 8270 DATA 173,-16,-1,208,1,96,
   173,-8
EA 8280 DATA -1,208,1,96,173,-6,-
   1,208,1
XK 8290 DATA 96,173,-10,-1,41,3,1
   41,-10
ZX 8300 DATA -1,173,-18,-1,133,20
   3,173
OR 8310 DATA -19,-1,133,204,169,0
   ,141
RY 8320 DATA -17,-1,238,-17,-1,56
   ,173
AT 8330 DATA -16,-1,237,-17,-1,16
   ,3,76
VC 8340 DATA -254,0,32,-32,-1,173
   ,-6,-1
HN 8350 DATA 141,-7,-1,206,-7,-1,
   174,-7
TM 8360 DATA -1,224,255,208,3,76,
   -251,0
NK 8370 DATA 172,-6,-1,32,-138,-1
   ,140,-3
AH 8380 DATA -1,173,-6,-1,174,-17
   ,-1,172
FO 8390 DATA -7,-1,32,-195,-1,140
   ,-20,-1
HM 8400 DATA 142,-21,-1,169,255,1
   41,-9
PB 8410 DATA -1,238,-9,-1,174,-9,
   -1,236
IR 8420 DATA -8,-1,208,3,76,-248,
   0,172
QS 8430 DATA -8,-1,32,-138,-1,140
   ,-5,-1
XJ 8440 DATA 172,-5,-1,177,205,17
   2,-3,-1
ZU 8450 DATA 57,-24,-1,240,2,169,
   1,141
QV 8460 DATA -1,-1,173,-135,-1,24
   0,9,56
BL 8470 DATA 169,1,237,-1,-1,141,
   -1,-1
ZM 8480 DATA 173,-1,-1,208,5,173,
   0,-1
WM 8490 DATA 240,46,174,-2,-1,173
   ,-1,-1
LH 8500 DATA 240,3,174,-4,-1,142,
   -212,-2
QY 8510 DATA 173,-8,-1,162,1,172,
   -9,-1
KU 8520 DATA 32,-195,-1,140,-22,-
   1,142
```

```
WU 8530 DATA -23,-1,32,-16,-2,173,-184
OH 8540 DATA -2,174,-183,-2,172,-185,-2
AT 8550 DATA 32,-187,-2,76,-134,0,76,-89
YT 8560 DATA 0,76,-65,0,96,0,0,0,0,0,0,0
UH 8570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0
GA 8580 DATA 0,0,0,0,1,2,4,8,16,32,64
CS 8590 DATA 128,169,0,141,-135,-1,172
NG 8600 DATA -17,-1,136,177,203,141,-136
GI 8610 DATA -1,16,13,169,1,141,-135,-1
IY 8620 DATA 173,-136,-1,41,127,141,-136
HF 8630 DATA -1,56,173,-136,-1,233,32,16
PA 8640 DATA 12,24,173,-136,-1,105,64
AE 8650 DATA 141,-137,-1,76,-107,-1,56
WO 8660 DATA 173,-136,-1,233,96,16,12,56
IK 8670 DATA 173,-136,-1,233,32,141,-137
RC 8680 DATA -1,76,-107,-1,173,-136,-1
EY 8690 DATA 141,-137,-1,169,0,133,206
BG 8700 DATA 173,-137,-1,133,205,162,3
AR 8710 DATA 24,38,205,38,206,202,208
CW 8720 DATA 248,24,165,206,109,244,2
BO 8730 DATA 133,206,96,0,0,0,142,-193
CC 8740 DATA -1,140,-194,-1,142,-191,-1
OM 8750 DATA 169,0,141,-192,-1,162,3,24
TY 8760 DATA 46,-191,-1,46,-192,-1,202
LJ 8770 DATA 208,246,160,255,200,56,173
ZV 8780 DATA -191,-1,237,-194,-1,141
RS 8790 DATA -191,-1,173,-192,-1,233,0
LX 8800 DATA 141,-192,-1,16,235,142,-191
KI 8810 DATA -1,96,0,0,0,0,141,-12,-2
OT 8820 DATA 140,-13,-2,169,0,141,-14,-2
XE 8830 DATA 141,-15,-2,24,173,-12,-2
WV 8840 DATA 109,-14,-2,141,-14,-2,169,0
JL 8850 DATA 109,-15,-2,141,-15,-2,202
FW 8860 DATA 208,235,56,173,-14,-2,233,1
PW 8870 DATA 141,-14,-2,173,-15,-2,233,0
EL 8880 DATA 141,-15,-2,56,173,-14,-2
WZ 8890 DATA 237,-13,-2,141,-14,-2,168
MT 8900 DATA 173,-15,-2,233,0,141,-15,-2
DO 8910 DATA 170,96,0,0,0,0,174,-10,-1
QX 8920 DATA 208,39,24,173,-14,-1,109
GK 8930 DATA -20,-1,141,-183,-2,173,-15
LJ 8940 DATA -1,109,-21,-1,141,-184,-2
ZC 8950 DATA 56,173,-12,-1,237,-22,-1
NF 8960 DATA 141,-185,-2,173,-13,-1,237
CE 8970 DATA -23,-1,141,-186,-2,96,202
XO 8980 DATA 208,39,56,173,-14,-1,237
IC 8990 DATA -22,-1,141,-183,-2,173,-15
MV 9000 DATA -1,237,-23,-1,141,-184,-2
VP 9010 DATA 56,173,-12,-1,237,-20,-1
MC 9020 DATA 141,-185,-2,173,-13,-1,237
AB 9030 DATA -21,-1,141,-186,-2,96,202
WL 9040 DATA 208,39,56,173,-14,-1,237
FZ 9050 DATA -20,-1,141,-183,-2,173,-15
LZ 9060 DATA -1,237,-21,-1,141,-184,-2
UP 9070 DATA 24,173,-12,-1,109,-22,-1
KF 9080 DATA 141,-185,-2,173,-13,-1,109
RR 9090 DATA -23,-1,141,-186,-2,96,24
KG 9100 DATA 173,-14,-1,109,-22,-1,141
GA 9110 DATA -183,-2,173,-15,-1,109,-23
UF 9120 DATA -1,141,-186,-2,24,173,-12
LD 9130 DATA -1,109,-20,-1,141,-185,-2
IZ 9140 DATA 173,-13,-1,109,-21,-1,141
ZO 9150 DATA -186,-2,96,0,0,0,0,134,85
NN 9160 DATA 133,86,132,84,162,96,169,11
GI 9170 DATA 157,66,3,169,0,157,72,3,173
EX 9180 DATA -212,-2,32,86,228,96,1
```

REVERSI

Jedną z ciekawszych gier, jakie mamy dla komputera Atari jest REVERSI. Nie odznacza się oszałamiającą grafiką ani fascynującym dźwiękiem. Nie przenosi nas również w inny, nieznany świat. Jednak każda rozgrywka jest interesująca, tym bardziej że jej przebieg zależy od nas. Gra nie wymaga biegłości w posługiwaniu się manipulatorem ani dobrego refleksu. Wymaga za to logicznego myślenia i przewidywania skutków posunięć.

Program jest komputerową wersją tradycyjnej gry, znanej od końca XVIII w., szczególnie chętnie uprawianej w drugiej połowie XIX wieku w całej Europie.

Do gry potrzebna jest plansza o 64 polach (jak zwykła szachownica, lecz jednobarwna) oraz 64 specjalne piony — każdy z nich jest z jednej strony biały, a z drugiej czarny, może więc być użyty jako biały lub czarny, w zależności od położenia na planszy.

Celem gry jest ustawienie, a dokładniej uzyskanie na planszy maksymalnej liczby własnych pionów.

Rozpoczynającego grę wyznacza się przez losowanie. Obydwaj gracze, na zmianę, kolejno ustawiają swoje piony na planszy. Pion raz postawiony nie wykonuje żadnych ruchów, kolejne posunięcia polegają na dostawianiu na planszy coraz to nowych pionów.

Obowiązkiem graczy jest w pierwszych czterech ruchach ustawić piony na czterech centralnych polach planszy (D4, D5, E4, E5). W następnych ruchach gracze ustawiają swoje piony już na dowolnie wybranych polach, przy czym w wyniku każdego posunięcia co najmniej jeden pion przeciwnika musi zostać zdobyty. Jeśli gracz, na którego przypada kolejność wykonania ruchu, nie jest w stanie zrobić tego, tak by zdobyć przynajmniej jeden pion przeciwnika, musi zrezygnować z wykonania ruchu.

Bicie pionów odbywa się przez oskrzydlenie z dwóch stron pól przeciwnika — pionowo, poziomo lub po przekątnej. Piony przeciwnika, znajdujące się pomiędzy naszymi, zostają zdobyte; odwrócone na drugą stronę stają się naszymi.

Jeden i ten sam pion, stojący na planszy, może więc w toku rozgrywki wiele razy zmieniać przynależność do jednego lub drugiego gracza. Często znaczna przewaga może być zniwelowana przez jeden strategicznie umieszczony pion.

W komputerowej wersji gry odpowiednia plansza pojawia się na ekranie z czterema pionami usta-

wionymi w centralnych polach planszy. Jeden gracz może grać przeciwko drugiemu lub komputerowi na trzech różnych poziomach, za pomocą manipulatora.

Na najłatwiejszym poziomie komputer zawsze stara się zabrać jak najwięcej pionów w jednym ruchu (podobnie grają początkujący gracze) i daje się łatwo ograć. Za to na najwyższym poziomie rozważa swoje ruchy, nasze odpowiedzi i swój następny ruch; bierze również pod uwagę każde pole, ponieważ niektóre z nich są traktowane jako bardziej wartościowe niż inne. Inna opcja oferuje niezwykłą własność — możemy zobaczyć, w jaki sposób komputer wyznacza swój następny ruch (dostępne na najwyższym poziomie). Wyświetla wtedy liczbę pól, które będą uzyskane dla dowolnego ruchu i wartość każdego kwadratu (pola) po rozważeniu innych możliwych ruchów.

Komputer jest wymagającym przeciwnikiem, zmuszającym nas do intensywnego myślenia. Umożliwia nam również, po nabraniu wprawy, rozegranie partii z prawdziwym ludzkim przeciwnikiem. Dostarcza nam planszę i piony, przestrzega zasad gry oraz na bieżąco podaje wynik.

Ludwik PIELA
Tomasz MROWIEC

PROGRAM 11

SHARP

KTO WINIEN?

Prezentowany program jest komputerową... *opowieścią kryminalną*. Ma ona 2295 odmian, które zapewnia funkcja RANDOM

A oto fabuła opowieści. Popełniono przestępstwo. Nie perfidne morderstwo, lecz „zwyczajną" kradzież. Jaką? Poda komputer! „Zaprezentuje" on ośmiu zatrzymanych. Należy zdecydować, *kto winien?*

Przesłanką są zaprezentowane „ustalenia", metodą zaś — dwa podstawowe pytania kryminalistyki:

1. — *KTO zostawił ślady?*
2. — *KTO skorzystał z kradzieży?*

„Śledztwo" jest dwuetapowe: najpierw uzyskujemy *część danych i wstępnie typujemy winnych.*

W drugim etapie dostajemy *pełną dokumentację i podejmujemy ostateczne decyzje.*

Czy były one słuszne? Odpowie... komputer.

Program wywołuje etykieta „X" (RUN 100).

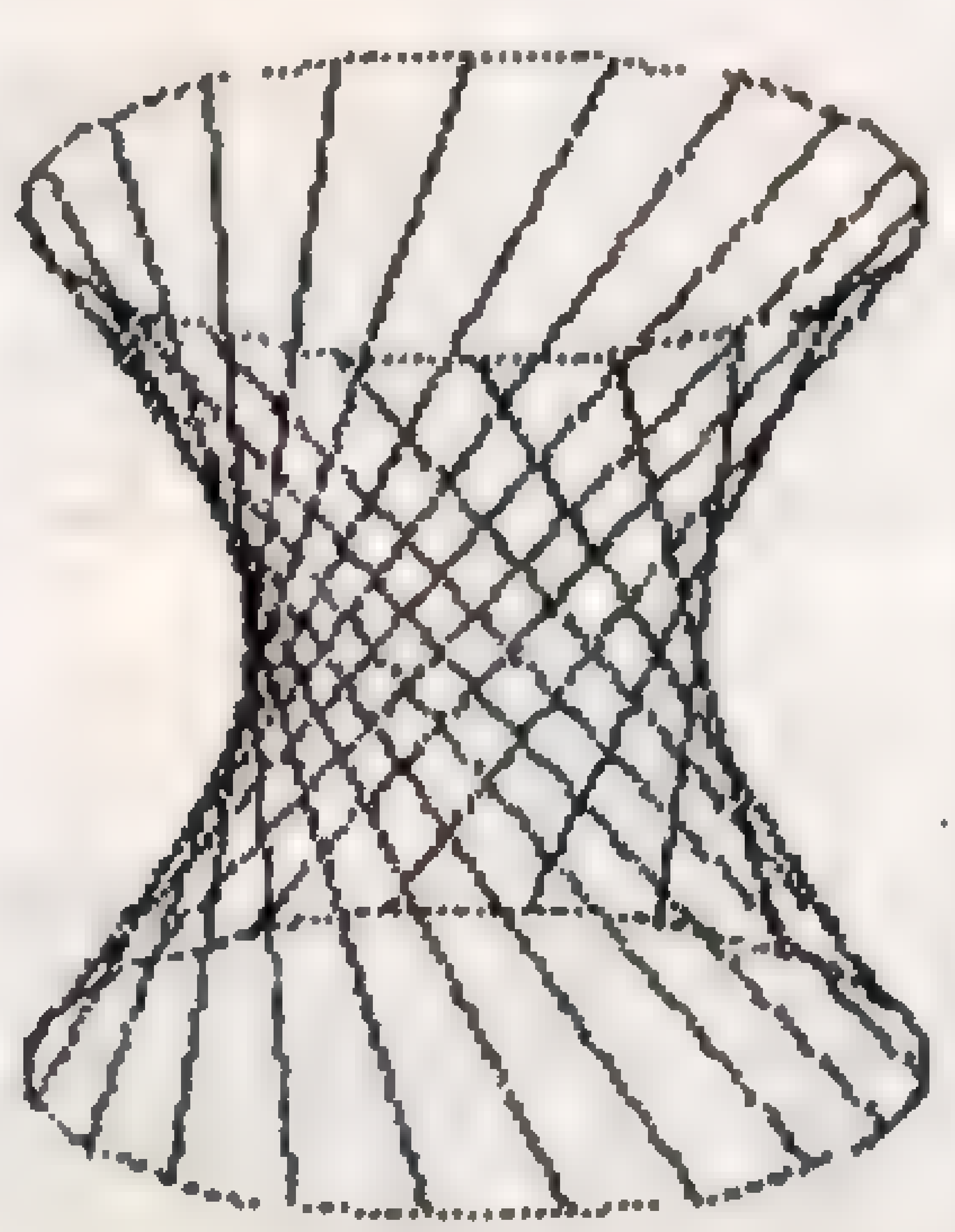
Uwaga! W programie wykorzystano *„japońskie delikatesy"*: 120 Restore 10 — oznacza „czytaj dane (Data) TYLKO linii 10". 280:ON A GOSUB 300, 320, 330 — "gdy A = 1, adresuj 300; gdy A = 2, to 320; gdy A = 3, to 330".

Życzymy osiągnięcia absolutnej biegłości...

Janusz MILLER

```
 10:DATA "ADAM","J
    AN","EDWARD","
    HENRYK","ZENON
    ","MARIAN","PA
    WEL","ROMAN"
 20:DATA " ZYJE UB
    OGO"," ZYJE SK
    ROMNIE"," ZYJE
     SREDNIO"
 30:DATA " CZESTE
    LIBACJE"," BUD
    UJE GARAZ"," S
    TAWIA DOM"
 40:DATA " WLAMANI
    E DO"," OGRABI
    ENIE"," WYNIES
    IENIE Z"
 50:DATA " PAWILON
    U"," MAGAZYNU"
    ," KASY"
 60:DATA " SLADY P
    ALCOW"," NITKI
     UBRANIA"," SL
    ADY OBUWIA"
 70:DATA " KILKA W
    LOSOW"," NIEDO
    PALEK"," BRAK
    SLADOW"
 80:A=INT Y:Y=(Y-A
    )*10:B=INT Y:Y
    =(Y-B)*10
 90:C=INT Y:Y=(Y-C
    )*10:D=Y:
    RETURN
100:"X":CLEAR :
    RANDOM :USING
    :WAIT 0:PRINT
    "S L E D Z T W
    O"
110:DIM X$(8),K$(6
    ),Z$(6),S$(6),
    X(8),Y(8)
120:RESTORE 10:FOR
    K=1TO 8:READ X
    $(K):NEXT K
130:RESTORE 20:FOR
```

```
   K=1TO 3:READ K
   $(K):NEXT K
140:RESTORE 30:FOR
   K=4TO 6:READ K
   $(K):NEXT K
150:RESTORE 40:FOR
   K=1TO 3:READ Z
   $(K):NEXT K
160:RESTORE 50:FOR
   K=4TO 6:READ Z
   $(K):NEXT K
170:RESTORE 60:FOR
   K=1TO 3:READ S
   $(K):NEXT K
180:RESTORE 70:FOR
   K=4TO 6:READ S
   $(K):NEXT K
190:G=RND 8:FOR K=
   1TO G
200:R=RND 8:IF X(R
   )>0GOTO 200
210:X(R)=3:NEXT K:
   WAIT
230:FOR K=1TO 8:IF
   X(K)>0GOTO 260
240:R=RND 2:IF R=2
   LET X(K)=1:
   GOTO 260
250:X(K)=2
260:NEXT K:R=RND 3
   :S=(RND 3)+3:
   PRINT Z$(R);Z$
   (S)
270:FOR K=1TO 8:A=
   INT X(K)
280:ON AGOSUB 300,
   320,330
290:NEXT K:PRINT "
   Z A T R Z Y M
   A N I":GOTO 36
   0
300:X(K)=X(K)+0.6
310:R=RND 6:X(K)=X
   (K)+R/100+R/1E
   3:RETURN
320:R=RND 5:X(K)=X
   (K)+R/10:GOTO
   310
330:R=RND 5:S=RND
   3:T=(RND 3)+3
340:X(K)=X(K)+R/10
   +S/100+T/1E3:
   RETURN
350:R=RND 3:S=(RND
   3)+3:PRINT Z$(
   R);Z$(S):Q=0
360:U=0:W=RND 8:
   FOR K=1TO 8
370:T=W+K:IF T>8
   LET T=T-8
380:IF (Q=1)AND (Y
   (T)<3)GOTO 460
390:Y=X(T):GOSUB 8
   0:U=U+1
400:PRINT U;" IMIE
   : ";X$(T)
410:PRINT "DOWODY:
   ";S$(B)
420:PRINT "ZWYKLE:
   ";K$(C):IF Q=
   0GOTO 440
430:PRINT "OSTATNI
   O: ";K$(D)
440:INPUT  WINIEN
   ? T=1 N=0 ";E:
   IF E=1LET Y(T)
   =3
450:IF E=0LET Y(T)
   =1
460:NEXT K:IF Q=1
   THEN PRINT "W
   Y N I K:I":
   GOTO 480
470:PRINT "ETAP DR
   UGI":Q=1:GOTO
   360
480:WAIT 0:U$=" UW
   AGA ! ":W$=" W
   INNY ":N$=" NI
   EWINNY ":D$="
   ZGODA: "
490:FOR K=1TO 8:A=
   INT X(K):B=Y(K
   ):C$=X$(K):
   PRINT STR$ K;:
   WAIT
500:IF (A<3)AND (B
   <3)THEN PRINT
   D$;C$;N$
510:IF (A=3)AND (B
   =3)THEN PRINT
   D$;C$;W$
520:IF (A=3)AND (B
   <3)THEN PRINT
   U$;C$;W$
530:IF (A<3)AND (B
   =3)THEN PRINT
   U$;C$;N$
540:WAIT 0:PRINT "
   ":NEXT K:END
```



ATARI

```
5 REM * GRAFIKA ATARI *
*HIPERBOLOIDA JEDNOOBRO
TOWA *
10 GRAPHICS 8:COLOR 1:P
I=3.14:N=-10
20 FOR A=-PI TO PI STEP
 0.01*PI:AA=A+2/3*PI
30 X=INT(COS(A)*45):XX=
INT(COS(AA)*45):Y=INT(S
IN(A)*17):YY=INT(SIN(AA
)*17)
40 IF A=N*PI/10 THEN N=
N+1:PLOT 160+X,30+Y:DRA
WTO 160+XX,120+YY
50 PLOT 160+X,30+Y:PLOT
 160+XX,120+YY:NEXT A
```

JANUSZ W. JANIEC

MIKROKOMPUTERY IBM PC (2)

System mikroprocesorowy, tworzący mikrokomputer IBM PC/XT, można zaliczyć do systemów wieloprocesorowych z dzieloną magistralą lokalną, a uściślając przypisać go do systemów wielomikroprocesorowych silnie powiązanych, gdyż komunikacja między mikroprocesorami odbywa się przez wspólną pamięć operacyjną.

Z punktu widzenia funkcjonalnego w systemie można wyróżnić cztery zasadnicze podsystemy: podsystem mikroprocesora i współpracujących z nim elementów, podsystem pamięci stałej ROM, podsystem pamięci zapisywalnej RAM, podsystem wejścia — wyjścia.

Z punktu widzenia konstrukcyjnego system można podzielić na pakiety między innymi: pakiet systemowy i pakiety dodatkowe (rozszerzające). Na rysunku 3 przedstawiono uproszczony schemat blokowy mikrokomputera IBM PC/XT.

W systemie IBM PC/XT jako jednostkę centralną wykorzystano INTEL 8088, będący wersją 16-bitowego mikroprocesora INTEL 8086, wersją z 8-bitową zewnętrzną magistralą danych. Mikroprocesor 8088 zastosowany w omawianym mikrokomputerze, pracuje z częstotliwością zegara 4,77 MHz. Sygnał zegarowy dla mikroprocesora dostarcza układ 8284, generator sygnału zegarowego. Źródłem częstotliwości sterującej pracą układu 8284 jest zewnętrzny rezonator kwarcowy o częstotliwości 14,31818 MHz, która to częstotliwość dzielona jest przez 3.

Częstotliwość 4,77 MHz pozwala na pracę mikroprocesora 8088 z cyklem magistrali trwającym cztery takty zegara po 0,210 μs lub 0,840 μs.

Układy wejścia — wyjścia pracują z cyklem pięciu taktów zegara po 0,210 μs lub 1,05 μs.

INTEL 8088 ma 20-bitową magistralę adresową (2^{20} = 1 MB = 1024 KB, przy czym 1 KB = 1024 bajty), a więc może adresować przestrzeń 1 048 576 bajtów. Z ośmioma mniej znaczącymi bitami adresu multipleksowane są bity danych. Bity adresowe i bity danych przekazywane są do układów współpracujących z mikroprocesorem za pomocą buforów adresowych i buforów danych.

Mikroprocesor 8088 steruje pracą pozostałych elementów systemu przez kontroler magistrali sterującej, układ 8288 nazywany również generatorem sygnałów magistrali sterującej w konfiguracji maksymalnej (mikroprocesora 8086 lub 8088).

Koprocesor INTEL 8087 jest procesorem numerycznym i przeznaczony jest do wykonywania operacji arytmetycznych. Pracuje równolegle z jednostką centralną (zwaną mikroprocesorem nadrzędnym), pobiera ten sam ciąg rozkazów, ale wykonuje rozkazy przeznaczone tylko dla niego.

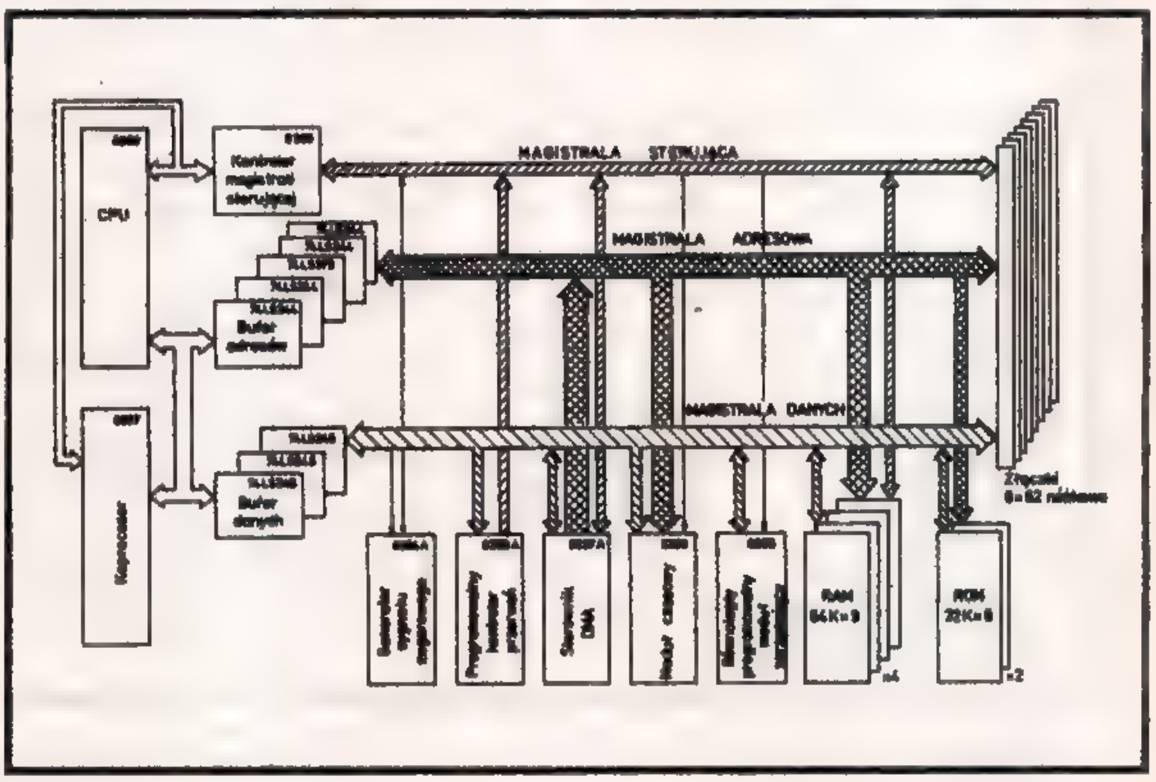
Programowalny kontroler przerwań 8259 A przeznaczony jest do nadzorowania przerwań (żądań obsługi) przekazywanych do mikroprocesora przez układy współpracujące. Układ 8259 A umożliwia realizację ośmiopoziomowych przerwań priorytetowych, tzw. przerwań maskowalnych INTR. Sześć przerwań przekazywanych jest do pakietu systemowego z pakietów dodatkowych, dołączonych do systemu za pomocą złączek (łączówek). Wśród sygnałów przerwań, zgłaszanych przez pakiety dodatkowe są sygnały z dysku stałego, dysku elastycznego, drukarki. Dwa kolejne przerwania zgłaszają układy z pakietu systemowego: moduł czasowy 8253 i klawiatura. W omawianym systemie realizowane jest również przerwanie niemaskowalne, po wykryciu błędu parzystości w pamięci operacyjnej.

Sterownik DMA (Direct Memory Access) 8237 A umożliwia prowadzenie szybkich transmisji między modułami zewnętrznymi mikrokomputera a pamięcią operacyjną, z komputera, przez równoległy programowany moduł sprzęgający 8255.

Poprzez porty (bramy wejścia — wyjścia) układu 8255 informacje z/do magistrali danych przekazywane są do/z różnych układów pakietu systemowego, m.in. do głośnika.

W systemie znajduje się 256 KB pamięci operacyjnej RAM, w czterech blokach 64 KB × 9, dziewiąty bit to kontrolny bit parzystości. W jednym bloku znajduje się dziewięć układów pamięci dynamicznej, o organizacji 64 K × 1 bit i czasie dostępu 0,200 μs. Pamięć operacyjną można zwiększyć do 640 KB po dołączeniu do pakietu systemowego, poprzez jedną ze złączek pakietu dodatkowego, zawierającego również układy organizujące pracę monitora monochromatycznego i drukarki matrycowej, mechanizmów dysków elastycznych, monitora kolorowego itp. Kolejne złączki umożliwiają dołączanie innych pakietów.

Ostatnim z omawianych elementów jest pamięć stała. Mogą to być pamięci ROM lub EPROM. W pakiecie znajdują się dwa układy o organizacji 32 K × 8 bitów. Mikrokomputer wykorzystuje do 40 KB pamięci stałej, w której zapisany jest ROM-BIOS, interpreter języka BASIC, diagnostyka.



pominięciem mikroprocesora. Steruje on transmisję danych odczytanych (zapisywanych) na dysku do/z pamięci operacyjnej.

Moduł czasowy (Timer/Cunter) 8253 wykorzystywany jest do generowania przebiegów wyjściowych o zadanej częstotliwości przebiegów sterujących głośnikiem mikro-

Dalszy opis mikrokomputera w następnym numerze „IKS-a".

Jacek WOJTALA

Spectrum Kryształ

(J. Rubinowicz)

```
10 FOR i=3 TO 20: CLS
20 DIM a(i): DIM b(i)
30 FOR n=1 TO i: FOR m=n TO 1
STEP -1: LET a(n)=128+INT (.5+86
*SIN (2*(n-1)*PI/i))
40 LET b(n)=88+INT (.5+86*COS
(2*(n-1)*PI/i))
50 PRINT #0;AT 1,15;i
60 PLOT a(n),b(n): DRAW a(m)-a
(n),b(m)-b(n)
70 BEEP .01,3*m: NEXT m: NEXT
n
80 PAUSE 50: NEXT i: GO TO 10
90 SAVE "*KRYSZTAL*" LINE 10
```





INFOSYSTEM '87







K R E S K A

BASIC CPC 464 (4)

MODUŁOWY PROGRAM GRAFICZNY

Obecnie przystąpimy do rozszerzenia programu KRESKA o następujące funkcje:

— kreślenie elipsy
— kreślenie trójkątów: równoramiennego i o dowolnym kształcie.

Kreślenie elipsy opiera się na tej samej zasadzie co rysowanie okręgu. Pierwsze naciśnięcie FIRE/COPY wyznacza środek elipsy. Dalsze przemieszczanie kursora graficznego po ekranie określa zmianę długości pierwszej osi elipsy. Ponowne wciśnięcie FIRE/COPY powoduje wyliczenie jej długości. Analogicznie wyznaczana jest długość drugiej osi. Samo kreślenie elipsy polega na wyznaczaniu na ekranie monitora wszystkich punktów o współrzędnych x,y spełniających warunki:

$x = xc\% + r * \cos(a)$
$y = yc\% + p * \sin(a)$

gdzie:
xc%, yc%, — współrzędne środka elipsy
r — połowa długości pierwszej osi
p — połowa długości drugiej osi
a — kąt od 0° do 360°.

W związku z podobieństwem algorytmu kreślenia elipsy i okręgu, dokonujemy modyfikacji procedury rysowania okręgu, m.in. wprowadzając linie programu odnoszące się do nowej funkcji. Wprowadzony parametr w$ steruje zmodyfikowaną procedurą w następujący sposób:
w$ = „E" — kreślenie elipsy
w$ = „O" — kreślenie okręgu.

W wypadku rysowania dowolnego trójkąta wyznaczamy kolejne jego wierzchołki, przemieszczając po ekranie kursor graficzny, oraz naciskając FIRE/COPY. Po określeniu współrzędnych trzeciego wierzchołka następuje wykreślenie figury. Aby otrzymać na ekranie trójkąt równoramienny, wyznaczamy punkty wierzchołkowe podstawy, a następnie z jej środka wyprowadzamy wysokość. Po jej wyznaczeniu (naciśnięcie FIRE/COPY) zostaje wykreślony trójkąt.

Prezentowany dodatkowo program RSX LOADER został zaczerpnięty z lutowego numeru pisma PC WORLD. Program ten rozszerza interpreter BASICa o dwie nowe komendy emulujące SCREENSWAP, znany z CPC 6128. Aby zapamiętać zawartość ekranu w pamięci RAM używamy komendy SCREENCOPY, a odtworzenie uzyskujemy stosując komendę: SCREENSHOW. Aby uaktywnić powyższe komendy należy:
a. wyzerować komputer
b. wczytać i uruchomić program RSX LOADER
c. usunąć linie programu RSX LOADER z pamięci np. komendą DELETE.

O możliwościach wykorzystania tych dodatkowych komend w programie KRESKA w następnym numerze.

JANEK

```
10 REM************************************
20 REM*              RSX LOADER          *
30 REM*  na podstawie PC WORLD 2/1987    *
40 REM************************************
50 IF PEEK(42750)=1 THEN MODE 2:PRINT "R
SX'y sa juz zaladowane...":END ELSE MEMO
RY 25999:wsk=0:FOR a=42750 TO 42816:READ
 a$:b=VAL("&"+a$):POKE a,b:wsk=wsk+b:NEX
T
60 IF wsk()6502 THEN MODE 2:PRINT "Blad
w DATA":END
70 CALL &A6FE:MODE 2:PRINT "RSX'y zostal
y zainicjalizowane"
80 DATA 01,08,A7,21,25,A7,C3,D1,BC,C9,10
,A7,C3,35,A7,C3,29,A7,53,43,52,45
90 DATA 45,4E,53,48,4F,D7,53,43,52,45,45
,4E,43,4F,50,D9,00,00,00,08,A7,11
100 DATA 90,65,21,00,C0,01,00,40,ED,B0,C
9,11,00,C0,21,90,65,01,00,40,ED,B0
110 DATA C9,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00,00,00,00,00,00
```

```
1990 CLS #2:PRINT #2," (O)KRAG (C)ZWOROK
AT (T)ROJKAT"
2000 LOCATE #2,2,2:PRINT #2,"(E)LIPSA
  (P)OWROT DO MENU "
2010 IF INKEY(34)=0 THEN w$="O": GOTO 20
80
2015 IF INKEY(58)=0 THEN w$="E": GOTO 20
80
2035 IF INKEY(51)=0 THEN 4000
2060 REM     KRESLENIE OKREGU I ELIPSY
2080 CLS #2:IF w$="O" THEN PRINT #2," WY
ZNACZ SRODEK OKREGU   LUB P"
2085 IF w$="E" THEN PRINT #2," WYZNACZ S
RODEK ELIPSY   LUB P"

2120 IF INKEY(F%)=0 AND w$="O" THEN cx%=
x%:cy%=y%:LOCATE #2,2,1:PRINT #2,"WYZNAC
Z PROMIEN OKREGU":FOR i=1 TO 150:NEXT i:
GOTO 2150
2125 IF INKEY(F%)=0 AND w$="E" THEN cx%=
x%:cy%=y%:LOCATE #2,2,1:PRINT #2,"WYZNAC
Z PIERWSZA OS ELIPSY   ":FOR i=1 TO 150:
NEXT i:GOTO 2150
2170 IF INKEY(F%)=0 AND w$="O" THEN ox%=
x%:oy%=y%:LOCATE #2,2,1:PRINT #2,"WYZNAC
Z SRODEK OKREGU   LUB P":GOTO 2200
2175 IF INKEY(F%)=0 AND w$="E" THEN ox%=
x%:oy%=y%:LOCATE #2,2,1:PRINT #2,"WYZNAC
Z DRUGA OS ELIPSY      ":PLOT ox%,oy%,1:
x%=cx%:y%=cy%:q%=0:MOVE cx%,cy%:GOTO 219
1
2191 GOSUB 1070:GOSUB 1210:GOSUB 1120:GO
SUB 1010:GOSUB 1070
2192 IF INKEY(F%)=0 THEN rx%=x%:ry%=y%:L
OCATE #2,2,1:PRINT #2,"WYZNACZ SRODEK EL
IPSY  LUB P":GOTO 2195
2193 IF q%=0 AND (x%()cx% OR y%()cy%) TH
EN PLOT cx%,cy%,1:MOVE x%,y%
2194 GOTO 2191
2195 d3%=cx%-rx%:d4%=cy%-ry%
2196 d3%=ABS(d3%):d4%=ABS(d4%)
2197 p=SQR((d3%^2)+(d4%^2))
2250 IF w$="O" THEN PLOT cx%+r*COS(a%),c
y%+r*SIN(a%),kolor
2255 IF w$="E" THEN PLOT cx%+r*COS(a%),c
y%+p*SIN(a%),kolor
```

Spectrum

```
 10 REM Program *KILIM*
 20 BORDER 0: LET n=1: LET a$="
ABABABABABABABABABABABABABABABAB
CDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCD
": LET b$="": FOR m=1 TO 11: LET
 b$=b$+a$: NEXT m: CLS
 30 FOR z=0 TO 7
 40 LET a=INT (RND*255)
 50 POKE 22432,a
 60 LET b=POINT (0,0)+2*POINT (
1,0)+4*POINT (2,0)+8*POINT (3,0)
+16*POINT (4,0)+32*POINT (5,0)+6
4*POINT (6,0)+128*POINT (7,0)
 70 POKE 65368+z,a
 80 POKE 65376+z,b
 90 POKE 65391-z,a
100 POKE 65399-z,b
110 NEXT z: REM AB CD = AB CD
120 PRINT #0;AT 1,15;n: LET n=n
+1: BEEP .02,a/8
130 PRINT AT 0,0;b$: GO TO 30
200 SAVE "*KILIM*" LINE 10
```

J. Rubinowicz

```
4000 REM********************************
4010 REM      KRESLENIE TROJKATA
4020 REM********************************
4025 CLS #2:PRINT #2," "+CHR$(244)+" (R)

OWNORAMIENNY   (D)OWOLNY";
4030 PRINT #2,"   (M)ENU KRESLENIE   (F)I
GURY";
4040 IF INKEY(38)=0 THEN 560
4050 IF INKEY(53)=0 THEN 1990
4060 IF INKEY(50)=0 THEN w$="R": GOTO 41
00
4080 IF INKEY(61)=0 THEN w$="D": GOTO 41
00
4090 GOTO 4040
4100 CLS #2
4110 PRINT #2," WYZNACZ PIERWSZY WIERZCH
OLEK ";
4120 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
4130 GOSUB 1720:GOSUB 1070:GOSUB 1210
4140 GOSUB 1010:GOSUB 1070:GOSUB 1120
4150 GOSUB 1010
4160 IF INKEY(F%)=0 THEN 4180
4170 GOTO 4130
4180 x1%=XPOS:y1%=YPOS:LOCATE #2,1,1:PRI
NT #2," WYZNACZ DRUGI WIERZCHOLEK    ";
4190 q%=0
4200 GOSUB 1720:GOSUB 1070:GOSUB 1210

4210 GOSUB 1010:GOSUB 1070:GOSUB 1120
4230 GOSUB 1010
4240 IF q%=0 THEN PLOT x1%,y1%,1:MOVE x,
y
4250 IF INKEY(F%)=0 THEN 4270
4260 GOTO 4200
4270 x2%=XPOS:y2%=YPOS
4275 IF w$="R" THEN LOCATE #2,1,1:PRINT
#2," WYZNACZ WYSOKOSC              ";

4280 IF w$="D" THEN LOCATE #2,1,1:PRINT
#2," WYZNACZ TRZECI WIERZCHOLEK   ";
4300 q%=0
4310 IF w$="R" THEN x4%=x1%+(x2%-x1%)/2:
y4%=y1%+(y2%-y1%)/2:MOVE x4%,y4%:x%=XPOS
:y%=YPOS
4320 GOSUB 1720:GOSUB 1070:GOSUB 1210

4330 GOSUB 1010:GOSUB 1070:GOSUB 1120

4340 GOSUB 1010
4345 IF q%=0 AND w$="D" THEN PLOT x2%,y2
%,1:MOVE x%,y%
4350 IF INKEY(F%)=0 THEN 4370
4360 GOTO 4320
4370 x3%=XPOS:y3%=YPOS
4380 MOVE x1%,y1%:DRAW x2%,y2%,kolor:DRA
W x3%,y3%,kolor:DRAW x1%,y1%,kolor
4390 GOTO 4000
```

* * *

W KLUBIE, KTÓRY BĘDZIE

Pierwszy mikrokomputer Spectrum Plus, pojawił się w Wyższej Szkole Oficerskiej Wojsk Łączności im. płk. B. Kowalskiego w październiku 1985 roku. Od siedmiu lat istniał już wówczas Ośrodek Obliczeniowy, który prócz usług informatycznych prowadził prace naukowo-badawcze dla potrzeb uczelni. Dzięki komputerowi „Odra 1305" utworzono całą sieć abonencką, z końcówkami, drukarkami w komendzie i cyklach nauczania. Na podstawie systemu opracowane były różne programy specjalistyczne i uniwersalne.

W miarę przybywania sprzętu, czyli mikrokomputerów, pracownikom ośrodka oraz podchorążym zamarzyło się powstanie klubu mikrokomputerowego. Pozostaje on nadal w sferze marzeń, choć już bardziej realnych. Klubu jako takiego nie ma, gdyż brak dla niego na razie pomieszczenia. Najprawdopodobniej już w tym roku znajdzie swoje lokum w Klubie Podchorążego. Tymczasem trwają poszukiwania nazwy dla klubu, który w świadomości sporej grupy zapaleńców już istnieje. Przy ośrodku powstało Koło Naukowe Informatyków, które wraz z funkcjonującym przy jednym z cyklów Kołem Naukowym Podchorążych Teleinformatyków skupia przyszłych członków klubu.

— Jest ich około czterdziestu — wyjaśnia kierownik ośrodka i zarazem opiekun komputerowej grupy, mjr **Tadeusz Szuszkiewicz**. — Połowa rekrutuje się spośród słuchaczy szkoły oficerskiej, druga połowa to podchorążowie SPR. Mamy możliwość korzystania z kilkunastu mikrokomputerów Spectrum, Unipolbrit 2086 i Amstrad 6128 M. Kilka spośród nich zawsze jest w ośrodku. Do tego dochodzi jeszcze kilka prywatnych mikrokomputerów, podchorążych, którzy nie potrafili rozstać się ze swoim sprzętem i zabrali go do szkoły.

Komputerowa „mania" nie okazała się słomianym ogniem, lecz zjawiskiem znacznie trwalszym. Od lat ma swoich wiernych zwolenników. Tu też wzbudza ogromne zainteresowanie. Podchorążowie podkreślają, że trudno dzisiaj oddzielić nowoczesną technikę od komputeryzacji. W programie zegrzyńskiej łącznościówki jest nauka posługi-

wania się mikrokomputerem i podstawowego programowania. Tę wiedzę można wzbogacić na klubowych spotkaniach.

Ważną rolę odgrywają tam słuchacze Szkoły Podchorążych Rezerwy. Są oni absolwentami renomowanych uczelni technicznych. Wielu z nich to informatycy lub cybernetycy. Oni organizują kursy języków, uczą zasad programowania i licznych sposobów wykorzystania mikrokomputera. Chętnych na te zajęcia jest tak dużo, że nie jest ich w stanie pomieścić sporych rozmiarów sala.

Podchorążowie SPR korzystają ze sprzętu zgromadzonego w Ośrodku Obliczeniowym i fachowej rady jego pracowników. Współpraca ta zresztą przeciąga się na dłuższy okres, trwa nawet już po przejściu podchorążych do cywila. Dobrymi tego przykładami są kontakty z informatykami Akademii Górniczo-Hutniczej czy Politechniki Gliwickiej. Do WOSWŁ dociera w ten sposób trudna do zdobycia literatura specjalistyczna, interesujące programy, a także sprawozdania z prowadzonych aktualnie prac naukowo-badawczych.

Etap zafascynowania grami komputerowymi szybko minął. To naturalna kolej rzeczy, że zaczyna się od rzeczy najprostszych, choć efektownych. Teraz podchorążowie rozwiązują już konkretne problemy, wyczytane w literaturze lub wynikłe z programu studiów. Za swoje największe osiągnięcie przyszli klubowicze uważają opracowanie egzaminatora z dowolnego przedmiotu. Za ten program zostali wyróżnieni przez ministra obrony narodowej. Poza tym większość czasu poświęcają na przygotowanie programów do zastosowania w wojskach łączności.

Sierż. pchor. WSO **Krzysztof Fąferek** jest właścicielem mikrokomputera Atari 800 XL. Mówi, że to jego wielkie hobby, któremu poświęca większość swego wolnego czasu. Jest w tej korzystnej sytuacji, że nie jest ograniczony czasem i po zamknięciu ośrodka ma swój sprzęt, który zresztą udostępnia kolegom.

— Najczęściej pracujemy nad konkretnymi problemami — stwierdza kpr. pchor. WSO, **Tomasz Szafranowski** — lecz w bardzo szerokim pojęciu. Rozwiązujemy bowiem skomplikowane kwestie techniczne, ale budujemy też pomoce naukowe do zdawania egzaminów, poczynając od regulaminów, a kończąc na taktyce. Oczywiście staramy się tworzyć coś nowego, indywidualnego, może nie mającego tak powszechnego, praktycznego zastosowania, ale przynoszącego za to najwięcej satysfakcji i nowych doświadczeń.

Absolwent Politechniki Warszawskiej, plut. pchor. SPR, **Jacek Kuśmierczyk** chwali sobie, że właśnie tutaj trafił do wojska. Jest informatykiem, ma więc szansę uzupełniać swoje wiadomości. Podczas studiów wiele się nauczył, został doskonale przygotowany teoretycznie, ale jeszcze dwa lata temu Politechnika nie dysponowała odpowiednim zapleczem sprzętowym. Tutaj więc weryfikuje swe umiejętności. Uczy kolegów, a przy okazji poznaje nowe typy mikrokomputerów.

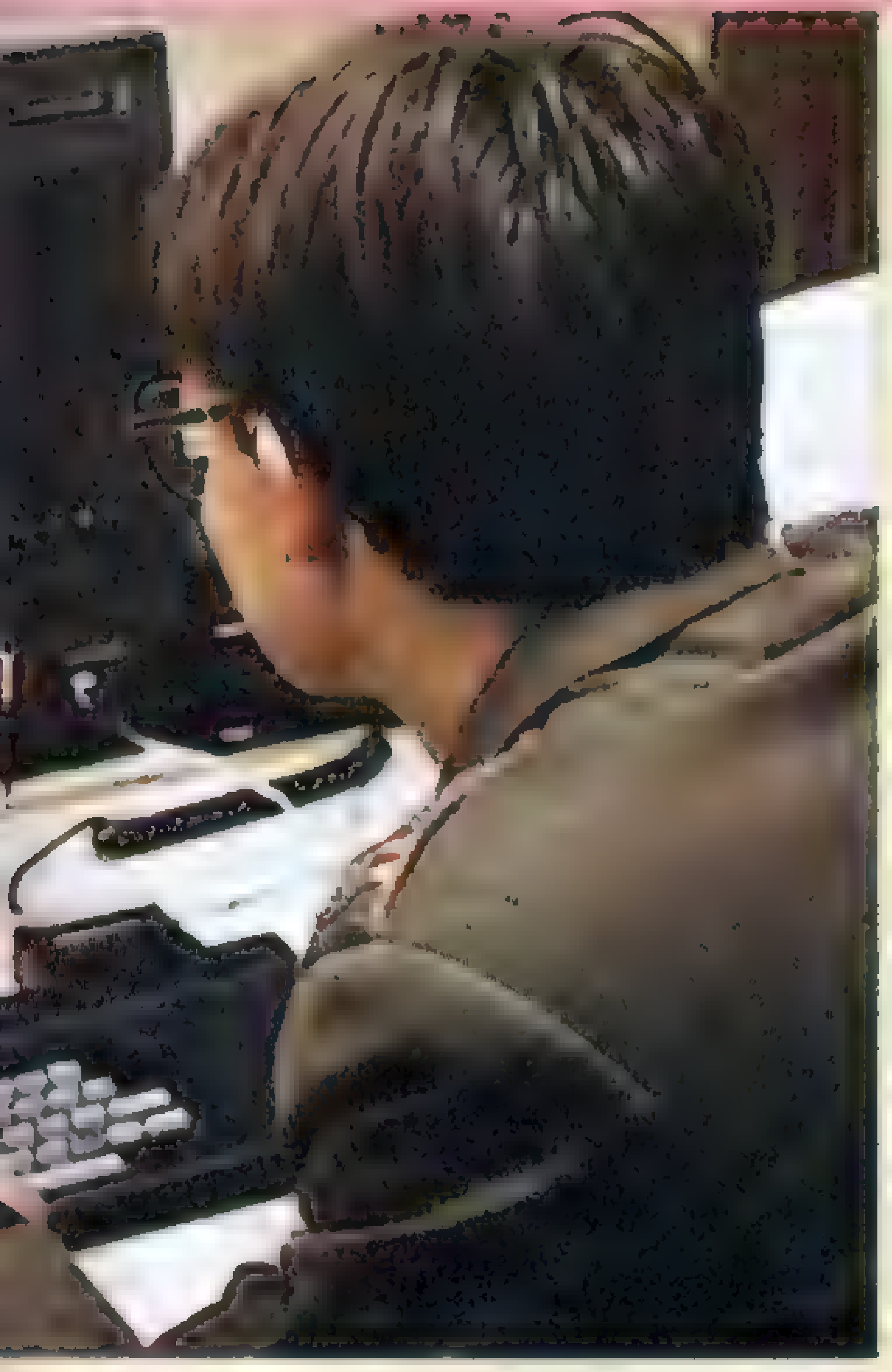
Widać, że ten nie istniejący jeszcze klub już działa. Ma ambicje oddziaływania nie tylko na słuchaczy szkoły, ale i na miejscową młodzież. Żeby tak było, niezbędny jest lokal i więcej sprzętu. Pomieszczenie wkrótce będzie, może znajdzie się i mecenas, który zwiększy stan posiadania o kilka mikrokomputerów.

J. RAJCH



W KLUB







BIE, KTÓRY BĘDZIE







Zmienne systemowe ROM Spectrum

Wskaźniki obszarów programu i zmiennych BASICa

LISTSP 23563/4
S—TOP 23660/1
E—PPC 23625/6
DEFADD 23563/4
DATADD 23639/40
DEST 23629/30
STRLEN 23666/7

Obszar, w którym przechowywana jest lista instrukcji do interpretacji, nazywą się obszarem programu. Ograniczają go wskaźniki **PROG** i **VARS**. W odpowiednim formacie może on być przekazany na ekran (lub drukarkę) po komendzie **LIST** (LLIST). Ciekawym zjawiskiem w Spectrum jest listing automatyczny. Instrukcje pojawiają się na ekranie po naciśnięciu ENTER bez potrzeby wczytania LIST. S—TOP trzyma tzw. automatyczny numer linii. E—PPC — numer linii podlegającej edycji po naciśnięciu EDIT. Jeśli E—PPC jest większe od S—TOP, to linia z kursorem programu wystąpi na górze ekranu, w przeciwnym razie na dole. LISTSP przechowuje na czas procesu wskaźnik stosu. Bezpośrednia zmiana zawartości E—PPC np.:

POKE 23625,100
POKE 23626,0

wywoła podobny efekt, co **LIST** 100, lecz oczywiście ekran nie zareaguje przewijaniem w celu pokazania linii z kursorem. Listing automatyczny, choć wygodny dla raczkujących, wydaje się jednak zbędny, czego potwierdzeniem jest nieobecność tego zjawiska w innych mikrokomputerach.

Dwie funkcje **Basic'a DEF FN** i **DATA** możemy umieścić w dowolnym miejscu programu, pozwalając interpreterowi na samodzielne ich odszukanie. Lepiej więc lokować je na początku, co przyśpiesza akcję. Są one związane ze zmiennymi systemowymi **DATADD** i **DEFADD**. W pierwszej z nich znajdziemy adres wskazujący miejsce, od którego będzie czytać dane instrukcja **READ**. Zwykle jest to adres przecinka separującego kolejne elementy w instrukcji DATA lub końca linii. Poniższy program zlicza dowolne dane umieszczone po słowie **DATA**:

```
 100 DEF FN p(a)=PEEK a +256*PEEK (a+1)
 110 LET elem=0
 120 RESTORE 200: REM numer linii
 130 LET ad= FN p(23639)+6
 140 IF PEEK ad<>34 THEN GOTO 160
 150 LET ad=ad+1: IF PEEK ad<>34 THEN G
OTO 150
 160 LET ad=ad+1+5*( PEEK ad=14)
 170 LET elem=elem+( PEEK ad=13 OR PEEK
ad=44)
 180 IF PEEK ad=13 THEN PRINT elem: STOP
 190 GOTO 140
 200 DATA SIN PI,"S",-3.14e-9, INT 1.1,
"""","""",PEEK 0,"koniec"
```

Niewątpliwym mankamentem Spectrum-Basic jest brak instrukcji pomocniczych np. do przenumerowywania linii, automatycznego ich generowania, niszczenia bloków programu itp., które użytkownikom innych komputerów ułatwiają życie. Niezbyt fortunne rozwiązanie klawiszy doprowadziło do ograniczenia liczby słów kluczowych, a co za tym idzie, do rezygnacji z wielu prostych w gruncie rzeczy funkcji.

Dla przykładu program kasowania bloków linii zajmuje tylko 45 bajtów. Wykorzystamy tu własność zmiennej **DEFADD** wskazującej na identyfikator pierwszego parametru wyrażenia **DEF FN** w momencie obliczania wartości funkcji. Określmy funkcję:

10 **DEF FN** d(p,o) = **USR** 62000

gdzie p jest pierwszą linią, zaś o ostatnią w bloku przeznaczonym do fizycznego zniszczenia. Oba parametry muszą być liczbami naturalnymi.

```
 10 ; ** DELETE '85
 20 ; GENS3M2 assembler
 30                ORG   62000
 40 DEFADD         EQU   23563
 50 PPC            EQU   23621
 60 RECL1          EQU   #19E5
 70 LINEAD         EQU   #196E
 80                LD    IX,(DEFADD)
 90                LD    L,(IX+4)
100                LD    H,(IX+5)
110                CALL  LINEAD
120                PUSH  HL
130                LD    L,(IX+12)
140                LD    H,(IX+13)
150                INC   HL
160                CALL  LINEAD
170                POP   DE
180                AND   A
190                SBC   HL,DE
200                RET   C
210                ADD   HL,DE
220                CALL  RECL1
230                LD    BC,(PPC)
240                INC   BC
250                BIT   7,B
260                RET   Z
270                LD    BC,10000
280                RET
```

Procedura jest dostosowana do współpracy z komendą GO TO, a więc kasowanie bloku powinno odbyć się po instrukcji np.

GO TO FN d (3000, 5000),
którą można użyć jako linię w programie, pozwalając nawet, aby skasowała samą siebie.

Opis tekstu źródłowego dla asemblera:

* *linie 80—170 — rejestr IX przechwytuje wskazanie* ***DEFADD****; numery linii podawane są do procedury* ***LINEAD****, która zwraca w* ***HL*** *ich adresy;*
* *linie 180—200 — sprawdzenie, czy pierwsza linia nie jest większa od ostatniej;*
* *linia 220 — fizyczne zniszczenie obszaru programu między adresami zawartymi w* ***HL*** *i* ***DE****;*
* *linie 230—260 — powrót, jeśli instrukcja odwołania jest linią w programie;*
* *linie 270—280 — powrót dla instrukcji wczytanej w trybie bezpośrednim (przyjmie ona postać* ***GO TO 10000****).*

Kod jest relokowalny

Następny problem dotyczy zapisu na nośniku tylko obszaru zmiennych BASIC. Najlepiej będzie pozbyć się obszaru programu, wykorzystując w tym celu określoną wcześniej funkcję FN d (p,o) lub krótszy kod:

```
10          ORG     62500
20 PROG     EQU     23635
30 VARS     EQU     23641
20          LD      HL, (VARS)
30          LD      DE, (PROG)
40          JP      6629
```

Po uruchomieniu tych 10 bajtów można zapisać na taśmie (dyskietce) wszystkie użyte w programie zmienne Basic'a przez:
SAVE *„zmienne”*,
a w odpowiednim momencie dołączyć je do programu (choćby w celu wymuszenia innych warunków jego działania przez: **MERGE** *„zmienne”*).

Spectrum od strony implementacji ma tylko dwa typy zmiennych: numeryczne rzeczywiste i tekstowe. Tablica liczb naturalnych (do 65535), która zajmowałaby przykładowo w pamięci 5KB jest w większości (ok. 3KB) nie wykorzystana. Brakuje też rozkazu, który usuwałby ją z pamięci, tworząc miejsce dla innej. Tablicę np. m (10, 100, 20, 10) można wyzerować przez ponowną jej deklarację, zredukować do minimalnych rozmiarów przez DIM m(1), ale fizycznie wyrugować jej z pamięci nie można. Chyba że nauczymy komputer...

Najpierw jednak musimy się dowiedzieć od niego, jak przechowuje zmienne w pamięci. Szczegółowe dane zawiera podręcznik firmowy, ale brakuje tu kodowania identyfikatora zmiennej.

bit	7 6 5	4 3 2 1 0
znaczenie	typ	litera

Typów szczegółowych jest sześć, przy czym typ 1 nie istnieje:
2) zmienna tekstowa (string);
3) zmienna numeryczna 0 1-literowym identyfikatorze;
4) tablice numeryczne (numeric array);
5) zmienna numeryczna o wieloliterowym identyfikatorze;
6) tablica tekstowa (string array);
7) zmienna numeryczna 1-literowa sterująca pętlą FOR-NEXT;

Teraz możemy przedstawić program ERASE, który służy do niszczenia tablic, ale po pewnych przeróbkach „zajmie się” każdym typem zmiennej.

```
 10 ; ** ERASE '86
 20 ;GENS3M2  assembler
 30               ORG   63000
 40 DEFADD        EQU   23563
 50 VARS          EQU   23627
 60 RECL          EQU   #19E5
 70 PARERR        EQU   #288B
 80 VARERR        EQU   #1C2E
 90 NEXT          EQU   #19B8
100 START         LD    HL,PARERR
110               PUSH  HL
120               LD    IX,(DEFADD)
130               LD    L,(IX+4)
140               LD    H,(IX+5)
150               LD    C,(HL)
160               SET   7,C
170               RES   5,C
180               RES   6,C
190               LD    A,(IX+7)
200               AND   A
210               RET   NZ
220               LD    A,(IX+6)
230               DEC   A
240               JR    Z,ERA
250               DEC   A
260               RET   NZ
270               INC   HL
280               LD    A,(HL)
290               CP    "x"
300               RET   NZ
310               SET   6,C
320 ERA           POP   HL
330               LD    HL,(VARS)
340 LOOP          PUSH  BC
350               CALL  NEXT
360               POP   BC
370               LD    A,(HL)
380               CP    128
390               JP    Z,VARERR
400               CP    C
410               EX    DE,HL
420               JP    Z,RECL
430               JR    LOOP
```

Opis:

* *linie 100—110 — umieszczenie na stosie adresu procedury błędu w przypadkach niewłaściwych danych w wyrażeniu* **DEF FN***;*
* *linie 120—320 — sprawdzenie poprawności podanych parametrów wejściowych oraz przygotowanie identyfikatora tablicy w rejestrze C (uwaga! w linii 290 powinien być w cudzysłowach znak dolara);*
* *linie 330—430 — przeszukanie obszaru zmiennych; w przypadku znalezienia identyfikatora zniszczenie tablicy, w przypadku przeciwnym wydruk raportu Variable not found;*

Kod jest również relokowalny. Do korzystania z niego potrzebne jest zdefiniowanie funkcji:
10 **DEF FN** e$(o$) = (**USR** 63000)

Zwróćmy uwagę, że będziemy przekazywać do procedury **ERASE** parametr w postaci zmiennej znakowej, czyli nazwę tablicy. Oto przykłady niszczenia tablic g oraz g$:

20 **IF VAL** (**FN** e$ („g")) **THEN**
30 **IF VAL** (**FN** e$ (gs")) **THEN**

DEST i **STRLEN** dotyczą zmiennych **BASIC**. Pierwsza wskazuje adres nazwy i wartości zmiennej w postaci zmiennoprzecinkowej (5 bajtów poprzedzonych znakiem kontrolnym.14), przy czym jeśli zmienna jest dopiero oznaczana, adres odnosi się do obszaru programu, jeśli zaś jej wartość jest modyfikowana, adres dotyczy obszaru zmiennych BASICa. **STRLEN** trzyma długość łańcucha tekstowego, jeśli w użyciu jest zmienna tekstowa. Zarówno **DEST**, jak **STRLEN** uczestniczą w tworzeniu +— bajtowej reprezentacji stringu, której używa kalkulator do obróbki tekstów w obszarze roboczym (**WORKSP**). W trakcie interpretacji **STRLEN** jest modyfikowane i na wyjściu oba bajty trzymają w zasadzie niezależne informacje. Pod 23666 znajdziemy pierwszą literę użytej zmiennej numerycznej lub nowo utworzonej zmiennej tekstowej. W przypadku przekształcania tej drugiej bajt zawiera liczbę znaków przed modyfikacją, co potwierdzi przykład:

```
10 LET a$="stringpierwotny"
20 PRINT LEN (a$)
30 LET a$=a$( TO 6)
40 PRINT PEEK 23666
```

Bajt 23667 podaje „utrudnioną" postać pierwszej litery wprowadzonej zmiennej, przy czym inny jest kod po komendzie LET, a inny po INPUT i READ. W wypadku wycinania tekstu (slicing) bajt ten zawiera zero. Za miesiąc o kanałach i strumieniach, ale bez wody...

Krzysztof MAMCARZ

W naszym komputerlandzie

Szaleństwo wybierania najpiękniejszej ogarnęło w końcu i komputerland. Sam Spektruś zapalił się najpóźniej, ale za to najgoręcej i z własnej pensji zafundował nagrodę, a nawet stanął na czele jury. Szarfę dla przyszłej Miss komputerlandu zrobiono z taśmy perforowanej.

Jury złożone z kompetentnych czynników postanowiło dopuścić do konkursu wszystkie komputerzyce, wychodząc z założenia, że wszystkie są urodziwe. Kiedy jednak do konkurencji zgłosiły się rzeczywiście wszystkie, jury znalazło się w niemałym kłopocie, bo wybrać trzeba było jedną.

Wśród plejady gwiazd uwagę zwróciły dwie piękności — niemłoda już, lecz nieźle utrzymana Odra i młodziutka, nieśmiała Mazovia.

Po prezentacji jury udało się na naradę, jak na prawdziwy kolektyw przystało i ustaliło kryteria oceny.

Po pierwsze: Miss komputerlandu powinna reprezentować wysokie walory osobiste, przy czym wymagane są wymiary 96—84—96.

Po drugie — Miss powinna być inteligentna, lecz na tyle, by nie przytłaczała członków jury, aby nie popadli w kompleksy.

Po trzecie: Miss powinna się umieć ubrać, żeby miała się z czego rozbierać.

Wreszcie po czwarte — Miss nie musi za dobrze władać polskim, angielskim, ani żadnym innym ludzkim językiem, bo jury nie lubi gadatliwych komputerzyc, szczególnie, gdy się przechwalają — która ładniejsza.

Niestety po długiej naradzie jury nie przyznało tytułu Miss komputerlandu, gdyż żadna z kandydatek nie spełniała czwartego warunku. Żadna nie chciała poddać się zakneblowaniu na czas pozostawania Miss...

Podglądał:
Eugeniusz MLECZAK

Spektrum

```
  30 BORDER 0: LET s=0: LET w=0
  40 FOR i=0 TO 7: FOR p=0 TO i
  50 BEEP .05,2*i: PRINT AT w+2,
s;i;"+";p; BRIGHT 1; INK i; PAPE
R p;"UUUUUUUUUUUU"
  60 LET w=w+1: IF w=16 THEN LET
 w=0: LET s=16
  70 NEXT p: NEXT i
  80 PAUSE 300: CLS : GO TO 20
  90 SAVE "36 kolorow" LINE 10:
PRINT '"Verify !": VERIFY "": GO
 TO 10
```

J. Rubinowicz

PASCAL (5)

Jedną z podstawowych dziedzin zastosowań komputerów jest przetwarzanie danych. Wiąże się ono z gromadzeniem i przechowywaniem informacji o wielu obiektach, z których każdy może być opisywany wieloma cechami. W mniejszej części omówiono podstawowe właściwości tych konstrukcji języka PASCAL, które umożliwiają opis obiektów charakteryzowanych przez wiele cech, gromadzenie informacji o wielu obiektach (ich ilość nie musi być z góry określona) i przechowywanie jej w pamięciach zewnętrznych.

1. Rekordy

Rekordy umożliwiają opisywanie obiektów, które można scharakteryzować wieloma różnymi cechami. Przykładem takiego obiektu może być opis mikrokomputera dostępnego na rynku krajowym. Przyjmując dla uproszczenia, że zawiera on 20-znakową nazwę mikrokomputera, 12-znakową nazwę firmy produkującej mikrokomputery oraz minimalną cenę mikrokomputera (w tys. zł), „rynkowy" mikrokomputer można opisać następująco:

TYPE *opmikrok* = **RECORD**
naznik: string [20];
firma: string [12];
cenamin: integer
END;

gdzie:
opmikrok — nazwa definiowanego rekordu,
naznik, firma, cenamin — nazwy składowych rekordu zwanych polami

RECORD—END — para słów kluczowych, oznaczająca definicję rekordu,

string — typ standardowy, umożliwiający definiowanie łańcuchów znaków (stałe typu „string" są już Czytelnikowi znane z instrukcji **WRITE**).

Powyższą definicja umożliwia deklarowanie zmiennych np. według formuły

VAR r1, r2,...rn: opmikrok;

z których każda opisuje inny mikrokomputer.

Sposoby odwoływania się do pól zmiennej rekordowej przez tzw. nazwy kwalifikowane wyjaśnia przykład 14, w którym poszczególnym polom nadaje się odpowiednie wartości.

Przykład 14

```
PROGRAM prog14;
  {zakladanie zbioru KOMPUTER.DTA
   na dyskietce}
 TYPE
  opmikrok=RECORD {opis mikrokomputera }
           nazmk:string[20];
           firma:string[12];
           cenamin:integer
          END;
  zbopmk=FILE OF opmikrok;
                 {definicja pliku}
 VAR
  zbior:zbopmk;  {deklaracja pliku}
  rek:opmikrok;
    {zmienna opisujaca mikrokomputer}
  ilop,   {liczba wprowadzanych opisow}
  i: integer;
 BEGIN
  READLN(ilop);
  ASSIGN(zbior,'komputer.dta');
    {dowiazanie fizycznej reprezentacji
    pliku-zbioru o nazwie KOMPUTER.DTA}
  REWRITE(zbior);
     {przygotowanie pliku do zapisu}
  FOR i:=1 TO ilop DO
   WITH rek DO
    BEGIN
     READLN(nazmk,firma,cenamin);
       {wczytanie opisu mikrokomputera
        z monitora}
     WRITE(zbior,rek)
        {zapisanie opisu mikrokomputera
         do pliku}
    END;
  CLOSE(zbior)     {zamkniecie pliku}
 END.
```

Po wykonaniu przedstawionej sekwencji wykonanie instrukcji

X:=rek. firma;

gdzie „X" jest typu „string [12]" spowoduje przypisanie zmiennej X wartości „SINCLAIR". Zauważmy, że łańcuch znaków o długości krótszej niż podana w definicji jest uzupełniany spacjami z prawej strony.

Przedstawiony sposób odwoływania się do pól zmiennej rekordowej może być uciążliwy (zwłaszcza, gdy jest ich dużo lub są to pola typu strukturalnego), gdyż każde pole musi być poprzedzone nazwą zmiennej.

Wygodniejszy dostęp do pól zmiennej rekordowej umożliwia instrukcja WITH, zwana instrukcją wiążącą lub instrukcją dostępu. Pozwala ona na odwoływanie się pól zmiennej rekordowej bezpośrednio przez nazwy pól, przy jednokrotnym podaniu nazwy zmiennej rekordowej. Wykorzystując instrukcję WITH fragment programu z przykładu 14 można zmodyfikować następująco:

Przykład 15

2. Pliki

Przechowywanie programów w pamięci zewnętrznej pozwala na ich wielokrotne wykorzystanie. Również dane, które podlegają przetwarzaniu co pewien czas (nieraz przez różne programy), muszą być przechowywane w pamięci zewnętrznej. Przykładem takich danych może być zarówno wykaz odbiorców energii elektrycznej, dla których nalicza się okresowo wysokość opłat oraz progam źródłowy, przetwarzany w zależności od potrzeb przez edytor lub kompilator.

W PASCALu istnieje możliwość korzystania z pamięci zewnętrznej przez tzw. pliki. Plik, podobnie jak tablica jest ciągiem elementów tego samego typu.

Od tablicy różni się metodą dostępu do elementów (w danej chwili może być dostępny tylko jeden element pliku oraz typu) i tym, że liczba elementów nie musi być z góry określona i może się zmieniać w trakcie wykonywania programu. Po zakończeniu działania programu informacje zgromadzone w pliku pozostają w pamięci zewnętrznej i mogą być wykorzystane przez ten sam lub inne programy. Z punktu widzenia fizycznej reprezentacji plik jest wyróżnionym obszarem pamięci zewnętrznej — zbiorem o zadanej nazwie.

Sposób zakładania na dyskietce zbioru podanych wcześniej opisów mikrokomputerowych ilustruje przykład 16.

Przykład 16

```
PROCEDURE wydruk(VAR zb:zbopmk;
                 naz:nazwazb);
  {procedura wyprowadzajaca na
   monitor zawartosc zbioru
   o zadanej nazwie}
 VAR
  rek:opmikrok;
 BEGIN
  ASSIGN(zb,naz);
  RESET(zb);
    {przygotowanie pliku do
     odczytu}
  WHILE NOT EOF(zb) DO
   BEGIN
    READ(zb,rek);
     {wczytanie kolejnego opisu
      mikrokomputera z pliku}
    WITH rek DO
     WRITELN(' ',nazmk,' ',firma,' ',
           cenamin)
   END;
  CLOSE(zb)
 END;
```

W przedstawionym programie wykorzystano procedury standardowe **ASSIGN, REWRITE, WRITE** i **CLOSE.**

ASSIGN dowiązuje do pliku fizyczny zbiór o podanej nazwie. Po wyniku procedury ASSIGN wszystkie operacje na pliku są równoważne działaniu na jego fizycznej reprezentacji. Nazwa zbioru jest łańcuchem znaków o postaci przyjętej dla zbiorów roboczych edytora.

REWRITE tworzy na dyskietce fizyczny zbiór o nazwie podanej w procedurze ASSIGN i udostępnia do zapisu pierwszy element pliku. Jeżeli zbiór o podanej nazwie już istnieje, to jest on kasowany. Po wykonaniu REWRITE plik jest pusty, tzn. nie zawiera żadnych elementów.

WRITE — dopisuje do pliku element umieszczony w zmiennej, podanej jako parametr procedury WRITE. Po zapisaniu danego elementu udostępniany jest kolejny element pliku.

CLOSE — zamyka plik. Procedura ta powinna być wykonana po zakończeniu działań na pliku.

Wprowadzając dane do programu należy pamiętać, że łańcuchy znaków muszą być łańcuchami o długości podanej w opisie rekordu (spacja też jest znakiem) oraz że kolejne łańcuchy nie są oddzielone od siebie żadnym separatorem. Program wykonano dla następujących danych:

ZX.Spectrum.Plus....SINCLAIR....110
ZX.Spectrum.48kB....SINCLAIR....75
130XE..............ATARI.......130
C-128..............COMMODORE...285
gdzie: . — oznacza spację.

Zawartość założonego zbioru można wyprowadzić na monitor wywołując procedurę „wydruk" podaną w przykładzie 17. W wywołaniu procedury parametr „" należy zastąpić łańcuchem znaków komputer dta:

Przykład 17

```
PROGRAM prog15;
  (wyprowadzenie zawartosci zbioru
   KOMPUTER.DTA na monitor)
 TYPE
  opsikrok=RECORD
            nazsk:string[20];
            firma:string[12];
            cenamin:integer
           END;
  zbopak=FILE OF opsikrok;
  nazwazb=string[12];
 VAR
  zbior:zbopak;
  nazwa:nazwazb;
    (nazwa wyprowadzanego zbioru)
 PROCEDURE wydruk(VAR zb:zbopak;
                  naz:nazwazb);
  (procedura wyprowadzajaca na
   monitor zawartosc zbioru
   o zadanej nazwie)
  VAR
   rek:opsikrok;
  BEGIN
   .
   .
   .
  END;
 PROCEDURE podkresl;
  BEGIN
   WRITELN('==============================',
           '================');
  END;
 BEGIN
  READ(nazwa);
  CLRSCR;(wykasowanie zawartosci ekranu)
  WRITELN;
  podkresl;
  WRITELN(' ':4,'NAZWA KOMPUTERA',' ':6,
          'FIRMA',' ':7,'CENA$');
  podkresl;
  wydruk(zbior,nazwa);
  podkresl;
  WRITELN('  $ cena minimalna w tys zl')
 END.
```

W procedurze „wydruk" poza już omówionymi procedurami standardowymi, wykorzystano procedury **RESET** i **READ**.

RESET udostępnia do przetwarzania (odczyt lub zapis) pierwszy element istniejącego pliku. Jeżeli brak jest fizycznej reprezentacji pliku, to wystąpi błąd wykonania.

READ odczytuje element pliku i umieszcza go w podanej zmiennej określonego typu. Po odczytaniu danego elementu, do przetwarzania zostaje udostępniony kolejny element pliku. Sprawdzenie czy wystąpił koniec pliku wykonuje funkcja standardowa EOF.

Program wyprowadzania zawartości założonego zbioru zamieszczono w przykładzie 17.

Przykład 17

Po wykonaniu programu „progu 15' uzyskano następujące wyniki:

NAZWA KOMPUTERA	FIRMA	CENA$
ZX Spectrum Plus	SINCLAIR	110
ZX Spectrum 48kB	SINCLAIR	75
130XE	ATARI	130
C-128	COMMODORE	285

$ cena minimalna w tys zl

Zawartość zbioru może być zmieniana m.in. przez dopisanie nowych elementów lub skasowanie już istniejących, wybranych elementów.

Program dopisywania nowych elementów na koniec założonego zbioru (tylko takie jest dopuszczalne) zamieszczono w przykładzie 18.

Przykład 18

```
PROGRAM prog16;
  (dopisanie rekordow na koniec
   zbioru KOMPUTER.DTA)
 TYPE
  opsikrok=RECORD
            nazsk:string[20];
            firma:string[12];
            cenamin:integer
           END;
  zbopak=FILE OF opsikrok;
 VAR
  zbior:zbopak;
  rek:opsikrok;
  ilop,
  i: integer;
 BEGIN
  READLN(ilop);
  ASSIGN(zbior,'komputer.dta');
  RESET(zbior);
  SEEK(zbior,FILESIZE(zbior));
   (przewiniecie pliku na koniec)
  FOR i:=1 TO ilop DO
   WITH rek DO
    BEGIN
     READLN(nazsk,firma,cenamin);
     WRITE(zbior,rek)
    END;
  CLOSE(zbior)
 END.
```

Przewinięcie pliku na koniec umożliwia procedura standardowa **SEEK** (parametr **FILESIZE** jest funkcją standardową obliczającą ilość elementów istniejącego pliku).

Po dopisaniu dla założonego zbioru elementu

800 XL......ATARI...70

zawartość zbioru była następująca (ponownie wykonano program „prog

NAZWA KOMPUTERA	FIRMA	CENA$
ZX Spectrum Plus	SINCLAIR	110
ZX Spectrum 48kB	SINCLAIR	75
130XE	ATARI	130
C-128	COMMODORE	285
800XL	ATARI	70

$ cena minimalna w tys zl

Prosty program kasowania wybranych elementów założonego zbioru zamieszczono w przykładzie 19.

Przykład 19

```
PROGRAM prog17;
   {kasowanie rekordow ze zbioru
    KOMPUTER.DTA}
 TYPE
  opmikrok=RECORD
            nazmk:string[20];
            firma:string[12];
            cenamin:integer
           END;
  zbopmk=FILE OF opmikrok;
 VAR
  zbior:zbopmk;
  nazwamk:string[20];
  ilopkas,
  i: integer;
 PROCEDURE kasrek;
  VAR
   zbpom:zbopmk;
   rek:opmikrok;
  BEGIN
   ASSIGN(zbior,'komputer.dta');
   RESET(zbior);
   ASSIGN(zbpom,'pom.dta');
   REWRITE(zbpom);
   WHILE NOT EOF(zbior) DO
    BEGIN
     READ(zbior,rek);
     IF rek.nazmk<>nazwamk THEN
              WRITE(zbpom,rek)
    END;
   CLOSE(zbior);
   CLOSE(zbpom);
   ERASE(zbior);
    {skasowanie zbioru KOMPUTER.DTA}
   RENAME(zbpom,'komputer.dta')
    {zmiana nazwy zbioru POM.DTA na
     KOMPUTER.DTA}
  END;
BEGIN
 READLN(ilopkas);
 FOR i:=1 TO ilopkas DO
  BEGIN
   READLN(nazwamk);
   kasrek
  END
END.
```

Przyjęto tutaj, że skasowanie nawet jednego elementu z pliku wymaga przejrzenia całego pliku. Ponieważ nie jest możliwe skasowanie elementu „ze środka" pliku, wykorzystuje się plik pomocniczy (pom, dla).

Elementy danego pliku są czytane kolejno i te z nich, które nie podlegają kasowaniu są zapisywane do pliku pomocniczego. Po przeglądnięciu całego pliku jest on kasowany (ERASE) a plik pomocniczy zostaje przemianowany na poprzednią nazwę pliku (RENAME). Identyfikatorem określającym, który element pliku ma być kasowany jest tutaj nazwa mikrokomputera

Po skasowaniu opisu mikrokomputera 130 XE zawartość zbioru komputer. dta była następująca:

NAZWA KOMPUTERA	FIRMA	CENA*
ZX Spectrum Plus	SINCLAIR	110
ZX Spectrum 48kB	SINCLAIR	75
C-128	COMMODORE	285
800XL	ATARI	70

* cena minimalna w tys zl

3. Zadania

Zadanie 12
Napisać program zakładania zbioru opisów pozycji domowej biblioteczki.

Zadanie 13
Napisać program porządkowania założonego zbioru alfabetycznie według nazwiska autora.
Uwaga: porządkowanie wykonać w pamięci operacyjnej. Można zmodyfikować poznaną już procedurę porządkowania ciągu.

Zadanie 14
Napisać program umożliwiający aktualizację zbioru opisów pozycji domowej biblioteczki uwzględniający dopisywanie lub kasowanie opisów.

Zadanie 15
Napisz program udostępniający informacje o danej pozycji domowej biblioteczki.

S. ROZMUS

PŁATEK KOCH'a

Każdy bok trójkąta równobocznego jest podzielony na trzy równe części i zamieniany na następującą łamaną:

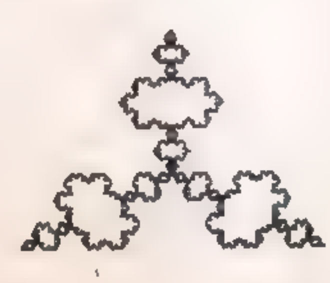


Każdy z odcinków łamanej ulega takiemu samemu przekształceniu. Operacji dokonuje procedura ZAMIANA metodą rekurencyjną. Proces kończy się w momencie, gdy długość boku jest odpowiednio mała (stała MINBOK).

Proponujemy również modyfikację procedury polegającą na wprowadzeniu zmiennego kąta. Zastąp parametr aktualny przy wywołaniu procedur LEFT i RIGHT zmienną typu REAL (zastanów się dlaczego REAL, a nie INTEGER), którą zadeklaruj i dopisz instrukcje jej wprowadzania.

Płatek Koch'a uzyskujemy dla kąta = 60 stopni, a modyfikacja pozwala otrzymać inne niezwykłe wzory.

```
 10 PROGRAM platek;
 20
 30 (********************************)
 40 (** PROGRAM rysuje platek      **)
 50 (**       sniegu Kocha         **)
 60 (**                            **)
 70 (** NAUKOWE KOLO PODCHORAZYCH  **)
 80 (**          WSOWL 1987        **)
 90 (********************************)
100
110 CONST minbok=5;
120 VAR
130  bok:real;
140  i:integer;
150
160 (********************************)
170 (**  tu nalezy wstawic pakiet  **)
180 (**  procedur grafiki zolwia   **)
190 (********************************)
200
210 PROCEDURE zamiana (bok:real);
220   BEGIN
230     IF bok>minbok THEN
240       BEGIN
250         zamiana(bok/3);
260         left(60);
270         zamiana(bok/3);
280         right(120);
290         zamiana(bok/3);
300         left(60);
310         zamiana(bok/3);
320       END
330     ELSE fwd(bok)
340   END; (* zamiana *)
350
360   BEGIN
370     write('bok=');
380     read(bok);
390     page;
400     turtle;
410     left(60);
420     xcor:=320-bok*sqrt(3)/3;
430     ycor:=200-bok*sqrt(3)/6;
440     FOR i:=1 TO 3 DO
450       BEGIN
460         zamiana(bok);
470         right(120);
480       END;
490   END.
```



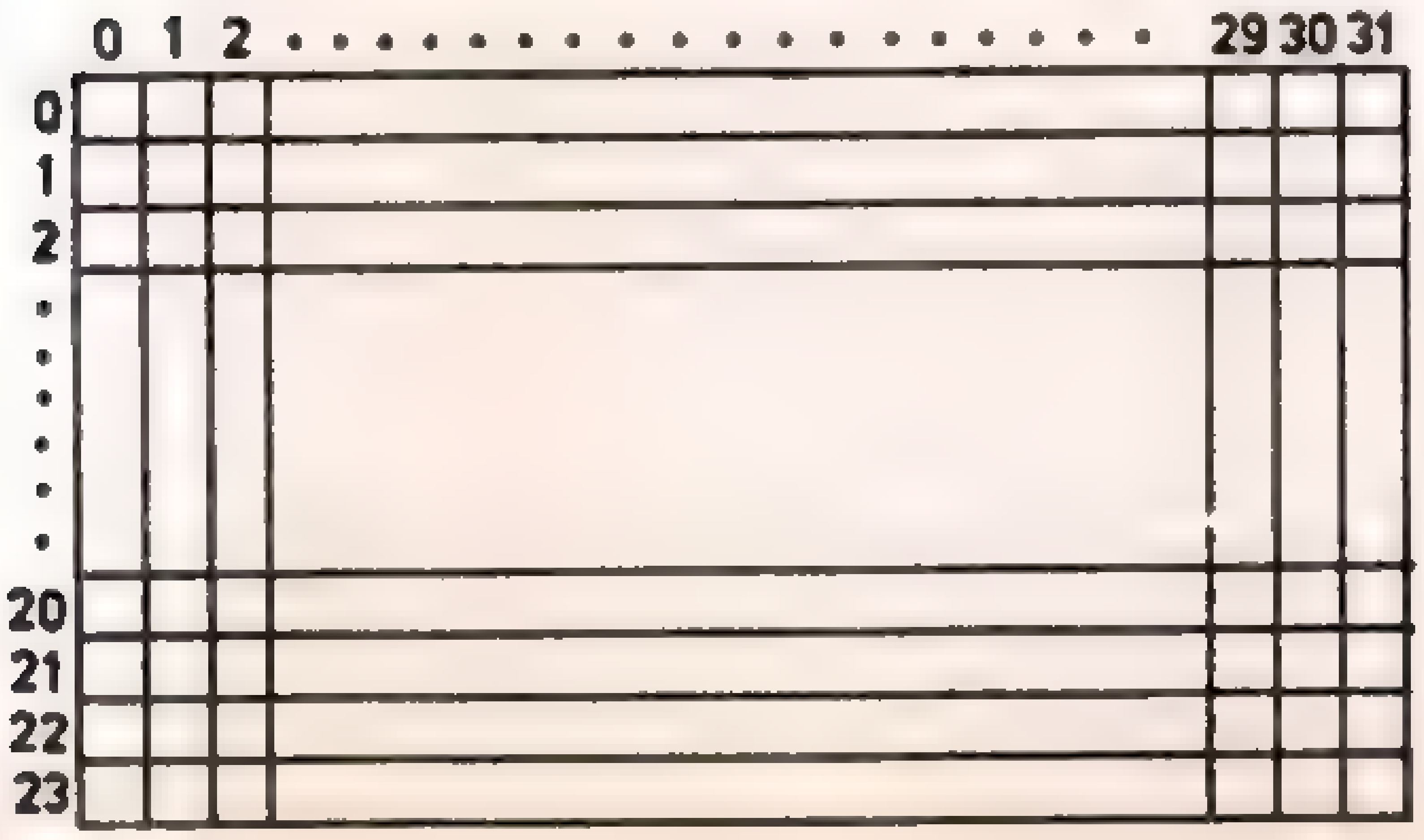
BASIC (7)

Podstawy programowania

Kontynuujemy wykład z programowania w języku BASIC na mikrokomputer ZX Spectrum. W wykładzie tym podajemy podstawowe wiadomości o grafice i kolorach wraz z omówieniem odpowiednich instrukcji BASICa dla ZX Spectrum.
Wykład siódmy:
grafiki ZX Spectrum; instrukcje PRINT AT n, l; PRINT TAB 1;
grafika: instrukcje PLOT, DRAW, CIRCLE

W wykładzie pierwszym omawialiśmy ekran telewizyjny (gdy ZX Spectrum podłączone jest do telewizora) lub ekran monitora (gdy podłączone jest do monitora). Przyjmijmy (jak w poprzednich wykładach) nazwę ekran telewizyjny. Powtórzmy to, co było powiedziane w pierwszym wykładzie. Ekran telewizyjny zawiera 24 linie po 32 znaki i jest podzielony na dwie części w poziomie i w pionie. Linie i znaki numerowane są od zera. Górna część ekranu zawiera 22 linie (0—21), wyświetlany jest na niej program albo wyniki i wykorzystywana jest w grafice. Dolna część ekranu jest wykorzystywana do wprowadzania rozkazów, wierszy programu i danych wejściowych, a także do wyświetlania komunikatów (zawiera ona dwie linie o numerach 22 i 23).

Narysujmy teraz schemat naszego ekranu w postaci:



Położenie każdego znaku na ekranie określamy przez podanie numeru linii (wiersza) i kolumny (innymi słowy współrzędne znaku).

Zajmiemy się teraz rysowaniem obrazków, wykorzystując symbole graficzne („grafiki") ZX Spectrum. **Symbole te wprowadzamy z klawiatury po uprzednim przejściu na tryb G** (kursor G — otrzymujemy naciskając jednocześnie klawisze CAPS SHIFT i 9). Naciśnięcie (gdy jest kursor G) klawiszy 1—8 spowoduje wypisanie na ekranie odpowiedniego symbolu graficznego w takiej postaci, w jakiej są narysowane na klawiszach, a z klawiszem SYMBOL SHIFT (sh) — odwrotnie tzn. kolor czarny stanie się biały, a kolor biały czarny. Poniżej podajemy szesnaście symboli graficznych (litera G — oznacza kursor G, sh — SYMBOL SHIFT):

symbol	sposób użycia	symbol	sposób użycia
□	G 8	█	G sh 8
▝	G 1	▙	G sh 1
▘	G 2	▟	G sh 2
▀	G 3	▄	G sh 3
▗	G 4	▛	G sh 4
▐	G 5	▌	G sh 5
▚	G 6	▞	G sh 6
▛	G 7	▗	G sh 7

Każdy znak zajmuje jedną pozycję na ekranie. Do wpisania takiego znaku w dowolne miejsce ekranu używana jest najczęściej instrukcja **PRINT** połączona z **AT**, postaci:

PRINT AT *n, l; „znak"*

gdzie: *n* — numer linii (0÷21)
l — numer kolumny (÷31)
znak — mogą to być znaki, teksty, a bez cudzysłowu nazwy zmiennych prostych, wyrażenia arytmetyczne itp.

AT *n, l* — przesuwa pozycję wydruku na wskazane miejsce ekranu (o współrzędnych (*n, l*).

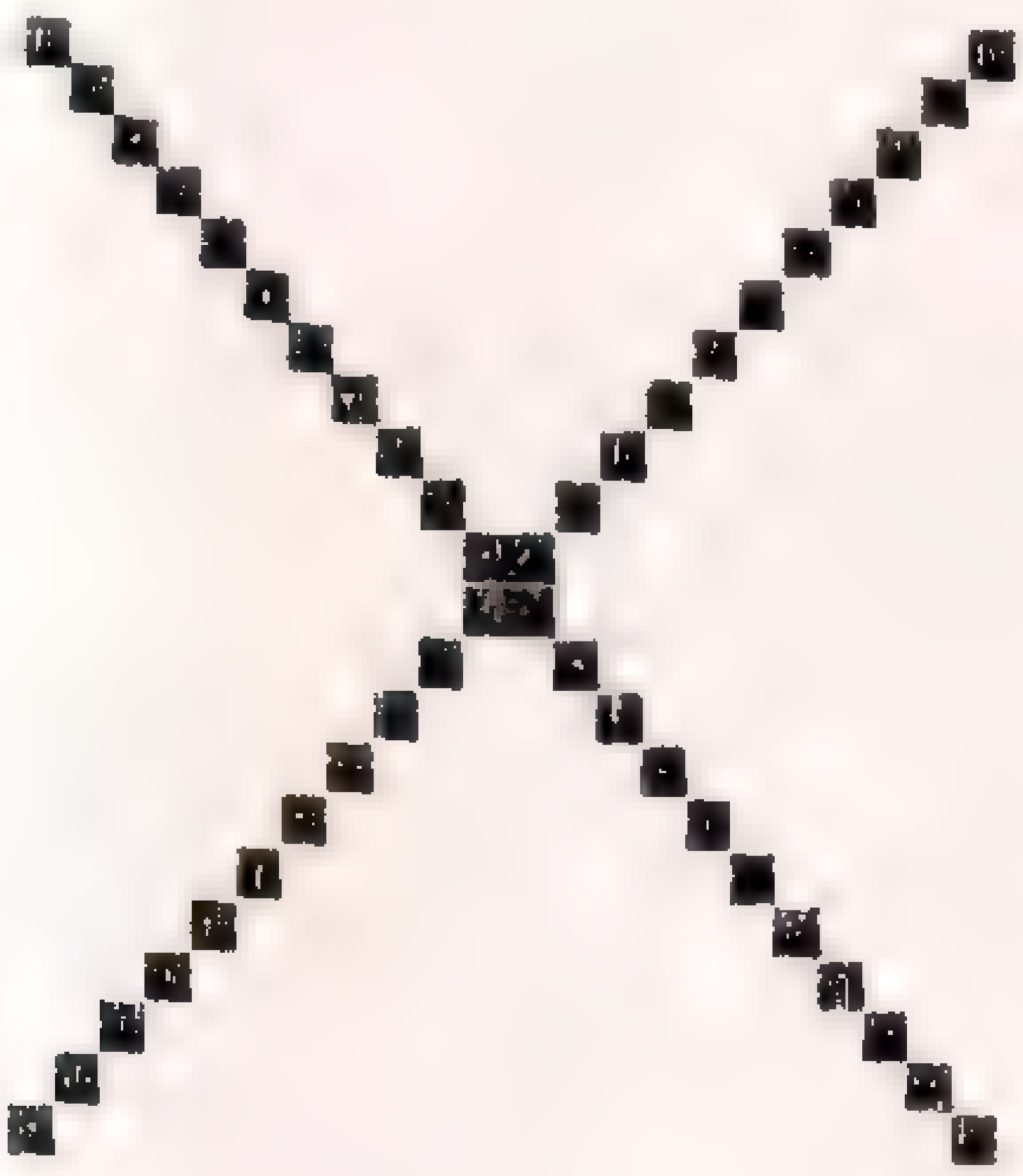
Po **AT** *n, l;* można pisać wszystko to, co było już omawiane w PRINT (średnik jest obowiązkowy).

Przykład 6.1.
W kwadracie o boku równym 22 znaki narysować przekątne używając symbolu „█"

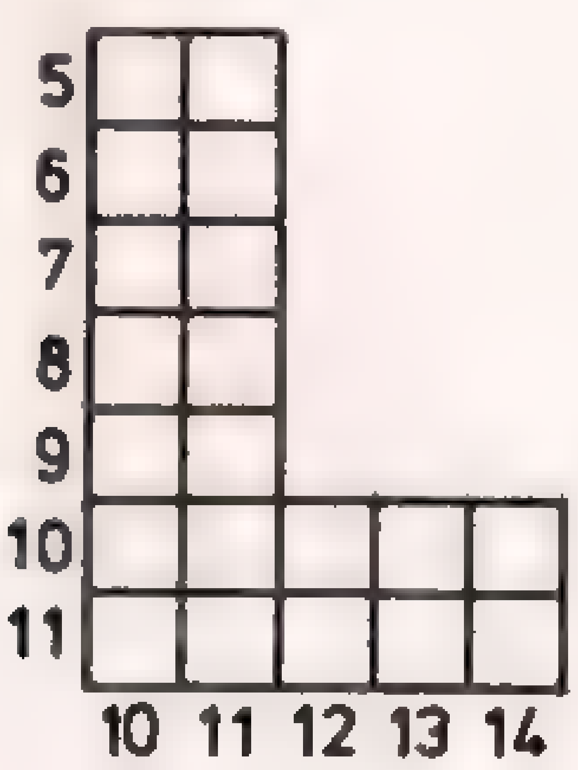
```
10 REM Przyklad 6.1
20 FOR i=0 TO 21
30 FOR j=0 TO 21
40 IF i=j THEN PRINT AT i,j,"■

45 IF i=21-j THEN PRINT AT i;j
"■"
50 NEXT j
60 NEXT i
```

Wprowadźmy i wykonajmy ten program. Otrzymane wyniki na ekranie są postaci:



Przykład 6.2.
Zaczynając od 5 wiersza i 10 kolumny wypisać dużą literę L, używając symbolu „■".
a) w pierwszym kroku „planujemy" obraz:



b) program:

```
 10 REM Przyklad 6.2
 20 REM rysowanie w liniach 5-9
 30 FOR i=5 TO 9
 40 PRINT AT i,10;"■■"
 50 NEXT i
 60 REM rysowanie w liniach 10,
11
 70 FOR j=10 TO 14
 80 PRINT AT 10,j;"■"
 90 PRINT AT 11,j;"■"
100 NEXT j
```

Wprowadźmy i wykonajmy ten program. Otrzymane wyniki na ekranie są postaci:



Przykład 6.3.
Dany jest program:

```
10 REM Przyklad 6.3"
20 PRINT AT 7,15; " ■ "
30 PRINT AT 8,15; "■■■"
40 PRINT AT 9,15; "■■■"
50 PRINT AT 10,15;" ■■ "
60 PRINT AT 11,15;" ■■ "
70 PRINT AT 12,15;"■■■"
```

Wprowadzając ten program, używamy symboli graficznych: □, ◨, ◳. Wprowadźmy i wykonajmy ten program. Otrzymane wyniki na ekranie są postaci:



Nauczyliśmy się przesuwać wydruk we wskazane miejsce ekranu, używając AT n, l. Za pomocą AT n, l umiemy np. pisać w dolnej części ekranu, a potem w środku albo w górnej części ekranu, nawet w miejscach uprzednio zapisanych.

W instrukcji **PRINT** można zastosować **TAB** *l* (*l* — nr kolumny) **TAB** *l* — dopisuje spacje, tak aby przesunąć miejsce wydruku do wymienionej kolumny, zostając w tej samej linii. Zdarza się, że trzeba przesunąć się do następnej linii i także do wskazanej kolumny. Komputer zmniejsza numer kolumny „modulo 32" (dzieli przez 32 i bierze resztę), więc TAB 33 oznacza to samo co TAB 1 (w następnej linii). Np. wprowadźmy i wykonajmy program ilustrujący działanie **TAB** *l*:

10 **FOR** n = 0 **TO** 20
20 **PRINT TAB** 6 * n; n;
30 **NEXT** n

Uwaga: używając w instrukcji PRINT: AT n, l lub TAB l zawsze piszemy średnik, tzn.
PRINT AT n, l; ....
lub PRINT TAB 1; ....

Sądzę, że podane zasady i przykłady powinny wystarczyć do samodzielnej zabawy w „grafikę" ZX Spectrum.

Teraz zajmiemy się rysowaniem obrazków na ZX Spectrum.

Część ekranu, którą możemy używać do tego celu, zawiera 22 linie i 32 kolumny, czyli 22 * 32 = 704 pozycje znakowe. Każda pozycja znakowa utworzona jest przez kwadracik 8 na 8 punktów. Punkty te nazwano pixelami (mamy 256 × 176 = 45 056 pixeli). Pixel jest określony przez dwie współrzędne:

— pierwszą x, która określa odległość pixela od lewego dolnego rogu ekranu (w prawo),
— drugą y, która określa wysokość pixela nad linią 22.

Na rysunku poniżej podane są współrzędne rogów i środka ekranu.



Współrzędne pixeli są inne niż linii i kolumn w AT. Punkty na ekranie rysujemy instrukcją postaci:

PLOT *x*, *y* (0 ≤ *x* ≤ 255, 0 ≤ *y* ≤175)

Punkty rysowane są w kolorze atramentu.

Linie proste na ekranie rysujemy instrukcją postaci:

DRAW *x*, *y* (—255 ≤ *x* ≤ 255, —175 ≤ *y* ≤175)

Linia rysowana przez instrukcję **DRAW** zaczyna się w miejscu, gdzie ostatnio skończyło się rysowanie instrukcji **PLOT**, **DRAW** lub **CIRCLE** (zwane też pozycją PLOT-u), kończy się w miejscu odległym o x pixeli w prawo i y pixeli w górę. Każda instrukcja rysowania przesuwa pozycję PLOT-u w miejsce, gdzie skończyło się rysowanie. Instrukcja DRAW nie określa punktu początkowego, tylko kierunek i długość.

Przykład 6.4.

Narysować dwie linie:

linię **A**: punkt początkowy o współrzędnych (56, 22)
punkt końcowy o współrzędnych (205, 106)

linię **B**: punkt początkowy o współrzędnych (226, 152)
punkt końcowy o współrzędnych (34, 22)

obliczamy współrzędne instrukcji DRAW:

linia **A**: 205 — 56 = 149 (x) 106 — 22 = 84 (y)
linia **B**: 34 — 226 = —192 (x) 22 — 152 = —130 (y)

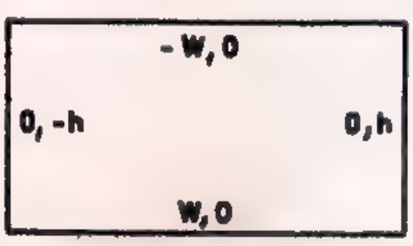
```
10 REM Przyklad 6.4
15 REM linia A
20 PLOT 56,22
30 DRAW 149,84
35 PAUSE 50
40 REM linia B
50 PLOT 226,152
60 DRAW -192,-130
```

Wprowadźmy i wykonajmy ten program. Sprawdźmy, czy otrzymaliśmy to, co poniżej:



Przykład 6.5.

Narysować prostokąt o bokach: w (równoległy do osi x) i h (równoległy do osi y) w środku ekranu:




```
10 REM Przyklad 6.5
15 REM PROSTOKAT
20 INPUT "podaj wartosc w (1<=
w<=255)";w
30 INPUT "podaj wartosc h (1<=
h<=175)";h
35 PLOT (255-w)/2,(175-h)/2
40 DRAW w,0
50 DRAW 0,h
60 DRAW -w,0
70 DRAW 0,-h
```

Wprowadźmy i wykonajmy ten program. Otrzymany rysunek na ekranie dla w = 150 i n = 100 jest postaci:



Instrukcję **DRAW** można użyć do rysowania fragmentów okręgów zamiast odcinków linii prostej. Postać tej instrukcji jest następująca:

DRAW *x, y, a*

x i *y* określają punkt końcowy jak normalnie,

a określa liczbę radianów, w jaką należy się obrócić w trakcie rysowania, jeżeli $a > 0$ to obrót jest w lewo, jeżeli $a < 0$ to obrót jest w prawo (cały okrąg ma 2π radianów, PI jest na klawiszu M).

Jeżeli $a = \pi$, to niezależnie od wartości x i y zostanie narysowana połówka okręgu.

Wykonanie ciągu instrukcji:

10 PLOT 100, 100
20 DRAW 50, 50, PI

spowoduje narysowanie połowy okręgu.

Przykład 6.6.

Narysować ciąg 6 okręgów za pomocą instrukcji DRAW x, y, a, zaczynając od punktu początkowego (40, 40).

Przyjmijmy np. x = y = 50

```
10 REM Przyklad 6.6
20 FOR i=10 TO 60 STEP 10
30 PLOT 40+i,40+i
40 DRAW 50,50,PI
50 DRAW -50,-50,PI
60 NEXT i
```

Wprowadźmy i wykonajmy ten program. Otrzymany rysunek na ekranie jest postaci:



Cały okrąg rysujemy stosując zwykle instrukcje **CIRCLE**, postaci:

CIRCLE *x, y, r*

x, y — współrzędna środka okręgu

r — promień okręgu.

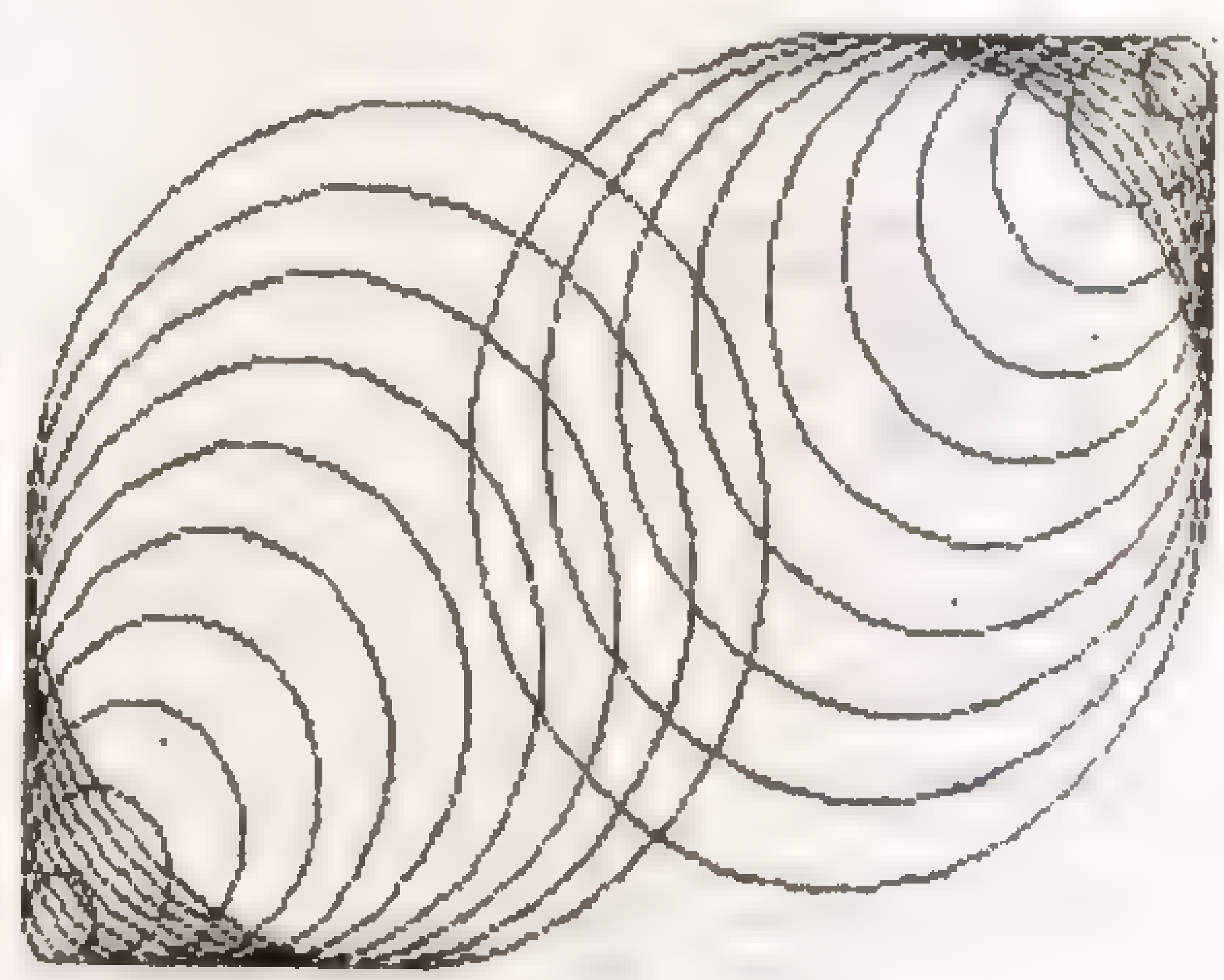
Przykład 6.7.

Narysować ciągi okręgów „wychodzące" z lewego dolnego i prawego górnego rogu ekranu, o współrzędnych środka i promieni danych wzorem:

$x_i = x_i + ih$ (dla $i = 0, 8, \ldots 80$, $h = 8$, $x_0 = 0$)

```
10 REM Przyklad 6.7
20 FOR x=0 TO 80 STEP 8
25 REM prawy gorny rog ekranu
30 CIRCLE 255-x,175-x,x
35 REM lewy dolny rog ekranu
40 CIRCLE x,x,x
50 NEXT x
```

Wprowadźmy i wykonajmy ten program. Otrzymany rysunek na ekranie jest postaci:



Przykład 6.8.

Narysować kilka wykresów funcji i * sin x oraz i * cos x, dla $0 \leq x \leq 4$ oraz i = 10,20,..80

```
10 REM Przyklad 6.8
15 REM rysowanie osi x
20 PLOT 0,88: DRAW 255,0
30 FOR i=10 TO 80 STEP 10
40 FOR x=0 TO 255
50 PLOT x,88+i*SIN (x/64*PI)
60 PLOT x,88+i*COS (x/64*PI)
70 NEXT x
80 NEXT i
```

zw.

Ankieta

Minął już rok od chwili ukazania się pierwszego numeru naszego pisma. W każdym kolejnym staramy się zaspokoić informacyjne potrzeby naszych Czytelników. Znamy je z listów nadchodzących do redakcji, ze spotkań. Tym razem, podobnie jak przed rokiem, prosimy o odpowiedź na poniższą ankietę. Z pewnością Wasze odpowiedzi pomogą nam w redagowaniu „IKSa". Wśród tych, którzy nadeślą pod naszym adresem swoje uwagi rozlosujemy nagrody.

A teraz odpowiedzcie.

Kto nas czyta?

1) Mam lat....

2) Mieszkam w mieście, które ma:
   a) więcej niż 150 tys. mieszkańców
   b) 50—150 tys. mieszkańców
   c) mniej niż 50 tys. mieszkańców
   d) mieszkam na wsi.

3) Jestem
   a) uczniem szkoły podstawowej
   b) uczniem szkoły średniej
   c) studentem uczelni cywilnej
   d) studentem uczelni wojskowej
   e) pracuję.

4) Posiadam lub mam dostęp do
   a) ZX Spectrum
   b) Atarii
   c) Commodore (jaki?)
   d) Amstrad (jaki?)
   e) inny — jaki? ....

5) Korzystam z komputera
   a) w domu (własny)
   b) w szkole
   c) w pracy
   d) w klubie
   e) w klubie garnizonowym
   f) u kolegi.

6) Czytam „IKSa"
   a) systematycznie
   b) czasami
   c) bardzo rzadko

7) Za pożyteczne uważam:		8) za niepotrzebne uważam:
a)	wykłady Basica	....
b)	wykłady Pascala	....
c)	wykłady Logo	....
d)	publikacje dot. systemów mikrokomputerowych	....
e)	programy gier	....
f)	programy użytkowe	....
g)	programy graficzne	....
h)	informacje o nowościach tech.	....
i)	zadania Ligi Myślących	....
j)	porady techniczne	....
k)	Twoje propozycje	....
....		....
....		....

Nasz adres:
Redakcja „IKS"
W-wa 00-950
ul. Grzybowska 77

Początkowo można było pracować. Byli klienci, były zlecenia, a w konsekwencji — pieniądze. Jeśli pracowało się szybko i skutecznie, zarobki przewyższały nie tylko stypendia, ale i średnią krajową płacę. Można było też mieć satysfakcję z wdrożeń swoich opracowań. Zmieniło się to nagle. Zmiana zarządu Studenckiej Spółdzielni spowodowała zmianę atmosfery i przepisów.

Włodzimierz Witkowski szukał miejsca dla siebie. Nie chciał się przekwalifikować, nie chciał podzielić losów dawnych kolegów z Mokotowskiej. Przed laty spotkali się w warszawskim Centrum Studenckiego Ruchu Naukowego. Na kursie języka APT. Potem robili oprogramowania dla obrabiarek sterowanych numerycznie.

Zamówienia mieli poważne

— ze Świdnika, „Nowotki", Mielca, Kalisza. Na początku dekady wszystko się rozpadło. Przyszła bessa dla rodzimych informatyków. Część wyjechała, za granicę i tam pracuje w zawodzie. Z tych co zostali jeden jest taksówkarzem, inny produkuje sztuczną biżuterię, a reszta jakoś przetrwała. Na dodatek nastąpiła rewolucja. Minikomputery wyparły „Odry" i RIADY. Wiernych informatyce pozostała garstka.

Entuzjaści — Witkowski tak określa ich, siebie.

Czy na entuzjaźmie można zarobić? Okazało się, że tak. Witkowski trafił do nie znanego wtedy nikomu Towarzystwa Wynalazców „RATIO".

Mimo szumnej nazwy firma to w istocie niewielka. Całe biuro mieści się w skromnym pomieszczeniu na warszawskim Bródnie przy ul. Ogińskiego 1. Kapitał zakładowy wszystkiego pół miliona. Tyle co na opłaty lokalowe i rachunki telefoniczne przez kilka miesięcy. Dwa i pół etatu przeznaczono na cały personel. Prezes, urzędujący wiceprezes i te pół dla głównego księgowego. Pensja: czterdzieści tysięcy. Ale żeby ją dostać trzeba było przyjść z ewentualnymi klientami towarzystwa. Z klientami, bo spółka od pewnego czasu praktycznie nie funkcjonowała.

Jedynym atutem jest koncesja na działalność. I to nawet rozległą. Prawie taką jaką prowadzi instytut naukowy i jeszcze kilka innych placówek różnego typu — wdrażanie projektów i opracowywanie systemów informatycznych, prace konstrukcyjne i ekspertyzy techniczne, ekonomiczne, prawne, consulting i marketing, analiza wyników badań i nadzór autorski. Słowem: dla każdego coś miłego, jeśli tylko pragnie wydać parę groszy za myślenie.

z administracji żyją.

Co prawda głównie tej gospodarczej.

Klient musi zobaczyć za co ma płacić...

Witkowski wrzuca Amstrada na tylne siedzenie fiata i jedzie do jakiegoś zakładu. Tam rozstawia wszystko w sali konferencyjnej, wczytuje program i...

Spektrum

```
   1 REM program TKANINA
  10 LET a$="": LET b$=""
  20 FOR n=1 TO 32:   LET a$=a$+
CHR$ 144: NEXT n
  30 FOR n=33 TO 64:  LET a$=a$+
CHR$ 145: NEXT n
  40 FOR n=1 TO 11: LET b$=b$+a$
: NEXT n
  50 PRINT BRIGHT 1;AT 0,0;b$
  60 FOR n=0 TO 7: LET a=RND*255
: POKE 65368+n,a: POKE 65383-n,a
: NEXT n: BEEP .02,a/5: GO TO 50
  99 SAVE "TKANINA" LINE 10
```

J. Rubinowicz



MYŚLENIE PO GODZINACH

— Proszę zobaczyć, jakie to proste... Ależ skąd, pani też potrafi... Proszę spróbować... Teraz tu naciskamy i już mamy wszystkie potrzebne dane... Nic nie ma prawa się zepsuć, chyba że komputer spadnie na podłogę... Niech pan, wprowadzi nowy program... Tak... ENTER!

Trwa mocno przyśpieszony kurs informatyki. Przeważnie pierwsi kończą go dyrektorzy. Sprawa mody? Chyba nie, tylko! Chociaż komputer to synonim postępu, a postępowy chce być każdy... Tym bardziej że pieniądze na ten cel są z Funduszu Postępu Techniczno-Ekonomicznego i nie każdy wie, jak je spożytkować.

— Jednak poważni menedżerowie widzą w tym przede wszystkim szansę uporządkowania funkcjonowania przedsiębiorstwa. A komputer do tego zmusza. I czynią to ostrożnie, małymi kroczkami. Gospodarka materiałowa. Finanse. Kadry. Zaczynamy od ustawienia Amstrada 6128 na biurku dyrektorskim, potem dostawiamy następne. Rozbudowywujemy oprogramowanie.

Z czasem powstają kompleksowe systemy. Tak jak to się dzieje w Ostrowi Mazowieckiej, gdzie „RATIO" ma w ciągu dwóch lat skomputeryzować cały zakład, „Bumar-Waryński". Na pierwszy ogień poszła gospodarka materiałowa, potem inne sfery życia zakładu, w tym optymalizacja produkcji. Całą siecią kierować będą profesjonalne komputery IBM. Nie byłoby to możliwe, gdyby dyrekcja zakładu, a zwłaszcza główny księgowy nie rozumieli wymogów nowoczesności w zarządzaniu przedsiębiorstwem.

Na podobne przedsięwzięcie chciały się porwać Państwowe Zakłady Zbożowe w Olsztynie. Niestety barierą stały się koszta. Co tu ukrywać, spore — choć Towarzystwo Wynalazców i tak stosuje narzuty o połowę niższe niż biura uspołecznione.

Konkurencyjne ceny sprawiają, że jak na razie Witkowski nie martwi się o brak klientów. Zresztą u niego obowiązuje naprawdę zasada: „Nasz klient — nasz pan". Zleceniodawca decyduje o tym czego chce (co nie wszędzie jest sprawą oczywistą!), kiedy ma to być wykonane, może sobie wybrać specjalistę, któremu zadanie zostanie powierzone.

Egzystencja spółki opiera się na współpracownikach. Na co dzień są to zatrudnieni etatowo w poważnych firmach informatycy, księgowi, radcy prawni, czasami nie w pełni doceniani i wykorzystywani, po godzinach dorabiają myśleniem. A myśleć muszą bardzo intensywnie skoro zleceniodawcy skłonni są im płacić kilkaset złotych za każde sześćdziesiąt minut pracy ich szarych komórek. Taka to

fucha — nie fucha.

Rozwiązania za każdym razem muszą być oryginalne. Nie można mechanicznie powielać programów, projektów, ekspertyz. W poszczególnych przedsiębiorstwach bowiem, nawet tej samej branży, o identycznym niemal profilu produkcji — odmienne panują zwyczaje, rozmaite obowiązują przepisy wewnętrzne, różny jest wreszcie stopień organizacyjnego bałaganu.

Właściciele rozliczają zarząd na dwóch i pół etacie z efektów jego starań. Witkowski wie, że jego zarobki zależą od obrotów spółki i że byle zastój może sprawić, iż będzie musiał sobie poszukać nowej pracy. Zarząd rozlicza wykonawców z rzetelności, terminowości. Nie ma w towarzystwie miejsca dla amatorów łatwych zarobków. Pieniądze tutaj są — to prawda, ale nie dostaje się ich za nic.

Czy to, co robią nie spotyka się z oporami?

— Spotyka się! Ludzie mimo wszystko nieufnie podchodzą do nowego. A komputery dla wielu jeszcze są tym nowym. Wymuszają zmianę sposobu myślenia i działania. Choć przecież nie ma się czego bać...

Nie sposób znaleźć sytuację, kiedy zainstalowanie maszyny elektronicznej pozbawia kogoś pracy. Owszem, zdarzają się redukcje etatów, ale tylko wówczas, kiedy ktoś „zastąpiony" komputerem przechodzi na emeryturę. Bywa, że czasami, gdy po kilku miesiącach pracy z komputerem następuje jakaś awaria, urywają się telefony, mnożą interwencje, aby ją usunąć jak najszybciej. Ludzie zaczynają sobie nie wyobrażać życia bez niego, choć kiedyś ponoć świetnie dawali sobie radę...

„RATIO" też skomputeryzował swoją działalność. Na dyskach zapisane są wszystkie dane, finansowe też. Główny księgowy jednak pieczołowicie prowadzi księgi rachunkowe. Wydruki wydrukami, ale jak na razie fiskus woli to, co wyliczy i zapisze człowiek.

Janusz B. GROCHOWSKI

Spektrum

```
  10 REM Program "*KILIM 2*"
  20 BORDER 0: LET n=1: LET a$="
ABABABABABABABABABABABABABABABAB
CDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCD
": LET b$="": FOR m=1 TO 11: LET
 b$=b$+a$: NEXT m: CLS
  25 PRINT AT 8,1; BRIGHT 1;"OBL
ICZANIE TABELI ODWRACAJACEJ"
  30 FOR s=0 TO 255
  40 POKE 22432,s
  50 LET t=POINT (0,0)+2*POINT (
1,0)+4*POINT (2,0)+8*POINT (3,0)
+16*POINT (4,0)+32*POINT (5,0)+6
4*POINT (6,0)+128*POINT (7,0)
  60 POKE 60000+s,t: NEXT s
  70 FOR z=0 TO 7
  80 LET a=INT (RND*255): LET b=
PEEK (60000+a)
  85 POKE 65368+z,a
  90 POKE 65376+z,b
  95 POKE 65391-z,a
 100 POKE 65399-z,b
 110 NEXT z: REM AB CD = AB CD
 120 PRINT #0;AT 1,15;n: LET n=n
+1: BEEP .02,a/8
 130 PRINT AT 0,0;b$: GO TO 70
 200 SAVE "*KILIM 2*" LINE 10
```

J. Rubinowicz

LIGA MYŚLĄCYCH

Zadanie 1

Ułamek $\frac{1}{2}$ można przedstawić w postaci ułamka, np. 6729/13458. W ułamku tym występują wszystkie cyfry (z wyjątkiem zera). Korzystając z tej zasady należy podać składające się z dziewięciu cyfr ułamki, o wartościach:
$\frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}$,
przy czym żadna z dziewięciu cyfr nie może się powtarzać

Zadanie 2

Jaki rok zeszłego stulecia zwiększy się $4\frac{1}{2}$-krotnie, jeśli popatrzymy na odbicie w lustrze cyfr zapisanych na kalkulatorze, tworzących tę datę? (Trzeba poeksperymentować)

Zadanie 3

Pewien wujek przyjechał w odwiedziny do swoich dwóch siostrzeńców i trzech siostrzenic. Najpierw wyszedł mu na spotkanie mały Krzyś z siostrzyczką Basią i od razu poinformował wujka, że jest dwa razy starszy od swej siostrzyczki. Następnie przybiegła Ela. Ojciec jej wyjaśnił gościowi, że suma lat obu dziewczynek jest dwa razy większa od wieku chłopca. Gdy powrócił ze szkoły Michał, ojciec powiedział, że suma lat obu chłopców jest dwa razy większa od sumy lat dziewczynek. Wreszcie zjawiła się Krysia i ujrzawszy gościa krzyknęła radośnie: Wujku przyjechałeś do nas akurat w moje urodziny, właśnie dzisiaj kończę dwadzieścia jeden lat. Ojciec jej zaś dodał: właśnie zauważyłem, że w chwili obecnej suma lat wszystkich mych córek jest dwa razy większa od sumy lat mych synów. Ile lat ma każda córka i każdy syn?

Zadanie 4

Przyrost produkcji w fabryce w porównaniu z rokiem poprzednim wynosi za pierwszy rok p%, za drugi g%. Jaki powinien być procent przyrostu produkcji w 3 roku, aby średni przyrost roczny produkcji za trzy lata był równy r%

Zadanie 5

Winda ma dwa systemy hamowania różniące się między sobą i niezależnie pracujące. W razie zerwania się liny pierwszy system zatrzymuje windę średnio w 995 przypadkach na 1000 a drugi system w 998 przypadkach na 1000. Obliczyć, jakie jest prawdopodobieństwo P, że w razie zerwania się liny winda zostanie zahamowana?

Nasz adres:
Redakcja „IKS"
W-wa 00-950
ul. Grzybowska 77

Pełne rozwiązania zadań prosimy przesyłać pod adresem redakcji do końca lipca br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Poczta „IKS-a"

„IKS" kończy PIERWSZY ROK. W ciągu tego (niezbyt długiego) czasu otrzymaliśmy wiele, bardzo wiele listów: pocztówek, starannie zaklejonych kopert, a nawet pracowicie sporządzonych maszynopisów.

Każdy, nawet najkrótszy, był uważnie przeczytany, a zawarte w nim myśli natychmiast wykorzystane, gdyż... „IKS" JEST DLA WAS!

Odpowiedzi było mało.

Mogliśmy bowiem albo odpowiadać, albo... ROBIĆ „IKSa"!

A nas było (i nadal jest) bardzo, bardzo mało.

Jan MŁYNARZ

Tomasz Nabożny:

„...Jestem czytelnikiem waszego czasopisma. Zbieram prospekty różnych firm, a także interesuję się informatyką. I właśnie pomyślałem sobie, czy nie moglibyście przysłać mi adresów firm komputerowych, takich jak Sinclair, Atari, Commodore, Sharp itd. ...Gdzie można zakupić drogą wysyłkową książki do nauki informatyki dla początkujących na ZX Spectrum +..."

Wydawnictwa Komunikacji i Łączności niedawno wydały książkę PRZEWODNIK PO ZX SPECTRUM. Nakład duży, cena wysoka (490 zł). Co się zaś tyczy „zbieractwa", to prawda jest taka: duże firmy są zainteresowane MASOWĄ SPRZEDAŻĄ swoich wyrobów i niechętnie (lub wcale) nie interesują się kolekcjonerami. Nasza rada: dużo się ucz (nie tylko angielskiego), a za parę lat powinny pojawić się polskie minikomputery i polskie prospekty tych minikomputerów.

Wojciech Zarzyka z Sanoka:

„...Interesuję się informatyką, chciałbym się dowiedzieć, ile kosztuje ZX SPECTRUM i gdzie go można kupić. A oto mój program..."

Dziękujemy za program. Komputery (za złotówki) można kupić trojako:
— „z ogłoszenia" (polecamy, gdyż znany jest sprzedawca),
— w komisie (sprzedawca pobiera dodatkową marżę),
— na giełdzie (bardziej ryzykowne — jest to typowe kupowanie kota w worku).

Sprzedaż komputerów prowadzi również Centralna Składnica Harcerska (Warszawa, Mokotowska), ale popyt dotychczas kilkakrotnie przewyższa podaż.

UWAGA!

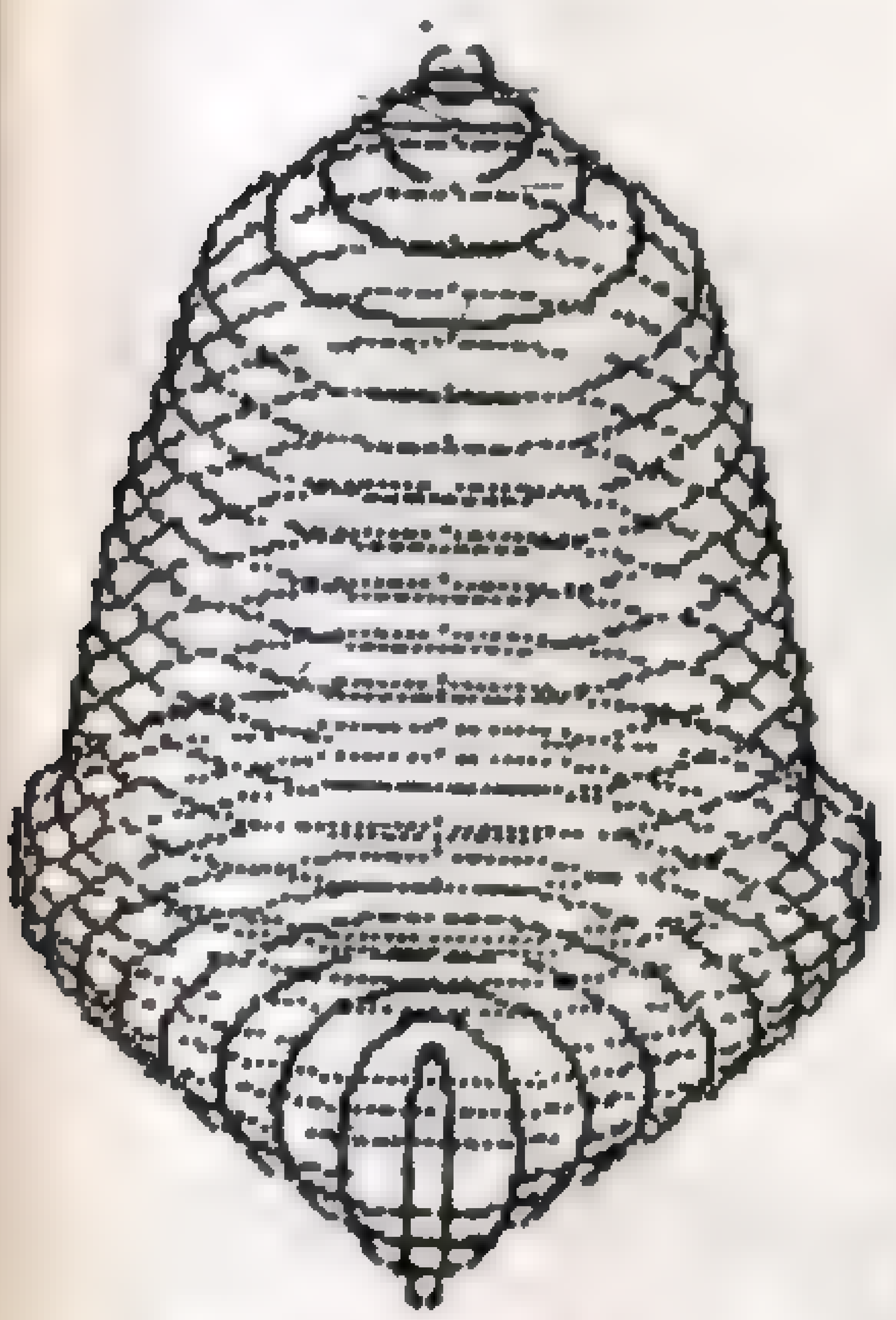
Zrzeszenie Studentów Polskich
M I K R O K O M P
93-590 Łódź
Al. Politechniki 3

organizuje w dniach 17—18 października br. konferencję użytkowników SHARP

telefony: ZSP 36-84-89
Prezes 32-57-83

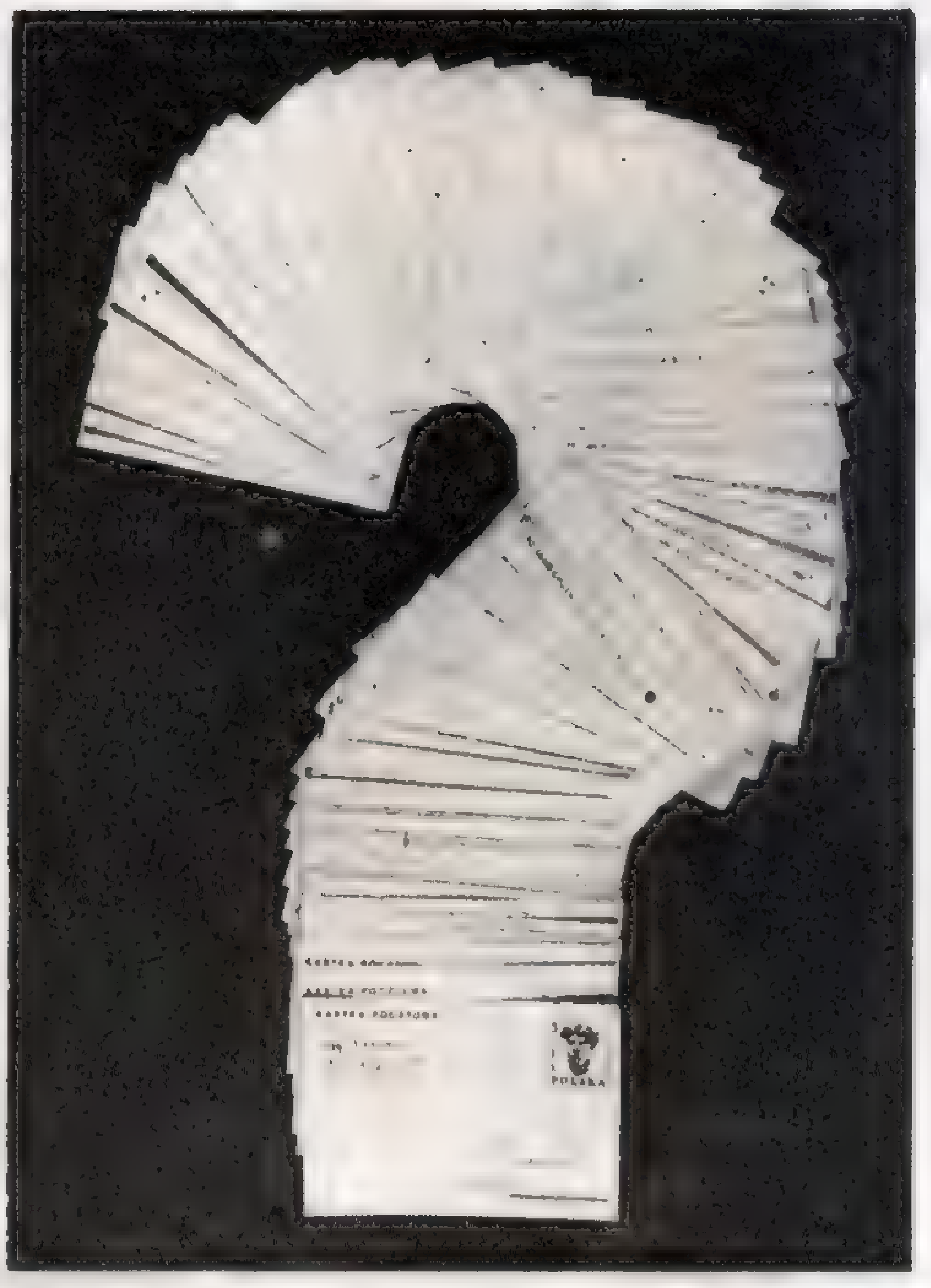
Pocztowa giełda

Rozwiązanie krzyżówki z nr. 2/87 „IKS-a"
Hasło brzmi:
„ELWRO to komputer dla Ciebie"
Bony pieniężne (1000 zł) wylosowali: Barbara Sułkowska — Wyszków, Jacek Golis —Zgierz, Stanisław Mączka — Leżajsk, Barbara Rejman — Kluczbork, Marcin Brzeziński — Lublin.
Nagrody książkowe otrzymują: Marcin Szalkowski — Skierniewice, Tomasz Staruszkiewicz — Przemyśl, Maciej Parzyszek — Warszawa-Anin, Ireneusz Nowak — Tarnowskie Góry, Halina Wojturska — Rzeszów, Jadwiga Zasada — Szczecin, Ewa Bonder — Świętochłowice, Zbigniew Słowik — Włoszczowa, Janusz Dołas — Częstochowa, Zdzisław Skiba — Szczecin.



```
5 REM GRAFIKA ATARI
10 GRAPHICS 8:COLOR
1:BY=16:BC=1:DEG
20 FOR B=31 TO 76 ST
EP 3:BB=COS(3*B):CC=
COS(1.1*(B-18)):IF B
>64 THEN BC=BC-0.2
30 FOR A=-180 TO 180
 STEP 2:X=INT(COS(A)
*30):Y=INT(SIN(A)*12
):IF B>64 THEN 70
40 PLOT 160+X*BB*1.1
,10+Y+B:PLOT 160+X*C
C,32+BY+Y+B
50 NEXT A:BY=BY+1.5:
NEXT B
70 PLOT 160+X*CC*BC,
32+Y*BC+B+BY:GOTO 50
```

Jeszcze raz konkurs



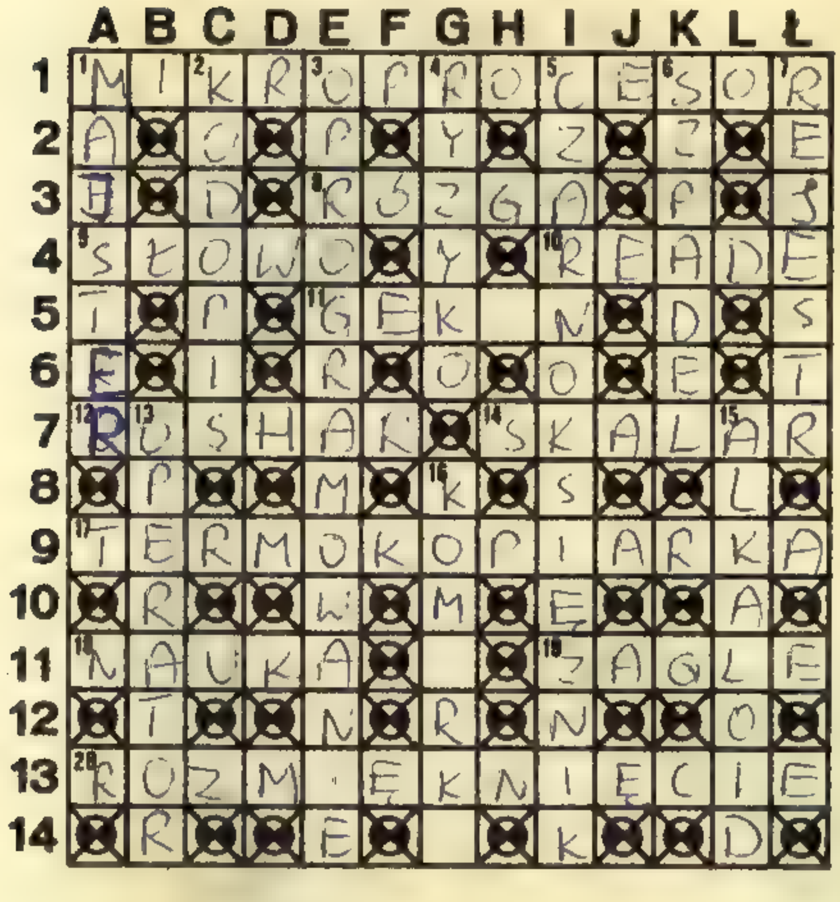
Oto plik kilkudziesięciu poprawnych oczywiście rozwiązań. Na odwrocie są różni nadawcy, ale jeden autor na pocztowe kartki naklejał powielone rozwiązanie. W losowaniu szczęścia niestety nie miał, ale należy mu się piątka z rachunku prawdopodobieństwa.

Spektrum

```
 10 REM Program "*KILIM 3*"
 20 BORDER 0: LET n*1: LET a$*"
FABCDEJIHGFABCDEJIHGFABCDEJIHGFA
TONMLKPQRSTONMLKPQRSTONMLKPQRSTO
": LET b$*"": FOR **1 TO 11: LET
 b$*b$+a$: NEXT *: CLS
 25 PRINT AT 8,1; BRIGHT 1;"OBL
ICZANIE TABELI ODWRACAJACEJ"
 30 FOR $*0 TO 255
 40 POKE 23432,$
 50 LET t*POINT (0,0)+2*POINT (
1,0)+4*POINT (2,0)+8*POINT (3,0)
+16*POINT (4,0)+32*POINT (5,0)+6
4*POINT (8,0)+128*POINT (7,0)
 60 POKE 60000+s,t: NEXT $
 70 FOR z*0 TO 39
 80 LET a*INT (RND*255): LET b*
PEEK (60000+a)
 85 POKE 65368+z,a
 90 POKE 65408+z,b
 95 POKE 65487-z,a
100 POKE 65527-z,b
110 NEXT z
120 PRINT #0;AT 1,15;n: LET n*n
+1: BEEP .02,a/8
130 PRINT AT 0,0;b$: PAUSE 140:
 BEEP .02,12+a/8: PRINT AT 0,0;
INVERSE 1;b$: GO TO 70
200 SAVE "*KILIM 3*" LINE 10
```

J. Rubinowicz

Krzyżówka nr 5



POZIOMO: 1. układ scalony wielkiej skali integracji, podstawowy element budowy mikrokomputerów,
8. witka, łoza,
9. ciąg bitów mieszczący się w pojedynczej komórce pamięci,
10. pisarz angielski (1814—84) rozwijający tradycje dickensowskie (skojarz z rozkazem „czytaj"),
11. owadożerna niewielka jaszczurka,
12. włosy z ogona i grzywy końskiej,
14. wielkość określona jednoznacznie jedną liczbą rzeczywistą (np. długość, temperatura),
17. urządzenie peryferyjne do termografii,
18. np. informatyka, cybernetyka,
19. latacz, kliwer, sztaksel,
20. utracenie twardości pod wpływem wilgoci, rozmoknięcie:

PIONOWO: 1. fachowiec, mistrz,
2. urządzenie klawiaturowe łączące funkcjonalne własności dalekopisu, elektrycznej maszyny do pisania i dziurkarki taśmy,
3. software,
4. przedsięwzięcie o niepewnym wyniku,
5. mag, czarodziej,
6. łopata, rydel,
7. urządzenie zapamiętujące pojedynczą wartość danej lub wyniku,
13. pracownik wyspecjalizowany w obsłudze komputera,
15. np. nikotyna, kofeina, strychnina,
16. element pamięci zawierający rozkaz lub dane i posiadający ściśle określony adres.

Rozwiązanie stanowi hasło utworzone z liter przeniesionych z następujących pól:
1B—1C—1K 5C—6E—6G—2E—4K—11K—9J—12B—12L—9C—2Ł—13D 4D—6C—5F—5K—3G—4G 13I 6Ł—11Ł—13K—7D—11A—13E—8G—9I 10G—13L—9F—12G—7B—7I—1H—9D—8B—11C—5A—4J—7F—2C—10E 4Ł—3A

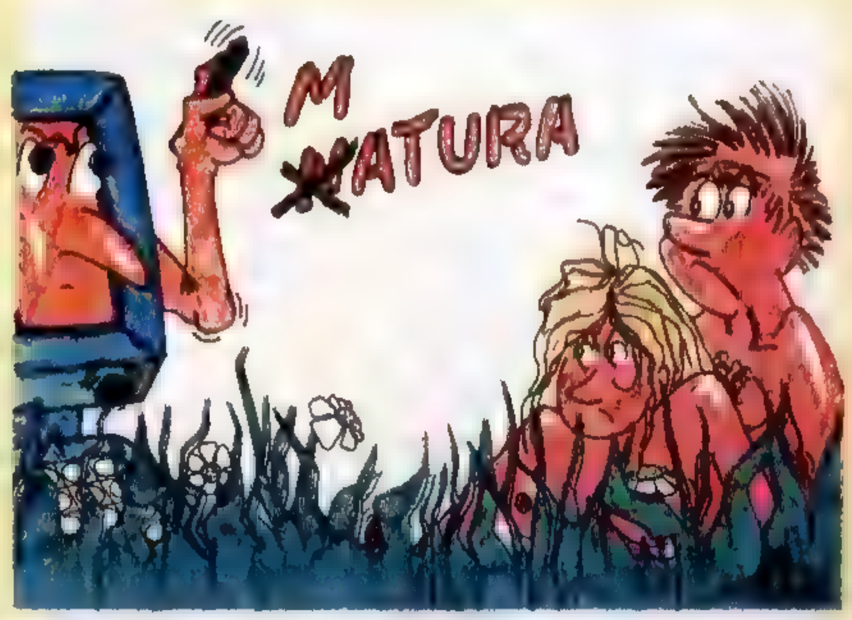
Rozwiązanie (tylko hasło) należy przesłać pod adresem redakcji do końca lipca br. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

Mikrociekawostki





Juki 5520 i 6500 ta sama firma, ale istotne różnice w cenie i możliwościach. Obie drukarki mają spory repertuar znaków, krojów i wielkości pisma. Istotnymi zaletami są przede wszystkim szybkość druku (ponad sto znaków na minutę) i cicha praca. Wada — niestety, z uwagi na cenę są to urządzenia dla amatorów w Polsce niedostępne.



Rys. Michał Przybyłowicz

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogon. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8814. Nr. ind. 361682. K-82

6 1987 (14)

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" ISSN 0860-2794 Cena 50 zł



WOJNA O KOMPUTERY
TAJEMNICE SPECTRUM
PROGRAMY

W numerze:

- Ciekawostki — str. 3
- Statystyka z ICL — str. 4
- W szponach Atari (Wielokolorowa grafika w trybie 8) — str. 6—9
- Komputer w medycynie Tomografia komputerowa — str. 13
- Fotoreportaż — str. 16—17
- „Wojna" o komputery — str. 18
- Mikrokomputery IBM PC — str. 19—20
- Basic — str. 21—22
- Grafika Commodore 64 — str. 23—24
- Sztuki i sztuczki (Zmienne systemowe ROM — Spectrum) — str. 25—26
- Pascal — str. 27—29
- Zrób to sam (Złośliwe ładunki) — str. 30
- Liga Myślących — str. 31
- Pocztowa giełda — str. 31
- Krzyżówka — str. 32
- Francuskie mikrokomputery — str. 32

PROGRAMY:

- Mój program (Sharp) — str. 5
- Mini-DOS (Atari) — str. 10
- Program kopiujący (Atari) — str. 11
- Gra w oczko (Sharp) — str. 14
(Spectrum) — str. 15

UWAGA CZYTELNICY IKSa!

ukaże się na przełomie lipca i sierpnia. Podwójna będzie również cena — 100 zł, ale sądzimy, że poziom zawartości będzie co najmniej tak wysoki jak i ona

I jeszcze o cenie. Od chwili, kiedy rozpoczęliśmy wydawać nasz miesięcznik (a to już 14 numerów) znacznie wzrosły koszty produkcji, podrożała poligrafia i papier. Stąd też rachunek ekonomiczny zmusza nas do zmiany ceny. Od września IKS będzie kosztował 80 złotych i choć jest to wzrost znaczny, pozostaniemy przecież najtańszym pismem mikrokomputerowym w kraju. Mamy też nadzieję, że dzięki pomocy Czytelników będziemy coraz lepsi — jesteśmy przecież dla Was. Chcemy być Waszym pismem

Rozpoczynając przed rokiem wydawanie „IKSa" zdawaliśmy sobie sprawę z olbrzymiej popularności mikrokomputerów. Urządzenia te napływały do kraju wartkim strumieniem, a ich cena sprawiła, że dla wielu stały się osiągalne. W ten sposób bariera dzieląca człowieka (z reguły nie zagłębiającego się w tajniki informatyki) z wielkimi komputerami zniknęła. Komputer przestał być już potworem mrugającym małymi żarówkami w niezrozumiałym języku — po prostu stał się bliższy. Nie oznacza to oczywiście, że wszystkie przeszkody zostały pokonane.

Najpierw była euforia nowości. Za „jedne" sto tysięcy można było stać się właścicielem najprostszego Z 81 czy Spectrum. W ślad za nimi przyszły nieco lepsze i droższe typy. Na informatyce chcieli znać się wszyscy; o komputerze trzeba było wiedzieć wszystko. W sytuacji najlepszej znaleźli się ci, którzy na swoim biurku mogli postawić owe cudo.

Po niedługim czasie sytuacja się wyjaśniła: informatyka nie jest dyscypliną trudną, ale na komputer nie każdy może sobie pozwolić. Po dużej fali zainteresowania przyszedł czas na refleksję. Z komputerów zrezygnowali ci, dla których były to jedynie zabawki. Sprzęt profesjonalny zaś jest jeszcze zbyt drogi, aby był powszechnie dostępny. Dlatego też tak ważne pozostają kluby i szkoły, skracające dystans do mikrokomputerów, bo jeśli nawet dziś nie stać nas na sprzęt, to możemy sobie zafundować wiedzę, a ta z pewnością się przyda.

Mamy zatem nadzieję, że Czytelnicy wybaczą nam, iż nie kokietujemy ich lawiną informacji o technicznych nowinkach — sądzimy, że ważniejsze jest ich wykorzystanie. Chcemy również ułatwić pierwszy kontakt z komputerem tym, którzy się go jeszcze lękają.

O wykorzystaniu mikrokomputerów i sprzętu wideo dużo mówiło się w czasie Infowideo '87, imprezy organizowanej przez Stowarzyszenie Mikrokomputerowe ABAKUS. I tu zaszły zmiany: w ubiegłym roku do Warszawskiego Ośrodka Kultury przy Elektoralnej — na Infowideo '86 — trudno było wejść — po roku zainteresowanie mimo wysiłków organizatorów było raczej mizerne. Jak będzie za rok?

REDAKCJA



— Mnie proszę nie dziękować. Wytypował pana komputer

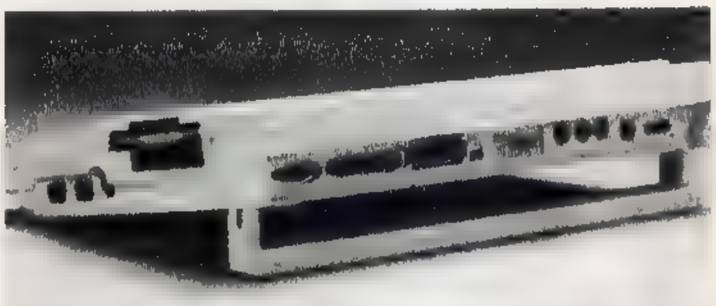
T U R B O   B A S I C

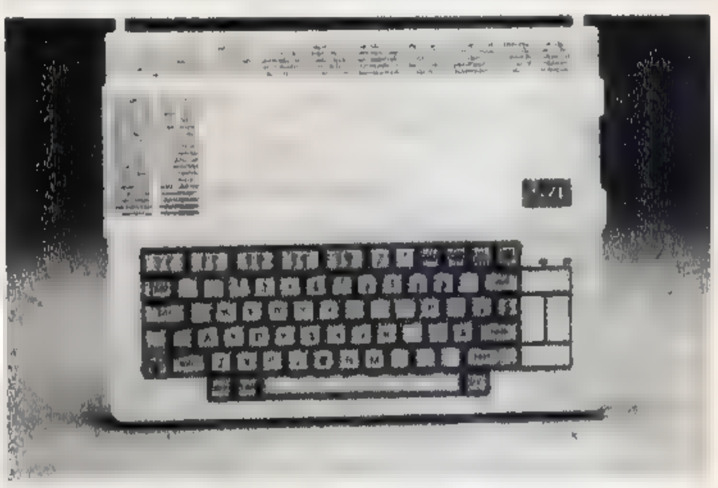
Firma Borland International wprowadziła na rynek nowy swój produkt - Turbo Basic (99.95 $) . Program kompiluje 12 .000 linii na minutę i tworzy zbiory typu EXE .Umożliwia on również kompilację w pamięci, posiada własny edytor ,program łączący oraz bibliotekę . Turbo Basic wykorzystuje instrukcje strukturalne IF ELSE IF , SELECT CASE , DO WHILE , DO UNTIL , LOOP WHILE , LOOP UNTIL . Program jest napisany w asemblerze i jest kompatybilny z IBM Advanced Basic oraz Microsoft GW-BASIC . Umożliwia obsługę karty graficznej EGA oraz koprocesora matematycznego 8087 / 80287 .Turbo Basic pozwala tworzyć programy , które przekraczają 64 KB ($ SEGMENT).

Turbo Prolog w medycynie

Pracownicy naukowi w Medical College of Ohio rozwijaja system ekspertowy w Turbo Prologu , przeznaczony dla lekarzy z oddziału intensywnej terapii , określajacy czy pacjent rzeczywiście uległ atakowi serca . Program ekspertowy , bedacy częścią składową systemu Expert Scane Segmentation , ocenia dane zgromadzone przez skaner ultradźwiekowy . Dane przesyłane są do komputera ,który wykorzystuje algorytmy napisane w jezyku C

Mikrokomputer domowy M S X X'press SVI.738





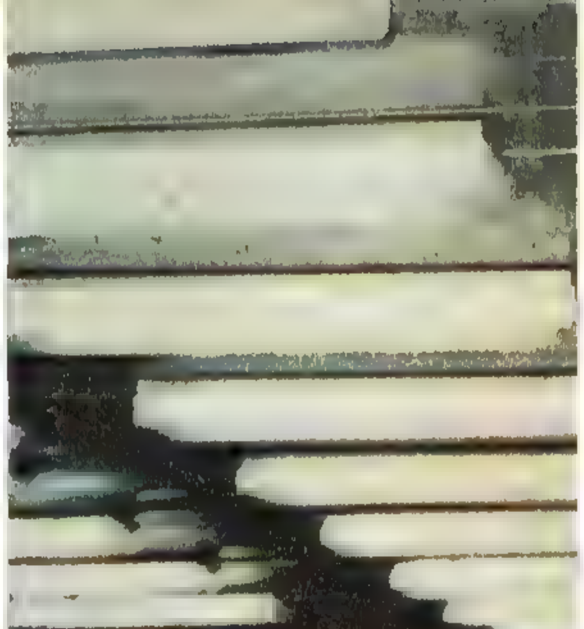
Mikrokomputer X'press nie jest tylko komputerem zabawowym , może korzystać z trzech systemów operacyjnych :CP/M, MSX-DOS i MSX Disk BASIC .Dzięki nim istnieje szeroki dostęp do biblioteki programów użytkowych . Mikrokomputer posiada pamięć zapisywalną RAM 80 KB. W komputerze wbudowano stację dysków 3 1/2 cala ,pozwalająca osiągnąć pojemność 360 KB . Video Display Processor umożliwia pracę monitora :
● w trybie alfanumerycznym 40 lub 80 znaków w wierszu;● w trybie graficznym w standardach małej , średniej i dużej rozdzielczości.Mikrokomputer posiada wyjścia RS-232C i Centronix. Klawiatura typu QWERTY składa się z 73 klawiszy ,w tym 10 funkcyjnych. Do mikrokomputera można dołączyć modem,drugą stację dysków, magnetofon, joystick,drukarkę.Mikrokomputer może pracować w lokalnej sieci komputerowej

□□ ■ □□ M I K R O D E P E S Z E □□ ■ □□

Wśród produktów azjatyckich można znaleźć mikrokomputer kompatybilny z IBM PC/AT ,wyposażony w procesor 80286 pracujący z częstotliwością 6 / 8 / 10 / 12 MHz i posiadający 4 MB pamięci RAM na płycie podstawowej . ■ □□ ■ □□ ■

opracowali: Mariusz Jarzębowski i Jacek Wojtala

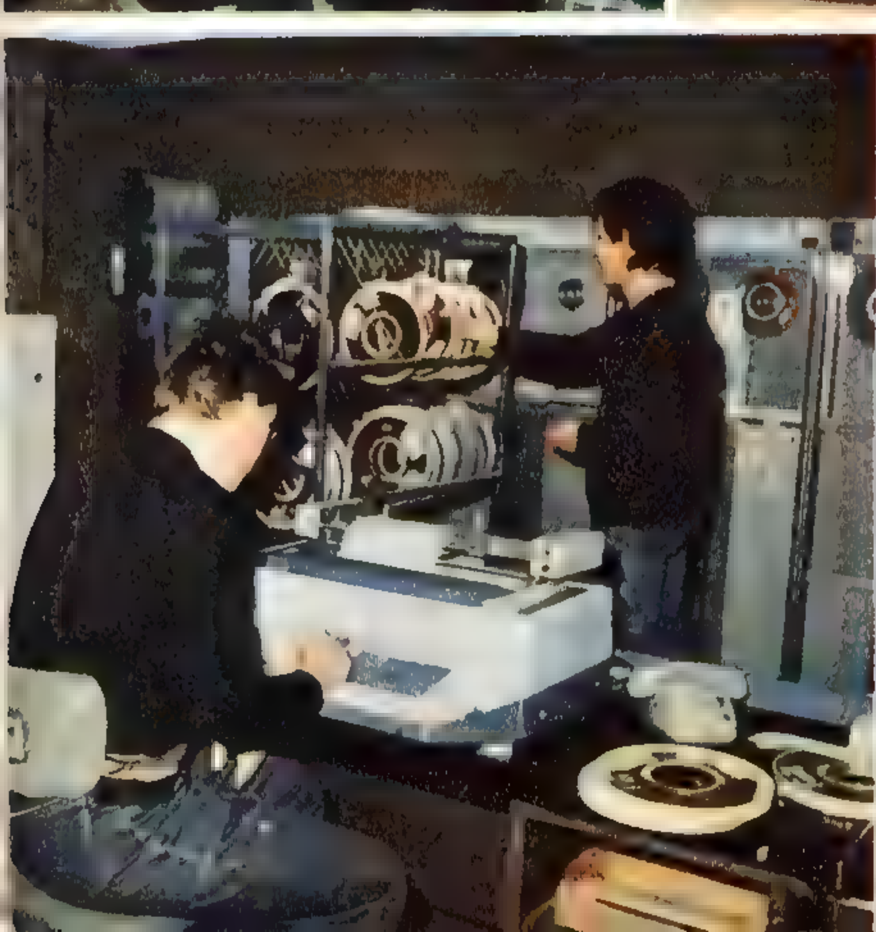












Statystyka z ICL

Na pierwsze powojenne wyposażenie Głównego Urzędu Statystycznego składały się nieliczne maszyny małej mechanizacji, przeniesione z okupacyjnego urzędu statystycznego w Krakowie. Od 1946 roku rozpoczęto użytkowanie mocno już wyeksploatowanych maszyn systemu kart dziurkowanych, wydzierżawionych od Ministerstwa Przemysłu. Istotnym postępem było dokonanie w 1949 roku zakupu we francuskiej firmie Bull 16 zestawów maszyn licząco-analitycznych, odznaczających się w owym czasie jednymi z najlepszych możliwościami eksploatacyjnymi. W 10 lat później — przed kolejnym Narodowym Spisem Powszechnym — zakupiono następną partię maszyn w firmie Bull, nowocześniejszych i wyposażonych w elektroniczne przystawki kalkulacyjne.

W 1963 roku dla zapewnienia zorganizowania warunków zakupu i uruchomienia elektronicznej techniki obliczeniowej w GUS, utworzono w Zakładzie Techniki Statystycznej Dział Studiów i Analiz. Stał się on zalążkiem przyszłego Ośrodka Elektronicznego. Prowadzone przez DSiA prace studialnobadawcze były ściśle powiązane z praktycznym uruchomieniem opracowań statystycznych na dostępnej wówczas w Polsce maszynie elektronicznej **ZAM 2**, zainstalowanej w Instytucie Maszyn Matematycznych oraz na maszynie **ICT-1300** pracującej w Centralnym Ośrodku Doskonalenia Kadr Kierowniczych.

W wyniku szczegółowej analizy parametrów techniczno-użytkowych, oprogramowania, cen, warunków dostaw itp., odnoszących się do różnych typów komputerów dostępnych na rynku, ostatecznie zdecydowano się zakupić dla GUS maszynę typu ICL 1905 w brytyjskiej firmie ICL (dawniej ICT). Komputer ten uruchomiony został w październiku 1967 i od pierwszych dni po odbiorze technicznym rozpoczęła się jego eksploatacja na dwie, a wkrótce na trzy zmiany.

Wybór tego typu komputera okazał się dla GUS szczęśliwy również z tego względu, że w kilka lat później polski przemysł komputerowy na podstawie porozumienia z firmą ICL zastosował ich system oprogramowania w polskich komputerach **Odra 1300**, dzięki czemu obie te rodziny maszyn zastosowały w pełni wymienialne oprogramowanie.

W 1970 roku nastąpiły kolejne komputerowe zakupy: **ICL 1902 A**, który już na miejscu przebudowano na większy model **1903 A** oraz polską **Odrę 1304** — pierwszy egzemplarz tej maszyny z produkcji seryjnej.

Obecnie Ośrodek Elektroniczny GUS ma 6 kompatybilnych komputerów **Odra 1305** i **ICL**. Poszczególne zestawy mogą pracować ze sobą, sprzęt jest wymienny, można łączyć odpowiednie moduły. Dyrektor ośrodka, Andrzej Karpiński, twierdzi, że jest to sprzęt dobry, w pełni odpowiadający stawianym wymaganiom. Firma jest znana, solidna, wywiązuje się z umów, rozporządza bogatym oprogramowaniem. W Warszawie znajduje się biuro ICL. Komputery cały czas są unowocześniane, ciągle dokupuje się nowe moduły. Jeśli pamięć pierwszego komputera wynosiła 32 tys. słów, to dzisiaj wynosi 512 tys. Dane rejestrowane są na taśmach magnetycznych. Można też robić to na kartach drukowanych, ale to mało ekonomiczna metoda. Do prowadzenia danych służą mikrokomputery **Mera 9150** z możliwością pracy równocześnie na 32 stanowiskach.

Jerzy RAJCH

(Foto: — Jan Zelman)

PROGRAM 12

Henryk Jasiński — MÓJ PROGRAM

SHARP

„IKS" był, jest i zawsze będzie Wasz! **Nie wszystkowiedzących, zadufanych w sobie redaktorów, lecz *pismem Czytelników*.**

Dlatego z przyjemnością publikujemy program, który opracował jeden z Was. CZEKAMY NA KOLEJNE!

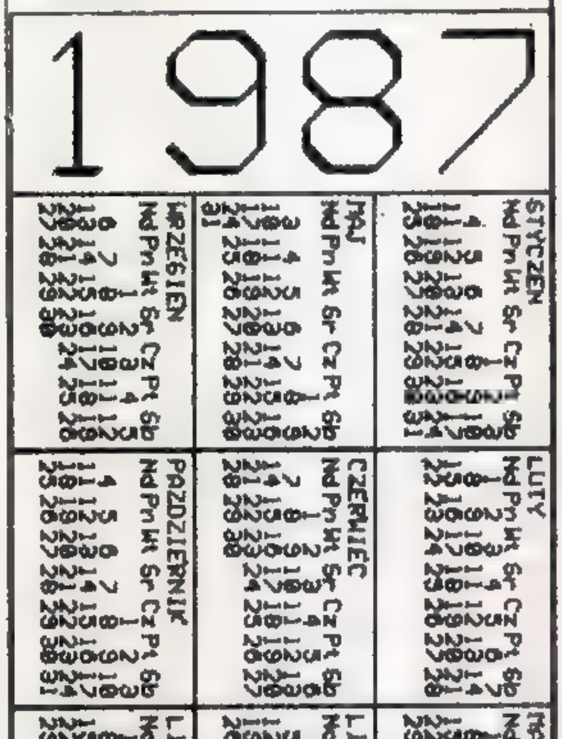
Henryk Jasiński z Warszawy pisze: „...Jesteście jedyną redakcją doceniającą fakt istnienia... tzw. Pocket Computers.

Pozostałe pisma, jeśli w ogóle je dostrzegają, to traktują komputery kieszonkowe jako ciekawostki, dodając obowiązkową uwagę, że nie można do nich dołączyć joysticka, jakby to decydowało o klasie komputera.

Pozwalam sobie przysłać... program drukujący kalendarz na dowolny rok...".

Tyle nasz Czytelnik. A oto jego program. Wywołuje go etykieta „A" (RUN 100).

```
100:"A":CLEAR :GRAPH
110:DIM A$(9)*1,B$(13)*1,M$(11)*11,L
   D(11),NM(11)
120:DATA "0","1","2","3","4","5","6"
   ,"7","8","9"
130:DATA "STYCZEN",31,1,"MAJ",31,5,"
   WRZESIEN",30,9,"LUTY",28,2
140:DATA "CZERWIEC",30,6,"PAZDZIERNI
   K",31,10,"MARZEC",31,3,"LIPIEC",
   31,7
150:DATA "LISTOPAD",30,11,"KWIECIEN"
   ,30,4,"SIERPIEN",31,8,"GRUDZIEN"
   ,31,12
160:DATA "N","d","P","n","W","t","S"
   ,"r","C","z","P","t","S","b"
170:FOR I=0TO 9
180:READ A$(I)
190:NEXT I
200:FOR M=0TO 11
210:READ M$(M),LD(M),NM(M)
220:NEXT M
230:INPUT "Rok ";R$:R=VAL R$
240:IF R<>INT ROR R<>ABS RGOTO 230
250:ROTATE 0:CSIZE 9:COLOR 1
260:GLCURSOR (0,-72):SORGN
270:FOR I=0TO 2
280:GLCURSOR (0+I,0+I):LPRINT R$
290:NEXT I
300:LINE (0,0)-(216,72),0,0,B
310:ROTATE 1:CSIZE 1
320:M=0:GLCURSOR (216,0):SORGN
330:FOR K=1TO 4
340:FOR L=1TO 3
350:M=M+1:B=NM(M-1):A=R
360:IF M<>4GOTO 410
370:X=0:IF A-INT (A/4)*4=0LET X=1
380:IF A-INT (A/100)*100<>0GOTO 400
390:X=0:IF A-INT (A/400)*400=0LET X=
   1
400:LD(M-1)=28+X
410:IF B-3<0LET A=R-1:B=B+13:GOTO 43
   0
420:B=B+1
430:C=INT (365.25*A)+INT (30.6*B)-
   INT (A/100)+INT (A/400)-427:O=C-
   INT (C/7)*7
440:GLCURSOR (-10,-4):LPRINT M$(M-1)
450:RESTORE 160
460:FOR I=0TO 13
470:READ B$(I)
480:NEXT I
490:E=-20:GLCURSOR (E,-4):GOSUB 700
500:FOR I=0TO 5
520:FOR J=1TO 7
530:Y=7*I+J-O:Z=INT (Y/10):U=Y-Z*10
540:IF Y<1OR Y>LD(M-1)LET B$(2*J-2)=
   " ":B$(2*J-1)=" ":GOTO 580
550:B$(2*J-2)=A$(Z)
560:IF Z=0LET B$(2*J-2)=" "
570:B$(2*J-1)=A$(U)
580:NEXT J
590:E=-20-I*9:GLCURSOR (E,-4):GOSUB
   700
600:NEXT I
610:LINE (0,0)-(-72,-102),0,0,B
620:GLCURSOR (-72,0):SORGN
630:NEXT L
640:GLCURSOR (216,-102):SORGN
650:NEXT K
660:ROTATE 0:TEXT :LF 4:END
700:FOR N=0TO 12STEP 2
710:IF N=0COLOR 3:GOTO 740
720:IF N=2COLOR 1:GOTO 740
730:IF N=12COLOR 2
740:B$=B$(N)+B$(N+1)
750:GLCURSOR (E,-4-N*7):LPRINT B$
760:NEXT N
770:RETURN
```



POCKET COMPUTER SHARP PC-1501

Tryb graficzny 8 dostarcza obrazów o najwyższej rozdzielczości, jakie możemy wytworzyć za pomocą Atari, z 320 punktami w poziomie i 192 (lub 160 dla rozszczepionego ekranu) w pionie. Ma jednak podstawową wadę: zezwala na używanie tylko jednego koloru. Instrukcja SETCOLOR 2 określa kolor i jaskrawość tła, a SETCOLOR 1 określa jaskrawość punktu lub linii wykreślonej za pomocą rozkazów PLOT i DRAWTO. Jedyną inną instrukcją SETCOLOR, która działa w trybie 8, jest SETCOLOR 4, ustawiająca kolor i jaskrawość ramki.

W szponach Atari (6)

Wielokolorowa grafika w trybie 8

Trudno kwestionować powyższe informacje, zaczerpnięte z podręcznika języka BASIC dla komputerów Atari, lecz nie zaszkodzi je sprawdzić. Dlatego po włączeniu komputera wprowadźmy instrukcję GRAPHICS 8. Otrzymamy rozszczepiony ekran z oknem tekstowym na dole. Wprowadźmy następnie **COLOR** 1: **PLOT** 5, **80:DRAWTO 315,80.** Na ekranie pojawi się linia pozioma, której kolor możemy zmieniać za pomocą instrukcji SETCOLOR

Spróbujmy poważnie podejść do badań. Za pomocą **SETCOLOR 2, 0, 0** ustawiamy czarne tło, a za pomocą **SETCOLOR 1, 0, 12** wybieramy linie mocno kontrastowe. Teraz **PLOT 160, 150:DRAWTO 160,5** kreśli linię pionową w kolorze niebieskim.

Jeśli otrzymany kolor nie jest niebieski, to pamiętajmy, że kolory w naszym telewizorze zależą od jego regulacji i warunków pracy. Chociaż kolory, które widzicie, mogą się różnić od opisanych, zasady i techniki ich otrzymywania pozostają takie same.

Wprowadźmy teraz instrukcje **PLOT 101, 150:DRAWTO 101,5.** Zobaczymy pionową linię w kolorze brązowym. Dlaczego otrzymaliśmy dwie linie w różnych kolorach? Jest to zależne od wartości współrzędnej **X**, czy jest parzysta, czy nieparzysta. Wszystkie linie pionowe z nieparzystą wartością **X** będą w jednym kolorze, a wszystkie z wartością **X** parzystą — w innym. Jest to prawdziwe także dla wszystkich punktów zapalonych za pomocą **PLOT**. Jeśli wykreślimy linię pod dowolnym kątem poprzez zapalenie tylko tych punktów, które mają wartości współrzędnej **X** parzyste lub nieparzyste, otrzymamy linię niebieską lub brązową.

Jeśli wykreślimy inną linię za pomocą **PLOT 160, 80 : DRAWTO 240, 120**, będzie ona czerwona. Wprowadzenie **PLOT 101, 80 : DRAWTO 180, 120** daje linię zieloną. Regułą jest, że wszystkie linie ukośne o kącie nachylenia określonym ułamkiem X/Y = 2/1 będą zielone lub czerwone, w zależności od współrzędnej X punktu początkowego.

Instrukcje **PLOT 10, 10 : DRAWTO 150, 150** dają linię szarą. Wszystkie linie wykreślone pod kątem X/Y = 1/1 będą szare.

Otrzymujemy zatem sześć podstawowych kolorów w trybie graficznym 8. Biały (linie poziome lub kilka linii jedna obok drugiej), brązowy i niebieski (linie pionowe), zielony i czerwony (linie ukośne) i szary. Pozostałe linie są mieszaniną tych podstawowych kolorów.

Jak to się ma zatem do informacji zawartych w podręczniku? Jaką wartość mogą mieć kolory, które otrzymuje się poprzez wykreślenie linii pod określonym kątem? Jak to się dzieje, że otrzymujemy tyle kolorów? Odpowiedzi na to należy szukać w strukturze kineskopu i sposobie tworzenia kolorów. Powierzchnia kineskopu pokryta jest szeregiem poziomych linii, które składają się z ciągów punktów niebieskich, zielonych i czerwonych. Punkty te zapalają się, jeśli trafi w nie strumień elektronów. Są one tak małe, że nie widzimy ich jako punkty, lecz tylko jako połączenie kolorów wielu punktów. Różnorodne barwy, jakie widzimy na ekranie telewizora uzyskuje się poprzez sterowanie intensywnością strumienia elektronów dla każdego punktu. Odpowiedni dobór niebieskich, zielonych i czerwonych punktów dostarcza pełnego zakresu kolorów

W trybie graficznym 8 linia pionowa jest tak wąska, że nie może pokryć wszystkich trzech kolorów jednocześnie. Kiedy współrzędna X linii pionowej jest parzysta, trafia głównie w punkty niebieskie. Kiedy współrzędna X jest nieparzysta, trafia w punkty czerwone i zielone, dając linię brązową. Rozdzielczość trybu graficznego 8 jest prawie tak dobra jak rozdzielczość ekranu telewizora! Ponieważ tak wąskie linie nie mogą uaktywnić wszystkich trzech punktów koloru, barwa linii nie może być efektywnie wyregulowana poprzez zrównoważenie intensywności świecenia punktów, jak to jest robione w trybach graficznych o niższej rozdzielczości (linie poziome zawsze pokrywają wszystkie kolory).

Podsumujmy zatem otrzymane wyniki:

1. Kreślenie jasnego koloru (o dużej jaskrawości) na ciemnym tle lub ciemnego na jasnym tle wzmacnia otrzymany efekt.
2. Kolor elementarnego punktu nie jest zmieniany przez jego współrzędną Y.
3. Kolor elementarnego punktu zależy nie tylko od ustawionego koloru, lecz także od tego, czy jego współrzędna X jest parzysta, czy nie i od koloru bezpośrednich sąsiadów w poziomie.
4. Rozdzielczość pozioma ma praktyczną granicę raczej 160 niż 320. Dwa sąsiednie poziome punkty elementarne zlewają się w jeden punkt w standardowym kolorze.

Podstawowe efekty kreślenia białych punktów na czarnym tle podsumowane są w poniższej tablicy. N jest ilością sąsiednich białych punktów w poziomie. X jest współrzędną tych punktów w terminach „parzysty", „nieparzysty". Kolor jest w przybliżeniu aktualnym kolorem tych punktów, przy założeniu normalnej regulacji monitora.

N	X	kolor
1	parzysty	niebieski
1	nieparzysty	brązowy
2	parzysty—nieparzysty	czerwony
2	nieparzysty—parzysty	szary
3	—	prawie biały
4	—	biały

Program 1 ilustruje występowanie kolorów poprzez kreślenie dwóch ciągów prawie pionowych białych linii na czarnym tle. Kolorowe poziome wstęgi utworzone są zgodnie z regułami z powyższej tablicy. W tym przykładzie nie powstaje biały kolor, ponieważ istnieją najwyżej pary sąsiednich „białych" punktów. Zauważmy, że paski o stałym kolorze tworzone są albo dlatego, że wszystkie „parzyste" punkty dają kolor niebieski, albo wszystkie „nieparzyste" dają kolor brązowy

```
PROGRAM NR 1

ZS 110 GRAPHICS 8:COLOR 1
RF 120 SETCOLOR 1,0,14:SETCOLOR 2
   ,0,0
JW 130 FOR X=0 TO 318 STEP 4
UF 140 PLOT X,0:DRAWTO X+1,159:NE
   XT X
JJ 150 FOR X=0 TO 308 STEP 4
BL 160 PLOT X,0:DRAWTO X+9,159:NE
   XT X
AV 199 REM ZMIANA KOLOROW
PB 200 OPEN #1,4,0,"K:"
NO 210 ? :? "WYBRANYM KOLOREM JES
   T TERAZ ";H
YH 220 ? "NACISNIJ KLAWISZ H DLA
   ZMIANY KOLORU"
VO 230 GET #1,X:IF X<>72 THEN 230
QX 240 H=H+1:IF H=16 THEN H=0
OH 250 SETCOLOR 2,H,0:GOTO 210
```

Aby umożliwić łatwą zmianę koloru, dodano linie 199 do 250, zachowując ustawienie zerowej jaskrawości dla tła i jaskrawości 14 dla kreślonych linii. Kolor tła może być prawie niewidoczny, lecz kolory poziomych wstęg mogą być zmieniane w znacznym stopniu. Najlepsze wyniki uzyskamy po wyregulowaniu jaskrawości i kontrastu telewizora na niską lub minimalną wartość

```
PROGRAM NR 2

IP 10 GRAPHICS 24
IA 20 SETCOLOR 1,0,14:SETCOLOR 2,
   0,0
Z 30 FOR X=0 TO 318 STEP 3
E 40 COLOR 1:PLOT 159,0:DRAWTO X
   ,191
NZ 60 NEXT X
TQ 70 GOTO 70
```

Program 2 wykorzystuje zmianę kolorów dla tworzenia wzorów mory. Aby zbadać nieco inny wzór z bardziej jednolitym rozkładem światła i zwiększonym kontrastem, dodajmy następującą linię:

50 COLOR 0:PLOT 159,0:DRAWTO X + 1, 191

Program krok po kroku kreśli czarną linię, która kasuje połowę (lub więcej) „białych" punktów, które były wykreślone w poprzednim kroku, w linii 40. Pomysł ten, po pewnych modyfikacjach, wykorzystany został w programie 3. Jest on tak zaprojektowany, aby osoby mało wprawione w posługiwaniu się klawiaturą nie musiały wprowadzać całego programu, aby zobaczyć co on robi.

PROGRAM NR 3

```
GC 100 GRAPHICS 0:POKE 82,5:POSIT
       ION 9,2:? "*** PIRAMIDA ***"
       :PRINT
RL 112 ? "USTAW KONTRAST SWOJEGO
       ODBIORNIKA"
GZ 113 ? "I JASKRAWOSC NA MINIMUM
       ."
LK 120 ? "CO WYBIERASZ:"
IU 130 ? "   0 PARAMETRY LOSOWE"
AX 140 ? "   1 PARAMETRY USTAWIAN
       E"
FM 150 TRAP 150:INPUT WYB:IF WYB=
       0 THEN GRAPHICS 24:GOTO 500
YK 160 IF WYB<>1 THEN 150
RM 170 ? :? "KRESLENIE TLA LINIAM
       I PIONOWYMI:"
FO 180 ? "   0 LINIE PARZYSTE"
SY 190 ? "   1 LINIE NIEPARZYSTE"
XA 200 ? "   2 BEZ TLA"
NA 210 TRAP 210:INPUT JASN:IF JAS
       N=2 THEN 230
UN 220 IF JASN<>0 AND JASN<>1 THE
       N 210
AP 230 ? :? "SPOSOB KRESLENIA PIR
       AMIDY:"
MD 240 ? "   0 OD SRODKA"
EK 250 ? "   1 OD LEWEJ NA PRAWO"
FG 260 TRAP 260:INPUT TRYB
XN 270 IF TRYB<>0 AND TRYB<>1 THE
       N 260
CM 280 ? :? "DLA WIERZCHOLKA PIRA
       MIDY UZYJ:"
TT 290 ? "   0 JEDNEGO PUNKTU"
JT 300 ? "   1 DWOCH PUNKTOW"
AN 310 TRAP 310:INPUT WIERZ
IZ 320 IF WIERZ<>0 AND WIERZ<>1 T
       HEN 310
JD 330 ? :? "ODSTEP PROMIENI OD W
       IERZCHOLKA:"
CR 340 ? "(LICZBA CALKOWITA: 2 DO
        6)"
TC 350 TRAP 350:INPUT ODST:IF ODS
       T<1 THEN 350
NS 360 ? :? "KRESLIC CIEMNE PIONO
       WE LINIE?"
FO 370 ? "   0 LINIE PARZYSTE"
SY 380 ? "   1 LINIE NIEPARZYSTE"
NP 390 ? "   2 BEZ LINII"
NG 400 TRAP 400:INPUT CIEM:IF CIE
       M=2 THEN 420
EL 410 IF CIEM<>1 AND CIEM<>0 THE
       N 400
LT 420 ? :? "JESLI KONIEC"
IX 430 ? "   0 PARAMETRY LOSOWE"
BO 440 ? "   1 UTRZYMAC WZORZEC"
XH 450 TRAP 450:INPUT UTRZ
QL 455 IF UTRZ=0 THEN GRAPHICS 24
       :GOTO 600
YX 460 IF UTRZ<>0 AND UTRZ<>1 THE
       N 450
SQ 470 POKE 82,2:? :? "WPROWADZAN
       IE WZORCA ZAKONCZONE"
PH 475 ? "NACISNIJ H ABY ZMIENIC
       KOLOR"
FB 480 ? :? "GOTOWE? NACISNIJ STA
       RT"
ET 490 IF PEEK(53279)<>6 THEN 490
VO 495 TRAP 40000:GRAPHICS 24:GOT
       O 600
SC 497 REM
CR 498 REM *** PARAMETRY LOSOWE *
       **
SI 499 REM
TI 500 JASN=INT(RND(0)*8)
SI 510 TRYB=INT(RND(0)*2)
BJ 520 WIERZ=INT(RND(0)*2)
ND 530 ODST=2+INT(RND(0)*5)
LD 535 IF TRYB=M AND WIERZ=A AND
       ODST=S THEN 510
QB 540 CIEM=INT(RND(0)*8)
FO 545 KOL=INT(RND(0)*16)
UX 550 POKE 77,0:REM ZEROWANIE TR
       YBU ZWRACANIA UWAGI
SD 597 REM
TF 598 REM *** WYKONANIE PROGRAMU
        ***
SJ 599 REM
HM 600 SETCOLOR 2,KOL,0:SETCOLOR
       1,0,14
DE 620 IF JASN>1 THEN 640
ZJ 630 COLOR 1:B=JASN:GOSUB 1000
DA 640 IF TRYB=0 THEN GOSUB 2000
EP 650 IF TRYB=1 THEN GOSUB 3000
RD 660 REM
UG 670 FOR K=191 TO 1 STEP -ODST
JN 680 COLOR 1:PLOT 159,WIERZ:DRA
       WTO 318,K:PLOT 159,WIERZ:DRAWT
       O 0,K
BO 690 COLOR 0:PLOT 159,0:DRAWTO
       318,K-1:PLOT 159,0:DRAWTO 0,K-
       1:NEXT K
XA 700 IF CIEM>1 THEN 720
PW 710 COLOR 0:B=CIEM:GOSUB 1000
DC 720 IF UTRZ=1 THEN 4000
XU 730 IF RND(0)<0.2 THEN GRAPHIC
       S 24
LU 740 M=TRYB:A=WIERZ:S=ODST
NK 750 GOTO 500
SH 997 REM
JZ 998 REM *** PODPROGRAM LINII P
       IONOWYCH
SN 999 REM
RI 1000 FOR K=B TO 319 STEP 2
HZ 1010 PLOT K,0:DRAWTO K,191:NEX
       T K
AF 1020 RETURN
KY 1998 REM
KZ 1999 REM
GM 2000 FOR K=0 TO 158 STEP ODST
ZX 2010 COLOR 1:PLOT 159,0:DRAWTO
        159+K,191:PLOT 159,0:DRAWTO 1
       59-K,191
UZ 2020 COLOR 0:PLOT 159,WIERZ:DR
       AWTO 160+K,191:PLOT 159,WIERZ:
       DRAWTO 158-K,191
HD 2030 NEXT K:RETURN
KW 2998 REM
LA 2999 REM
FB 3000 FOR K=0 TO 318 STEP ODST
JC 3010 COLOR 1:PLOT 159,WIERZ:DR
       AWTO K,191
NE 3020 COLOR 0:PLOT 159,0:DRAWTO
        K+1,191:NEXT K:RETURN
HZ 4000 REM
ID 4001 REM
ET 4010 OPEN #1,4,0,"K:"
BJ 4020 GET #1,X:IF X=72 THEN KOL
       =KOL+1
EQ 4030 IF KOL=16 THEN KOL=0
PI 4040 SETCOLOR 2,KOL,0:GOTO 402
       0
```

Część programu (linie 100 do 470) prawie w całości służy do ustawiania parametrów wzoru przez użytkownika, aby lepiej poznać w jaki sposób osiągane są różne efekty. W celu zmniejszenia liczby wprowadzanych linii, proszę podmienić pierwszą część programu pojedynczą linią: 100 GRAPHICS 24 Wprowadzenie rozpoczynamy od linii 500

Program w zasadzie nie wymaga dodatkowych wyjaśnień. Najważniejsze informacje są w nim zawarte. Należy jednak podkreślić kilka rzeczy. Linie 500 do 545 wybierają zestaw parametrów losowych dla wzoru, który ma być kreślony. Zmienne JASN i CIEM skojarzone są z podprogramem kreślenia zestawu linii pionowych w wierszu 1000 programu. Są to zmienne o podwójnym przeznaczeniu: jeśli są równe 0 lub 1, wtedy kreślony będzie zestaw linii „parzystych" lub „nieparzystych", lecz jeśli większa niż 1, podprogram nie będzie wywołany Zatem prawdopodobieństwo, że JASN spowoduje wywołanie podprogramu wynosi 0, 25, ponieważ losujemy liczbę całkowitą z przedziału 0 do 7. To samo dotyczy zmiennej CIEM

Zmienna JASN ustala kolorowe tło dla wzoru, lecz daje nieco inny efekt, jeśli wzór nie został zniszczony przez linię 730 Zmienna CIEM powoduje wykasowanie wszystkich kolorów za wyjątkiem czarnego zanim program ponownie przejdzie do wyboru nowego zestawu parametrów losowych. Linia 535 wraz z 740 umożliwiają upewnienie się, że nowe wartości zmiennych TRYB, WIERZ i ODST są inne niż stare. Linia 550 chroni przed trybem zwracania uwagi (attract mode) dopóki program działa na nowych wartościach zmiennych. Za wyjątkiem możliwości zatrzymania wzoru przez użytkownika (linie 450 i 720), nie ma żadnych opóźnień czasowych. Jeden cykl programu trwa około minuty, musi to być wystarczające dla zaobserwowania aktualnego wzoru. Jeśli chcemy przyjrzeć się dokładniej, działanie programu może być wstrzymane i wznowione za pomocą klawisza **CTRL 1**

PROGRAM NR 4

```
FC 50 GRAPHICS 0:POSITION 16,2:?
      "KLEJNOT":? :?
AY 55 ? "PO ZAKONCZENIU KRESLENIA
      --":?
FX 60 ? "WYLACZ SWIATLO W POKOJU,
       ABY"
YB 70 ? "UZYSKAC EFEKT ZABRUDZONE
      GO"
TR 80 ? "OKNA.":?
UH 90 ? "NACISNIJ KLAWISZ H DLA Z
      MIANY"
BY 95 ? "KOLORU I/LUB WYREGULUJ N
      ASYCENIE":? "KOLOROW.":?
EG 100 ? "BOK KWADRATU?"
GO 110 ? "(MAKSIMUM 4*47--TRY 4*4
       6)"
QQ 120 TRAP 150:? "4*";:INPUT BOK
       :BOK=4*BOK
YL 130 IF BOK<1 OR BOK>188 THEN 1
       50
MN 140 TRAP 40000:GOTO 300
DE 150 PRINT CHR$(253):GOTO 50
YO 300 REM *** DEFINIOWANIE ROGOW
        KWADRATU
HD 310 XDO=INT(159.5-BOK/2):XGO=X
       DO+BOK
GK 320 YDO=INT(95.5-BOK/2):YGO=YD
       O+BOK
RX 349 REM
UQ 350 REM *** KRESLENIE TLA ***
RN 355 REM
HN 360 GRAPHICS 24:SETCOLOR 2,0,0
       :SETCOLOR 1,0,14:COLOR 1
ES 410 FOR X=XDO TO XGO
IR 420 PLOT X,YDO:DRAWTO X,YGO:NE
       XT X
RY 449 REM
KU 450 REM *** KRESLENIE KLEJNOTU
        ***
RO 455 REM
VJ 460 COLOR 0:FOR K=0 TO BOK STE
       P 4
DJ 470 PLOT XDO,YDO:DRAWTO XGO,YG
       O-K
GT 480 PLOT XDO,YDO:DRAWTO XGO-K,
       YGO
BU 490 PLOT XGO,YDO:DRAWTO XDO,YG
       O-K
CH 500 PLOT XGO,YDO:DRAWTO XDO+K,
       YGO
XA 510 PLOT XGO,YGO:DRAWTO XDO,YD
       O+K
AM 520 PLOT XGO,YGO:DRAWTO XDO+K,
       YDO
YX 530 PLOT XDO,YGO:DRAWTO XGO,YD
       O+K
EN 540 PLOT XDO,YGO:DRAWTO XGO-K,
       YDO
GZ 550 NEXT K
SJ 599 REM
KG 600 REM *** ZMIANA KOLORU:NACI
       SNIJ KLAWISZ H ***
RG 605 REM
PH 610 OPEN #1,4,0,"K:"
FS 620 GET #1,X:IF X=72 THEN KOLO
       R=KOLOR+1
RM 630 IF KOLOR=16 THEN KOLOR=0
HR 640 SETCOLOR 2,KOLOR,0:GOTO 62
       0
MO 1000 GOTO 1000
```

Program 4 ilustruje zmiany kolorów za pomocą gęstego wzoru ciemnych linii kreślonych na jasnym tle. Równoodległe promienie kreślone są z każdego z czterech rogów kwadratu do dwóch przeciwległych boków. Zapalone punkty, pozostające po zakończeniu tego procesu tworzą figurę składającą się z czterech segmentów w kształcie trąby. Obraz staje się interesujący, jeżeli najpierw wyregulujemy telewizor na przyjemną kombinację kolorów, a następnie zgasimy światło w pokoju.

Uwaga: Litery z lewej strony numerów linii nie wchodzą w skład programu. Są to sumy kontrolne dla Edytora Basic'u.

Atari Writer — edytor tekstów

Szacuje się, że najbardziej uniwersalnym i najczęściej wykorzystywanym profesjonalnym zastosowaniem mikrokomputerów jest ich używanie do pisania i redagowania tekstów. Służące do tego programy noszą nazwę edytorów tekstowych (ang. text editors) lub programów przetwarzania tekstów (ang. word processors). Do typowych zastosowań takich programów należą: pisanie artykułów, referatów, pism urzędowych, tłumaczeń, itp.

Typowy program przetwarzania tekstów przekształca komputer w specyficzną maszynę do pisania. Tekst wprowadzamy do pamięci komputera za pomocą klawiatury, obserwując na ekranie czy nie ma w nim pomyłek. Gotowy tekst przeglądamy na ekranie, a po naniesieniu ewentualnych poprawek oraz sprawdzeniu poprawności układu (stronice, tytuły, wcięcia, odstępy itp.) przenosimy na papier za pomocą drukarki. Możemy go również przechowywać w pamięci zewnętrznej (głównie dyskowej) w celu sporządzenia nowej wersji lub też jako część większej całości.

Programy przetwarzania tekstów oferują dodatkowe możliwości, znacznie ułatwiające pracę.

- Program wykonuje za nas pewne czynności redakcyjne — dotyczy to głównie decyzji o przejściu do nowego wiersza, wyrównanie lewego i prawego marginesu, itp.
- Po wprowadzeniu zmian (poprawieniu błędów, zmianie układu zdań, przeniesieniu fragmentów tekstu w inne miejsca, itp.) nie przepisuje się całego tekstu od nowa. Wystarczy podać odpowiednie rozkazy oraz wpisać w określone miejsca nowe fragmenty tekstu.
- Ze zgromadzonych tekstów można tworzyć różne kombinacje, na przykład przed wydrukowaniem zmienić tę część, która dotyczy konkretnego adresata.
- Wydruk kolejnej wersji tekstu uzyskuje się bardzo szybko — jest to niezwykle istotne przy projektowaniu wyglądu zewnętrznego poszczególnych stron. Dodatkowych możliwości dostarczają drukarki mozaikowe: różne kroje czcionek, druk wytłuszczony, rozszerzony i inne.

Do dobrego tonu należy, aby każdy komputer posiadał własny edytor tekstów (najczęściej unikalny, niepodobny do innych). Dzięki temu również większość mikrokomputerów domowych wyposażona jest w takie programy. Są one zwykle mniej wyrafinowane niż te do zastosowań profesjonalnych, lecz realizują podobny zestaw podstawowych funkcji. Dla komputerów Atari 800 XL, 130 XE i 65 XE takie funkcje wykonuje Atari Writer. Nie jest on jedynym edytorem tekstów dla mikrokomputerów tej firmy. Jest jednym z prostszych. Dodatkową zaletą, ważną dla większości posiadaczy tych popularnych komputerów, jest możliwość wykorzystania pamięci taśmowej do przechowywania programu oraz zbiorów tekstowych.

W dalszej części spróbujemy przedstawić funkcje realizowane przez program Atari Writer oraz sposoby ich wykorzystania.

Podstawowym elementem na ekranie jest kursor, migająca pozioma kreska o szerokości znaku, który wskazuje miejsce działania programu: pisanie nowych liter, usuwanie lub zamianę już wprowadzonych, tworzenie akapitu, itp. Położenie kursora określane jest przez jego współrzędne — numer wiersza i numer kolumny w wierszu (L i C w oknie komunikatu — w prawym dolnym rogu ekranu). Kursor może poruszać się po całym tekście. Można przemieszczać go o wiersz (**CTRL↓** i **CTRL↑**) lub całą stronę (**OPTION↓** lub **OPTION↑**), o jeden znak w prawo lub w lewo (**CTRL→** lub **CTRL←**), na początek tekstu (**SELECT T**) lub jego koniec (**SELECT B**).

Tekst pisze się jak na zwykłej maszynie do pisania, linia po linii. Nie używa się klawisza **RETURN**. Podczas wprowadzania słowa, które przekracza prawy margines ekranu, program automatycznie przenosi to słowo na początek następnej linii. Klawisz RETURN ma specjalne znaczenie, używany jest do zakończenia akapitu (którego początek oznaczamy poprzez **CTRL P**). Może być również używany do wstawiania do tekstu pustych linii.

W każdej chwili wprowadzony tekst można zmienić lub poprawić. Dopiero przy tym można docenić zalety redagowania tekstu za pomocą komputera. Podczas pisania na maszynie można poprawić tylko pojedyncze litery w gotowym maszynopisie (za pomocą taśmy korektorskiej). Wstawienie dodatkowych wyrazów lub zdań wymaga przepisania tekstu. Za pomocą edytora tekstu można kasować, wstawiać, wymieniać dowolne litery, wyrazy lub całe zdania. W celu poprawienia błędu lub wymiany słowa lub zdania, należy najpierw wykasować błędne. Używamy do tego klawiszy **CTRL DELETE**. Po wykasowaniu wprowadzamy poprawkę. Program automatycznie robi miejsce na tyle znaków, ile potrzebujemy dodać. Komputer odsuwa tekst napisany uprzednio, nie wymazując go i robiąc miejsce dla wprowadzanych znaków.

Kolejne funkcje odnoszą się do bloków tekstu. Za pomocą **CTRL X** oznaczamy początek i koniec takiego bloku. Tak zaznaczony blok może zostać przeniesiony (**OPTION M**) lub powtórzony (**OPTION D**) w innym miejscu, wskazanym przez kursor, lub wykasowany (**OPTION DEL. BACK SP.**). Interesującą właściwością programu jest przechowywanie przenoszonego, powtarzanego lub kasowanego bloku tekstu w specjalnym buforze. Dzięki temu możliwe jest odtworzenie go (**START INSERT**) w przypadku błędnego wykasowania. Pojemność tego bufora jest jednak ograniczona (maksymalnie 30 linii ekranu) — dlatego wybrane bloki nie powinny przekraczać tej wielkości.

Przydatną funkcją jest wyszukiwanie podanego ciągu (o długości nie większej niż 25 znaków) i zastępowanie go innym. W celu użycia tej funkcji należy ustawić kursor na początku tekstu i nacisnąć **SELECT S**. Komputer pyta, jaki ciąg ma znaleźć (**SEARCH FOR:...**), czy zastąpić go innym (**REPLACE STRING, Y/N?**) i czym go ewentualnie zastąpić (**REPLACE WITH:...**). Można wymieniać podany ciąg krok po kroku lub w całym zbiorze naraz (**REPLACE GLOBAL, Y/N?**). Do pracy krokowej należy nacisnąć **N i RETURN**, wtedy program podmieni tylko pierwsze wystąpienie ciągu i zapyta, czy kontynuować poszukiwania (**CONTINUE SEARCHING, Y/N?**). Po naciśnięciu **Y i RETURN** program odnajduje następne wystąpienie podanego ciągu i daje możliwość wyboru. Możemy przerwać lub kontynuować poszukiwanie i wymianę do końca tekstu. Opisana funkcja może być używana do wyszukiwania i kasowania. W tym celu po pytaniu o nowy ciąg należy nacisnąć **RETURN**. Znaleziony ciąg zostanie zastąpiony odstępem (spacją). Wyszukiwanie i wymiana mogą być przerwane w dowolnej chwili poprzez naciśnięcie klawisza BREAK. Należy pamiętać, że program rozpoczyna poszukiwania od aktualnej pozycji kursora.

Długość tekstu, który możemy wprowadzić do komputera, zależy od ilości wolnej pamięci. Podczas wprowadzania lub poprawiania tekstu dobrze jest sprawdzać od czasu do czasu, ile mamy jeszcze wolnego miejsca. W tym celu należy nacisnąć **OPTION F**. Program wyświetla odpowiedź wyrażoną w bajtach. W celu oszacowania ile to stronic, możemy przyjąć, że każda strona z podwójnymi odstępami wymaga 1500 bajtów.

Kolejną grupą funkcji są operacje na zbiorach. Tekst wprowadzony do komputera może być zapisany w pamięci zewnętrznej (dyskowej lub taśmowej) w celu przechowania dla późniejszego poprawienia, albo jako część większej całości. W tym celu należy przejść do menu (klawisz **ESC**), wybrać opcję przechowania zbioru (Save File). Program poprosi nas o podanie nazwy urządzenia i zbioru (**SAVE DEVICE: FILENAME**). Do stacji dysków trzeba włożyć sformatowaną dyskietkę i napisać **D**: nazwa zbioru. Nazwa może mieć długość do ośmiu znaków. Jeśli dyskietka nie jest sformatowana, program dysponuje opcją Format Disk, którą należy wywołać przed zapisaniem zbioru tekstowego.

W celu przechowania zbioru na taśmie magnetycznej musimy podać C: w odpowiedzi na propozycję podania nazwy urządzenia. Program zarząda włączenia magnetofonu na zapis. Należy włożyć kasetę, przewinąć w odpowiednie miejsce (zanotować stan licznika), wcisnąć klawisze **RECORD i PLAY** oraz **RETURN**.

Wprowadzanie zbioru tekstowego do pamięci komputera odbywa się podobnie. Z menu programu należy wybrać opcję wprowadzania (Load File). Po propozycji podania nazwy urządzenia i zbioru (**LOAD DEVICE: FILENAME**) musimy sprawdzić, czy odpowiednia dyskietka jest włożona do stacji dysków. Wtedy piszemy **D: nazwa zbioru**. Jeśli zbiór tekstowy przechowywany jest na taśmie magnetycznej, niezbędne jest ustawienie taśmy na początek zbioru (według stanu licznika) i zapisanie C:. Po sygnale dźwiękowym należy nacisnąć klawisze **PLAY** oraz **RETURN**.

W przypadku posiadania stacji dysków

przydatne mogą być dwie dodatkowe funkcje programu wybierane z menu: katalog zbiorów i kasowanie zbioru. W celu otrzymania katalogu zbiorów należy włożyć dyskietkę do stacji dysków i wybrać opcję **„Index of Disk Files"**. Program wyświetli katalog zbiorów w kolejności alfabetycznej nazw. Pokaże także, ile sektorów zajmuje każdy zbiór i ilość wolnych sektorów na dyskietce. Jednocześnie może być wyświetlone 21 nazw zbiorów, jeśli jest ich więcej następuje przesuwanie tekstu do góry, można je zatrzymać przez naciśnięcie klawisza spacji. Ponowny przesuw uruchamiany jest również klawiszem spacji. Katalog zbiorów może być wydrukowany po naciśnięciu **Y i RETURN** w odpowiedzi na pytanie **PRINT DIRECTORY, Y/N?**

W celu wykasowania zbioru musimy upewnić się, czy jest on na dyskietce włożonej do stacji dysków, przejść do menu programu i wybrać opcję kasowania zbioru. Następnie trzeba podać nazwę urządzenia i nazwę zbioru (**DELETE DEVICE: FILENAME**). Program upewnia się, czy na pewno chcemy wykasować ten zbiór (**FILE DELETE — ARE YOU SURE, Y/N?**). Po naciśnięciu **Y i RETURN** program wymazuje określony zbiór z dyskietki.

Możliwe jest również łączenie zbiorów tekstowych zapamiętanych na dyskietce lub kasecie z tekstem przechowywanym aktualnie w pamięci komputera. W tym celu należy ustawić kursor w miejscu, gdzie chcemy dołączyć tekst ze zbioru i nacisnąć **OPTION L**. Następnie trzeba podać nazwę urządzenia i zbioru. Dalsze postępowanie jest takie samo jak podczas wprowadzania zbioru.

Trzecią grupę tworzą funkcje przygotowania wydruku. Za pomocą programu Atari Writer możemy wydrukować tekst w dowolnej postaci, dzięki użyciu różnorodnych rozkazów formowania. Pewne z nich, wprowadzane w bloku formowania wydruku u góry ekranu, sterują globalnym kształtowaniem tekstu. Inne, wprowadzane wewnątrz tekstu, podczas tworzenia go lub poprawiania, używane są do określania odstępstw od ogólnego formatu.

Każda litera w negatywie, występująca u góry ekranu, reprezentuje rozkaz, który zmienia formowanie całego tekstu. Liczba z prawej strony litery jest wartością redagowania dla tego rozkazu. W celu ustawienia własnych wartości musimy przesunąć kursor do bloku formowania wydruku, wykasować liczbę, którą chcemy zmienić i wprowadzić nową. W ten sposób możemy ustawić własne wartości dla marginesu dolnego (**B**) i górnego (**T**), odstępu między akapitami (**D**), wcięcie akapitu (**I**), wyrównanie do prawego marginesu (**J**), margines lewy (**L**) i prawy (**R**), odstęp między liniami (**S**), długość stronicy (**Y**) oraz sposób wydruku (**G**).

Powyższe rozkazy mogą być również wprowadzone bezpośrednio do tekstu, można je wprowadzać pojedynczo lub łącznie, podczas tworzenia lub poprawiania tekstu. Program wyświetla odpowiednią literę i liczbę w negatywie (nie występują one na wydruku).

Możliwe jest umieszczenie, na każdej stronicy wydruku, jedno- lub dwuliniowego nagłówka lub stopki. W szczególności można otrzymać numery drukowanych stronic. Dla określenia nagłówka musimy nacisnąć **CTRL H**, wprowadzić tekst i nacisnąć **RETURN**. Definiowanie stopki odbywa się podobnie, tylko naciskamy **CTRL F**. Domyślnie są one wyrównane do lewego marginesu drukowanej stronicy. Możemy jednak umieścić je na środku (po wprowadzeniu **CTRL C**) lub wyrównać do prawego marginesu (po wprowadzeniu **CTRL C CTRL C**).

W celu numerowania drukowanych stronic musimy nacisnąć @ (Shift 8) w tym miejscu nagłówka lub stopki, w którym ma wystąpić numer stronicy. Jeśli chcemy mieć tylko numery stronic, wprowadzamy @ jako cały tekst nagłówka lub stopki.

Możliwe jest obejrzenie uformowanego tekstu, w takiej postaci, jaka będzie na wydruku, bez drukowania. Służy do tego funkcja przeglądania wydruku, do której przechodzimy po naciśnięciu **OPTION P**. Najpierw musimy określić typ posiadanej drukarki. Następnie program pyta, czy chcemy przeglądać cały dokument, czy tylko wybrane stronice (**PREVIEW WHOLE DOCUMENT, Y/N?**). Naciskamy **Y i RETURN**, gdy chcemy przeglądać cały tekst lub **N i RETURN**, aby obejrzeć wybrane stronice (musimy wprowadzić numery stronic początkowej i końcowej). Po uformowaniu tekstu program wyświetla na ekranie lewy, górny róg pierwszej stronicy. Używając klawiszy sterowania kursorem możemy obejrzeć cały tekst (wskaźniki **L** i **C** pokazują położenie kursora na zredagowanej stronie, nie na ekranie). Należy pamiętać, że pewne tryby wydruku i odstępy między liniami i akapitami nie są dokładnie takie jak na wydruku.

W czasie przeglądania wydruku nie można poprawiać tekstu. Można to zrobić dopiero po przejściu do trybu poprawiania (poprzez naciśnięcie **START**).

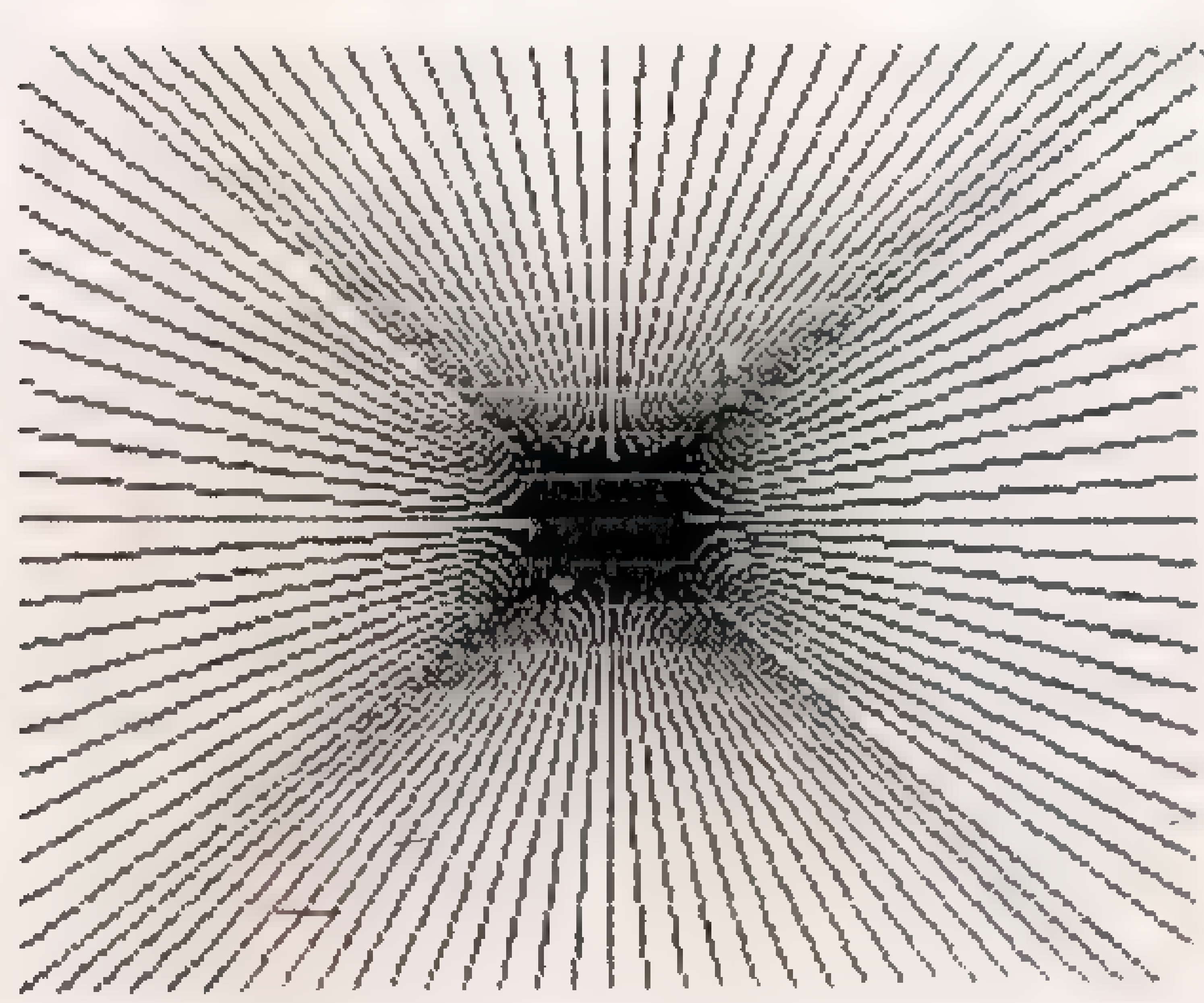
Po wszechstronnym sprawdzeniu przygotowanego tekstu możemy przystąpić do drukowania. W tym celu musimy przejść do menu programu i wybrać opcję **Print File**. Podczas pierwszego wybrania tej pozycji musimy określić typ posiadanej drukarki. Następnie udzielamy odpowiedzi, czy chcemy drukować cały tekst (**PRINT WHILE DOCUMENT, Y/N?**). Po naciśnięciu **N i RETURN** możemy drukować wybrane stronice po podaniu numeru początkowego i końcowego. Musimy podać jeszcze liczbę drukowanych egzemplarzy (od 1 do 99).

Z powyższego, dosyć pobieżnego, przeglądu funkcji programu wynika, że posiada on spore możliwości w zakresie redagowania tekstów. Pewną niedogodnością jest konieczność pamiętania wszystkich funkcji i odpowiadających im poleceń. Program w niewielkim tylko stopniu służy nam pomocą. Poważniejszymi wadami, zresztą powszechnymi wśród zachodnich edytorów tekstów są: brak polskich liter na ekranie oraz w repertuarze znaków firmowej drukarki (Atari 1027) i poważne kłopoty w ich użyskaniu w tym programie. Pomijamy już takie drobiazgi, jak niezbyt ładny krój czcionki (tworzony w polu 7 × 7), podniesienie do góry małych liter z ogonkami, mała liczba znaków w linii ekranu (40), itp. Pomimo tych wad program może pomóc w poznawaniu możliwości edytorów tekstu oraz służyć do modyfikowania programów w języku **BASIC** (przechowanych za pomocą instrukcji **LIST**).

Tomasz MROWIEC,
Ludwik PIELA

GRAFIKA NA SPECTRUM

```
 20 LET z=8
 30 FOR i=0 TO 255 STEP z
 40 PLOT i,0
 50 DRAW 255-2*i,175
 60 NEXT i
 70 FOR i=0 TO 175 STEP z
 80 PLOT 0,i
 90 DRAW 255 ,175-2*i
100 NEXT i
```



ATARI

PROGRAM 13

MINI-DOS

Każdy programista, wykorzystujący wbudowany w ROM mikrokomputera ATARI 800XL translator języka BASIC z dyskowym systemem operacyjnym (DOS), styka się często z irytującą sytuacją. Dochodzi do niej wtedy, gdy po zakończeniu pisania programu, programista chce zapisać go do zbioru na dyskietce, lecz nie wie, czy jest na niej wystarczająco dużo miejsca, lub musi użyć nowej dyskietki, lecz ta nie jest sformatowana. Mało tego, czasami powstaje problem, jaką nazwę nadać nowo tworzonemu zbiorowi, aby nie powtórzyć nazwy zbioru zapisanego już na dyskietce. W takiej sytuacji nie pozostaje nic innego, jak „wejść pod DOS-a", składując najpierw zawartość pamięci do zbioru MEM.SAV, sformatować nową dyskietkę, bądź „obejrzeć" listę zbiorów na dyskietce, na której chcemy zapisać nowy program, następnie wrócić do BASIC-a, odtwarzając zawartość zbioru MEM.SAV. Wszystkie te operacje pochłaniają wiele cennego czasu. Analogicznie wygląda problem, gdy w opisanej sytuacji chcemy skasować zbiór, zmienić jego nazwę, zabezpieczyć zbiór przed przypadkowym zniszczeniem (LOCK), lub zdjąć zabezpieczenie ze zbioru (UNLOCK).

Bardzo pomocnym w takich sytuacjach jest prezentowany poniżej program o nazwie MINI — DOS. Umożliwia on wykonywanie z poziomu BASICa funkcji normalnie dostępnych w pakiecie DUP (Disc Utility Package) DOS-a. Zasada działania programu jest bardzo prosta. Gdy zostanie on uruchomiony (RUN) zapisuje na dyskietce pięć zbiorów o nazwach: „DIR", „DELETE", „LOCK", „UNLOCK", „RENAME". Do każdego z wymienionych zbiorów program wprowadza po kilka linii programu w języku BASIC, realizujących odpowiednią funkcję. Wprowadzone do zbiorów linie programowe nie są numerowane. Efekt tego rozwiązania jest taki, że gdy zapiszemy komendę np:

ENTER „D:DIR" RETURN w negatywie to komputer zacznie wprowadzać do pamięci RAM linie programowe, zapisane w zbiorze „DIR", a że nie mają one numerów, komputer będzie je natychmiast wykonywał, nie niszcząc aktualnej zawartości RAM-u.

MINI — DOS ma następujące opcje:

E. „D:DIR" — listuje na ekranie wykaz zbiorów na dyskietce.
E. „D:RENAME" — zmienia nazwę zbioru.
E. „D:DELETE" — kasuje zbiór.
E. „D:LOCK" — „nakłada zabezpieczenie" na zbiór.
E. „D:UNLOCK" — „zdejmuje zabezpieczenie" ze zbioru.

Nazwy opcji zostały specjalnie dobrane w języku angielskim, ponieważ najbardziej kojarzą się z odpowiednimi opcjami z MENU DOS-a.

Program bardzo łatwo można rozbudować, wprowadzając do niego dodatkowe funkcje. Np., gdy zastąpimy linię 130 linią poniższą:

130 RESTORE 120:GOSUB 140GOTO 200

i dodamy dodatkowe linie 200—240

200 CLOSE#1:OPEN#1,8,0, „D:FORMAT"
210 DATA ?:? „WŁÓZ DYSKIETKE DO FORMATOWANIA":
? „I NACISNIJ START ! CZEKAM..."
220 DATA FOR I=1 TO 3000:IF PEEK (532
79) <> 6 THEN NEXT I:END
230 DATA ?„FORMATUJE...";XIO 254&#1&0&0&"D: „:?:?"...
FORMATOWANIE ZAKONCZONE !„ , —1
240 RESTORE 210:GOSUB 140

to program zapisze na dyskietce sześć zbiorów (pięć wcześniej wymienionych i szósty o nazwie „FORMAT"), a komenda E. „D:FORMAT RETURN umożliwi formatowanie dyskietki. Ale UWAGA! zalecam szczególną uwagę przy wprowadzaniu linii 200—240 i ostrożność przy ich testowaniu. Jeden drobny błąd może spowodować zniszczenie cennych informacji zapisanych na dyskietce. Postępując podobnie, można dobudować do MINI—DOS-a funkcję kopiowania zbiorów na dyskietce.

Po uważnym wprowadzeniu do pamięci mikrokomputera wszystkich linii programu należy zapisać go na oddzielną dyskietkę. Program najlepiej wykorzystywać według poniższej zasady. Zawsze po sformatowaniu nowej dyskietki, należy załadować MINI—DOS do pamięci i uruchomić go wkładając do stacji dysków nowo sformatowaną dyskietkę. MINI—DOS zapisze na niej pięć lub sześć (gdy dopisane zostaną linie 200—240) zbiorów o wcześniej wymienionych nazwach. Od tego momentu, gdy dyskietka ta będzie znajdowała się w napędzie dyskowym, będzie można używać opcji MINI—DOS-a z poziomu translatora BASIC-a. Wynika stąd wniosek, że chcąc mieć zawsze „pod ręką" powyższe opcje, zbiory tworzone przez MINI—DOS powinny znajdować się na każdej dyskietce, której będziemy używać pracując z translatorem BASIC-a. Jest to wniosek słuszny. Nie należy z tego powodu martwić się o racjonalne gospodarowanie pojemnością dyskietki. Zbiory zapisywane przez MINI—DOS zajmują razem 10—12 sektorów (dla DOS 2.5 każdy zbiór zajmuje dwa sektory), jest to naprawdę niewielka strata, zważywszy na wygodę programowania, jakiej dostarcza ten program.

opracował H. KRASUSKI

```
1 REM **************************
2 REM MINI - DOS
3 REM **************************
20 GRAPHICS 0:? "PROSZE CHWILE P
OCZEKAC..."
100 DIM A$(200)
110 OPEN #1,8,0,"D:DIR"
120 DATA CLR:CLOSE#1:DIM A$(20):
OPEN #1&6&0&"D:*.*"
125 DATA ? CHR$(28);"ZBIORY NA D
YSKIETCE":?:FOR X=1 TO 64:INPUT
#1;A$:? A$
126 DATA :IF A$(1&1)=" " OR A$(1
&1)="*" THEN NEXT X,-1
130 RESTORE 120:GOSUB 140:GOTO 2
60
140 READ A$:I=LEN(A$):IF A$="-1"
 THEN RETURN
150 FOR X=1 TO I
160 IF A$(X,X)="&" THEN A$(X,X)=
","
170 NEXT X
180 ? #1;A$:GOTO 140
250 REM
260 CLOSE #1:OPEN #1,8,0,"D:DELE
TE"
270 DATA CLR:CL.#1:CL.#2:DIM A$(
12)&B$(14):O.#1&4&0&"E:":
271 DATA ? CHR$(29);"PODAJ NAZWE
 ZBIORU":? "KTORY MA BYC SKASOWA
NY!"
280 DATA ?CHR$(28);"NAZWA ZBIORU
: ";:I.#1;A$:B$(1&2)="D:":X=LEN(
A$):B$(3&X+2)=A$(1&X)
281 DATA :X.33&#2&0&0&B$:?:?A$;"
 SKASOWANY!"
285 DATA -1
290 RESTORE 270:GOSUB 140
310 CLOSE #1:OPEN #1,8,0,"D:LOCK
"
320 DATA CLR:CL.#1:CL.#2:DIM A$(
12)&B$(14):O.#1&4&0&"E:":?CHR$(2
9);"PODAJ NAZWE ZBIORU"
321 DATA :?"KTORY MA BYC ZABEZPI
ECZONY"
330 DATA ?CHR$(28);"NAZWA ZBIORU
: ";:I.#1;A$:B$(1&2)="D:":X=LEN(
A$):B$(3&X+2)=A$(1&X)
331 DATA :X.35&#2&0&0&B$:?:?A$;"
 ZABEZPIECZONY
340 DATA -1
350 RESTORE 320:GOSUB 140
370 CLOSE #1:OPEN #1,8,0,"D:UNLO
CK"
380 DATA CLR:CL.#1:CL.#2:DIM A$(
12)&B$(14):O.#1&4&0&"E:":? CHR$(
29);"PODAJ NAZWE ZBIORU"
381 DATA :?"KTORY MA BYC ODBEZPI
ECZONY"
390 DATA ? CHR$(28);"NAZWA ZBIOR
U: ";:I.#1;A$:B$(1&2)="D:":X=LEN
(A$):B$(3&X+2)=A$(1&X)
391 DATA :X.36&#2&0&0&B$:?:?A$;"
 ODBEZPIECZONY
400 DATA -1
410 RESTORE 380:GOSUB 140
430 CLOSE #1:OPEN #1,8,0,"D:RENA
ME"
440 DATA CLR:CL.#1:CL.#2:DIM A$(
25)&B$(27):OPEN#1&4&0&"E:":B$(1&
2)="D:"
445 DATA ? CHR$(29);"PODAJ&ODDZI
ELAJAC PRZECINKIEM&":?"STARA I N
OWA NAZWE ZBIORU!"
450 DATA ? CHR$(28);"STARA NAZ.,
NOWA NAZ.: ";:I.#1;A$:X=LEN(A$):
B$(3&X+2)=A$(1&X):X.32&#2&0&0&B$
451 DATA :?:?"NAZWA ZBIORU ZMIEN
NA
455 DATA -1
460 RESTORE 440:GOSUB 140
500 END
```

PROGRAM 14

ATARI

Program kopiujący

Program umożliwia kopiowanie programów napisanych w języku wewnętrznym komputera. Po uruchomieniu instrukcją RUN, program sygnalizuje pojedynczym sygnałem (beep) gotowość do wczytywania oryginału z kasety. Naciśnięcie, np. klawisza RETURN, powoduje otwarcie kanału 1 w L.175 na czytanie. Gdy czytanie programu zostanie zakończone, komputer powiadomi podwójnym sygnałem dźwiękowym, że może rozpocząć wykonywanie kopii. Nastąpi to w momencie otwarcia kanału 1 jako wyjścia w L.260, przez naciśnięcie np. klawisza RETURN. Po wykonaniu kopii następuje powrót do sytuacji umożliwiającej wprowadzenie następnego programu.

J. J.

```
65 REM ********************
70 REM *                  *
75 REM *PROGRAM KOPIUJACY*
80 REM *                  *
85 REM *  KASETA-KASETA   *
90 REM *                  *
95 REM ********************
100 DATA 104,174,138,2,134,61,16
0,0,162,0,185,0,4,129,203,200,23
0
105 DATA 203,208,2,230,204,196,6
1,240,3,76,10,6,96
110 DATA 104,169,128,133,61,160,
0,162,0,161,203,153,0,4,200,230
115 DATA 203,208,2,230,204,196,6
1;240,3,76,9,6,198,61,96
120 GRAPHICS 17
125 P=PEEK(742)*256+PEEK(741)-PE
EK(145)*256+PEEK(144)-1500
130 IF P>32767 THEN P=32767
135 DIM P$(P):P$(1)=CHR$(0):P$(P
)=CHR$(0):P$(2)=P$
140 POKE 203,ADR(P$)-(INT(ADR(P$
)/256)*256):POKE 204,INT(ADR(P$)
/256)
145 FOR A=1536 TO 1565:READ D:PO
KE A,D:NEXT A
150 TRAP 200
155 ? #6;" CZYTANIE PROGRAMU"
160 ? #6;" Z KASETY"
165 ? #6:? #6;" wcisnij dowolny
"
170 ? #6;" klawisz"
175 OPEN #1,4,255,"C:":B1=0
180 FOR A=B1 TO P STEP 128
185 GET #1,W:B1=B1+128:X=USR(153
6)
190 NEXT A:? #6;"brak pamieci RA
M do"
195 ? #6;"skopiowania programu":
END
200 IF PEEK(195)=138 THEN CLOSE
#1
205 GRAPHICS 17
210 ? #6;" WGRYWANIE PROGRAMU"
215 ? #6;" NA KASETE"
220 ? #6:? #6;" wcisnij dowolny"
222 ? #6;" klawisz"
225 GOTO 235
230 ? "BLAD - ";PEEK(195):END
235 RESTORE 110
240 FOR A=1536 TO 1566
245 READ D:POKE A,D:NEXT A
250 POKE 203,ADR(P$)-(INT(ADR(P$
)/256)*256)
255 POKE 204,INT(ADR(P$)/256)
260 CLOSE #1:OPEN #1,8,255,"C:"
265 FOR A=0 TO B1 STEP 128
270 X=USR(1536)
275 P=ASC(P$(A+128)):PUT #1,P
280 NEXT A
285 CLOSE #1:GRAPHICS 17
290 ? #6:? #6;" PROGRAM"
295 ? #6;" SKOPIOWANY !"
300 FOR C=0 TO 1000:NEXT C
305 RUN
```

Prawie trzy lata temu, wiosną 1984 roku wdarł się przebojem na europejski rynek mikrokomputerowy AMSTRAD CPC 464. Mało znana firma, która dotąd zajmowała się produkcją domowego sprzętu elektronicznego, zmieniła swoje zainteresowania i natychmiast odniosła ogromny sukces. AMSTRAD stał się popularny w Wielkiej Brytanii, Francji, w Hiszpanii, a nawet w Stanach Zjednoczonych. A niemieckojęzyczną częścią Europy „zajął się" równolegle jego RFN-owski bliźniak SCHNEIDER.

POGWARKI O CPC 464

Znakomicie pamiętam moje pierwsze spotkanie z tymi urządzeniami. Odbyło się ono podczas Dni Kultury Mikroinformatycznej w klubie Abakus. Młodzi ludzie prezentujący owe komputery budzili ogromne zainteresowanie oraz cichą zazdrość zarówno zaawansowanych „mikrofanów", jak i przybyłych incognito informatyków. Te dwadzieścia kilka miesięcy, które minęły od tamtego czasu — prawie epoka w świecie mikrokomputerów — sprawiły, że poczciwa „464" mocno się zestarzała, zdążyła jednak zdobyć dużą popularność również w Polsce. Nic w tym dziwnego, gdyż maszyna to nie byle jaka, bez nadzwyczajnych, nie zawsze sprawdzonych nowinek technicznych, ale i bez większości błędów, jakie dotąd popełniali konkurenci. Słowem — „licząca solidność". W owym czasie parametry techniczne stawiały ją w czołówce mikrokomputerów domowych. Bardzo dobry interpreter BASICa, niewygórowana cena oraz to, że był sprzedawany w zestawie z magnetofonem i monitorem (co ogranicza do niezbędnego minimum nieporozumienia z domownikami i sprzętem) pozwalały sądzić, że długo jeszcze świat (a przynajmniej Europa) będzie stać przed nią otworem. Tym bardziej iż wspomniane zalety AMSTRADa oraz termin promocji sprzętu przyczyniły się w sporym stopniu do chłodnego przyjęcia mikrokomputerów standard MSX. Zagrożenie nadeszło z najmniej oczekiwanej strony. W kilka miesięcy później AMSTRAD wypuścił kolejny model — CPC 664, a po roku CPC 6128. *„Chcąc wygrywać z innymi, musimy być konkurencyjni dla siebie samych"* — usprawiedliwiał się w jednym z wywiadów Alan Sugar — naczelny animator firmy... i w ciągu trzech lat przygotował do produkcji kolejne typy mikrokomputerów: PCW 8256, PCW 8512 oraz „hit" ostatnich miesięcy IBM i podobny PC 1512. We wszystkich modelach, z wyjątkiem ostatniego, widać duży wpływ rozwiązań przyjętych w „czterysta sześćdziesiątce czwórce". Szczególnie zbliżone do niej są modele CPC 664 i CPC 6128, będące dla niej coraz mocniejszą konkurencją.

Nie będę pisał, jaka to porządna komputerowa rodzina — po prostu trochę się zżyliśmy. Sądzę zresztą, że nie tylko my... Pozwolę sobie zatem, zadedykować wszystkim *„wspaniałym mężczyznom (i nie tylko) na tych wspaniałych maszynach"* kilka kolejnych pogawędek poświęconych AMSTRADowi CPC 464. Ta niezobowiązująca forma umożliwia mi dosyć swobodne potraktowanie tematu

POGAWĘDKA PIERWSZA — COŚ NIECOŚ O „ŻELASTWIE"

Na początku, zamiast cytować starożytnych, przytoczę wypowiedź Bruce'a Goddena z Locomotive Software, dotyczącą założeń, jakie postawiono konstruktorom pierwszego AMSTRADa. *„Naszym celem było stworzenie wyrafinowanego komputera, łączącego dwie rzadko chodzące w parze cechy: jak najmniejszy koszt HARDWAREu i możliwie największą szybkość pracy. Byliśmy także zobowiązani udostępnić wszystkie możliwości maszyny programistom piszącym w BASICu".*

Przyjęto, że sercem (a może raczej mózgiem) komputera będzie tani, bardzo popularny i co istotne, dobrze znany mikroprocesor Z80A. Wybór tego układu był podyktowany nie tylko jego zaletami technicznymi, chociaż znam kilku fachowców, którzy do dziś uważają go za najlepszy, popularny procesor ośmiobitowy. Liczono przede wszystkim na szybkie pozyskanie oprogramowania. Już na etapie projektowania zakładano, że w zestawach z dołączoną pamięcią dyskową będzie powszechnie wykorzystywana jedna z wersji systemu operacyjnego CP/M. Konsekwentnie — dyskietka z CP/M 2.2 była dodawana przy zakupie stacji dysków. Niestety, brak konsekwencji dał znać o sobie przy doborze standardu nośnika. Zastosowano zupełnie nowy i drogi (co prawda dobrze zabezpieczony) dysk 3-calowy, zamiast powszechnie używanej „piątki" Muszę wyznać, że do dziś nie wybaczyłem tego posunięcia Alanowi Sugarowi.

Wróćmy jednak do środka, do „kości" — ich powiązania funkcjonalne przedstawione zostały na zamieszczonym obok schemacie.

Ponieważ Z80 jest układem powszechnie znanym i doczekał się odpowiedniej „literatury" w języku polskim, przypomnę tylko, że jego praca w AMSTRADzie jest synchronizowana sygnałem zegarowym o częstotliwości 4 MHz (±0,1%).

Najważniejszym, oczywiście po mikroprocesorze, układem komputera jest **Video** GATE ARRAY (często zdrobniale nazywany GATE ARRAY). Można by przyrównać go do ULA w ZX SPECTRUM. Zawarte w nim układy wykonują elementarne działania logiczne i realizują funkcję, od których bezpośrednio zależą niemal wszystkie operacje komputera. GATE ARRAY po prostu koordynuje pracę systemu. W szczególności zaś uaktywnia lub blokuje pamięć ROM, nadzoruje pracę w określonym trybie graficznym, a co się z tym wiąże ponosi odpowiedzialność za „gospodarowanie". W powiązaniu z kontrolerem monitora ekranowego (układ HD 6845S CRTC) generuje sygnały video, synchronizuje i wymusza zmiany obrazu, zgodnie z potrzebami aktualnie realizowanego programu. Istotną funkcją GATE ARRAY jest także zerowanie odpowiednich rejestrów i uaktywnianie procedur obsługi w razie wyłączenia zasilania lub zerowania mikrokomputera przez jednoczesne naciśnięcie klawiszy CTRL, SHIFT i ESC.

Problem generowania sygnałów akustycznych rozwiązano w AMSTRADzie, stosując układ scalony AY—3—8912 firmy General Instruments, oznaczony na schematach literami PSG (Programable Sund Generator). Ma on trzy niezależne kanały dźwięku oznaczone literami A, B i C oraz pseudolosowy generator szumu, dołączany do dowolnego z nich. Kanały A i C to dwa oddzielne tory stereofoniczne, a dźwięk z kanału B jest miksowany z sygnałami w obydwu torach. Efekt stereofonii można uzyskać jedynie przy użyciu zewnętrznego wzmacniacza. Sygnał wprowadzany jest wówczas przez łącze zawsze „małym JACKiem" Wbudowany w komputer głośnik współpracuje jednocześnie ze wszystkimi kanałami, a jego parametry znacznie obniżają jakość generowanego dźwięku. I jeszcze jedno, sygnały akustyczne, jakie powstają podczas wczytywania programów z taśmy kasetowej, nie są niestety wyprowadzane na zewnątrz.

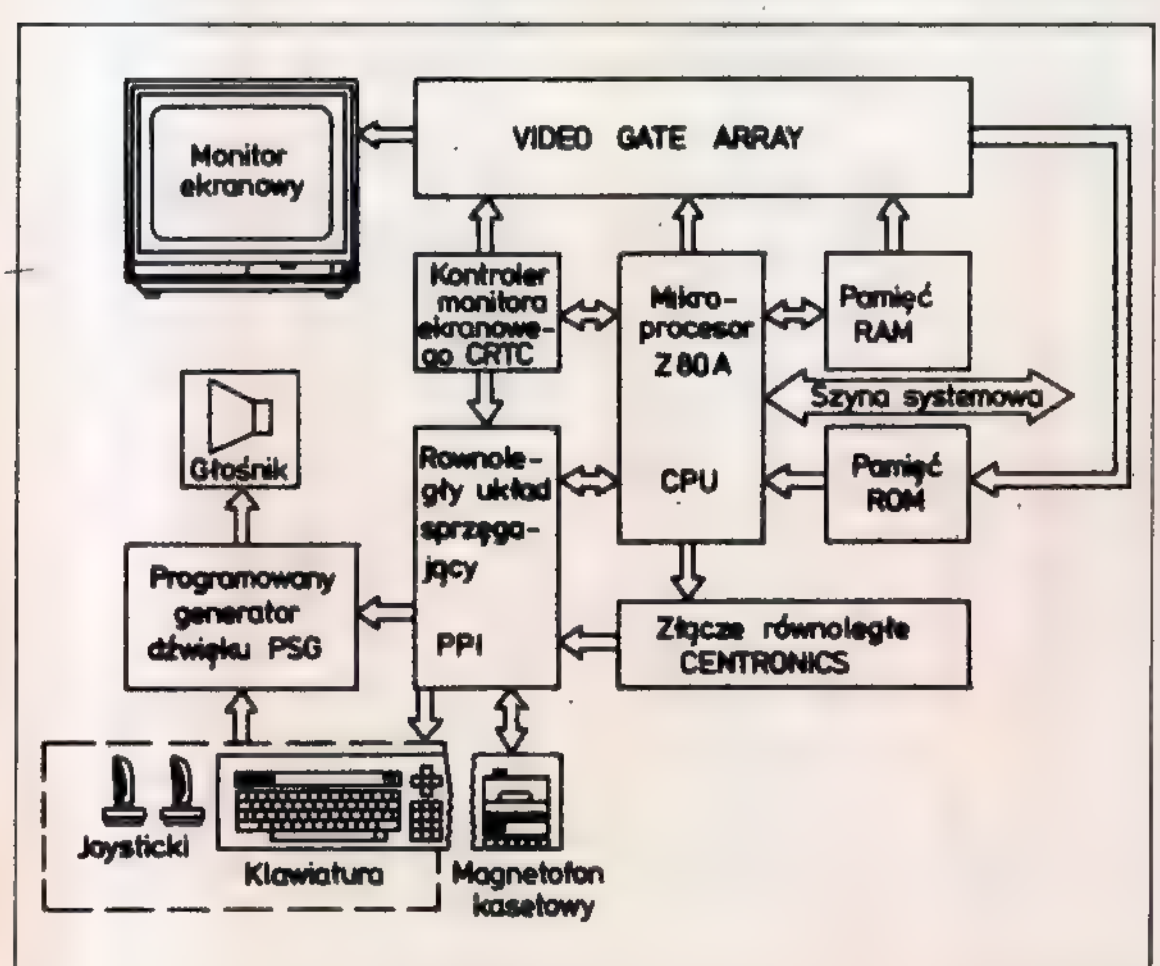
W układzie scalonym programowanego generatora dźwięku znajduje się również I/O PORT — element wejścia-wyjścia, przez który są wprowadzane informacje z klawiatury i JOYSTICKów Port ten współpracuje z równoległym układem sprzęgającym PPI (Parallel Paripheral Interface — układ scalony 8255). Nazwa równoległy oznacza, że pozwala on przekazywać dane po ośmiu liniach równocześnie. Za pośrednictwem PPI komputer korzysta z pamięci na taśmie kasetowej, steruje pracą silnika magnetofonu, a przy użyciu połączonego z nim i mikroprocesorem złącza równoległego CENTRONICS może współpracować z większością drukarek, przesyłając informacje w przyjętym jako standard międzynarodowy amerykańskim kodzie ASCII.

Konstruktorzy AMSTRADa przewidzieli także potrzebę rozbudowy systemu komputerowego o różnego rodzaju urządzenia dodatkowe, jak np.: modem, łącze szeregowe, moduły pamięci, pióro świetlne, a przede wszystkim pamięć dyskowa. Do tego celu przeznaczona została łącząca się bezpośrednio z układem mikroprocesora szyna systemowa. Podłączyć do niej można równocześnie wiele różnych elementów, nakładając jedna na drugą łączówki kolejnych urządzeń.

Niestety, kiedy wyobrażę sobie, co mógłbym połączyć do zaprzyjaźnionego CPC i jaką to sprawiłoby nam obydwu radość, zaczynają drżeć mi ręce i zupełnie nie mogę się skupić. Nie pomagają proszki od bólu głowy. Dopiero wielokrotne powtarzanie półgłosem, że: nawet najprostszy AMSTRAD może dać mi wiele radości — tylko należy poświęcić mu trochę czasu — przynosi pewną ulgę.

Siadamy zatem do przygotowania **POGAWĘDKI DRUGIEJ — FIRMWARE, ALEŻ TO NIE TAKIE PROSTE.**

ROM-man



Komputer w medycynie

Tomografia komputerowa

W 1895 roku niemiecki fizyk Wilhelm Roentgen wykonał, przy użyciu promieniowania o nieznanej wówczas naturze, zdjęcie układu kostnego dłoni człowieka, otwierając tym samym drogę do bezpośredniej penetracji ludzkiego ciała bez konieczności użycia skalpela. Uzyskana w ten sposób fotografia, oczywiście czarno-biała i nieostra, była efektem utrwalenia na kliszy fotograficznej zjawiska pochłaniania promieniowania X przez duże cząstki, w tym wypadku wapnia — efekt cienia na 1 części rysunku.

Przy takiej technice prześwietleń obraz uzależniony jest od kąta widzenia, czyli kierunku padania wiązki promieniowania. Ponadto zdjęcie rentgenowskie, przy bardziej złożonej strukturze wewnętrznej organizmu może być trudne do interpretacji, gdyż bliższe organy, pochłaniające promieniowanie, mogą zakrywać następne, głębiej położone.

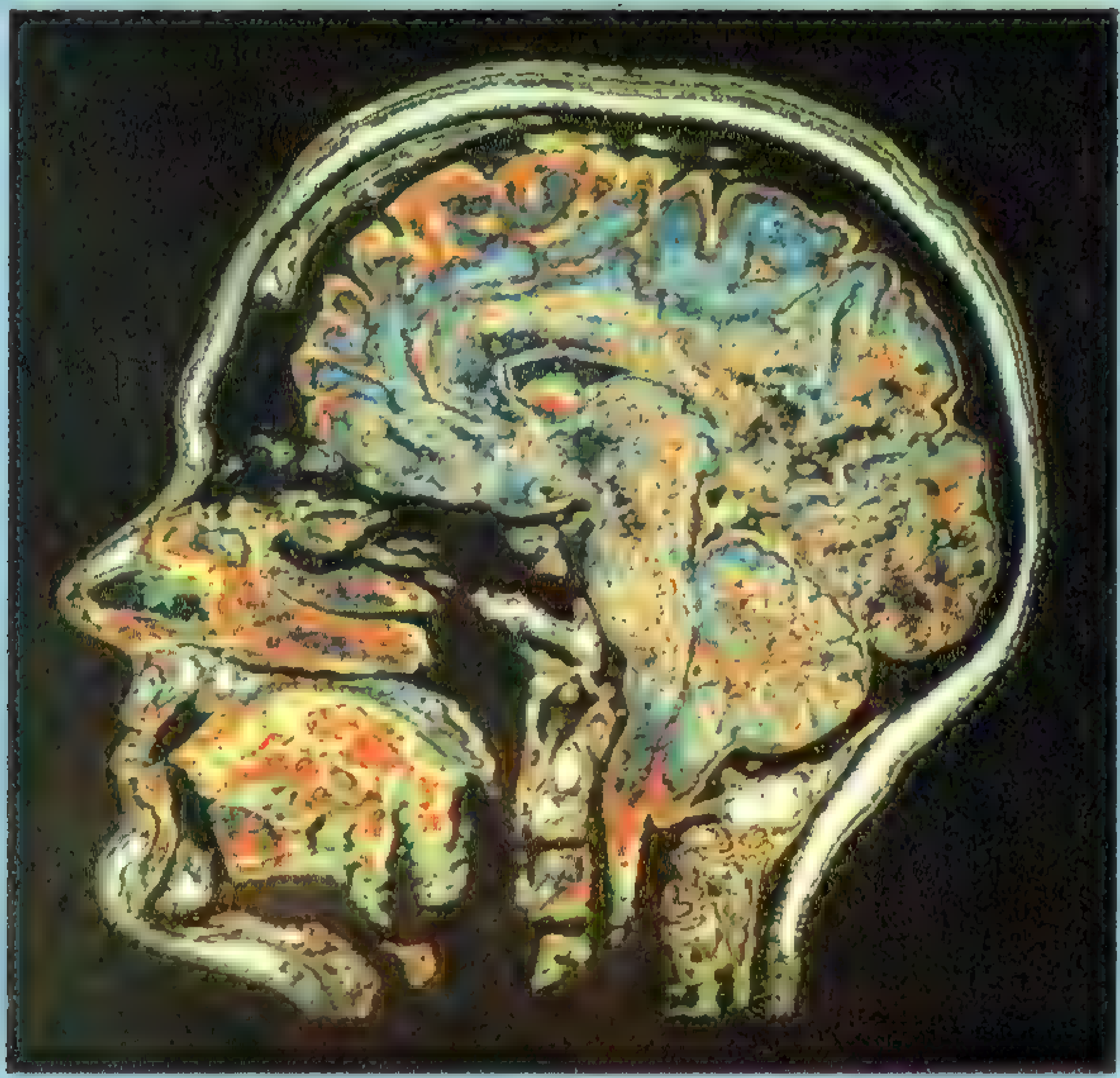
Kolejne dziesięciolecia badań naukowych przyniosły wyjaśnienie natury użytego promieniowania, poprawę jakości uzyskiwanej fotografii i coraz bliższą urzeczywistnienia myśl o nowej metodzie rentgenografii, nie obarczonej brakami metody klasycznej. Myślano zatem coraz śmielej o kolorowym obrazie trójwymiarowym badanego narządu wewnętrznego, przy czym już nie w formie zdjęcia, lecz na ekranie telewizora. Trzeba było jednak na to czekać aż do 1972 roku, gdy geniusz ludzki połączył pozornie odległe techniki: badań rentgenowskich, otrzymywania obrazu video i komputerowego przetwarzania danych. Rezultat, który przyniósł autorowi nagrodę Nobla nazwano tomografem komputerowym. Centralną częścią tego urządzenia jest bęben obrotowy (na rysunku część 2) z trzema źródłami promieniowania rentgenowskiego i setkami czujników tego promieniowania umieszczonymi symetrycznie na jego obwodzie. Pacjent poddawany badaniu umieszczany jest wewnątrz bębna, który powoli obracając się i przesuwając wzdłuż ciała człowieka umożliwia wykonanie serii zdjęć rejestrujących milimetr po milimetrze organy ludzkiego ciała w przekroju poprzecznym. Wyniki kolejnych prześwietleń, zapisane w kodzie cyfrowym wraz z dokładnym określeniem położenia urządzeń naświetlających względem pacjenta, są przechowywane w pamięci komputera. Komputer dysponuje zatem setkami zdjęć badanego narządu wewnętrznego, widzianego z każdej możliwej strony. W tym momencie decydujący staje się kunszt programistów, którzy stworzyli oprogramowanie komputera, będącego częścią składową tomografu. Zebrane dane, stanowiące ułamkowe obrazy anatomiczne w kodzie maszynowym, przetwarzane są bowiem w jeden trójwymiarowy obraz widoczny na monitorze operatora tomografu. Dla podniesienia czytelności obrazu jest on automatycznie nasycany kontrastującymi kolorami. Uwydatnia to subtelne różnice między narządami, doskonale widocznymi na przyciemnionym tle.

Efekt podróży do wnętrza ciała ludzkiego przy użyciu tomografu komputerowego przypomina do złudzenia badanie przez płetwonurków dna morskiego z jego złożoną florą i fauną. Bogata paleta grafiki komputerowej pozwala przykładowo na eksplorację kręgosłupa przypominającego na ekranie wielką rafę koralową, spękaną i pełną ubytków w przypadku ciężkich urazów.

Lekarz śledząc obraz tomograficzny pacjenta bez trudu natrafi na ślad postępującego zniekształcenia organów, wywołanego nowotworem, co umożliwia szybką diagnozę i bezbłędną akcję chirurgiczną. Niestety tą metodą nie można wykryć wczesnej fazy choroby nowotworowej, fazy, w której nie występują jeszcze zmiany kształtu zaatakowanego narządu — choć i na to jest już metoda, której poświęcimy następną relację ze skomputeryzowanego świata medycyny. Dla podkreślenia wagi tomografii komputerowej nie można nie wspomnieć o innym bardzo ważnym jej zastosowaniu. Dane przechowywane w pamięci komputera zebrane przez tomograf podczas prześwietlenia pacjenta mogą być bardzo przydatne w chirurgii, nie bez przyczyny nazywanej rekonstruującą. Sprzężenie precyzyjnej obrabiarki z komputerem pamiętającym wyniki badania tomograficznego umożliwiają stworzenie plastycznej kopii dowolnej części kości. Chirurg planujący przykładowo operację miednicy dysponuje zatem na podstawie obrazu tomograficznego zdrowej części ciała precyzyjną kopią operowanego elementu, co niezmiernie ułatwia mu pracę.

I pomyśleć, że Roentgen opowiadanie o tomografie komputerowym potraktowałby jako powiastkę science-fiction!

AT



PROGRAM 15—16

SHARP

Gra w oczko

Proponujemy Wam grę hazardową. Hasło „hazard" pochodzi z francuskiego, gdzie oznacza: przypadek, traf, zdarzenie losowe. Tak więc formalnie: gra hazardowa = gra przypadkowa.

Rzetelną grą przypadkową powinien rządzić „ślepy los". Na tej zasadzie działają kolektury loteryjne i Toto-Lotek. Ale bywa, że ktoś świadomie *„pomaga losowi"*. W takim wypadku mamy do czynienia z *oszustwem* (vide: Kodeks karny).

Gra w „oczko" wymaga talii z 24 kart (od asów do dziewiątek). Mają one następujące wagi: as — 11 punktów („0 — oczek"), dziesiątka — 10 pkt., król — 4, dama — 3, walet — 2, zaś dziewiątka — zero punktów. Kolor nie gra roli.

Jeden gracz (bankier) wydaje po jednej karcie, zaś drugi gracz starannie kalkuluje:

— jeśli ma 21 punktów (czyli „oczko") to zawsze wygrywa,

— jeśli ma ponad 21 punktów (czyli „furę"), to zawsze przegrywa.

— jeśli ma poniżej 21 punktów (przykładowo 17) to „czeka" i wtedy losuje bankier.

W tym pojedynku zwycięża ten, kto ma więcej punktów.

W przypadku remisu (dwa razy po 17 lub dwie „fury") wygrywa bankier.

Jeśli bankier wylosuje kolejno dwa asy i ma 22 punkty to wygrywa („bankierskie oko").

Program wywołuje etykieta „X" (RUN 10).

Ludowe porzekadło głosi „...nie za to ojciec prał syna, że grał w karty, ale za to, że się ogrywał...".

Zweryfikujcie: u nas nie ma ograniczeń stawek!

Janusz MILLER

SHARP 1500-PC

```
10:"X":CLEAR :
   RANDOM :USING
   :WAIT 0
20:DIM T(6,4),J(2
   4),P(24),H$(6)
   ,K$(4)
30:DATA " 9 "," W
   ALET "," DAMA
   "," KROL "," 1
   0 "," AS "
40:DATA "KIER","K
   ARO","TREFL","
   PIK"
50:RESTORE 30:FOR
   L=1TO 6:READ H
   $(L):NEXT L
60:RESTORE 40:FOR
   L=1TO 4:READ K
   $(L):NEXT L
70:U=0:V=0:F$="FU
   RA ! ":P$=" PK
   T.":O$="*OKO*
   "
80:INPUT "JAKA ST
   AWKA ? ";S
90:IF S<=0THEN
   BEEP 7,100,300
   :GOTO 80
100:WAIT 0:D=0:
   INPUT "DAC KAR
   TE ? T=1 N=0 "
   ;D
110:IF D=0GOTO 150
120:GOSUB "A":X=X+
   1:J(X)=Z:PRINT
   "TY ";H$(H);K$
   (K);
130:GOSUB "C":U=U+
   Q:IF U=21THEN
   CLS :PRINT O$;
   :GOTO 300
140:WAIT 100:PRINT
   " :";U;P$:CLS
   :WAIT 0:GOTO 1
   00
150:GOSUB "A":Y=Y+
   1:P(Y)=Z:PRINT
   "ON ";H$(H);K$
   (K);
160:GOSUB "C":V=V+
   Q:IF V=21THEN
   CLS :PRINT O$:
   GOTO 310
170:WAIT 100:PRINT
   " :";V;P$:CLS
   :WAIT 0:IF (U>
   VAND V<17)GOTO
   150
180:IF Y=0GOTO 200
190:IF (INT P(Y)+
   INT P(Y-1))=12
   THEN PRINT O$;
   :GOTO 310
200:IF U>21THEN
   PRINT F$;:GOTO
   310
210:IF V>21THEN
   PRINT F$;:GOTO
   300
220:IF U>VGOTO 300
230:IF V>=UGOTO 31
   0
300:WAIT 0:PRINT "
   W Y G R A L E
   S ":BEEP 20,25
   ,25:M=M+S:GOTO
   320
310:WAIT 0:PRINT "
   P R Z E G R A
   L E S ":BEEP 3
   ,100,300:M=M-S
320:WAIT 100:PRINT
   "R E Z U L T A
    T"
330:IF M<0THEN
   PRINT "S T R A
    T A :";ABS M:
   GOTO 350
340:PRINT "Z  Y  S
     K :";M
350:IF X>23OR Y>23
   GOTO 500
360:D=0:INPUT "GRA
   MY ? T=1 N=0 "
   ;D
370:IF D=1GOTO 70
380:END
390:"B":H=INT Z:K=
   (Z-H)*10:
   RETURN
400:"A":H=RND 6:K=
   RND 4:IF T(H,K
   )=1AND F<>24
   GOTO 400
410:Z=(K/10)+H:T(H
   ,K)=1:F=F+1:IF
   F<25RETURN
420:WAIT 0:PRINT "
   KONIEC TALII":
   BEEP 10,50,100
   :GOTO "D"
430:"C":IF H=1LET
   Q=0
440:IF (H=2)OR (H=
   3)OR (H=4)LET
   Q=H
450:IF H=5LET Q=10
460:IF H=6LET Q=11
470:RETURN
500:"D":WAIT 50:T$
   =" TY ":O$=" O
   N ":PRINT "K
   A  R  T  Y"
510:FOR L=1TO 24
```

```
520:IF J(L)=0GOTO
    540
530:Z=J(L):GOSUB "
    B":PRINT STR$
    L;T$;H$(H);K$(
    K)
540:IF P(L)=0GOTO
    560
550:Z=P(L):GOSUB "
    B":PRINT STR$
    L;O$;H$(H);K$(
    K)
560:NEXT L:GOTO 10
```

Przykład

```
K  A   R  T  Y
1  TY   WALET TREFL
1  ON   DAMA KIER
2  TY   AS TREFL
2  ON   AS PIK
3  TY   9 KIER
3  ON   DAMA KARO
4  TY   KROL KIER
4  ON   KROL PIK
5  TY   AS KARO
5  ON   9 PIK
6  TY   10 PIK
6  ON   10 KIER
7  TY   WALET KARO
7  ON   WALET PIK
8  TY   10 TREFL
8  ON   KROL TREFL
9  TY   9 TREFL
9  ON   AS KIER
10 TY   WALET KIER
10 ON   KROL KARO
11 TY   10 KARO
11 ON   DAMA TREFL
12 TY   DAMA PIK
13 TY   9 KARO
```

GRA W OCZKO SPECTRUM

```
  5 LET m=0
 10 DIM t(6,4): DIM j(24)
 15 DIM p(24): DIM h$(6,6)
 20 DIM k$(4,5)
 25 DATA "9 ","WALET ","DAMA "
 30 DATA "KROL ","10 ","AS "
 35 DATA "KIER","KARO","TREFL","PIK"
 40 RESTORE
 45 FOR L=1 TO 6: READ H$(L): NEXT L
 50 FOR L=1 TO 4: READ K$(L): NEXT L
 55 LET f=0: LET x=0: LET y=0
 60 LET u=0: LET v=0
 65 LET f$="FURA! ": LET P$=" PKT."
 70 LET o$="*OKO*"
 80 INPUT "Jaka stawka ? ";s
 90 IF s<=0 THEN  BEEP .2,-15: GO TO 80
100 INPUT "Dac karte? T/N ";a$
110 IF a$<>"t" AND a$<>"T" THEN  GO TO 150
120 GO SUB 400: LET x=x+1: LET j(x)=z
125 PRINT "TY "; INVERSE 1;h$(h);k$(k);
130 GO SUB 430: LET u=u+q
135 IF u=21 THEN  PRINT o$: GO TO 300
140 PRINT " :";u;p$: GO TO 100
150 GO SUB 400: LET y=y+1: LET p(y)=z
155 PRINT  INVERSE 1;"ON" ; INVERSE 0;h$(h);k$(k);
160 GO SUB 430: LET v=v+q
165 IF v=21 THEN  PRINT o$: GO TO 310
 70 PRINT  INVERSE 1;" :";v;p$
175 IF u>v AND v<17 THEN  GO TO 150
180 IF y=0 THEN  GO TO 200
190 IF y>1 THEN  GO TO 250
200 IF u>21 THEN  PRINT f$;: GO TO 310
210 IF v>21 THEN  PRINT f$;: GO TO 300
220 IF u>v THEN  GO TO 300
230 GO TO 310
250 IF INT p(y)+INT p(y-1)=12 THEN  PRINT o$;
260 GO TO 310
299 REM
300 PRINT "W Y G R A L E S "
305 BEEP .1,10: BEEP .7,10
307 LET M=M+S: GO TO 320
310 PRINT "P R Z E G R A L E S "
315 BEEP .2,-14: PAUSE 5: BEEP .2,-14
317 LET M=M-S
320 PRINT "R E Z U L T A T"
330 IF m<0 THEN  PRINT "S T R A T A :";-m: GO TO 350
340 PRINT "Z Y S K :";m
350 IF x>23 OR y>23 THEN  GO TO 500
360 INPUT "gramy ? T/N ";a$
370 IF a$="t" OR a$="T" THEN  GO TO 60
380 STOP
390 LET h=INT z: LET k=(z-h)*10: RETURN
400 LET h=INT (RND*6)+1: LET k=INT (RND*4)+1
405 IF t(h,k)=1 AND f<>24 THEN  GO TO 400
410 LET z=k/10+h: LET t(h,k)=1
415 LET f=f+1: IF f<25 THEN  RETURN
420 PRINT " KONIEC TALII"
425 BEEP .2,0: GO TO 500
430 IF h=1 THEN  LET q=0
440 IF h=2 OR h=3 OR h=4 THEN  LET q=h
450 IF h=5 THEN  LET q=10
460 IF h=6 THEN  LET q=11
470 RETURN
500 LET t$=" TY ": LET O$=" ON "
505 PRINT "K A R T Y"
510 FOR l=1  TO 24
520 IF j(l)=0 THEN  GO TO 540
530 LET z=j(l): GO SUB 390
535 PRINT STR$ l;t$;h$(h);k$(k)
540 IF p(l)=0 THEN  GO TO 560
550 LET z=p(l): GO SUB 390
555 PRINT STR$ l;o$;h$(h);k$(k)
560 NEXT l: PAUSE 0: RUN
```

AMSTRAD

PROGRAM ZNOSI PROTEKCJĘ ODCZYTU

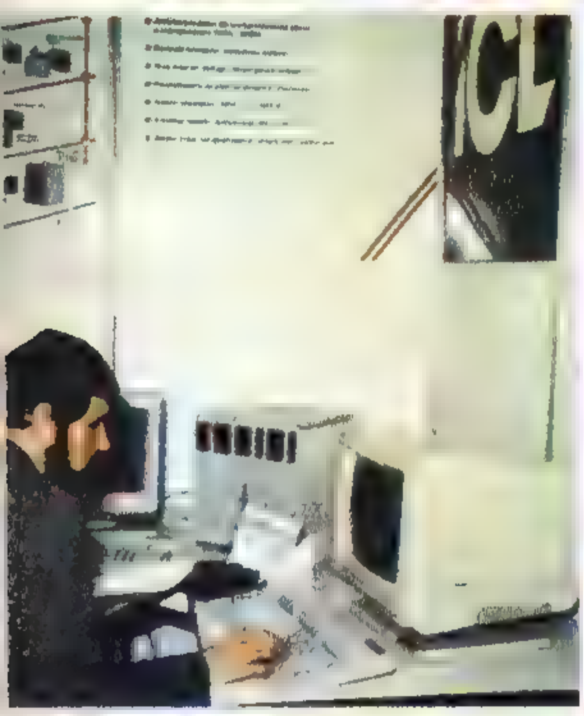
Przedstawiony program znosi „protekcję odczytu". Często chcemy podejrzeć komendą LIST sposób, w jaki dany program został napisany w języku BASIC. Niestety nie zawsze jest to możliwe. Prezentowany poniżej program pozwoli uporać się z tym problemem. Chcąc odczytać program napisany w języku Basic należy najpierw załadować i uruchomić komendą RUN poniższy program, a następnie załadować komendą LOAD interesujący nas program. Możecie być pewni, że większość programów da się teraz przejrzeć komendą LIST.

```
10 SYMBOL AFTER 256:MEMORY &A640:SYMBOL AFTER 240
20 FOR x=0 TO 54:READ a:POKE &A641+x,a:NEXT
30 IF PEEK(&AC01)=0 THEN POKE &A669,44
40 CALL &A641
50 CALL &BBBA:CALL &BC02:MODE 1:PEN 1:PAPER 0
60 MODE 2:INK 0,13:INK 1,1:BORDER 13:PRINT"PROGRAM
 ZAŁADOWANY ":PRINT CHR$(164)" POWODZENIA":PRINT
70 NEW
80 DATA 58,122,188,50,120,166,62,195,50,122,188,42
,123,188,34,121,166,33,89,166,34,123,188,201,245,2
29,58,120,166,50,122,188,42,121,166,34,123,188,175
,50,69,174,225,241,205,122,188,245,229,205,65,166,
225,241,201
```





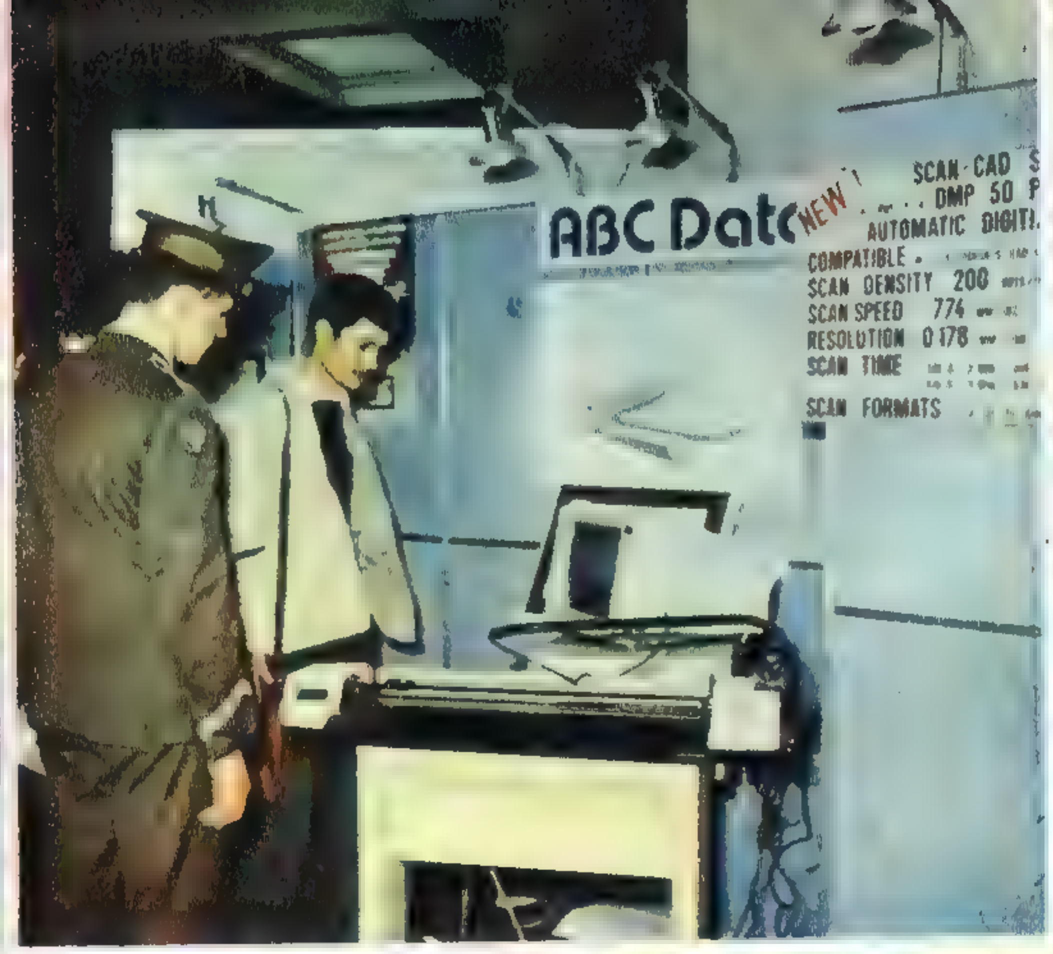
SPOTKANIA na „Infosystem '8













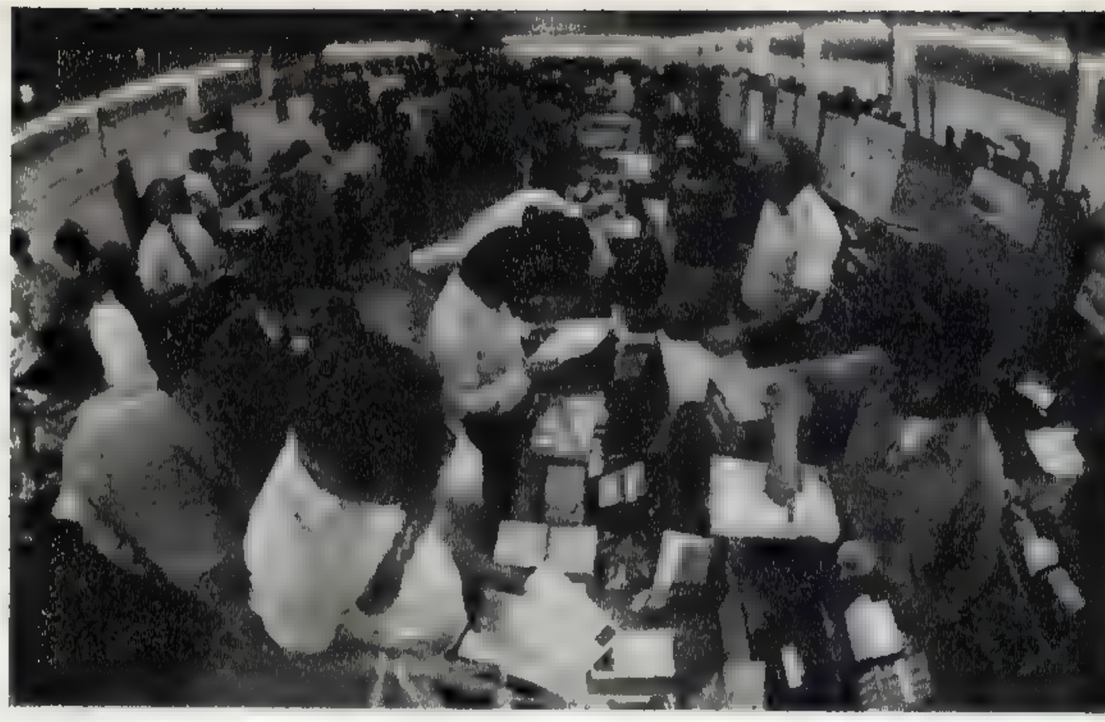
7”





Foto: Jan Zelman

Kaizen zen-ja — przeddzień wojny. Takie słowa słyszy dziś Japończyk w telewizji, w radio. Czyta je także na łamach gazet. Rozbrzmiewało ono przed laty, niedługo przed atakiem japońskiego lotnictwa na bazę amerykańską w Pearl Harbour, w grudniu 1941 r. Hasło to zwiastowało Japończykom początek „Wielkiej wojny o Pacyfik". A co zwiastuje dziś?



Wielkie centrum sprzedaży komputerów pod San Francisco. Zabrakło tanich japońskich cudów techniki.

(Foto: — „Stern")

Wojna o komputery

USA obłożyły wysokimi cłami wyroby japońskiego przemysłu elektronicznego. Od paru miesięcy wszyscy, i w Japonii i w USA, studiują długą listę produktów, które obciążono cłem, a jak mawiają Japończycy, „karnymi opłatami", w wysokości 100 proc. ich wartości. Spis ten obejmuje m.in. stacje dysków elastycznych do komputerów, mikroprocesory, odbiorniki telewizyjne, radiomagnetofony, lodówki, klimatyzatory, satelity komunikacyjne, silniki elektryczne małej mocy, przyrządy pomiarowe, pompy, filmy światłoczułe, ręczne narzędzia o napędzie elektrycznym. Natomiast cłem nie są obciążone półprzewodniki, których amerykański przemysł komputerowy potrzebuje, a nie może zastąpić własną produkcją. Tak więc zarządzenia ochronne dotyczą przede wszystkim urządzeń elektronicznych. Oblicza się, że Japonia straci na tym ok 300 mln dolarów rocznie.

Czy Japończycy musieli zgodzić się na te restrykcje?

Obserwatorzy zagraniczni w Tokio relacjonują, że postępowanie Amerykanów przypomina mieszkańcom Kraju Wschodzącego Słońca rok 1853, kiedy to amerykański admirał Matthew Perry pojawił się u wybrzeży japońskich ze swą flotą i bez większych ceregieli zażądał przerwania ścisłej izolacji Japonii, którą kraj ten stosował wobec siebie. Władcy jego uznali bowiem, że lepiej nie wiedzieć, co dzieje się u innych, niż narażać się na kontakty z „barbarzyńcami". Tylko że admirał Perry rozporządzał ogniem swych dział i był gotów zrealizować groźby zawarte w ultimatum: otwieracie granice albo...

W Waszyngtonie uznano wówczas, że rynek japoński jest tak nęcący, iż USA nie mogą czekać, aż konkurenci ich ubiegną. Skończyło się na protestach szogunów, ale wymowa straszliwych dział była aż nadto przekonująca.

Dziś jakby sytuacja się „odwróciła". To Amerykanie zamykają drzwi przed japońskim „nosem", przynajmniej, jeśli chodzi o elektronikę. Można powiedzieć: jakże to, USA i Japonia są sojusznikami, współpraca militarna rozwija się doskonale. To wszystko prawda, ale jednocześnie USA mają 170 miliardów dolarów deficytu handlowego, a nadwyżka eksportu japońskiego do USA wynosi 52 mld dolarów. Potwierdza się stare porzekadło: tam, gdzie chodzi o pieniądze, kończą się żarty, a zaczyna... cło!

A czy „żartować" mogą tylko Amerykanie, przecież w końcu, to nie połowa XIX wieku, ale koniec XX!

Ustępliwość Japończyków tłumaczy się zależnością ich kraju od źródeł surowcowych. Wiemy, co potrafią produkować i jak, ale muszą mieć z czego. USA pozostają na razie czołowym producentem w świecie kapitalistycznym wielu rodzajów surowców i są potężnym eksporterem węgla, molibdenu, magnezu, fosfatów, materiałów z zawartością boru. Dla prawie bezsurowcowej Japonii amerykański partner jest tym, na którego przypada jedna trzecia ogólnego importu paliw mineralnych i jedna czwarta surowców nieenergetycznych. USA odgrywają decydującą rolę w zaopatrzeniu gospodarki japońskiej w surowce. I taka jest prawda. I takie są konsekwencje.

Japończycy protestują, ale na tym się skończy. Nie skończyły się natomiast nadzieje Japończyków, że „załatwią" Amerykanów sposobem.

Koniec rozrzutności

Dotychczas USA mogły sobie pozwolić na tolerowanie deficytu budżetowego i handlowego. Olbrzymi potencjał gospodarczy jeszcze znosi brzemię ogromnych wydatków zbrojeniowych. Dolar był i jest międzynarodową walutą rezerwową, w której dokonuje się ponad 60 proc. wszystkich rozliczeń międzynarodowych. Ponadto powiedzmy zwyczajnie i wprost: nie stać już dziś gospodarkę amerykańską na wymienność dolara na złoto, ale stać ją na to, by „przełknęła" wprowadzenie kilku czy kilkunastu miliardów dolarów na rynek, za pomocą maszyn drukujących „zielone", bez komplikacji w postaci inflacji. Przy czym obecnie na skutek nieprawdopodobnego zadłużenia wewnętrznego (deficyt budżetowy) pojawiły się oznaki niepokoju na rynku. Dziś w Białym Domu zdają sobie sprawę, że USA muszą konkurować na rynkach zagranicznych i prezentować atrakcyjną ofertę eksportową. Także i w zakresie elektroniki.

Japończycy obliczają, obserwują i wiedzą. Świadomi są tego, iż roczny wskaźnik wzrostu produkcji w USA w przeliczeniu na godzinę pracy zmniejszył się już od połowy lat sześćdziesiątych. Realne koszty jednostki pracy są wyższe niż w Japonii. Wydajność pracy również wzrasta w tempie wolniejszym. Jakość wyrobów „made in USA" jest gorsza niż japońskich. Występuje zjawisko opóźnienia w inwestycjach technologicznych. W zakresie maszyn numerycznych USA utraciły przodującą pozycję na rzecz Japonii. Amerykańscy producenci skoncentrowali się na osiąganiu szybkich zysków, co wypaczyło ich politykę inwestycyjną w dziedzinie produkcji artykułów elektronicznych codziennego użytku. Ustąpienie miejsca Japończykom, czyli rezygnacja z rynku wewnętrznego, odnośnie tych artykułów jest równoznaczna z osłabieniem eksportu. Teraz próbuje się cłami ochronnymi odwrócić tę sytuację.

Decyduje technologia

Uwagę specjalistów zwraca problem

technologiczny. W ostatnich latach przemysł elektroniczny w USA przeznaczył 67 miliardów dolarów na badania i opracowanie nowych technologii. Połowa tej kwoty pochodzi z kas rządowych. Jest to akurat trzy razy tyle, ile mogą wydać na te cele japońscy konkurenci. A jednak ich wyroby elektroniczne, a szczególnie komputery o pojemności pamięciowej 1 megabita oraz mikroprocesory o pojemności 32 bitów skutecznie rywalizują z analogicznymi urządzeniami „made in USA". Dlaczego?

Japończycy twierdzą, że ani Europejczycy, ani Amerykanie nie potrafią handlować. Towar trzeba umieć nie tylko wyprodukować, ale i sprzedać. Mentalność ludzi Wschodu jest inna niż Zachodu. Na to Amerykanie powiadają: to co, mamy sprzedawać nasze komputery w kimonie? Kłaniać się klientom? Obie strony wiedzą doskonale, że nie o to chodzi, lecz zwyczajnie o „szmal". Godzinę drogi na południe od San Francisco, w pobliżu wzgórz Santa Cruz znajduje się ogromne centrum sprzedaży urządzeń elektronicznych. Można tu nabyć komputery do koloru i wyboru, podzespoły, monitory monochromatyczne, drukarki mozaikowe, laserowe, plottery, digitizery, streamery, zespoły indywidualne, sieci komputerowe, zestawy specjalistyczne z wyposażeniem, moduły do mikrokomputerów, układy scalone itp. Na wieść o podniesionych cłach na wyroby japońskie rozległ się tu (gdzie indziej również) płacz i zgrzytanie zębów. Ceny japońskich produktów elektronicznych były do tej pory tańsze. Niektórzy klienci mówią, że amerykańskie nie są gorsze. Inni, że japońskie lepsze, ale ceny, te ceny!

Skorzysta klient

USA bronią się przed japońską, i nie tylko, konkurencją, machinacjami na giełdach, które powodują kierowany spadek wartości dolara. Dziś za 1 „zielonego" otrzymuje się więc mniej jenów, niż przed rokiem. Czyli że Amerykanin musi wydać więcej dolarów, by kupić japońskie cuda elektroniczne. Zwyżka wartości jena japońskiego już daje o sobie znać w postaci jeszcze nieznacznego, ale wyraźnego spadku eksportu. A przecież Japonia żyje z tego, co sprzeda. W Tokio złośliwie przypominają amerykańskim sprzymierzeńcom: jak w tym świetle wygląda wasze zachwalanie wolnej, nieskrępowanej gospodarki? Najpierw tyle się gada o liberaliźmie, handlu wolnym od barier i restrykcji, a jak są lepsi od was, to sięgacie do metod, które obłożyliście klątwą!

Tzw. wojna handlowa między USA a Japonią nie ogranicza się jedynie do niezadowolenia amerykańskich i japońskich konkurentów „elektronicznych". Trudno sobie wyobrazić, aby jedna ze stron miała skapitulować. W trakcie tej niewątpliwej walki o rynki może dojść do taktycznych kompromisów, ale nic więcej. Z drugiej strony amatorzy domowej elektroniki zyskają na tym. Japończycy wymyślą kolejne nadzwyczajne rzeczy w elektronice. Amerykanie i Europejczycy nie zostaną w tyle.

Henryk KAWKA

W poprzednim odcinku przedstawiony został uproszczony schemat blokowy pakietu systemowego mikrokomputera IBM PC/XT, na którym znajdują się: mikroprocesor INTEL 8088 z koprocesorem INTEL 8087 i ze współpracującymi układami m.in. 8284, 8259A, 8237A, 8253, 8255; pamięć zapisywalna RAM o pojemności 256 KB; pamięć stała ROM.

MIKROKOMPUTERY IBM PC (3)

W mikrokomputerze do pakietu systemowego muszą być dołączone pakiety umożliwiające współpracę z monitorem, drukarką lub plotterem, stacją dysków elastycznych, dyskiem stałym (twardym, sztywnym — ang. hard disk, fixed disk). Nie wyczerpuje to listy możliwych pakietów. Na fot. 1 przedstawiono niektóre z nich

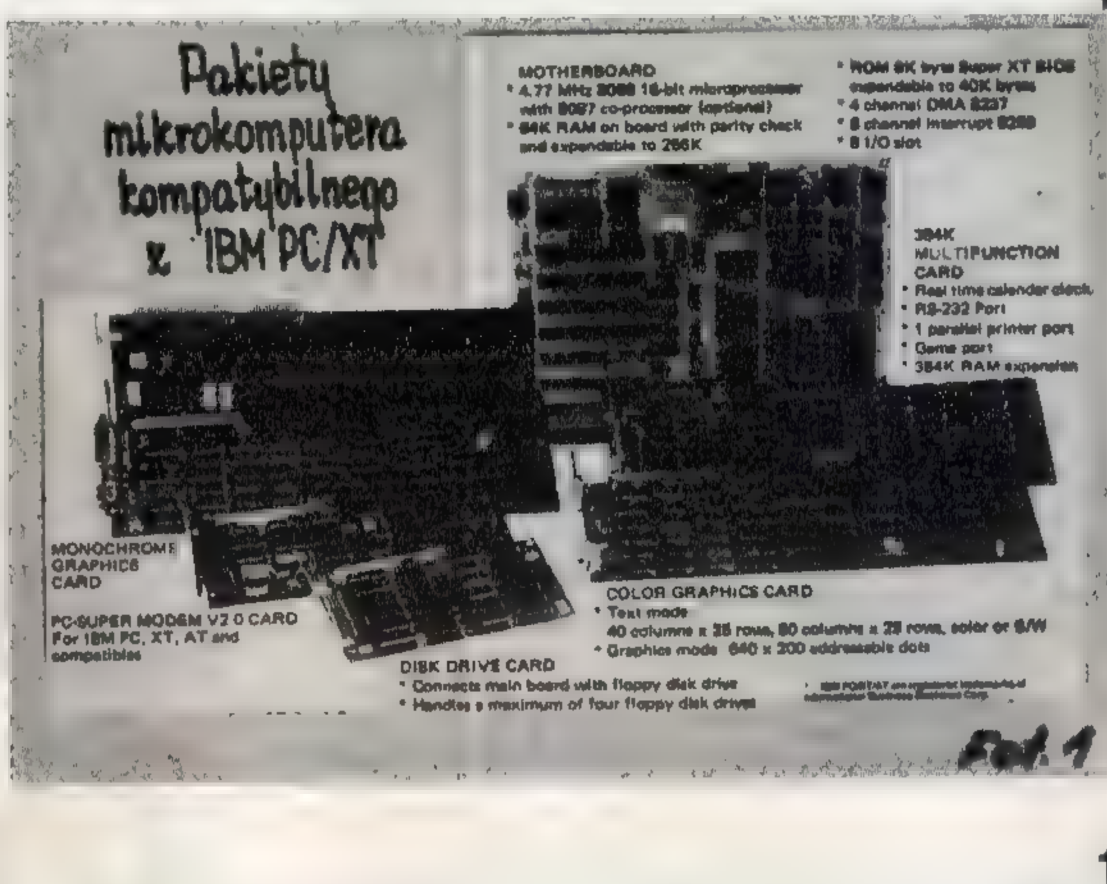
Karta graficzna służy do organizacji ekranu monitora. Możliwe są dwa mody pracy: **mod alfanumeryczny** 40 lub 80 znaków w 25 liniach oraz **mod graficzny** 160x100, 320x200, 640x200, 640x350 lub 720x348 punktów adresowych bezpośrednio, odpowiednio w standardzie małej, średniej i dużej rozdzielczości. Możliwe mody pracy ekranu monitora zależą od karty graficznej (kart graficznych) przyłączonej do pakietu systemowego.

W mikrokomputerach IBM PC/XT (i kompatybilnych) można spotkać następujące karty graficzne: Monochrome Display Adaptor, Color Graphics Adaptor, Enhanced Graphics Adaptor, Hercules.

Monochrome Display Adaptor (MDA) umożliwia pracę w modzie alfanumerycznym 80 znaków w 25 liniach, w modzie graficznym 720x350 punktów adresowanych bezpośrednio. Podstawowym układem karty jest sterownik MOTOROLA 6845 CRT, karta ma pamięć ekranu 4 KB. MDA może zawierać złącze równoległe, do połączenia drukarki lub plottera.

Color Graphics Adaptor (CGA) umożliwia następującą organizację ekranu: w modzie alfanumerycznym 40 lub 80 znaków w 25 liniach w szesnastu kolorach; w modzie graficznym: małej rozdzielczości 160x100 punktów adresowanych bezpośrednio w szesnastu kolorach, średniej rozdzielczości 320x200 punktów w czterech kolorach, dużej rozdzielczości 640x200 punktów w dwóch kolorach. CGA ma pamięć ekranu 16 KB, podstawowym jej układem jest sterownik 6845 CRT. Istnieje możliwość dołączenia do karty pióra świetlnego.

Enhanced Graphics Adaptor (EGA) w modzie alfanumerycznym organizuje ekran monitora identycznie jak CGA. W trybie graficznym EGA umożliwia: średnią rozdzielczość 320x200 punktów adresowanych bezpośrednio w szesnastu kolorach, dużą rozdzielczość 640x200 w szesnastu kolorach lub 640x350 w dwóch kolorach lub 640x350 w sześćdziesięciu czterech kolorach. EGA ma w wersji firmy IBM pamięć ekranu 64 KB, niektóre wersje kompatybilne mają pamięć ekranu 256 KB oraz umożliwiają jeszcze dodatkowo rozdzielczość 720x348 punktów w dwóch kolorach. Instalacja karty EGA wymaga monitora o dużej rozdzielczości i specjalnej wersji ROM—BIOS-u.



Fot. 1

W mikrokomputerach kompatybilnych z IBM PC/XT można spotkać również Hercules Graphics Card o rozdzielczości w modzie graficznym 720x348 punktów adresowych bezpośrednio w dwóch kolorach.

Kolejnym pakietem, który może być dołączony do pakietu systemowego, jest **karta wielofunkcyjna**. Karta ta umożliwia rozszerzenie pamięci RAM do 640 KB, ma złącze szeregowe RS—232 C i złącze równoległe CENTRONICS oraz zegar czasu rzeczywistego z baterylką.

Kolejny pakiet, który musi być w mikrokomputerze IBM PC/XT, służy do **sterowania stacjami dysków elastycznych,** takimi jak np. na foto 2. Pakiet może równocześnie sterować dyskiem stałym. W pakiecie służącym jedynie do sterowania stacjami dysków elastycznych głównym elementem jest sterownik NEC µPD 764, mogący obsłużyć dwie stacje dysków.

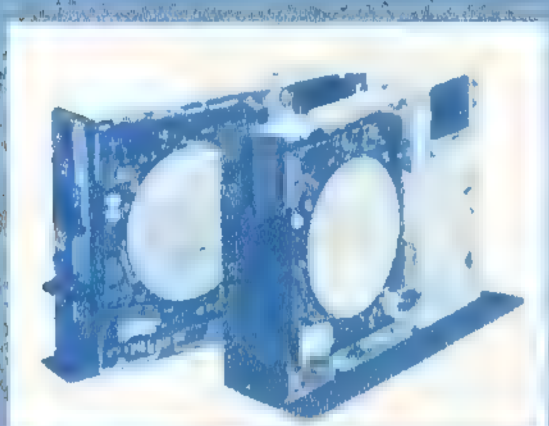


Foto. 2 Stacja dysków elastycznych 5 ¼ cala

Stacja dysków (ang. disk drive, diskette drive) może być jedno- lub dwustronna. Stacje dysków jednostronne umożliwiają formatowanie dyskietek (dysków elastycznych, ang. diskette) jednostronnie, w standardzie 8 lub 9 sektorów po 512 bajtów na ścieżce, przy 40 ścieżkach. Umożliwia to osiągnięcie pojemności 160 lub 180 KB. Dla stacji dysków dwustronnych pojemność wynosi odpowiednio 320 lub 360 KB. Ważniejsze parametry stacji: szybkość transmisji danych 250 K bitów/s, czas dostępu między ścieżkami dyskietki 6 ms, czas niezawodnej pracy stacji 20 000 godzin, przy trwałości nośnika 3.10^6 przejść po ścieżce. W zależności od typu stacji wykorzystywane są **dyskietki** jednostronne (single — side, dauble — density) lub dwustronne (double — side, double — density).

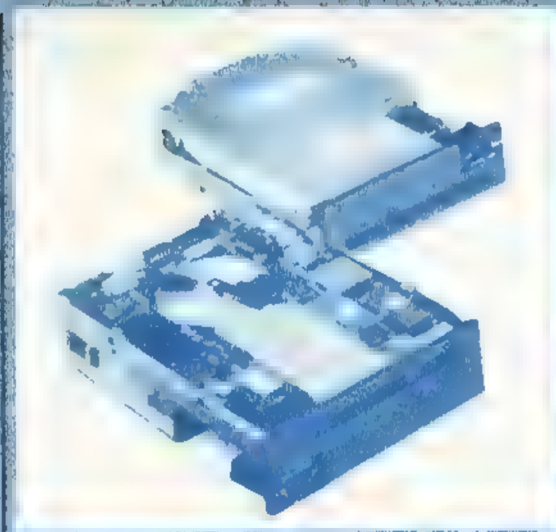


Foto. 3 Dyski stałe typu Winchester o pojemności 20 MB

W mikrokomputerze IBM PC/AT dyskietki formatowane są wg standardu dwustronnie 9 sektorów na ścieżce, przy 80 ścieżkach na stronie, co umożliwia osiągnięcie pojemności 720 KB lub wg standardu dwustronnie, 15 sektorów na ścieżce, przy 80 ścieżkach na stronie, co pozwala osiągnąć pojemność 1200 KB.

Dysk stały, taki jak np. na fot. 3, z reguły ma pojemność 10 lub 20 MB. Dysk 10 MB formowany jest czterostronnie, po 17 sektorów 512-bajtowych na ścieżce, przy maksymalnie 1226 ścieżkach i 306 cylindrach. Jego najważniejsze parametry to: szybkość transmisji 5 M bitów/s, czas niezawodnej pracy 5 lat, czas dostępu między ścieżkami 3 ms. Dysk o pojemności 20 MB formatowany jest identycznie, ma 615 cylindrów.

W mikrokomputerze można zainstalować pakiet umożliwiający pracę w sieci lokalnej.

Kolejnym elementem mikrokomputera jest **głośnik,** z którym można pracować za pomocą układów 8253 lub 8255.

Mikrokomputer może zawierać złączkę dla **joystick'a.**

Następnym, bardzo ważnym elementem jest **zasilacz** o mocy minimum 130 watów. Obecnie w PC/XT wykorzystywane są zasilacze o mocach 135 watów, 150 watów. W mikrokomputerach PC/AT wykorzystywane są zasilacze o mocach 180 watów lub 200 watów.

Klawiatura umożliwia kontakt z systemem za pomocą 83 klawiszy, które można podzielić na trzy grupy: 10 klawiszy funkcyjnych, klawisze znakowe oraz klawisze ekranowe (numeryczne i specjalne). Obsługą klawiatury zajmuje się mikrokomputer jednoukładowy INTEL 8084, który wykonuje również test po włączeniu zasilania. Obecnie w mikrokomputerach kompatybilnych z IBM PC/XT stosowane są jeszcze dwa inne typy klawiatur: z 84 klawiszami i z 101 klawiszami.

Monitor jest bardzo ważnym elementem systemu, musi on być dopasowany do karty graficznej znajdującej się w komputerze.

Charakterystykę budowy mikrokomputera PC/XT uzupełnia tab. 1, przedstawiająca mapę adresową.

Tablica 1.

adres	przeznaczenie
00000	RAM
10000	RAM
20000	RAM
30000	RAM
40000	RAM
50000	RAM
60000	RAM
70000	RAM
80000	RAM
90000	RAM
A0000	rozszerzenie pamięci ekranu dla EGA
B0000	pamięć ekranu
C0000	rozszerzenie ROM — BIOS, sterownik
D0000	—
E0000	—
F0000	ROM — BIOS, interpreter BASIC, diagnostyka systemu

Następny odcinek dotyczyć będzie dyskowego systemu operacyjnego, który zostanie omówiony na przykładzie MS DOS 3.10.

Jacek WOJTALA



Czym skorupka za młodu nasiąknie ... Fot. Ryszard Rogoń

BASIC 8

Na ekranie możemy rysować w kolorach. Kolory odnoszą się do całej pozycji literowej (kwadracika 8 x 8 pixeli) i nie mogą być określone dla poszczególnych pixeli. Gdy w danym pixelu stawiamy punkt, to jest on w kolorze atramentu, odpowiednim do całego kwadracika 8 na 8 zawierającego ten punkt.

Na ZX Spectrum dostępnych mamy osiem kolorów.

Kolor	Klawisz	
czarny	0	(BLACK)
ciemnoniebieski	1	(BLUE)
czerwony	2	(RED)
purpurowy	3	(MGNTA)
zielony	4	(GREEN)
bladoniebieski (lazurowy)	5	(CYAN)
żółty	6	(YELLOW)
biały	7	(WHITE)

Kolejność, w jakiej są ustawione, odpowiada intensywności szarości na ekranie telewizora czarno-białego

Z pozycją każdego znaku związane są dwa kolory:
— kolor atramentu (INK) odpowiadający czarnym kropkom,
— kolor papieru (PAPER) czyli tło, odpowiadający kropkom białym.

W pisanych i wykonywanych przez nas programach każda pozycja na ekranie miała kolor biały i czarny atrament (pismo czarne na białym papierze). Za pomocą instrukcji postaci

PAPER *n* n = 0,1,...,7 (numery kolorów)

ustawiamy kolory papieru kwadracików (mogą to być pojedyncze, lub ciąg kwadracików, lub cały ekran), a instrukcją postaci:

INK *n* n = 0,1,...,7 (numery kolorów)

nadajemy kolory pisma (całych kwadracików)

Jasność obrazu zwiększamy za pomocą instrukcji:

BRIGHT 0 lub 1

gdzie: 0 — jasność normalna, 1 — jasność zwiększona.

Przykład 6.9.

Poniższy program pokazuje 8 kolorów (łącznie z białym i czarnym) i dwa poziomy jasności, jakie można otrzymać na kolorowym odbiorniku. Na telewizorze czarno-białym otrzymamy różne odcienie szarości

Wprowadźmy i wykonajmy ten program

```
   5 REM Przykład 6.9
  10 FOR m=0 TO 1: BRIGHT m
  20 FOR n=1 TO 7
  30 FOR c=0 TO 7
  35 REM 4 barwne spacje
  40 PAPER c: PRINT "    ";
  50 NEXT c: NEXT n: NEXT m
  55 FOR i=1 TO 7
  60 FOR m=0 TO 1: BRIGHT m: PAP
ER 7
  70 FOR c=0 TO 3
  80 INK c: PRINT c;" ";
  90 NEXT c: PAPER 0
 100 FOR c=4 TO 7
 110 INK c: PRINT c;" ";
 120 NEXT c: NEXT m
 125 NEXT i
 130 PAPER 7: INK 0: BRIGHT 0
```

Oglądając obraz na ekranie, prześledźmy sami działanie instrukcji **PAPER, INK, BRIGHT.**

Można także wprowadzić błyskanie (miganie) obrazu na ekranie za pomocą instrukcji

FLASH 0 lub 1

gdzie: 0 — brak błyskania, 1 — błyskanie

Przykład 6.10.

Poniższy program pokazuje, jak działa instrukcja FLASH 1 umieszczona w dwóch różnych miejscach programu

Wprowadźmy i wykonajmy ten program. Prześledźmy sami „błyskanie" obrazu na ekranie w zależności od miejsca w programie instrukcji **FLASH 1.** Kolory jasności i błyskanie w danym miejscu nazywamy atrybutem

```
   5 REM Przykład 6.10
  10 REM błyskanie całego ekranu
  20 BORDER 2: INK 1: PAPER 6: F
LASH 1: CLS
  30 DRAW 255,175,1
  40 DRAW -255,-175,1
  50 PAUSE 400: FLASH 0: CLS
  55 REM błyskanie rysunku
  60 BORDER 4: INK 3: PAPER 7: C
LS
  70 DRAW FLASH 1;255,175,1
  80 DRAW FLASH 1;-255,-175,1
  90 PAUSE 400: CLS : FLASH 0
```

W powyższych instrukcjach można napisać liczbę 8 lub 9. Liczba 8 może być użyta we wszystkich instrukcjach. **PAPER 8, INK 8, BRIGHT 8, FLASH 8** i oznacza „przezroczysty", w znaczeniu takim, że nie zmienia poprzedniego atrybutu (pozostają poprzednie atrybuty).

Liczba 9 może być użyta tylko z PAPER i INK i oznacza „kontrast", tj. aktualny kolor (atramentu lub papieru) jest zmieniany na biały, jeśli drugi jest ciemny (czarny, granatowy, czerwony lub purpurowy) lub na czarny, jeśli drugi jest jasny (zielony, lazurowy, żółty, lub biały). Kolor atramentu jest dobierany tak, aby kontrastował ze starym kolorem papieru.

Przykład 6.11.

Poniższy przykład pokazuje nam, jak działa liczba 9 umieszczona w instrukcji **INK** („kontrast").

Wprowadźmy i wykonajmy ten program

```
  10 REM Przykład 6.11
  15 INK 9
  20 FOR c=1 TO 7
  30 PAPER c
  40 FOR i=1 TO 15
  50 PRINT c+i;"  ";
  60 NEXT i
  70 NEXT c
```

Oglądając obraz na ekranie, prześledźmy sami działanie INK 9

Kropkowy wzór wypisany na ekranie można „kontrolować" za pomocą dwóch instrukcji postaci:

INVERSE 0 lub *1* *oraz* **OVER** 0 lub *1*

0 — wyłączenie, 1 — włączenie

Instrukcja **INVERSE 1** powoduje, że rysunek staje się negatywem zwykłej formy, kropki w kolorze papieru zostaną zastąpione przez kropki w kolorze atramentu i vice versa. Np. jeżeli mamy biały papier i czarny atrament, wówczas odpowiedni rysunek pojawi się jako biały na czarnym, ale kolor papieru i atramentu pozostają nie zmienione, zmieniają się tylko kropki

Instrukcja **OVER 1** powoduje szczególny rodzaj nadrukowywania, tzn. jeżeli coś jest napisane w danym miejscu, to zawartość poprzednia zostaje całkowicie usunięta po wykonaniu **OVER 1.**

Omówiliśmy dotychczas kolory górnej części ekranu. Dolna część ekranu ma jako kolor — kolor papieru, a dla atramentu obowiązuje kod 9, wyłączone jest błyskanie, jasność jest normalna. Kolor ramki, otaczającej ekran, składający się z 24 linii i 32 kolumn, można zmienić instrukcją postaci

BORDER *n* (*n*=0,1,...,7).

Przy wprowadzaniu danych przestrzegana jest reguła o użyciu kontrastującego atramentu dobieranego do papieru

Poniższy program ustala kontrastowe kolory ekranu:
— część górna · **PAPER** *n*
— część dolna — ramka, **BORDER** 7-n (*n*=0,1,..., 7)

```
20 FOR n = 0 TO 7
30 BORDER n BORDER 7 - n · CSL
40 PAUSE 100/ NEXT n
```

sprawdzenie pozostawiamy czytelnikowi

Instrukcje **PAPER, INK** itd. można stosować jako elementy instrukcji **PRINT** (stawiając po nich średniki) — ich efekt jest chwilowy (do pierwszego dwukropka lub do następnej instrukcji). Można je także stosować jako elementy instrukcji **PLOT, DRAW, CIRCLE.**

Poniżej podajemy kilka możliwych kombinacji:

PLOT ... — kropka w kolorze atramentu, tzn. ustawia pixel tak, aby pokazywał kolor atramentu,
PLOT INVERSE *1*;... — daje pixel (kropka) w kolorze papieru
PLOT OVER *1*;... — pixel przechodzi z koloru papieru w kolor atramentu lub odwrotnie
PLOT INVERSE *1*; **OVER** *1*;... — zostawia pixel w stanie, w jakim był, ale zmienia pozycję **PLOT**-u (czyni to niezauważalnie)
DRAW OVER *1*;... — pixele przechodzą z koloru papieru w kolor atramentu lub odwrotnie (jeśli coś było napisane, to poprzednia zawartość zostaje usunięta)
DRAW INVERSE *1*;... — pixele (kropki) w kolorze atramentu przechodzą na pixele (kropki) w kolorze papieru i odwrotnie (otrzymujemy negatyw).

Instrukcje **OVER** 1 i **INVERSE** 1 można stosować np. razem z **DRAW** do „wymazywania" rysunków

Przykład 6.12.

Wprowadźmy i wykonajmy następujący program:

```
  5 REM Przyklad 6.12
 10 FOR i=1 TO 22
 20 FOR j=1 TO 16
 30 PRINT "■";
 40 NEXT j: NEXT i
 50 PLOT 0,0
 55 DRAW OVER 1,255,175
 60 PAUSE 200
 70 PLOT 0,0
 80 DRAW OVER 1;255,175
```

Na ekranie otrzymamy czarno-białą kratkę, a następnie zostanie wykreślona linia prosta o pixelach koloru białego i czarnego (instr. 55 DRAW OVER 1; 255, 175) po chwilowej przerwie (PAUSE 200) linia zniknie — została zamazana instrukcją 80 DRAW OVER 1; 255, 175. Dlaczego? Odpowiedź zostawiamy czytelnikowi. Rysunek został nienaruszony

Wprowadźmy instrukcję

80 **DRAW INVERSE** 1, 255, 175

i wykonajmy program.

Zauważymy, że instr. INVERSE 1, „wymazuje" linię, ale „narusza" rysunek. Nie zaleca się wymazywać linii w odwrotną stronę, niż była rysowana, np. w naszym programie nie należałoby napisać:

50 **PLOT** 0, 0
55 **DRAW OVER** 1; 255, 175
60 **PAUSE** 200
70 **PLOT** 255, 175
80 **DRAW OVER** 1; — 255,— 175

Przykład 6.13.

Dany jest program:

```
  5 REM Przyklad 6.13
 10 FOR i=1 TO 22
 20 FOR j=1 TO 16
 30 PRINT "■ ";
 40 NEXT j: NEXT i
 50 FOR i=1 TO 81 STEP 6
 55 CIRCLE OVER 1;127,87,i
 60 NEXT i
 70 PAUSE 200
 80 FOR i=1 TO 81 STEP 6
 90 CIRCLE OVER 1;127,87,i
100 NEXT i
```

Wprowadźmy i wykonajmy ten program. Ilustruje on rysowanie okręgów instrukcją **CIRCLE** połączoną z instr. **OVER** 1, a następnie w identyczny sposób narysowanych okręgów. Zauważamy, że tło, na którym rysowane były okręgi, po ich wymazaniu jest takie samo jak przed rysowaniem.

Wymieniając instr. 90 na instr. postaci

90 **CIRCLE INVERSE** 1; 127, 87, i

po wykonaniu programu na poprzednim tle otrzymamy białe okręgi. Na zakończenie podajemy cztery programy do wprowadzenia i wykonania

Przykład 6.14.

Napisać program rysujący na ekranie ciąg prostokątów (od największego do najmniejszego), ale tak aby poprzedni prostokąt „znikał".

```
 10 REM Przyklad 6.14
 15 PAPER 2: INK 4: BORDER 0
 20 FOR n=0 TO 80 STEP 10
 30 LET x=255-2*n
 40 LET y=175-2*n
 50 PLOT n,n
 60 DRAW x,0: DRAW 0,y
 70 DRAW -x,0: DRAW 0,-y
 80 PAUSE 200
100 DRAW INVERSE 1;x,0
110 DRAW INVERSE 1;0,y
120 DRAW INVERSE 1;-x,0
130 DRAW INVERSE 1;0,-y
135 PAUSE 50
140 NEXT n
```

Wprowadźmy i wykonajmy ten program. Jest on dobrą ilustracją działania instrukcji INVERSE 1;

Przykład 6.15.

```
 10 REM Przyklad 6.15
 20 PAPER 2: CLS
 30 LET a$="■"
 40 FOR i=1 TO 14
 50 LET a$=a$+"■"
 60 NEXT i
 80 FOR i=1 TO 22
 90 PRINT a$
100 NEXT i
110 REM lewa czesc ekranu zosta
ła zamalowana na czarno,prawa je
st w kolorze papieru
120 FOR i=1 TO 500
130 PLOT INK 7;RND*255,RND*175
140 NEXT i
150 REM na ekranie wypisano los
owe punkty w kolorze atramentu
160 REM RND - generator liczb p
seudolosowych z przedzialu <0,1)
```

Wprowadźmy i wykonajmy ten program. Opis programu zamieszczony jest w komentarzach (po słowie REM). Każde uruchomienie programu może dać inny wynik

Przykład 6.16.

Dany jest program

```
 10 REM Przyklad 6.16
 20 BORDER 0: PAPER 7: CLS : BR
IGHT 1
 30 FOR i=1 TO 400
 40 PRINT INK 7*RND;AT 21*RND,1
5*RND;"■"
 50 PLOT FLASH 1; INK 7*RND;130
+125*RND,175*RND
 60 NEXT i
```

Wprowadźmy i wykonajmy ten program. Odpowiedź na pytanie dlaczego tak, a nie inaczej wykonuje się ten program, pozostawiamy czytelnikowi

Przykład 6.17.

Dany jest program

```
 10 REM Przyklad 6.17
 20 BORDER 0: PAPER 5. CLS
 30 LET x=1
 40 FOR l=0 TO 21
 50 FOR c=1 TO 6
 60 PRINT INK c;AT l,c+x;"■"
 70 NEXT c
 80 LET x=x+1
 90 NEXT l
```

Wprowadźmy i wykonajmy ten program. Na telewizorze kolorowym otrzymamy ładny rysunek, a na czarno-białym rysunek w kolorach biało-szaro-czarnych

ZW

GRAFIKA (3)
COMMODORE 64

W poprzednim artykule z tego cyklu przedstawiony został znakowy tryb pracy mikrokomputera C-64. Grafika w tym trybie polega na tworzeniu określonych kompozycji ze znaków standardowych lub samodzielnie zaprojektowanych. Możliwości graficzne są jednak ograniczone. Trudności te znikają, gdy potrafimy korzystać z trybu graficznego o dużej rozdzielczości (nazywanego dalej trybem graficznym). Podstawowe wiadomości z tego zakresu zostaną przedstawione w tym artykule. Tryb graficzny opisany zostanie na przykładzie programu FUNKCJA, służącego do kreślenia funkcji na ekranie.

Mapa bitowa ekranu

Jak pamiętamy z poprzedniej części, obraz w trybie znakowym **VIC** tworzy na podstawie zawartości **PAMIĘCI EKRANU** (i **GENERATORA ZNAKÓW**). W trybie graficznym wykorzystywany jest obszar pamięci zwany **MAPA BITOWA EKRANU** (**MBE**). Każdemu punktowi ekranu odpowiada określony bit pamięci **MBE**. Strukturę ekranu i jej odzwierciedlenie w pamięci przedstawia rys. 1

Cały ekran podzielony jest na 1000 sekcji rozmieszczonych w 25 wierszach i w 40 kolumnach. Każda sekcja składa się z 64 punktów (8 x 8). W wyniku takiego podziału dysponujemy 320 punktami w poziomie (40 x 8) oraz 200 punktami w pionie (25 x 8). Sekcje w pamięci mikrokomputera zajmuje 8 bajtów (64 bity). Rozmieszczenie sekcji w pamięci przedstawia rysunek. Numery sekcji na ekranie odpowiadają numerom w pamięci. Cała MBE zajmuje 8000 bajtów (1000 x 8).

Istnieją dwie podstawowe metody programowania grafiki w trybie graficznym

1. Projektujemy obraz na siatce odpowiadającej punktom ekranu. Wyznaczamy wartości liczbowe odpowiednich bajtów MBE. Zapisujemy je w MBE
2. Dla dowolnego punktu ekranu wyznaczamy odpowiadający mu bit MBE i ustawiamy na określoną wartość (0 lub 1)

Wykorzystanie drugiej metody wiąże się z interpretacją ekranu jako ćwiartki układu współrzędnych kartezjańskich. W naszych rozważaniach przyjmiemy za środek układu lewy dolny róg, oś X jest pozioma i skierowana w prawo, oś Y jest pionowa i skierowana do góry. Na osi Y wyróżniamy 200 punktów ponumerowanych od 0 do 199. Na osi X wyróżniamy 320 punktów w trybie dwukolorowym lub 160 punktów w trybie wielobarwnym, ponumerowanych od 0 do 319 lub 159. Dla przypomnienia: w trybie wielobarwnym jednemu punktowi na ekranie odpowiada para bitów w pamięci.

Wyznaczając bit MBE, odpowiadający wybranemu punktowi ekranu o współrzędnych X i Y, stosujemy procedurę (w trybie dwukolorowym)

```
200 REM * PUNKT *
210 YY = 199 — Y
220 SW = INT (YY/8): REM NR WIERSZA 0 ... 42
230 SK = INT (X/8): REM NR KOLUMNY 0 ... 39
240 S = SW * 40 + SK: REM NR SEKCJI 0 ... 999
250 SL = YYAND7: REM NR LINII NR BAJTU W SEKCJI 0 ... 7
260 BAJT = S * 8 + SL + MAP: REM NR BAJTU PAMIĘCI
270 SP = XAND7: REM NR PUNKTU W LINII 0 ... 7
280 BIT = 7 — SP: REM NR BITU W BAJCIE 7... 0
300 RETURN
```

Dla trybu wielobarwnego należy wprowadzić następujące zmiany:

```
230 SK = INT (X/4): REM NR KOLUMNY 0 ... 39
270 SP = XAND3: REM NR PUNKTU W LINII 0 ... 3
280 BIT = 3 — SP
290 B2 = 2 * BIT: B1 = B2 + 1: REM NR BITÓW W BAJCIE
```

Parametrami procedury **PUNKT** są
— X, Y — współrzędne punktu
— MAP — adres początkowy MBE w pamięci mikrokomputera

Wykonując tę procedurę otrzymujemy adres bajtu pamięci BAJT i numer (y) bitu (ów) BIT (B1 i B2) odpowiadające punktowi na ekranie. Na wyznaczonym bicie możemy dokonać operacji:
— ustawić wartość bitu na 1
POKE BAJT, **PEEK** (BAJT) **OR** (2↑BIT)
— ustawić wartość bitu na 0
POKE BAJT, **PEEK** (BAJT) **AND** (255 — 2↑BIT)

Dwukolorowy tryb graficzny

Opis pracy w trybie graficznym przedstawiony jest na przykładzie programu **FUNKCJA** i dlatego w instrukcjach zachowana jest numeracja wierszy programu

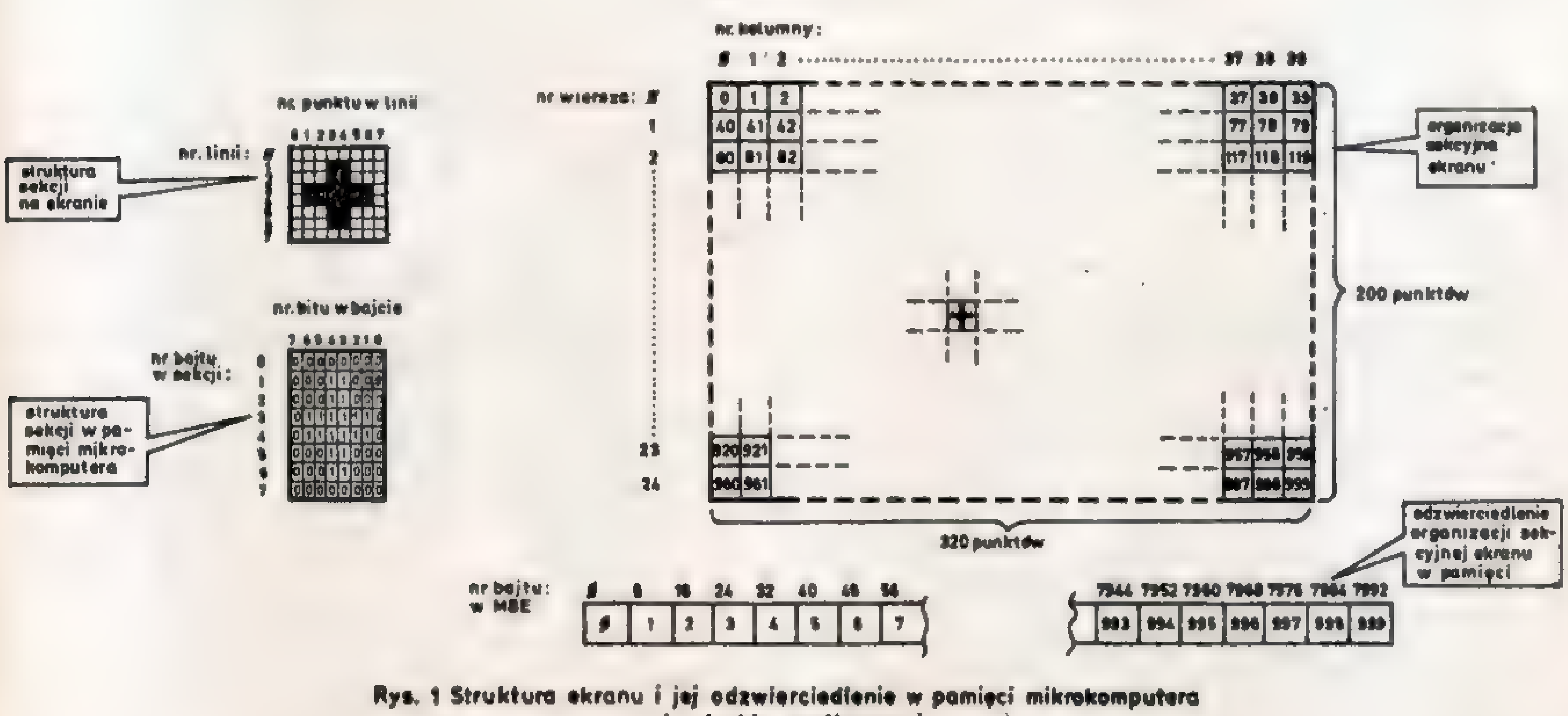
Tryb graficzny dużej rozdzielczości inicjujemy ustawiając piąty bit w 53265 bajcie na 1

20 **POKE** 53265, **PEEK** (53265) **OR** 32

Powrót do trybu znakowego otrzymujemy po wykonaniu

120 **POKE** 53265, **PEEK** (53265) **AND** 223

W trybie dwukolorowym definiujemy kolory w każdej sekcji osobno. Jako pamięć kolorów VIC wykorzystuje PAMIĘĆ EKRA-



Rys. 1 Struktura ekranu i jej odzwierciedlenie w pamięci mikrokomputera (w trybie graficznym)

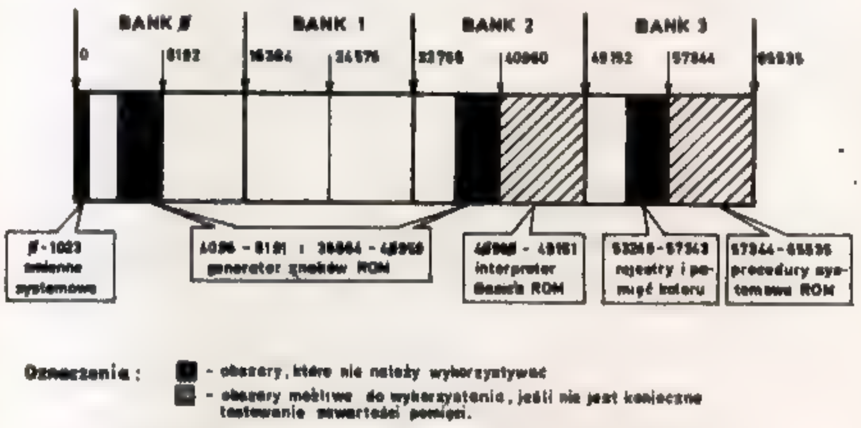
NU (PE). Każdej sekcji przyporządkowany jest bajt PE (kolejność występowania sekcji w MBE odpowiada kolejności bajtów PE). Kolor bitu o wartości 0 określają 4 najmłodsze bity w przyporządkowanym mu bajcie PE. Kolor bitu o wartości 1 określają 4 najstarsze bity.

W programie FUNKCJA tło ma być jednolite na całym ekranie (bity zerowe w MBE) koloru białego — kod koloru 1. Funkcja ma być kreślona (bity o wartości 1) w kolorze czarnym — kod 0. Należy zatem wykonać

40 **FOR** I = MAP **TO** MAP + 7999: **POKE** I,0 **NEXT**
50 **FOR** I = KAP **TO** KAP + 999: **POKE** I,1: **NEXT**

KAP jest adresem początkowym PE.

Musimy obecnie wybrać obszary pamięci, w których rezydować będzie MAPA BITOWA EKRANU i PAMIĘĆ EKRANU. Na rys. 2 przedstawiony jest zarys podstawowej konfiguracji pamięci C-64 z podziałem na BANKi i ewentualne obszary, w których można umieścić MBE i PE.



Rys. 2 Obszary pamięci C64 do wykorzystania na lokalizację mapy bitowej ekranu i pamięci kolorów (pamięci ekranu)

Jak wynika z rysunku, MBE może zajmować jedną z dwóch połówek BANK-u. PE powinna być umieszczona w innej połowie niż MBE. Określając położenie MBE i PE, możemy skorzystać z tabelek zamieszczonych w I części artykułu („IKS" nr 2/87). Wyboru w BANK-u dokonujemy, ustawiając odpowiednią wartość w 56576 bajcie. Lokalizację ustalamy wprowadzając do rejestru 53272 wartość z kolumny wyznaczającej adres początkowy MBE i wiersza wyznaczającego adres początkowy PE.

30 **POKE** 56572, 150: **POKE** 53272, 9: MAP = 24576: KAP = 16384

Funkcję do wykreślenia definiować będziemy w postaci wyrażenia arytmetycznego

11 **DEF FN** F1 (XX) = SIN (XX)

Konieczne jest również podanie zakresu zmienności argumentu (XP, XK) i wartości funkcji (YP, YK):

10 XP = 0: XK = 8: YP = —2: YK = 2

Funkcję będziemy kreślić następującą sekwencją instrukcji:

60 DX = 320 {(XK — XP): DY = 200} (YK — YP)
70 FOR I = XP TO XK STEP ABS (1/DX)
80 X = INT (I — XP) DX
90 Y = INT {(FN F1 (I) — YP) DY}: GOSUB200: POKE BAJT, 2↑BIT
100 NEXT

Dla otrzymania gotowego programu FUNKCJA dołączamy procedurę PUNKT i jeszcze dwie linie programu:

100 GETZ$: IF Z$ = "" THEN110
140 END

Możemy teraz uruchomić nasz program komendą RUN. Po zakończeniu kreślenia funkcji program czeka na wciśnięcie dowolnego klawisza, co spowoduje powrót do trybu znakowego i zakończenie pracy.

Wielobarwny tryb graficzny

Wykonanie dwóch lub trzech różnokolorowych wykresów funkcji na tym samym ekranie w trybie dwukolorowym jest niemożliwe. Możemy rozwiązać ten problem wykorzystując tryb wielobarwnej pracy VIC-a, który pozwala nam na zdefiniowanie czterech różnych kolorów w każej sekcji.

Tryb wielobarwny uzyskamy instrukcją
25 POKE 53270, PEEK (53270) OR 16

Powrót do trybu dwukolorowego umożliwia nam:
125 POKE 53270, PEEK (53270) AND 239

Jak już wspomniałem wcześniej, jednemu punktowi na ekranie odpowiada para bitów wskazująca, skąd pobrać kod barwy tego punktu. Odbywa się to według zależności:

PARA BITÓW	LOKALIZACJA KOLORU
00	rejestr 53281
01	najstarsze 4 bity odpowiedniego bajtu PE
10	najmłodsze 4 bity odpowiedniego bajtu PE
11	najmłodsze 4 bity PAMIĘCI KOLORU

Wprowadźmy do naszego programu definicje nowych funkcji
12 DEF FN F2 (XX) = COS (XX)
13 DEF FN F3 (XX) = SIN (XX) + COS (XX)

Każda z funkcji ma być w innym kolorze: **F1** — czarna; **F2** — biała. **F3** — zielona. Tło w kolorze niebieskim. W tym celu należy wykonać instrukcję
55 POKE 53281, 6: FOR I = 55296 TO 56295: POKE I,5: NEXT

Ponieważ w trybie wielobarwnym mamy w poziomie tylko 160 punktów, na ekranie należy zmienić wiersz
60 DX = 160/(XK — XP): DY = 200/(YK — YP)

Do sekwencji instrukcji kreślących funkcję wprowadzamy nowe wiersze rysujące poszczególne krzywe

81 Y = INT {(FN F1 (I) — YP)/DY}: GOSUB200: POKE BAJT, PEEK (BAJT OR (2↑B2) AND (255 — 2↑B1)
82 Y = INT {(FN F2 (I) — YP)/DY}: GOSUB200: POKE BAJT, PEEK (BAJT) OR (2↑B1) AND (255 — 2↑B2)
83 Y = INT {(FN F3 (I) — YP)/DY}: GOSUB200: POKE BAJT, PEEK (BAJT) OR (2↑B1 + 2↑B2)

Musimy jeszcze tylko usunąć wiersz 90 i dokonać modyfikacji procedury PUNKT i już mamy gotowy program FUNKCJA, pozwalający na wykreślenie trzech różnokolorowych krzywych.

O innych ciekawych właściwościach VIC-a będzie można przeczytać w kolejnym artykule.

S. WASIELEWSKI



— ZAKAŁAPUĆKAŁ SIĘ !!!

Zmienne systemowe ROM-Spectrum

Wejście-wyjście

CHANS 23631/2
CURCHL 23633/4
STRMS 123568 605
P-POSN 23679
PRCC 23680/1
ECHO-E 23682/3

Interpretacja komend obsługujących urządzenia wejścia--wyjścia opiera się na dwóch metodach komunikacji: przez porty i kanały. **Porty** to w zasadzie dodatkowa przestrzeń adresowa, z którą wymiana informacji odbywa się rozkazami IN (odpowiednik PEEK) oraz OUT (odpowiednik POKE). Za pomocą tych instrukcji możemy kontrolować stan urządzenia zewnętrznego. I tak, drukarka korzysta z portu 254, joystick Kempstona z portu 31, a Interface I pracuje na portach 254, 239 i 247. Głośnik i magnetofon są również obsługiwane przez 254.

Kanały są formą określenia kierunku i sposobu prowadzenia korespondencji z urządzeniem zewnętrznym. Firmowo określono ich cztery: K-klawiatura (keyboard), S-monitor (screen), P-drukarka (printer) oraz R-bufor edycyjny (editing buffer channel record). Dane kanałowe umieszczone są w ROM-ie, lecz podczas inicjalizacji systemu następuje ich przerzucenie do RAM, a więc można je modyfikować. Obejmują 21 bajtów od adresu 23734, który zmienia CHANS.

Dla przykładu definicja kanału S zawiera.

adres	zawartość	znaczenie
23739	244	2548 — adres procedury wyjścia —
23740	9	wydruk znaku na ekranie;
23741	196	5572 — adres wejścia — w tym wy-
23742	21	padku jest to procedura powodująca raport Invalid I/O device;
23743	83	„s" — identyfikator kanału;

Kanał **K** jest jedynym, który akceptuje zarówno dane wejściowe (dekodowanie przyciśniętego klawisza), jak też dane wyjściowe (wydruk znaku ASCII), co wynika z jego specyfiki. Jeśli kanał nie ma wejścia, to drukowany jest komunikat *Invalid I/O device*.

Tajemniczy kanał **R** jest używany wewnętrznie przez interpreter do przesyłania danych do bufora edycyjnego, co odbywa się po naciśnięciu **EDIT** lub do przestrzeni roboczej (z reguły operacje z łańcuchami tekstowymi). Nie można się nim posługiwać, programując w Basic'u, natomiast można go tak zmodyfikować, aby edycja linii była niemożliwa:

POKE 23744,124
POKE 23745,0

Dane kanałowe kończy bajt 128 będący jednocześnie sygnałem rozpoczęcia obszaru Basic.

Jeśli nie manipulujemy obszarem danych kanałowych (a zwykle nie), można się go pozbyć zwiększając pamięć przeznaczoną dla Basic'a:

```
10 POKE 23635, PEEK 23631
20 POKE 23636, PEEK 23632
30 POKE 23631, 175
40 POKE 23632, 21
50 REM PROG=23734, CHANS=5551
```

Zmienna CHANS wskazuje teraz obszar tych samych danych kanałowych, ale już w ROM-ie.

W danym momencie można przesłać dane tylko jednym kanałem, z którego korzystać można przez wybór skojarzonego z nim strumienia. Dane o strumieniach umieszczone są w zmiennej **STRMS** liczącej sobie 38 bajtów, co pozwala na zdefiniowanie 19 sposobów przesłań. Dwa bajty dla każdego strumienia zawierają liczbę, która dodana do 23733 (CHANS-1) wskazuje początek definicji wybranego kanału. Po włączeniu systemu do STRMS zostaje wysłane 14 bajtów identyfikujących 7 strumieni początkowych:

nr strumienia	kanał skojarzony	przesunięcie
253	K	1
254	S	6
255	R	11
0	K	1
1	K	1
2	S	6
3	P	16

Pierwsze dwa są zarezerwowane dla procedur obsługujących magnetofon i jeszcze do nich wrócimy. Strumień 255 współpracuje z kanałem **R**, strumień **0** z edytorem, 1 z wprowadzaniem danych (INPUT), 2 wysyła dane na ekran, 3 na drukarkę. Reszta 4..15 pozostaje do dyspozycji użytkownika.

Przed przesłaniem danych strumień musi być otwarty, tzn. skojarzony z odpowiednim kanałem. W chwili przesyłania kanał ten staje się dla systemu kanałem aktualnym i początek jego danych zostaje wpisany do zmiennej **CURCHL**. Teraz przykład:

```
10 BORDER 4
20 OPEN #4,"K"
30 LIST
40 LIST #4
50 CLOSE #4
60 GOTO 60
```

Otworzyliśmy (OPEN #) strumień 4 wg parametrów kanału **K**, aby wysłać listing na ekran edycyjny. Fizycznie otwarcie to polega na wpisaniu wspomnianego już przesunięcia do odpowiednich bajtów **STRMS**. Można to uczynić również komendami **POKE**. Zamknięcie strumienia (CLOSE #) polega na wpisaniu przesunięcia równego zeru. Zabronione jest zamykanie strumieni standardowych (0..3).

Po otwarciu przesyłamy dane komendą **PRINT**, a odbieramy rozkazem **INPUT**. Przy pracy w warunkach początkowych nie wyszczególniamy numerów strumieni przy tych instrukcjach, gdyż maszyna robi to automatycznie. **PRINT** oznacza więc naprawdę **PRINT** # 2, a **INPUT** posługuje się strumieniem 1 (INPUT # 1). LLIST znaczy to samo, co **LIST** # 3.

Wychodzić można na wszystkie standardowe kanały, lecz **INPUT** działa tylko w porozumieniu z kanałem K. Próba zaburzenia tego rozwiązania prostymi sposobami będzie nieudana.

```
10 POKE 23741,168
20 POKE 23742,16
30 INPUT #2;AT 1,0; LINE d$
```

Modyfikacja adresu wyjścia się udała, ale tylko na pierwszy rzut oka. Zwykle, jeśli użytkownik ma chęć na swoisty typ przesłania, wtedy definiuje własny kanał i pisze procedury wejścia i wyjścia. Dokładnie pokazuje to wybrany przykład **PRINT 16**. Program ten pozwala otrzymywać na monitorze znaki dwukrotnie szersze od firmowych przy zachowaniu możliwości korzystania ze wszystkich kodów kontrolnych.

Opis programu: (str. 26)

linie 160..240 — dołączenie danych kanału **D** na koniec danych kanałowych przez przesunięcie wskaźników RAM o 5 bajtów w górę (PROG = 23760);
linie 250..260 — otwarcie strumienia 5 przez skojarzenie z kanałem D; linie 280..300 — dane kanału **D**,
linie 310..550 — obsługa kodów kontrolnych w procedurze wyjścia;
linie 560...950 — obsługa kodów drukowanych w ASCII, (grafiki, słowa kluczowe);
linia 970 — obszar tworzenia szerokiego znaku.

Po kompilacji (lub wczytaniu bajtów już skompilowanych) należy zainstalować kanał przez np. **RANDOMIZE USR 6000**, a potem korzystać, stosując komendę wyjścia np.:

PRINT # 5; **AT** 5,5; **INVERSE** 1; *„prusakolep"*

Kod nie jest relokowalny. Zmiana adresu początkowego może odbyć się tylko po wczytaniu tekstu źródłowego do asemblera (linia 10) i ponownej kompilacji. Jeśli uda się Wam „podłączyć" znany program **PRINT 64** do strumienia 4, to będziecie dysponować możliwością jednoczesnego drukowania na ekranie w trzech trybach znakowych.

W zasadzie metoda kanałowo-strumieniowa staje się użyteczna w programowaniu dopiero po dołączeniu INTERFACE 1. Mamy tu rozszerzenie o cztery typy kanałów dających dużo większe możliwości. Pierwszy dotyczy microdrive'u i pozwala zapisywać dane w postaci plików.

```
10 OPEN #4,"m";1;"nazwa pliku"
20 PRINT #4,x
30 CLOSE #4
```

Zapisaliśmy na kasetę (cartridge) zmienną x, ale możemy też zapisać listing (LIST # 4), katalog (CAT # 4,1) lub dowolny zbiór liczb, czy znaków nawet bezpośrednio z klawiatury (INKEY$ # 4).

Następne dwa kanały dotyczą złącza szeregowego RS 232. Pozwala ono przesyłać dane w transmisji szeregowej, tzn. bit po bicie. Kanał **B** akceptuje wszystkie kody, lecz kanał **T** tylko podstawowy zbiór drukowanych znaków ASCII ignorując 8 bit. Ich zaletą jest możliwość jednoczesnego otwarcia, co można wykorzystać np. do wysyłania na drukarkę znaków kontrolnych kanałem **B**, a znaków ASCII kanałem **T**.

Czwarty kanał **N** dotyczy pracy w sieci, a więc pozwala na komunikację między poszczególnymi jednostkami Spectrum.

Powróćmy teraz do strumieni wykorzystywanych przez procedury kasetowe. Strumień 253 służy do wydruku napisu *Start tape, then press any key*, a więc używa wyjścia kanału **K**. W wypadku zapisu grafiki ekranowej napis ten niszczy dwie dolne linie obrazka. Jeśli chcemy zapisać cały ekran, postępujemy następująco:

```
10 POKE 23734,124: POKE 23735,0
20 PAUSE 1: PAUSE 0
30 SAVE "nazwaobrazka"SCREEN$
40 POKE 23734,244: POKE 23735,9
```

Zmieniono tu adres procedury wyjścia kanału **K**, który w krytycznym momencie wynosi 124 (pod tym adresem jest tylko instrukcja RET). Strumień 254 jest używany do wydruku typu i nazwy programu po przyjęciu nagłówka. Czasem informacje te nie są istotne, lecz niszczą wczytany wcześniej obrazek pilotujący. Przykładowy loader rozwiązujący tę kwestię:

Tu dla odmiany nastąpiła modyfikacja wyjścia dla kanału s. Jeżeli program wczytywany przez linię 30 ma autostart, to linię 40 trzeba zawrzeć wewnątrz niego, gdyż w przeciwnym razie nie będzie wyjścia na ekran główny.

```
10 LOAD "nazwa"SCREEN$
20 POKE 23739,124: POKE 23740,0
30 LOAD "program"
40 POKE 23739,244: POKE 23740,9
50 RUN
```

Na zakończenie gwoli ścisłości o trzech pozostałych zmiennych wyszczególnionych na początku.

P-POSN i **PR-CC** dotyczą drukarki. Pierwsza trzyma aktualną pozycję znaku w buforze, który rezerwuje obszar od adresu 23296 do 23551 włącznie. Po przesłaniu 32 znaków wysyłany jest kod nowej linii.

Druga natomiast dla tejże pozycji przechowuje adres w buforze. Starszy bajt **PR-CC** (23681) figuruje w instrukcji firmowej jako wolny, co w razie korzystania z drukarki czyni to stwierdzenie nieprawdziwym.

Zmienna **ECHO-E** przechowuje kopię pozycji wydruku w ekranie edycyjnym, czyli zmiennej SPOSNL. W momencie operacji w liniach edycji uaktualniana jest jednak tylko ta ostatnia, więc interpreter otrzymuje informacje skąd i dokąd drukowano. Ma to znaczenie w wypadku dodrukowywania w każdej linii ekranu edycyjnego spacji uzupełniających w kolorach ATTR-P.

Kolejnym tematem będą zmienne stanu systemu...

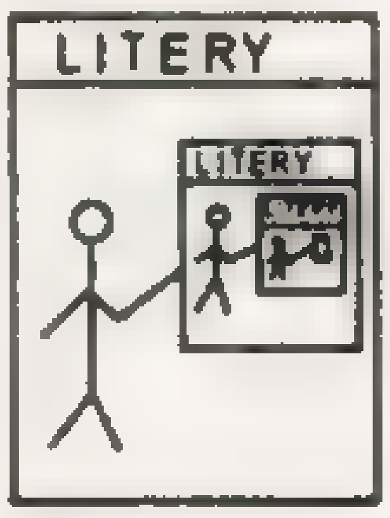
```
 10 ; ** PRINT16
 20 ; BEN83M2 assembler
 30             ORG  60000
 40 STR_5       EQU  23584
 50 CHARS       EQU  23606
 60 UDG         EQU  23675
 70 TVDATA      EQU  23566
 80 KANALD      EQU  23754
 90 GETPOS      EQU  #0B03
100 CTRL        EQU  #0A04
110 PRT?        EQU  #0A69
120 RAPORT      EQU  #15C4
130 EXKAN       EQU  #0A80
140 POCONT      EQU  #0A87
150 MAKERM      EQU  5717
160
170 INSTAL      LD   HL,KANALD
180             PUSH HL
190             LD   BC,5
200             CALL MAKERM
210             LD   HL,DEFKAN
220             POP  DE
230             LD   BC,5
240             LDIR
250             LD   HL,21
260             LD   (STR_5),HL
270             RET
280 DEFKAN      DEFW OUT_5
290             DEFW RAPORT
300             DEFB "D"
310 OUT_5       CP   32
320             JR   NC,ZNAKI
330             CALL GETPOS
340             CP   6
350             JP   C,PRT?
360             CP   #18
370             JP   NC,PRT?
380             CP   16
390             JP   C,CTRL
400             LD   HL,KODPOS
410             CP   22
420             JR   NC,POS
430             LD   HL,KODKOL
440 POS         PUSH HL
450             JP   GETPOS
460 KODPOS      LD   DE,KANPOS
470             JR   KANKOL
480 KODKOL      LD   DE,WYJ
490 KANKOL      LD   (TVDATA),A
500             JP   EXKAN
510 WYJ         LD   DE,OUT_5
520             JP   POCONT+3
530 KANPOS      LD   DE,WYJ
540             LD   (TVDATA+1),A
550             JP   EXKAN
560 ZNAKI       LD   DE,(CHARS)
570             LD   H,0
580             LD   L,A
590             ADD  HL,HL
600             ADD  HL,HL
610             ADD  HL,HL
620             ADD  HL,DE
630             EX   DE,HL
640             LD   HL,OBSZAR
650             PUSH HL
660             POP  IX
670             LD   B,8
680 PETLA       LD   A,(DE)
690             PUSH BC
700             CALL TWZNAK
710             POP  BC
720             LD   (IX+0),L
730             LD   (IX+8),H
740             INC  IX
750             INC  DE
760             DJNZ PETLA
770             LD   HL,(UDG)
780             PUSH HL
790             LD   HL,OBSZAR
800             LD   (UDG),HL
810             LD   A,145
820             CALL #09F4
830             LD   A,144
840             CALL #09F4
850             POP  HL
860             LD   (UDG),HL
870             RET
880 TWZNAK      CALL BITY4
890             LD   L,H
900 BITY4       LD   B,4
910 BIT1        RRCA
920             RR   C
930             SRA  C
940             DJNZ BIT1
950             LD   H,C
960             RET
970 OBSZAR      DEFS 16
```

Krzysztof MAMCARZ

PASCAL (6)

Ten i kolejny odcinek adresowane są do Czytelników zainteresowanych głębiej możliwościami języka PASCAL: algorytmami rekurencyjnymi, przetwarzaniem tekstów, grafiką. Przedmiotem prezentowanego odcinka są programy rekurencyjne.

Obiekt nazywamy rekurencyjnym, *jeżeli częściowo składa się z siebie samego lub jego definicja odwołuje się do niego samego.* Rekurencję spotykamy nie tylko w matematyce, ale także w życiu codziennym. Wielu Czytelników pamięta jeszcze zapewne pierwszy elementarz, na którego okładce znajdował się następujący obrazek (rekurencyjny):



Rekurencja jest szczególnie przydatnym narzędziem w definicjach matematycznych. Pozwala ona na definiowanie funkcji przez określenie jej wartości w kroku pierwszym i wyznaczenie wartości funkcji dzięki określeniu jej wartości w kroku pierwszym i wyznaczenie wartości funkcji w każdym kolejnym kroku na podstawie wartości otrzymanej w kroku poprzednim. Klasycznym przykładem jest następująca definicja silni: $n! = \begin{cases} 1 & \text{dla } n = 0 \\ n \cdot (n-1)! & \text{dla } n > 0 \end{cases}$

Opis taki można niemal automatycznie przekształcić na deklarację funkcji pascalowej. Program wyznaczania wartości n! dla $n \leq 7$ został zamieszczony w przykładzie 19.

Przykład 19

Występujący w programie zapis {$a—} jest niezbędny dla programów rekurencyjnych. W praktyce wiele problemów można opisać za pomocą mechanizmu rekurencji. Weźmy pod uwagę następujące zadanie: *„należy wyznaczyć wszystkie możliwe rozmieszczenia k obiektów w n pudełkach, tak aby w każdym pudełku znalazł się jeden obiekt (zakładamy, że $k \leq n$)".*

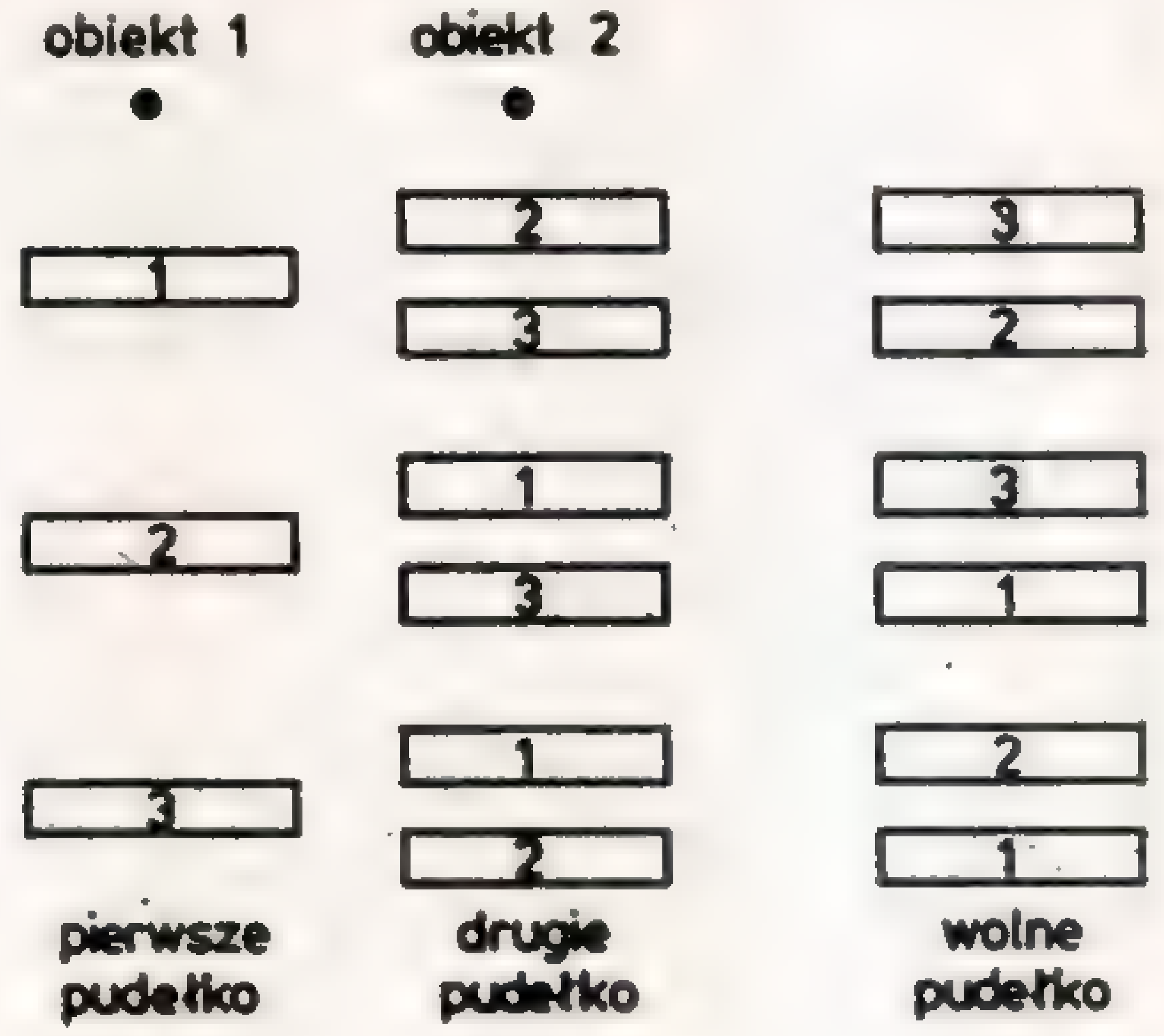
```
PROGRAM prog18:
{$a-}
 VAR
  n:integer;
 FUNCTION silnia(n:integer):integer;
  BEGIN
   IF n=0 THEN
        silnia:=1
   ELSE
        silnia:=n*silnia(n-1)
  END;
 BEGIN
  READ(n);
  WRITELN:
  IF (n>=0) AND (n<=7) THEN
   WRITELN(n:1,'!=',silnia(n))
  ELSE
   WRITELN('za duza wartosc n')
 END.
```

Metodę wyznaczania takich rozmieszczeń prześledzimy dla $k = 2$ i $n = 3$. Można przyjąć następujący sposób postępowania

a) spośród wszystkich pudełek wybieramy jedno pudełko (zwane dalej pierwszym) i wkładamy do niego pierwszy obiekt.
b) spośród dwóch pozostałych pudełek wybieramy jedno pudełko (zwane drugim) i wkładamy do niego drugi obiekt.
c) nie zmieniając pierwszego pudełka, przekładamy drugi obiekt do następnego pudełka.

Wszystkie możliwe rozmieszczenia uzyskamy, powtarzając czynności a ÷ c dla każdego z możliwych sposobów wyboru pierwszego pudełka

Jeżeli ponumerujemy pudełka od 1 do 3 i przyjmiemy zasadę, że spośród wolnych pudełek wybieramy pudełko o najmniejszym numerze, to dla tego przypadku uzyskamy następujące kolejne rozmieszczenia obiektów:



Jeżeli przez **P** oznaczymy zbiór numerów pudełek — FP = {1, 2,...,n} — to nasze zadanie sprowadza się do znalezienia wszystkich ciągów równowartościowych $<P_1, P_2, ..., P_k>$ o wyrazach ze zbioru **P** (są to tzw. k-elementowe wariacje bez powtórzeń ze zbioru n-elementowego). Wyraz P_1 takiego ciągu możemy wybrać na n sposobów, wraz P_2, na n-1 sposobów, ogólnie: przy ustalonych wyrazach $P_1, P_2, ..., P_{i-1}$ możemy jako pi wybrać dowolny spośród n-j + 1 elementów zbioru $P \setminus \{P_1, P_2, ..., P_{i-1}\}$.

Program wyznaczania k-elementowych wariacji bez powtórzeń ze zbioru n-elementowego zamieszczonego w przykładzie 20.

Przykład 20

```
PROGRAM prog19;
{$a-}
 CONST
  nn=9;
  kk=7;
 TYPE
  elem=ARRAY[1..nn] OF char;
  kontr=ARRAY[1..nn] OF boolean;
  sukc=ARRAY[1..kk] OF char;
 VAR
  dowybr:elem;
  praw:kontr;
  wynik:sukc;
  i,n,k,liczw:integer;
 PROCEDURE wyprowadz;
  VAR
   m:integer;
  BEGIN
   IF liczw>20 THEN
    BEGIN
     REPEAT UNTIL keypressed;
     clrscr;
     liczw:=2
    END
   ELSE
    liczw:=liczw+1;
   FOR m:=1 TO k-1 DO
    WRITE(wynik[m]);
   WRITELN(wynik[k])
  END;
```

```
PROCEDURE wybierz(i,n,k:integer; dowybr:ele_,
                  praw:kontr; VAR wynik:sukc);
 VAR
  l:integer;
 BEGIN
  IF i<k THEN
   BEGIN
    FOR l:=1 TO n DO
     IF praw[l]=true THEN
      BEGIN
       wynik[i]:=dowybr[l];
       praw[l]:=false;
       wybierz(i+1,n,k,dowybr,praw,wynik);
       praw[l]:=true
      END
   END
  ELSE
   FOR l:=1 TO n DO
    IF praw[l]=true THEN
     BEGIN
      wynik[k]:=dowybr[l];
      wyprowadz;
     END
 END;
BEGIN
 REPEAT
  READLN(k,n)
 UNTIL (k<=n);
 FOR i:=1 TO n DO
  BEGIN
   praw[i]:=true;
   READ(dowybr[i])
  END;
 clrscr;
 WRITELN;
 WRITE(' ',k,'-elementowe wariacje z ',
       n,'-elementowego zbioru:');
 FOR i:=1 TO n DO
  WRITE(' ',dowybr[i]);
 WRITELN;
 WRITELN;
 liczw:=1;
 wybierz(1,n,k,dowybr,praw,wynik)
END.
```

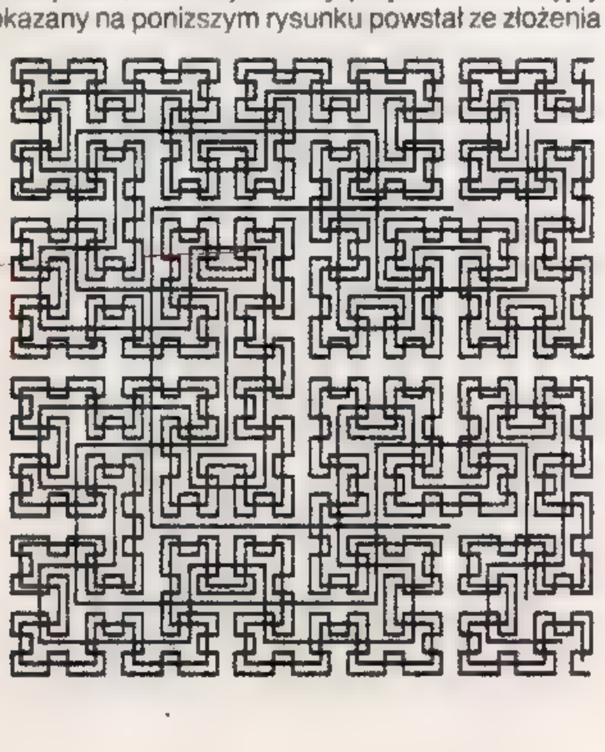
W programie wykorzystano:

— tablice znaków *dowybr* (do przechowania nazw elementów zbioru n-elementowego) i *wynik* (do przechowania kolejnych wariacji); elementami tych tablic są pojedyncze znaki,

— tablice wartości logicznych *praw* (do określania, spośród których elementów zbioru można wybierać kolejny wyraz ciągu),

— funkcję standardową *keypressed* przyjmującą wartość *true*, jeśli zostanie wciśnięty klawisz; w tym wypadku funkcja ta umożliwia stronicowanie wyświetlanych wariacji (po 20 na stronie) — zmiana strony następuje po wciśnięciu klawisza.

Za Wirthem *(Algorytmy + Struktury danych = Programy, s. 142)* przytoczymy jeszcze jeden, bardziej złożony przykład. Atrakcyjny układ graficzny pokazany na poniższym rysunku powstał ze złożenia pięciu krzywych. Regularny kształt tych krzywych sugeruje możliwość kreślenia ich za pomocą pisaka sterowanego przez komputer. Naszym celem jest określenie schematu rekurencji, zgodnie z którym mógłby być skonstruowany program kreślący. Trzy spośród pięciu nakładających się krzywych mają kształt przedstawiony na poniższym rysunku; oznaczamy je przez H_1 H_2 H_3.



Rys. 1



Rys. 2

Po bliższym przyjrzeniu okazuje się, że krzywą H_{i+1} otrzymano ze złożenia 4 krzywych H_i dwukrotnie zmniejszonych, odpowiednio obróconych i powiązanych razem trzema liniami. Zauważmy, że H_1 może być traktowana jako złożenie 4 pustych krzywych H_0, powiązanych ze sobą trzema liniami prostymi. H_i jest zwana *krzywą Hilberta rzędu i*, od nazwiska wynalazcy D. Hilberta (1891).

Załóżmy, że naszymi podstawowymi narzędziami do kreślenia są dwie zmienne współrzędne *x* i *y*, procedura *ustaw pióro* (ustawiająca pióro pisaka w punkcie o współrzędnych *x* i *y*) i procedura *kreśl* (przesuwająca pióro z jego aktualnej pozycji do pozycji wskazanej przez x i y).

Ponieważ każda krzywa H_i składa się z 4 dwukrotnie zmniejszonych kopii krzywej H_{i-1}, wydaje się naturalne podzielenie procedury rysującej H_i na cztery części, z których każda rysuje krzywą H_{i-1} odpowiednio zmniejszoną i obróconą.

Jeśli te cztery części oznaczamy przez *A, B, C* i *D*, a procedury kreślące linie łączące będziemy oznaczać strzałkami odpowiednio zorientowanymi, to uzyskamy następujący schemat rekurencji:

A : D ← A ↓ A → B

B : C ↑ B → B ↓ A

C : B → C ↑ C ← D

D : A ↓ D ← D ↑ C

Program kreślący n krzywych Hilberta $H_1, H_2, ..., H_n$ zamieszczono w przykładzie 21.

Przykład 21

```
 uGRAM prog20;
  {program kresli krzywe Hilberta }
  {zadanego rzedu                 }
{$a-}
 CONST
  h0=256;
 VAR
  i,h,x,y,x0,y0,n:integer;
 PROCEDURE b(i:integer); FORWARD;
 PROCEDURE c(i:integer); FORWARD;
 PROCEDURE d(i:integer): FORWARD;
 PROCEDURE InitGrafik;
  BEGIN
   WRITE(#$1b,'0',#$1b,'y');
  END;
 PROCEDURE leaveGrafik;
  BEGIN
   WRITE(#$1b,'1',#$1b,'x');
  END;
 PROCEDURE ustawpioro;
  BEGIN
   INLINE($2A/x/$EB/$2A/y/$cd/
          $5a/$fc/$EA/$BB);
  END;
```

```
PROCEDURE kresl:
 BEGIN
  INLINE($2A/x/$EB/$2A/y/$cd/$5a/
        $fc/$F6/$BB);
 END;
PROCEDURE a(i:integer);
 BEGIN
  IF i>0 THEN
   BEGIN
    d(i-1); x:=x-h; kresl;
    a(i-1); y:=y-h; kresl;
    a(i-1); x:=x+h; kresl;
    b(i-1)
   END
 END;
PROCEDURE b;
 BEGIN
  IF i>0 THEN
   BEGIN
    c(i-1); y:=y+h; kresl;
    b(i-1); x:=x+h; kresl;
    b(i-1); y:=y-h; kresl;
    a(i-1)
   END
 END;
PROCEDURE c;
 BEGIN
  IF i>0 THEN
   BEGIN
    b(i-1); x:=x+h; kresl;
    c(i-1); y:=y+h; kresl;
    c(i-1); x:=x-h; kresl;
    d(i-1)
   END
 END;
PROCEDURE d;
 BEGIN
  IF i>0 THEN
   BEGIN
    a(i-1); y:=y-h; kresl;
    d(i-1); x:=x-h; kresl;
    d(i-1); y:=y+h; kresl;
    c(i-1)
   END
 END;
BEGIN
 initgrafik;
 i:=0; h:=h0;
 x0:=(h DIV 2)+75;
 y0:=x0+50;
 READ(n);
 REPEAT
  i:=i+1; h:=h DIV 2;
  x0:=x0+(h DIV 2);
  y0:=y0+(h DIV 2);
  x:=x0; y:=y0;
  ustawpioro;
  a(i);
 UNTIL i=n;
 REPEAT UNTIL keypressed;
 leavegrafik
END.
```

Procedura *A* jest wywoływana w programie głównym po jednym razie dla każdej z nakładanych na siebie krzywych Hilberta Program główny określa również punkt początkowy krzywej, tj. początkowe wartości zmiennych *x* i *y* oraz przyrost jednostkowy *h* Przez *ho* oznaczono całą szerokość strony, na której będą kreślone krzywe

Ponieważ w PASCAL-u obowiązuje zasada, że pierwsze wystąpienie obiektu powinno być jego definicją, zastosowano tzw. deklaracje wyprzedzające procedur *B*, *C*, i *D*

W deklaracji takiej podaje się pełny nagłówek procedury n blok procedury zastępuje się słowem kluczowym *FORWARD*

W dalszej części programu deklaruje się blok procedury, poprzedzony jej identyfikatorem (parametrów się nie podaje).

Procedury graficzne omówimy w jednym z kolejnych odcinków

S. ROZMUS

Tym razem nie było sensacji — umiarkowana frekwencja jest rzeczywistym obrazem zainteresowania informatyką w kraju.

Infowideo'87



Największą frajdę na tegorocznym Infowideo miały maluchy. Pierwszy kontakt z komputerem z pewnością na długo pozostanie w pamięci przedszkolakom.

Foto: Jan Zelman





ZŁOŚLIWE ŁADUNKI

Obwody scalone („mądre" kostki) zawierają coraz więcej coraz mniejszych funktorów zerojedynkowych. Ta miniaturyzacja powoduje, że te najnowocześniejsze, a więc MOS (Metal Oxide Semiconductor) wymagają szczególnej ostrożności: są uczulone na ŁADUNKI ELEKTROSTATYCZNE.

Przypominamy: ładunki elektrostatyczne powstają nie tylko przy pocieraniu bursztynu o wełnę, ale również, gdy trzemy SYNTETYK o... WSZYSTKO!

Kto nie wierzy, niechaj gwałtownie ściągnie sweter z anilany: może usłyszeć trzeszczenie iskierek, a jeśli będzie ciemno, to je zobaczy.

Ładunki elektrostatyczne skutecznie BLOKUJĄ obwody scalone MOS. Efekt jest taki, że komputer (kalkulator) nagle... głupieje. Coś pokazuje, ale źle!

Niektórzy „dowcipni" radzą w takiej sytuacji... wyjazd do Japonii.

Nie jest to konieczne, gdyż często możemy zreperować samodzielnie. Wymagana tu jest zręczność radioamatora i jego narzędzia (wkrętaki, pinceta, lutownica, lutowie...). Jeśli tego nie mamy, to radzimy poszukać zaprzyjaźnionego radiomajsterkowicza.

Dodatkowe wyposażenie to arkusik blachy (duży talerz), 30 centymetrów przewodu izolowanego (tzw. linka LY), lampa „kreślarska", spodeczek i ewentualnie lupka.

Z przewodu usuwamy po 10 milimetrów izolacji. Z jednego końca robimy „pędzelek", a drugi bielimy cyną.

Teraz strój: zdejmujemy SYNTETYCZNĄ odzież (sweter z anilany, nylonową koszulę...) i zakładamy bawełniane. Bardzo starannie myjemy ręce i dokładnie suszymy.

Uwaga: suche, czyste ręce bardzo często kładziemy na arkuszu blachy, aby usunąć ładunki. **Lutownice małej mocy starannie uziemiamy (korpus). Wyjmujemy komputerowe bateryjki.**

Na blachę kładziemy „zwariowany" komputer. Uważnie lokalizujemy wkręty (warto zrobić szkic rozmieszczenia). Ostrożnie wykręcamy — niektóre są przyklejone lakierem. Wykręcone kładziemy na spodeczek.

Gdy wszystkie wykręciliśmy, bardzo ostrożnie „otwieramy" komputer: *bez użycia noża, szczypiec — samymi palcami.* **Jeśli napotkamy opór, to szukajmy zapomnianego wkrętu.**

Po „otwarciu" warto zrobić szkic wnętrza: wynotować symbole obwodów scalonych.

Teraz często kładźmy ręce na blachę. Często róbmy przerwy i dużo, dużo oglądajmy: „połówki" komputera są z reguły połączone cienkimi linkami lub kruchymi taśmami.

Wyszukujemy „masę": ma ona liczne punkty lutownicze połączone zazwyczaj CZARNYM przewodem.

Do masy lutujemy jeden koniec naszej linki. Rozładowujemy ręce i drugim końcem linki „pędzlujemy" po WSZYSTKICH METALOWYCH elementach: końcówkach obwodów scalonych, metalowych ścieżkach druku, wszystkich metalowych punktach (jeśli jest wygodny dostęp, to również po drugiej stronie płytki drukowanej).

Ten zabieg LIKWIDUJE ŁADUNKI ELEKTROSTATYCZNE. Radzimy zrobić krótką przerwę, obejrzeć notatki i znów... rozładować ręce, kładąc je na blachę.

Zabieg „pędzlowania" powtarzamy (może coś przeoczyliśmy).

Znów przerwa i rozładowanie rąk. Przylutowany koniec odlutowujemy i bardzo uważnie oglądamy, czy „coś" nie odłączyliśmy.

Powoli i co najmniej uważnie składamy „połówki". Ze spodeczka pincetką bierzemy po jednym wkręcie i „dociągamy częściowo". Gdy spodeczek pusty, to „z wyczuciem" dokręcamy wkręty.

Bardzo uważnie oglądamy złożony komputer: NIE POWINNO BYĆ ŚLADÓW!

Zakładamy bateryjki (jeśli je ma).

Jeszcze raz oglądamy i START.

Życzymy „ożywienia komputera".

Janusz MILLER

AMSTRAD

GRA „VIA MALA"

Jest to prosta gra zręcznościowa, polegająca na przejechaniu jak najdłuższego odcinka krętej drogi. Wygrywa ten, kto zdobędzie większą liczbę punktów. Zetknięcie się w czasie jazdy z krawędzią drogi grozi katastrofą Życzymy udanej zabawy

```
10 CLS
20 REM 'VIA MALA'
30 PRINT:PRINT:PRINT SPC(12),"OPIS GRY"
40 PRINT:PRINT:PRINT:PRINT
50 PRINT SPC(8);CHR$(242);" - RUCH W LEWO"
60 PRINT:PRINT
70 PRINT SPC(8);CHR$(243);" - RUCH W PRAWO"
80 FOR I=1 TO 4000:NEXT.
90 ENV 1,1,15,30,15,-1,12:MODE 1:INK 0,0:PAPER 0:B
ORDER 0:INK 1,6:INK 2,26:INK 3,21
100 CLS:P=100:Z=50:X=15:PEN 2:          FOR I=1 T
O 25:PRINT TAB(X);CHR$(143);SPACE$(10);CHR$(143):N
EXT I:WX=20:LOCATE WX,1:PEN 1:PRINT CHR$(239)
110 Z=Z-1:IF Z<1 THEN Z=1
120 FOR I=1 TO Z:SOUND 2,500+Z*10,1,15-(Z\8),0,0,4
:NEXT I:SOUND 2,200+Z*20,17,15-(Z/8),0,0,4
130 LOCATE WX,1:PRINT" ":T$=INKEY$:IF T$="" THEN 1
50
140 IF T$=CHR$(243) THEN WX=WX+1 ELSE IF T$=CHR$(2
42) THEN WX=WX-1
150 P=P+(50-Z):X=X+SGN(RND-RND):IF X<1 THEN X=1 EL
SE IF X>29 THEN X=29
160 IF WX>X AND WX<X+11 THEN PEN 2:LOCATE 1,1:PRIN
T CHR$(11):LOCATE 1,1:PRINT CHR$(30);TAB(X);CHR$(1
43);SPACE$(10);CHR$(143):PEN 1:LOCATE WX,1:PRINT C
HR$(239):GOTO 110
170 CLS:FOR I=26 TO 0 STEP -1:INK 0,I:BORDER I:SOU
ND 1,0,4,15,0,0,I:NEXT         I:SOUND 1,0,100,0,1
,0,26:PEN 3:PRINT"PUNKTY:";P:PRINT:INPUT "GRASZ DA
LEJ? T/N",NO$:IF UPPER$(LEFT$(NO$,1))="T" THEN 100
```

LIGA MYŚLĄCYCH

Zadanie 1

Z pełnego naczynia zawierającego kwas o stężeniu 96 proc. odlano 2,5 litra i dopełniono naczynie kwasem o stężeniu 80 proc., a następnie odlano 2,5 litra i znowu dopełniono naczynie kwasem o stężeniu 80 proc. W wyniku tych czynności otrzymano kwas 89-procentowy.

Należy znaleźć pojemność naczynia.

Zadanie 2

Cyfry liczby trzycyfrowej tworzą postęp geometryczny. Suma cyfr jedności i setek stanowi $\frac{5}{2}$ cyfry dziesiątek. Jeżeli od szukanej liczby odejmiemy liczbę ułożoną z tych samych cyfr, lecz napisanych w odwrotnym porządku, to otrzymamy 297. Należy znaleźć tę liczbę.

Zadanie 3

Dziadek i babcia mają w sumie 154 lata. Dziadek ma 2 razy tyle lat, ile babcia miała wtedy, kiedy dziadek miał tyle, ile babcia ma obecnie.

Proszę podać, ile lat ma dziadek, a ile babcia?

Zadanie 4

Na szalki wagi postawiono dwa wiadra napełnione po brzegi wodą. W jednym z wiader pływa kawałek drewna. Które z wiader jest cięższe? Odpowiedź należy uzasadnić.

Zadanie 5

Marek jadąc autobusem zauważył Pawła, który szedł wzdłuż ulicy po chodniku, ale w kierunku przeciwnym do ruchu autobusu. Autobus niebawem zatrzymał się na przystanku, a Marek po wyjściu z niego pobiegł w kierunku oddalającego się Pawła. Od momentu zauważenia Pawła przez Marka do chwili wyjścia Marka z autobusu minęło 10 sekund. Przyjmując, że prędkość biegu Marka jest dwa razy większa od prędkości chodu Pawła i pięć razy mniejsza od prędkości autobusu, obliczyć po upływie ilu sekund Marek dogoni Pawła?

Zadanie 6

Pewien robotnik kopie dół. Na zapytanie przechodnia, jak głęboki będzie dół, odpowiedział: „mój wzrost wynosi 180 cm, gdy wykopię dół do końca, moja głowa będzie o tyle poniżej powierzchni ziemi, o ile teraz, gdy już wykopałem połowę głębokości dołu, jest powyżej niej". Jaka będzie głębokość wykopanego dołu?

Rozwiązania zadań prosimy przesyłać do redakcji do końca września br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody niespodzianki.

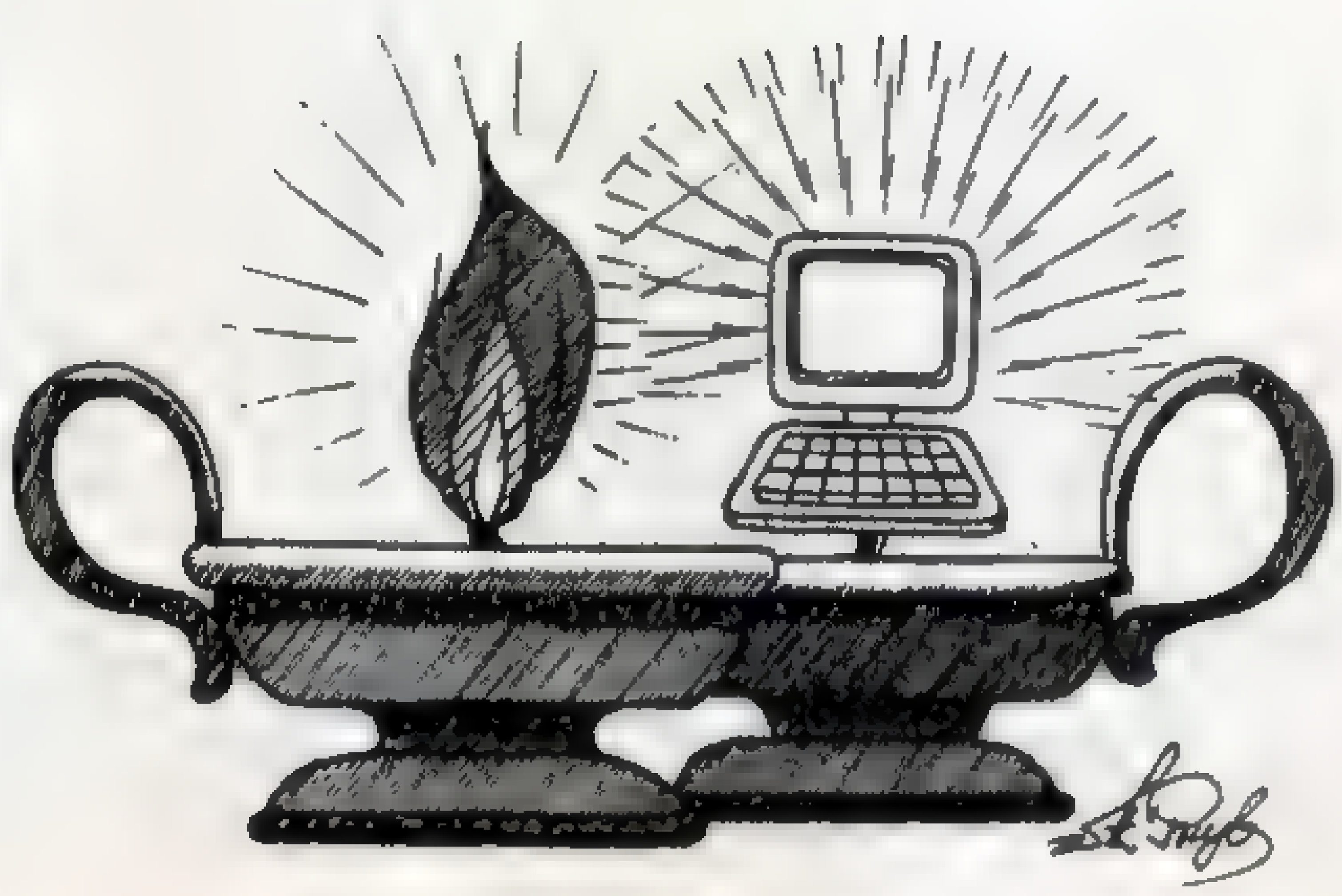
Uwaga! **Do programu „Sumy kontrolne" (IKS nr 4) wkradł się błąd. Poniżej zamieszczamy prawidłowy wydruk:**

```
310 GOSUB 600
345 NM$=STR$(HOLD):POSITION
  28-LEN(NM$),1:PRINT
  NM$:NM$="":SYN$=H1$:DOT=0
  :NUMALL =0:GOTO 130
```

Pocztowa giełda

Rozwiązanie krzyżówki z "IKS-a" nr 3/87
Hasło brzmi: „IKS" — PISMO DLA CIEBIE
Bony pieniężne (1000 zł) wylosowali: Maria Wawrzyniak — Konin, Dariusz Jankowski — Siedlce, Zdzisław Ziółkowski — Łódź, Piotr Nowak — Sosnowiec, Marek Zemfler — Gdańsk.
Nagrody książkowe otrzymują: Ryszard Galiński — Warszawa, Sławomir Kossakowski — Wałbrzych, Mieczysław Cetnar — Czchów, Marek Oleszczuk — Lublin, Zbigniew Kaliszyk — Parczew, Joanna Pasternak — Brzeg, Eugeniusz Wojciechowski — Poznań, Witold Chwiałkowski — Sandomierz, Jacek Piska — Płock, Tadeusz Hermanowski — Białogard.

- **Kupię nową klawiaturę do ZX-81. Tomasz Sadzikowski, ul. Grzybowa 1/30 32-503 Chrzanów 5.**
- **Wymienię programy na Commodore 64 oraz zakupię. Wioletta Kryza, ul. Żeromskiego 36/11, 97-425 Zelów.**
- **Kupię komputer Commodore 16 + magnetofon 1531. Adam Wojcieszyk, ul. Świerczewskiego 5 m 43, 86-200 Chełmno.**
- **Literatura na Atari, informacja — koperta zwrotna + znaczek. Adam Gawlikowski, ul. Z. Wolskiego 5/1 m 6, 09-400 Płock.**
- **Pilnie poszukuję literatury, programów do mikrokomputera ZX-81 (1K). Piotr Kempys, ul. Bociania 9, 43-316 Bielsko-Biała.**



Rys. Michał Przybyłowski

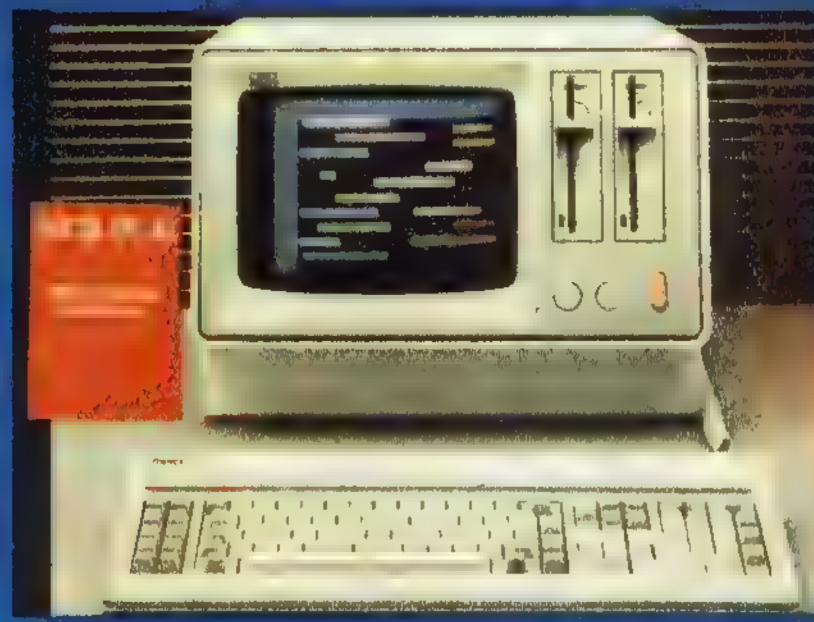
KRZYŻÓWKA NR 6



POZIOMO: 5) twarda, cienka tektura, stosowana w elektrotechnice jako materiał izolacyjny, 6) zaprawa do zupy z tłuszczu roztartego z mąką i zasmażonego, 7) niemożność odbierania wrażeń słuchowych wskutek wrodzonych lub nabytych zmian w narządzie słuchu, 8) oddzielanie się czerwonych krwinek od osocza, 13) język literacki starożytnych i średniowiecznych Indii, 14) Homo sapiens, 15) tech. okrągła tarcza, której środek nie leży na osi jej obrotu (ekscenter).

PIONOWO: 1) urządzenie do łączenia dwóch elementów w celu przeniesienia ruchu z jednego na drugi, 2) tech. układ ziarn, warstw lub włókien w wewnętrznej budowie materiału, 3) stan społeczny wywodzący się z rycerstwa, 4) dotykanie ciała w miejscach wrażliwych i pobudzanie tym do śmiechu, 9) hodowca, sprzedawca ptaków lub gat. pająka, 10) obelga, zniewaga, 11) amator „białego szaleństwa", 12) pomost na słupach do przeprowadzenia trasy komunikacyjnej ponad poziomem terenu.

Litery z kratek ponumerowanych dodatkowo w prawym dolnym rogu od 1 do 14 utworzą hasło, które wystarczy nadesłać jako rozwiązanie zadania, pod adresem redakcji do końca września 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.



NCR PC4I, NCR PC6 i NCR PC8 są szesnastobitowymi komputerami profesjonalnymi. Pierwszy z nich oparty jest na mikroprocesorze 8088 (4.77MHz). Może być wyposażony opcjonalnie w koprocesor arytmetyczny 8087. Pamięć operacyjna ma pojemność 256 Kb (z możliwością rozszerzenia do 640 Kb). Na kolorowym bądź czarno-białym monitorze o rozdzielczości 640 na 400 punktów standardowo mieści się 25 80-znakowych wierszy. W obudowie monitora znajdują się dwie dyskietki 5"1/4 (po 360 Kb).

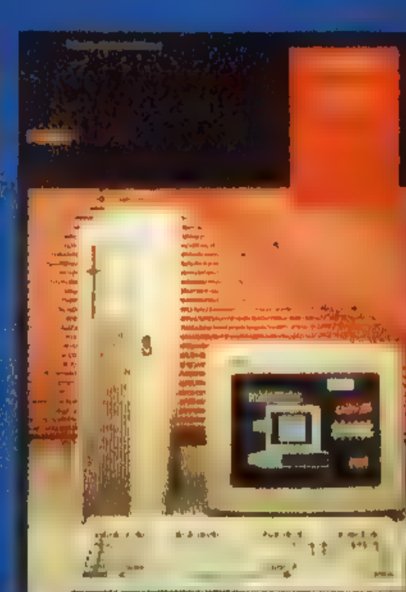
NCR PC6 (mikroprocesor 8088-2; opcjonalnie 8087) wyposażony jest w dwa poziomy grafiki. Standard dla monitora monochromatycznego 12" wynosi 720x350 punktów, a dla kolorowego 14" 640 na 200.

Podobne parametry ma NCR PC8. Systemem operacyjnym, pod którym pracują wszystkie te modele, jest DOS (NCR—DOS 2.11 lub NCR—DOS 3.1). NCR PC8 jest również wyposażany w XENIXa V. Oczywiście, istnieją możliwości rozbudowy konfiguracji podstawowej tych mikrokomputerów: zwiększenia ich pamięci zewnętrznych o floppy dyski i dyski twarde.

Francuskie mikrokomputery (oparte na sprawdzonych już mikroprocesorach) są całkowicie zgodne ze standardem IBM, a produkowane są głównie na francuski rynek.

c.





„IKS" — dodatek „Żołnierza Wolności". Redagują Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8896 Nr. ind. 361682 K-62

INFORMATYKA KOMPUTERY SYSTEMY

CENA — 100 zł



W NUMERZE:

ICL — str. 4
Innowacje • opinie • prognozy — str. 5
Czy komputer może uczyć — str. 6
Procesor obrazu — str. 7—14
O zerowaniu zmiennych — str. 14
Pogwarki o CPC 464 — str. 15
Od zabawy do nauki — str. 16
Starty i lądowania bez samolotów — str. 17
Sztuki i sztuczki (9) — str. 18
Komputer w medycynie — str. 20
Intel 8080 — str. 21
Tajny zapis — str. 23
Liga Myślących — str. 30
Giełda pomysłów — str. 31





Komputer w rękach zwycięzcy

Szyfr już złamany. Na ogłoszony w pierwszym numerze „IKS'a", informatycznego dodatku „Żołnierza Wolności", konkurs wpłynęło kilkanaście tysięcy prawidłowych rozwiązań. Zwycięzcę wybrał los. Pięcioletnia Monika wylosowała trzy koperty. Nadawcą pierwszej i zarazem triumfatorem naszego konkursu okazał się Bogdan Olszewski z Sosnowca. Dwie drugie nagrody otrzymują Bożena Silezin-Budek i Michał Stajszczak — oboje z Warszawy.

Z tej to okazji odbyła się w redakcji miła uroczystość wręczenia nagród. Redaktor naczelny „Żołnierza Wolności" płk Zdzisław Janoś wręczył zwycięzcy zestaw komputerowy ufundowany przez znaną na naszym rynku firmę ICL. Sprzęt i kilkanaście programów narzędziowych służyć będą inżynierowi z Sosnowca i jak zapowiedział pan Bogdan spróbuje na nim rozwiązać nawet niektóre zadania profesjonalne (oczywiście na miarę możliwości).

Natomiast dwa radioodbiorniki „Maria" trafiły do rąk warszawiaków, którym przysłowiowego łutu szczęścia zabrakło do wylosowania głównej nagrody.

A teraz jeszcze kilka słów o konkursie. Naszym zdaniem odczytanie kryptogramu nie było zadaniem prostym. Najważniejszą sprawą było oczywiście odnalezienie klucza, a potem krok po kroku należało odczytać niezrozumiały ciąg znaków, w którym znalazły się dodatkowe utrudnienia. Zatem najważniejszy był klucz. W kolejnych numerach staraliśmy się podać o nim bliższe informacje. Ale ten trud okazał się niepotrzebny, już wkrótce bowiem po ogłoszeniu konkursu zaczęły napływać rozwiązania — prawidłowe oczywiście!!!

Po odczytaniu nasz kryptogram miał postać:

WSZYSTKIE BOJOWE
OPERACJE MIAŁY
JEDEN AZYMUT
+ WOLNA POLSKA +



Redaktor naczelny „ŻW" płk Zdzisław Janoś przekazuje komputer zwycięzcy — Bogdanowi Olszewskiemu.



A to już wszyscy laureaci w komplecie i ostatnie gratulacje.

****
NOWOŚCI ************ M O C D L A B I U R ********** NOWOŚCI
****

OFFICEPOWER - uznany został za najlepszy na świecie system komputerowy dla biur, którego autorem jest brytyjska ICL - International Computers Limited. System został stworzony dla amerykańskiego Ministerstwa Transportu w wyniku kilkuletniej pracy grupy fachowców z ICL. System - po dwóch latach eksploatacji - okazał się bardzo dobry. Obecnie sprzedaje się go masowo dla dużych biur na całym świecie. OFFICEPOWER jest główną częścią ICL-owskiej oferty w zakresie systemów biurowych (Office Systems), których sprzedaż w 1991 roku szacuje się na 10 miliardów dolarów rocznie.

OFFICEPOWER jest kompletnym, zintegrowanym systemem służącym do automatyzacji prac biurowych. W przypadku pracy lokalnej obsługuje od 4 do 128 użytkowników (lub więcej w przypadku sieci). Software wykorzystuje system operacyjny UNIX V stający się międzynarodowym standardem, natomiast hardware stanowią 32-bitowe komputery ICL-owskie CLAN3 i CLAN7 (od jesieni tego roku również DRS300). System wykorzystuje również jako terminale ICL-owskie: DRS300, PC QUATTRO i OPD, IBM-owskie PC (i kompatibilne, w szczególności ITT-XTRA). System komunikuje się ze wszystkimi dużymi komputerami ICL i IBM. Główną jego zaletą jest bardzo wysoka jakość wszystkich jego aplikacji.

<u>Na OFFICEPOWER składają się:</u>

* dwupoziomowe przetwarzanie tekstów z wszystkimi znanymi obecnie możliwościami. Przetwarzanie dwupoziomowe oznacza, że pierwszy poziom przeznaczony jest dla osób, które usiadły po raz pierwszy do komputera, natomiast poziom drugi dla bardziej doświadczonych,
* poczta elektroniczna o nieograniczonej ilości "skrzynek pocztowych",
* notes i kalendarz na dowolną ilość lat,
* "przypominacz" pamiętający terminy zebrań i spotkań,
* listy nazwisk i adresów układanych wg. dowolnych kluczy,
* automatyczna adresacja pism i listów do wielu adresatów wyznaczonych z listy,
* arkusz elektroniczny zintegrowany z edytorem tekstów i grafiką wykresową,
* baza danych, raportowanie i kalkulator wielozadaniowy,



* informator telefoniczny zawiadamiający, że podczas naszej nieobecności ktoś, kto odebrał telefon ma dla nas wiadomość (lub wręcz tą wiadomość),
* możliwość zakładania folderów zawierających zbiory uszeregowanych plików,
* możliwość przesyłania informacji drogą telekomunikacyjną oraz zabezpieczania ich według własnych kluczy,
* możliwość definiowania własnych aplikacji dla tych, których pociąga eksperymentowanie z systemem.

OFFICEPOWER okazał się systemem na tyle "przyjaznym" w użyciu że, użytkujący go szybko pozbyli się barier psychologicznych związanych z zastosowaniem komputerów w codziennej pracy biurowej.

****

(sch)

ICL

ICL — tej firmy przedstawiać nie trzeba. Od lat znana jest na polskim rynku. Polscy informatycy jeszcze dziś często pracują na elwrowskiej Odrze — młodszej siostrze ICL 1900. Ale to już historia.

Firma rozwija się dynamicznie, opanowuje nowe technologie i zdobywa nowe rynki zbytu. Warszawskie przedstawicielstwo oferuje DRS 20 i 30 — systemy przeznaczone do prac biurowych. ICL może też kompletnie wyposażyć placówki handlowe począwszy od kas, a na komputerach skończywszy. To oczywiście nie wszystko. Szlagierem jest DRS 300, którego już przedstawialiśmy. Istotną zaletą tego urządzenia jest jego bardzo dobre oprogramowanie. Użytkownik pracuje w pełnym komforcie, komputer podpowiada komendy — zresztą wybór odpowiednich funkcji odbywa się za pomocą myszki, czyli jest bardzo prosty. Jedyna wada to wysoka cena — nawet bogate zakłady nie będą sobie mogły na niego pozwolić.

ICL jest dobrym narzędziem nie tylko dla sekretarek i statystyków, ale także i dla inżynierów i zawodowców zajmujących się projektowaniem systemów informatycznych.



DRS 300 należy do komputerów „łatwych i przyjemnych".

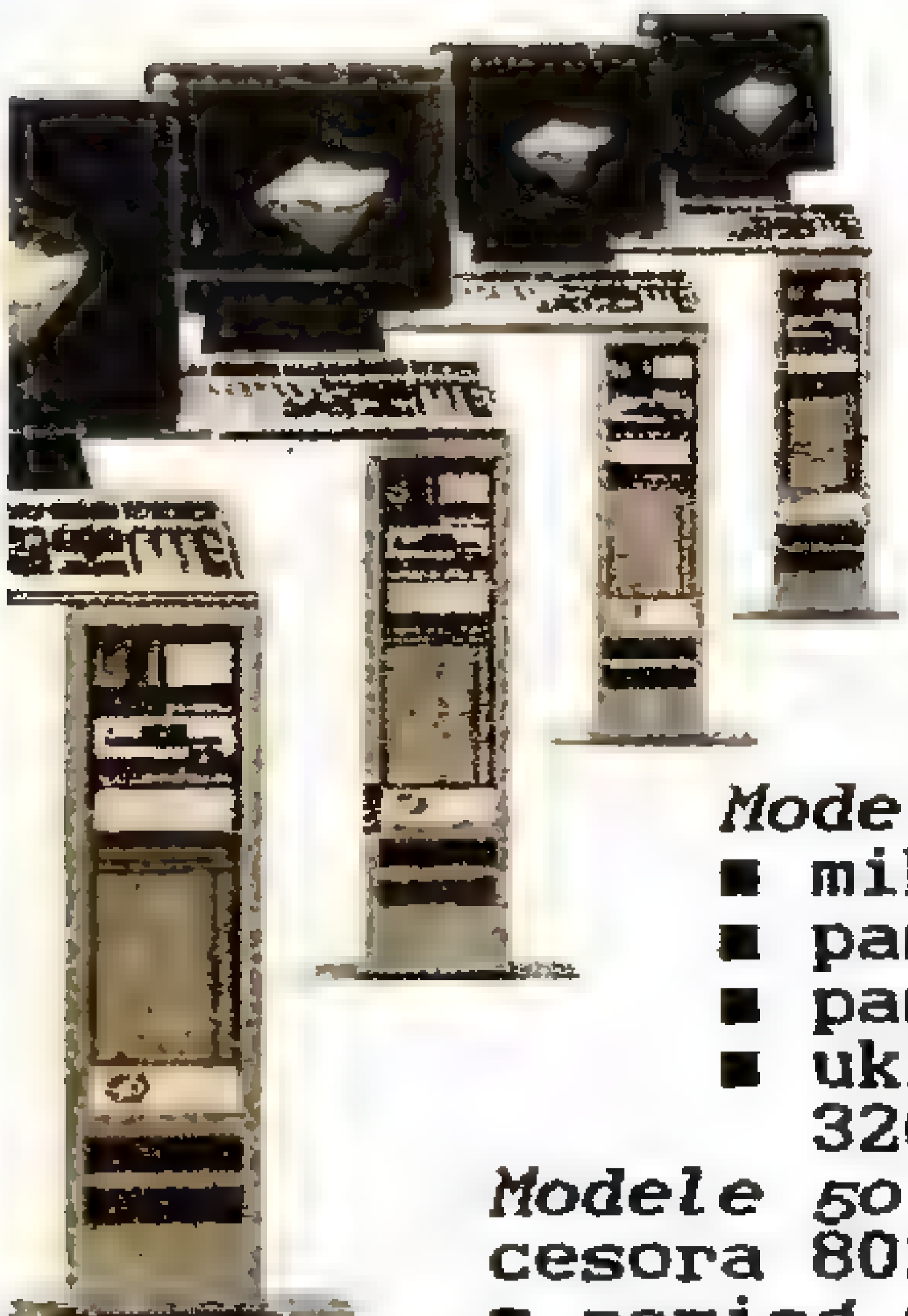




W siedzibie firmy DRS 20 wykorzystywane są w praktyce.



Elektroniczne kasy są jednym z elementów proponowanych przez ICL systemów komputerowych.



I B M Personal System / 2

IBM PS/2 to prawdopodobnie kolejny milowy krok w historii mikrokomputerów ,podobnie jak sześć lat temu IBM PC . Firma *IBM* proponuje cztery modele: 30,50,60,80 .*IBM PS/2* posiadają stacje dysków elastycznych 3 1/2 cala o pojemności 720 KB lub 1,44 MB oraz dyski stałe 3 1/2 cala 20,44,70 lub 115 MB .Mogą mieć dołączoną zew - nętrzną stację dysków 5 1/4 cala oraz stację z zapisem laserowym WORM(write-once read-many) o pojemności 200 MB .

Model 30 jest ewolucją IBM PC/XT ,posiada :

- mikroprocesor 8086 o częstotliwości zegara 8 MHz;
- pamięć zapisywalną 640 KB ;
- pamięć stałą 64 KB z CBIOS (kompatybilne z BIOS);
- układ MCGA zapewniający grafikę o rozdzielczości 320 x 200 pixeli w 256 kolorach .

Modele 50,60 lepiej wykorzystujące możliwości mikropro - cesora 80286 wyposażono w :

- pamięć zapisywalną 1 MB o czasie dostępu 150 ns ,roz - szerzalną do 7 lub 15 MB ;
- pamięć stałą 128 KB z CBIOS i ABIOS ;
- układ VGA zapewniający grafikę o rozdzielczości 640 x 480 pixeli w 256 kolorach ,a po kolejnej rozbudowie 1024 x 768 pixeli.

Model 80 różni się od modeli 50 i 60 tym ,że posiada mikroprocesor 80386 (16 lub 20 MHz) oraz pamięć zapisywalną o czasie dostępu 80 ns i pojemności do 16 MB . Modele *60,80* posiadają jednostkę cent - ralną w obudowie typu *wieża* ,umieszczanej pionowo na podłodze.Ceny modeli w standardowej konfiguracji:£ 1106 ,£ 2658 ,£ 4464 ,£ 7056.

C O M P A Q P O R T A B L E I I I

Jest mikromokputerem przenośnym ,wyposażonym w mikroprocesor 80286 (12 lub 8 MHz) i koprocesor 80287 (8MHz).Pamięć standardowo 640 KB może być rozszerzona do 6,6 MB .Pamięć zewnętrzną stanowią : dwie stacje dysków 5 1/4 cala o pojemności 360KB lub 1,2MB i dysk stały 20 lub 40 MB .Mikrokomputer posiada kartę graficzną pozwalającą na ekranie ciekłokrystalicznym LCD na rozdzielczość w trybie graficz- nym 640 x 400 pixeli (zgodna z IBM CGA) lub kartę EGA . COMPAQ PORTABLE III posiada trzy porty we-wy :równoległy CENTRONICS ,sze- regowy RS 232 C oraz typu DIN do podłączenia kolorowego monitora zewnętrznego .Klawiatura posiada 84 klawisze .System operacyjny to MS DOS v.3.20 .

□□ ■ □□ M I K R O D E P E S Z E □□ ■ □□

■□ *W bieżącym roku należy spodziewać się prototypu mikroprocesora INTEL 80486 , w związku z tym firmy IBM i Microsoft pracują nad nową wersją systemu operacyjnego DOS.* ■□ *Najnowsza propozycja firmy BORLAND to kompilator języka C o nazwie Turbo C.* ■□ *DMP 4000 to ostatni model drukarki firmy AMSTRAD kompatybilnej z drukarkami IBM i EPSON .* ■□ *Kolejnym modelem plotera firmy ROLAND jest DPX3300 pozwalający wykonywać rysunki formatu A1 na papierze i przeźroczy- stej folii w ośmiu kolorach .* □□ ■ □□ ■ □□ ■ □□

Nr 5 1987/06/05 *opracował : Jacek Wojtala*

Czy komputer może uczyć?

(Rozmowa druga)

Włodzimierz GOGOŁEK

Kontynuując relację rozmowy z Robertem — jednym z absolwentów „skomputeryzowanej uczelni", uwagę moją zwróciła kwestia doboru informacji wyświetlanych uczącemu się przez komputer. Wykorzystywane są w tym celu elementy sztucznej inteligencji — komputer „rozmawia" z użytkownikiem, korzysta jednak z pewnych niezmiennych całości — umownych porcji. Porcje te zbudowane są zarówno z fragmentów treści nauczania, jak i innych komunikatów (np. aktywizujących pracę studenta — naukę).

Podczas konwersacji (komputer — uczący się) mającej na celu nauczenie umiejętnego wykorzystywania zwrotów w języku niemieckim, kontrola aktywności emocjonalnej (np. czasy reakcji) studenta wskazują, iż zbliża się on do stanu, który popularnie nazywamy nudą. Dowodzi tego znaczne wydłużenie się czasów odpowiedzi na zadawane przez maszynę pytania, a także treść i forma tych odpowiedzi. W tej sytuacji komputer sprawdza (w dostępnym mu zbiorze) dokładnie, z kim ma do czynienia — wiek, płeć, zainteresowania itp. Wartości tych danych to „drogowskazy" w wędrówce po podzbiorach „komunikatów aktywizujących". Mężczyzna 21 lat, kibic sportowy — to dane z kartoteki — nudzi się — pozwala to na identyfikację podzbioru, który zawiera komunikaty aktywizujące, w tym wypadku o treści łatwej do wyobrażenia — np. wyniki ostatnich rozgrywek ligi wraz z komputerowymi przewidywaniami wyników następnej rundy. Podczas uzgadniania poglądów, między komputerem a uczącym się (na temat meczów), maszyna kontroluje zmianę aktywności emocjonalnej i po przekroczeniu pewnego progu (student już się nie nudzi) temat zostaje zmieniony z „ligi" na naukę języka.

Oczywiście w praktyce „drogowskazami" drogi do tematu „pobudzającego rozmowę" są dziesiątki innych wartości między innymi: iloraz inteligencji, stan cywilny, wyniki w nauce, sprawność pamięci, oczu.

Podobnie liczba zbiorów tematów „pobudzających rozmów" jest znaczna, przewiduje bowiem wędkarzy, koneserów kuchni, kina, wielbicieli gwiazd (także śpiewających) itp.

Są to przykłady prostej modyfikacji sprawności uczenia się, stosowane jako ostateczne. Wcześniej istnieje bogaty zestaw innych środków „pobudzania", jakie daje nie treść, lecz forma wyświetlanych na ekranie monitora komputera obrazów (tekstów, rycin, cyfrowych odwzorowań np. fotografii). Okazuje się bowiem, że niewielkie modyfikacje postaci obrazu dostarczają cały arsenał drobnych (nieustannie stosowanych, jak przysłowiowe krople wody spadające na kamień) bodźców, które mają na celu aktywizowanie pracy studentów. Wykorzystano tu udział kolorów, słów i zwrotów na nasze samopoczucie. W praktyce polega to na ciągłym „poznawaniu przez komputer „swoich możliwości".

Na początku eksploatacji programu edukacyjnego (sterującego pracą komputera wspomagającego proces nauczania), wprowadzone zostały do niego między innymi ogólne prawidłowości oddziaływania kolorów na człowieka, na przykład: czerwień pobudza, zieleń uspokaja, niebieski z czernią drażni. Stosując te prawidłowości w praktyce (zmiana barw obrazu w pamięci komputera jest bardzo prostym zabiegiem) maszyna rejestruje reakcje uczących się — różnicę stanu aktywności przed i po zmianie kolorów — w odpowiednich „szufladkach". Przykładowe oznaczenie (identyfikator) takiej szufladki (podzbioru): 19 lat, niewiasta, nie lubi sportu, lubi książki — pamiątki. Mieszczą się w niej kolory: zielony z białym i czarny, jako najskuteczniejszy podczas zwiększania aktywności emocjonalnej studentek mających 19 lat, przedkładających książki nad sport — wniosek ten uzasadniają wyniki 634 prób stosowania tych i innych kolorów w podobnych okolicznościach. Innymi słowy komputer (centralny) kontrolując codziennie pracę 750 uczących się, prezentuje każdemu z nich co najmniej 20 kolorowych obrazów — zdobywa zatem doświadczenie ze stosowania 15 000 obrazów — „widzi", jakie kolory i jak pobudzają użytkowników. Zważywszy na różnice uczących się (płci, wieku itp.) kilka tygodni eksploatacji programu edukacyjnego wystarcza, by „maszyna wiedziała", komu, kiedy i jak ubarwić obraz, by poza treścią niósł określony bodziec (modyfikacji stanu aktywności emocjonalnej).

Aby usprawnić nauczanie, w którym komputer wyręcza pracę nauczyciela, podobnie jak w wypadku kolorów, następuje klasyfikacja niektórych słów, zwrotów. Tworzą one komunikaty, „wtrącane" przez maszynę podczas nauki, modyfikujące aktywność emocjonalną studenta. Oto niektóre przykłady: „Osiągasz wyraźne postępy", „Uczysz się bardzo efektywnie", „Pośpiesz się!", „Szybciej!", „Skoncentruj się!" itp. Dziesiątki tego typu komunikatów komputer wykorzystuje, "poznając" ich skuteczność wobec określonych osób i sytuacji (szufladkuje komunikaty min. wg wieku, płci, zamiłowań).

Przedstawione uwagi dowodzą, że podczas nauki wspomaganej przez komputer — na równi z treścią — forma i sposób przekazywania informacji przez komputer decyduje o efektach współpracy uczącego się z maszyną. Zadbano o to z wielką skrupulatnością — dając uczącym się swego rodzaju rekompensatę za brak „żywego słowa" wypowiadane-



go przez nauczyciela, który poza przekazywaniem wiedzy także się uśmiecha i gestykuluje... Doceniając ten problem część zajęć w uczelni Roberta prowadzą nauczyciele, którzy „uzbrojeni" w komputery dysponują między innymi ekranami monitorów wielkości tablicy szkolnej. Stworzyło to warunki do niemal nieograniczonych możliwości ilustrowania wykładów. Tablica-ekran poza tradycyjnym przeznaczeniem (kredę zastępuje „kolorowe pióro") jest pośrednikiem prezentacji tysięcy obrazów zapisanych w pamięci komputera. Każdy z nich nauczyciel może odszukać i wyświetlić (oczywiście za pomocą maszyny) w czasie krótszym niż trwa wydanie polecenia wykonania tej czynności. Temat ten wiąże się z budową i tak zwanym przetwarzaniem obrazów, które po przetwarzaniu danych i tekstów stało się znaczącym obszarem zastosowań komputerów w ogóle, a w uczelniach w szczególności. Zważywszy na atrakcyjność tego zagadnienia — poprawnej konstrukcji i modyfikacji obrazów w pamięci komputera — stało się ono przedmiotem naszej kolejnej rozmowy z absolwentem skomputeryzowanej szkoły.

W szponach Atari (6)

Procesor obrazu

Tomasz MROWIEC, Ludwik PIELA

Rastrowy system wyświetlania

Dla zrozumienia możliwości graficznych komputera Atari, musimy przypomnieć sobie, jak wyświetlany jest obraz na ekranie telewizora. Z tyłu kineskopu generowany jest strumień elektronów i wystrzeliwany w kierunku ekranu. Wzdłuż swojej drogi przelatuje on między zespołem cewek odchylania poziomego i pionowego, które mogą odchylić strumień. Dzięki temu może on być skierowany do dowolnego punktu ekranu. Elektronika zawarta wewnątrz odbiornika powoduje, że strumień elektronów przemieszcza się po ekranie w sposób regularny. Startuje on w lewym górnym rogu i przesuwa się poziomo. Zmiany intensywności strumienia, występujące w miarę jego ruchu, malują obraz na ekranie. Po osiągnięciu prawego brzegu jest wygaszany i przechodzi do lewego brzegu następnej linii (nie uwzględniamy w opisie wybierania międzyliniowego). Następnie jest włączany i znowu przesuwa się poziomo. Po wypełnieniu całego ekranu i osiągnięciu jego dolnego brzegu strumień elektronów jest wygaszany i wraca do lewego górnego rogu. Następnie cały cykl zaczyna się od nowa. Powtarzany jest 50 razy na sekundę. Taki sposób tworzenia obrazu nazywamy **rastrowym systemem wyświetlania**.

Zdefiniujmy teraz pewne pojęcia. Pojedyncze przejście strumienia przez ekran nazywamy *„poziomą linią ekranu"*. Pozioma linia ekranu jest podstawową jednostką miary odległości pionowej na ekranie. Wysokość obrazu określamy przez podanie liczby poziomych linii ekranu, które go tworzą. Okres, w którym strumień elektronów wraca od prawego do lewego brzegu, nazywamy *„powrotem poziomym"*. Okres, w którym strumień wraca na górę ekranu, nazywamy *„powrotem pionowym"*. Cały proces kreślenia obrazu na ekranie zajmuje 20 tysięcy mikrosekund. Powrót pionowy trwa około 1500 μs, a powrót poziomy — około 14 μs.

Większość odbiorników telewizyjnych projektowana jest z nadmiarem linii; oznacza to, że kreślą one obraz w taki sposób, że brzegi obrazu wykraczają poza ekran. W efekcie nie mamy ramki na ekranie. Jest to jednak niewygodne dla komputerów, ponieważ informacja, która jest poza brzegami ekranu, jest tracona. Z tego powodu obraz tworzony przez komputer musi być trochę mniejszy, niż telewizor może teoretycznie wyświetlić. Atari wykorzystuje tylko 192 poziome linie ekranu. Dlatego granicą rozdzielczości pionowej odbiornika telewizyjnego, używanego z tym komputerem, są 192 punkty elementarne.

Standardową jednostką odległości poziomej jest *„takt koloru"*. Szerokość obrazu określamy przez podanie liczby taktów koloru, które zajmuje. W pojedynczej poziomej linii ekranu istnieje 228 taktów koloru, z których maksymalnie 176 jest widocznych. Dlatego granicą rozdzielczości poziomej standardowego odbiornika telewizyjnego jest 176 punktów elementarnych. Możliwe jest także sterowanie półtaktami koloru, co daje rozdzielczość poziomą 352 punktów elementarnych.

Procesor obrazu — ANTIC

Z tego krótkiego przypomnienia widać, jak skomplikowane i czasochłonne jest tworzenie obrazu na ekranie telewizora lub monitora. Gdyby tym wszystkim zajmował się mikroprocesor, to miałby niewiele czasu na realizację innych funkcji. Dlatego zaprojektowany został specjalizowany układ scalony — procesor obrazu o nazwie **ANTIC** (Alpha-Numeric Television Interface Controller), który zajmuje się tworzeniem obrazu na ekranie i odciąża mikroprocesor. ANTIC jest prawdziwym mikroprocesorem: ma własny zestaw rozkazów. Wykorzystując je możemy napisać program, zwany programem wyświetlania (ang. display list), który steruje wyjściem graficznym komputera. Dostarcza on następujących informacji: gdzie mogą być znalezione dane do wyświetlania, jakich trybów wyświetlania należy używać do interpretacji trybów ekranu i jakie specjalne opcje wyświetlania powinny być używane.

Normalnie program wyświtlania tworzony jest przez system operacyjny podczas realizacji instrukcji **GRAPHICS**. ANTIC wykonje kolejne rozkazy programu wyświetlania. W zależności od kodu rozkazu zawartość pamięci ekranu interpretowana jest jako dane tekstowe lub graficzne. ANTIC wysyła informacje sterujące obrazem do innego procesora (układ GTIA — Graphics Television Interface Adaptor). Jak każdy procesor, ANTIC wykonuje rozkazy programu wyświetlania w następujących etapach:

1. Pobranie kodu rozkazu programu wyświetlania i zapamiętanie go w rejestrze rozkazów
2. Interpretacja zawartości pamięci jako danych graficznych lub znakowych, na podstawie kodu rozkazu.

➧

3. Jeśli rozkaz wskazuje dane znakowe, ANTIC pobiera bajty z pamięci ekranu, przeszukuje zestaw wzorców znaków i przesyła wzory znaków do wyświetlenia.
4. Jeśli rozkaz wskazuje dane graficzne, ANTIC przesyła je bezpośrednio do wyświetlania.
5. Zwiększenie licznika programu wyświetlania, który wskazuje następny rozkaz do wykonania
6. Zwiększenie licznika linii ekranu o liczbę bajtów przesłanych z pamięci ekranu do wyświetlania.
7. Powtórzenie tych kroków od początku.

W tym miejscu winni jesteśmy pewne wyjaśnienie. Rozkazy programu wyświetlania określają sposób wyświetlania jednej linii. Lecz nie linii ekranu, ale tak zwanej linii obrazu, która w zależności od używanego trybu, składa się z kilku linii ekranu. Na przykład, w trybie graficznym 2 linia obrazu składa się z 16 linii ekranu, a w trybie 7 — z dwóch linii ekranu. Jak widać, każda linia obrazu może być różnie wyświetlana, w zależności od użytego rozkazu procesora ANTIC. Jednak w trybach graficznych, definiowanych w języku BASIC, wszystkie rozkazy programu wyświetlania określają zwykle ten sam tryb wyświetlania. W efekcie na ekranie otrzymujemy jednolite linie obrazu. Jednak nic nie stoi na przeszkodzie, aby utworzyć dowolny ciąg linii obrazu na ekranie, możemy je zdefiniować za pomocą odpowiednich rozkazów programu wyświetlania.

Program wyświetlania powinien być tak skonstruowany, aby zapewnić:

- zrównoważenie linii obrazu nie mieszczących się na ekranie, do tego przeznaczone są rozkazy wyświetlania pustych linii,
- załadowanie licznika pamięci obrazu adresem początkowym obszaru pamięci, zawierającej aktualne dane graficzne lub tekstowe do wyświetlania,
- posiadanie aktualnych instrukcji wyświetlania określających, który tryb lub tryby graficzne są używane,
- przejście na początek programu wyświetlania (rozkazy skoku) po wykonaniu całego; w pewnych wypadkach rozkazy skoku niezbędne są do kontynuowania programu wyświetlania lub pamięci ekranu poza adresy graniczne.

Zestaw rozkazów procesora ANTIC

Istnieją cztery klasy rozkazów, umożliwiające realizację wymienionych zadań: wyświetlanie grafiki, wyświetlanie znaków, wyświetlanie pustej linii i skoki. Dodatkowo, zgodnie z tymi rozkazami możliwe są następujące opcje: ładowanie licznika pamięci obrazu (LMS), przesuwanie obrazu i przerwania programu wyświetlania (DLI).

Rozkazy wyświetlania grafiki powodują wyświetlanie przez ANTIC linii obrazu zawierającej punkty elementarne w jednolitym kolorze. Wyświetlany kolor pochodzi z rejestru koloru. Wybór tego rejestru określony jest przez wartość danych ekranu. W trybach kreślenia czterokolorowych (tryby BASIC'a 3, 5, 7 i 15 oraz tryby ANTIC'a 8, A, D i E) do wybrania rejestru koloru potrzebna jest para bitów.

Ze względu na to, że do określania jednego punktu elementarnego potrzebne są dwa bity, z każdego bajtu danych ekranu dekodowane są cztery punkty. Na przykład, bajt danych ekranu zawierający wartość $1B (00011011) może wyświetlić 4 punkty; pierwszy będzie w kolorze tła, drugi w kolorze rejestru 0, trzeci w kolorze rejestru 1 i czwarty w kolorze rejestru 2.

W dwukolorowych trybach kreślenia (tryby BASIC'a 4, 6, 14 i 8 i tryby ANTIC'a 9, B, C i F) każdy bit określa jeden z dwóch rejestrów koloru. Wartość 0 wybiera kolor tła dla punktu, a wartość 1 wybiera rejestr koloru 0. W jednym bajcie ekranu może być zawarta informacja o ośmiu punktach.

Istnieje osiem różnych trybów wyświetlania grafiki. Różnią się one liczbą kolorów (2 lub 4), wysokością linii obrazu (1, 2, 4 lub 8 linii ekranu) i liczbą punktów elementarnych,

```
JY 3 REM ************************
PG 4 REM * Program #1: umozliwia*
JV 5 REM * utworzenie dowolnego *
TP 6 REM * programu wyswietlania*
KC 7 REM ************************
WB 8 ? CHR$(125);"   PODAJ ILOSC SEGMENTOW EKRANU":INPUT N:N=N+1
WP 10 DIM GEKR(N),NRGR(2*N)
PQ 20 GRAPHICS 8
XB 40 FOR I=1 TO N-1:? CHR$(125):? " SEGMENT nr: ";I:? "      Podaj tryb ANTIC'u: ";:INPUT A:NRGR(1+(I-1)*2)=A
PM 42 ? "      Podaj ilosc linii: ";:INPUT A:NRGR(2*I)=A:NEXT I
DP 43 NRGR(2*N-1)=-1:NRGR(2*N)=0
TJ 140 LST=1536
TG 150 GOSUB 30000
GC 160 ? "PROGRAM WYSWIETLANIA GOTOWY"
TO 420 STOP
ZT 20000 REM *******************
EL 20001 REM * Podprogram nr 1: *
AI 20002 REM *  uklada program  *
JD 20003 REM *   wyswietlania   *
AN 20004 REM *******************
RO 30000 SEG=1
IA 30010 LOK=LST
OD 30020 GEKR(0)=PEEK(88)+PEEK(89)*256
UX 30030 GRAN=INT((LST/1024+1)*1024)
UX 30040 GRAN2=INT(GEKR(0)/4096+1)*4096
DQ 30050 FOR X=LOK TO LOK+2
NB 30060 POKE X,112:LOK=LOK+1
RC 30070 NEXT X:I=1
PM 30080 GEKR(SEG)=GEKR(SEG-1)
UP 30100 TRYB=NRGR(1+2*(I-1)):POWT=NRGR(2*I):I=I+1
YU 30110 IF TRYB<0 THEN OP=65:ADDR=LST:GOSUB 30330:RETURN
CM 30120 PRZYR=40
UM 30130 IF TRYB>=6 AND TRYB<=12 THEN PRZYR=20
FM 30140 IF TRYB=8 OR TRYB=9 THEN PRZYR=10
UY 30150 FOR X=1 TO POWT
VO 30160 IF LOK<>GRAN-3 THEN 30200
QN 30170 OP=1:ADDR=GRAN:GOSUB 30330
TT 30180 LOK=LOK+1:GRAN=GRAN+1023
YV 30190 GOTO 30200
IV 30200 POKE LOK,TRYB
SR 30220 IF (SEG<>1 OR X<>1) AND (GRAN2-GEKR(SEG)>=PRZYR) THEN 30280
```

```
SL 30240 OP=TRYB+64
AO 30250 ADDR=GEKR(SEG)
FT 30260 IF GRAN2-GEKR(SEG)<PRZYR
   THEN GRAN2=GRAN2+4096:ADDR=AD
   DR+PRZYR
PP 30270 GOSUB 30330
CK 30280 LOK=LOK+1
TR 30290 GEKR(SEG)=GEKR(SEG)+PRZY
   R
MS 30300 NEXT X
HH 30305 IF TRYB=15 THEN GEKR(SEG
   )=GEKR(SEG)+PRZYR
VF 30310 SEG=SEG+1
BN 30320 GOTO 30080
ZE 30330 POKE LOK,OP
BX 30340 LOK=LOK+1
UW 30350 POKE LOK,ADDR-(INT(ADDR/
   256)*256)
CF 30360 LOK=LOK+1
PW 30370 POKE LOK,INT(ADDR/256)
EO 30380 RETURN
AV 30900 REM ********************
GN 30910 REM * Podprogram nr 2: *
KZ 30920 REM *   ustala adres   *
TD 30930 REM *  pamieci ekranu  *
BL 30940 REM ********************
PT 31000 POKE 88,GEKR(X)-(INT(GEK
   R(X)/256)*256)
RB 31010 POKE 89,INT(GEKR(X)/256)
   :RETURN
```

```
JX 2 REM ************************
UE 3 REM * Program nr 2  nalezy *
ED 4 REM *dograc do poprzedniego*
FV 5 REM *przy pomocy komendy   *
SJ 6 REM *      ENTER "C:"      *
KC 7 REM ************************
HF 8 N=5
WB 40 FOR I=1 TO N-1:READ A,B:NRG
   R(1+(I-1)*2)=A:NRGR(2*I)=B:NEX
   T I
RD 42 DATA 2,2,7,1,15,144,6,2
QY 160 REM
LE 200 POKE 559,0
HN 210 POKE 560,0
LH 220 POKE 561,6
RA 230 POKE 559,34
AZ 240 X=0:GOSUB 31000
ZM 250 POKE 87,0:POKE 752,1
EN 260 POSITION 13,0:? #6;"Wykres
    funkcji"
FR 265 POSITION 13,1:? #6;"------
   --------"
BM 270 X=1:GOSUB 31000
KY 280 POKE 87,2
CZ 290 POSITION 6,0:? #6;"y=sin(x
   )"
BG 300 X=2:GOSUB 31000
WL 310 POKE 87,8:COLOR 1
EU 320 PLOT 0,0:DRAWTO 0,140:PLOT
    0,70:DRAWTO 319,70
```

które mieszczą się w jednej poziomej linii obrazu (40, 80, 160 lub 320). Dlatego pewne tryby dają lepszą rozdzielczość; będą one oczywiście wymagać więcej pamięci RAM ekranu. W tablicy 1 prezentujemy zestawienie charakterystyk poszczególnych trybów wyświetlania grafiki.

Rozkazy wyświetlania znaków zmuszają ANTIC do wyświetlania linii obrazu ze znakami. Każdy bajt pamięci ekranu określa jeden znak. Istnieje sześć trybów wyświetlania znaków. W tablicy 2 prezentujemy zestawienie ich charakterystyk.

Rozkazy kreślenia pustych linii powodują wyświetlenie pewnej liczby poziomych linii ekranu w kolorze tła. ANTIC dysponuje ośmioma takimi rozkazami różniącymi się ilością pustych linii ekranu. Najczęściej używany jest rozkaz wysłania ośmiu pustych linii (kod 112 lub 70 szesnastkowo) na ekran. Tablica 3 zawiera zestawienie charakterystyk poszczególnych rozkazów.

Tablica nr 1

Kod rozkazu		Tryb BASIC a	Ilość punktów w poziomie	Ilość bajtów na linię	Ilość linii ekranu	Ilość bitów na punkt
Dziesiętny	Szesnastkowy					
8	08	3	40	10	8	2
9	09	4	80	10	4	1
10	0A	5	80	20	4	2
11	0B	6	160	20	2	1
12	CC	14	160	20	1	1
13	0D	7	160	40	2	2
14	0E	15	160	40	1	2
15	0F	8	320	40	1	1

Tablica nr 2

Kod rozkazu		Tryb BASIC a	Ilość punktów w poziomie	Ilość bajtów na linię	Ilość linii ekranu	Ilość bitów na punkt
Dziesiętny	Szesnastkowy					
2	02	0	40	40	8	8
3	03	—	40	40	10	8
4	04	12	40	40	8	8
5	05	13	40	40	16	8
6	06	1	20	20	8	8
7	07	2	20	20	16	8

Tablica nr 3

Kod rozkazu		Liczba linii ekranu
Dziesiętny	Szesnastkowy	
0	00	1
16	10	2
32	20	3
48	30	4
64	40	5
80	50	6
96	60	7
112	70	8

```
MZ 330 DEG :PLOT 0,70
GE 340 FOR I=1 TO 319
DM 350 DRAWTO I,70-65*SIN(2*I)
GF 360 NEXT I
CB 370 X=3:GOSUB 31000
KM 380 POKE 87,1
RA 390 POSITION 2,0:? #6;"wykonal
   :    TM&LP"
NF 395 POSITION 2,1:? #6;"WARSZAW
   A   1987"
MN 400 GOTO 400


JW 1 REM ************************
JJ 2 REM *                      *
KW 3 REM *     PROGRAM NR 3     *
JL 4 REM *                      *
KA 5 REM ************************
HF 8 N=5
AI 10 DIM GEKR(N),NRGR(2*N),A$(1)
CU 20 GRAPHICS 8:START=0
WB 40 FOR I=1 TO N-1:READ A,B:NRG
   R(1+(I-1)*2)=A:NRGR(2*I)=B:NEX
   T I
TM 42 DATA 2,2,7,1,15,192,6,2
DP 43 NRGR(2*N-1)=-1:NRGR(2*N)=0
SG 90 GRAPHICS 0:? CHR$(125):CLOS
   E #1
ZF 100 POSITION 6,1:? "WYKRESY FU
   NKCJI BIEGUNOWYCH"
DY 110 POSITION 3,3:? "MENU FUNKC
   JI:":?
RM 115 ? "     1> R=B*Q
      SPIRALA"
DC 120 ? "     2> R=A*(1+COS(Q))
      CARDIOIDA"
XR 125 ? "     3> R=A*(1-SIN(Q))"
AP 130 ? "     4> R=A*SIN(B*Q)
      ROZETA"
KL 135 ? "     5> R=A*COS(B*Q)"
OU 140 ? "     6> R=COS(A*SIN(B*Q)
   )"
OR 145 ? "     7> R=SIN(A*COS(B*Q)
   )"
EH 150 ? "     8> R=A
      WIELOKAT"
AS 155 ? :? "WPROWADZ:":TRAP 90
CV 160 ? "        NUMER FUNKCJI: ("
   ;N;")";:POSITION 27,15:INPUT N
   :? "        PARAMETR A    : (";A;
   ")";:POSITION 27,16:INPU
EM 162 ? "        PARAMETR B    : ("
   ;B;")";:POSITION 27,17:INPUT B
   :? "        PRZYROST      : (";KR
   OK;")";:POSITION 27,18:I
AA 163 ? "        SKALA         : ("
   ;SK;")";:POSITION 27,19:INPUT
   SK
ML 165 N=N+1*(N=0)
DI 170 IF N=1 THEN U=4
AN 175 A=A+1*(A=0)
AY 180 B=B+1*(B=0)
```

ANTIC używa dwóch typów rozkazów skoku. Pierwszy (JMP) jest prostym skokiem bezwarunkowym, który umieszcza w liczniku programu nowy adres, występujący w dwóch bajtach bezpośrednio po kodzie rozkazu i kontynuuje program wyświetlania od nowego adresu. Jego jedyną funkcją jest dostarczenie rozwiązania następującego problemu: rejestr licznika programu ANTIC ma tylko 10 bitów, dlatego program wyświetlania nie może przekroczyć 1KB. Jeśli musi przekroczyć tę granicę, to musi używać rozkazu JMP dla jej przekroczenia. Oznacza to, że program wyświetlania nie jest w pełni przemieszczalny.

Drugi rozkaz skoku (JVB) jest częściej używany. Wprowadza on wartość argumentu do licznika programu ANTIC i czeka na sygnał synchronizacji pionowej, po zakończeniu powrotu pionowego, który wykonuje odbiornik telewizyjny po wykreśleniu ostatniej linii ekranu. Rozkaz ten jest normalnie używany na końcu programu wyświetlania i kontynuuje od nowego adresu (najczęściej początku programu wyświetlania). Przejście na początek programu tworzy nieskończoną pętlę; oczekiwanie na koniec powrotu pionowego sprawia, że pętla jest zsynchronizowana z cyklem wyświetlania obrazu odbiornika telewizyjnego.

JMB i JVB są rozkazami 3-bajtowymi; pierwszy bajt jest kodem rozkazu, drugi i trzeci są adresem skoku.

Opcja LMS wymaga pewnego wyjaśnienia. Wybierana jest przez ustawienie 6 bitu bajtu rozkazu trybu wyświetlania grafiki lub znaków. Kiedy ANTIC spotyka taki rozkaz, wprowadza do licznika pamięci ekranu bajty argumentu. Licznik ten informuje procesor obrazu, gdzie zaczyna się RAM ekranu. Następnie ANTIC rozpoczyna pobieranie z tego obszaru danych do wyświetlania. Rozkaz LMS jest 3--bajtowy: jeden bajt kodu operacji i dwa bajty argumentu. W prostych programach wyświetlania rozkaz LMS używany jest tylko raz, na początku programu wyświetlania. Czasami może być konieczne użycie drugiego rozkazu LMS. Taka potrzeba występuje, kiedy obszar pamięci ekranu przekracza granicę 4 KB. Rejestr licznika pamięci ekranu ma długość 12 bitów, dlatego dane do wyświetlania nie mogą przekroczyć granicy 4 KB. Jeśli przekroczą, musi być użyty dodatkowy rozkaz LMS. Oznacza to, że dane do wyświetlania nie są w pełni przemieszczalne (przynajmniej dla niektórych trybów wyświetlania).

Tworzenie programu wyświetlania

Każdy program wyświetlania powinien rozpoczynać się od trzech rozkazów *„pustych ośmiu linii"*. Wykonuje się to w celu przesunięcia początku obrazu o 24 linie ekranu w dół. Po tym powinna być określona pierwsza linia. Jednocześnie powinno być użyte LMS do przekazania ANTIC, gdzie znajdzie on pamięć ekranu. Następnie zaczyna się właściwy program wyświetlania, który zawiera bajty dla linii obrazu na ekranie. Ogólna liczba poziomych linii ekranu wytworzonych przez program wyświetlania nie powinna przekraczać 192. ANTIC nie zachowuje wymagań czasowych ekranu telewizyjnego. Jeśli damy ANTIC zbyt wiele linii ekranu do wyświetlenia, zrobi to, lecz obraz prawdopodobnie będzie się przesuwał. Wyświetlanie mniej niż 192 linii ekranu nie spowoduje problemów. Programista musi obliczyć sumę poziomych linii ekranu wytworzonych przez program wyświetlania i zweryfikować ją. Program wyświetlania kończy się rozkazem JVB.

W celu utworzenia własnego programu wyświetlania musimy najpierw zaprojektować format obrazu. Wszystkie podręczniki zalecają zrobienie tego na papierze. Na razie zróbmy tak samo. Naszkicujmy obraz ekranu i przetłumaczmy go na ciąg linii obrazu. Przetłumaczmy ciąg linii obrazu na ciąg bajtów określających rozkazy ANTIC. Wstawmy trzy bajty *„pustych 8 linii"* na początku programu. Ustawmy 6 bit pierwszego bajtu programu. Utworzy to rozkaz ładowania licznika pamięci obrazu (LMS). Następne dwa bajty określają adres RAM ekranu. Po nich występują

pozostałe bajty programu. Na końcu programu wyświetlania wstawimy rozkaz JVB i adres początku programu wyświetlania. Teraz zapamiętujemy te bajty w RAM. Mogą one być w dowolnym miejscu; upewnijmy się tylko, że nie pokrywają się z obszarem grafiki gracz—pocisk lub wzorcami znaków i nasz JVB wskazuje początek programu wyświetlania. Następnie musimy wyłączyć ANTIC na ułamek sekundy, na czas przepisania adresu programu wyświetlania. Zrobimy to przez wstawienie 0 do SDMCTL pod adresem 559 ($22F). Wtedy zapamiętamy adres nowego programu wyświetlania do komórek o adresach 560 i 561 ($230 i $231). W końcu włączymy ANTIC wstawiając 34 ($22) do SDMCTL. W czasie powrotu pionowego, gdy ANTIC jest nieaktywny, system operacyjny wprowadzi te nowe wartości do jego licznika programu.

Działanie z własnym programem wyświetlania

Pamięć ekranu może być umieszczona gdziekolwiek w przestrzeni adresowej komputera. Normalnie program wyświetlania określa początek pamięci ekranu w pierwszym rozkazie wyświetlania — rozkaz LMS. Jednak ANTIC może wykonać nowy rozkaz LMS w każdej linii programu wyświetlania. W ten sposób informacja z całej przestrzeni adresowej komputera może być wyświetlona na jednym ekranie. Może to być cenne przy ustawianiu niezależnych okien tekstowych.

Istnieje kilka ograniczeń w umieszczaniu pamięci ekranu w RAM komputera. Po pierwsze, nie może ona przekraczać granicy adresu 4 KB. Jeśli nie możemy tego uniknąć (jak w trybie 8 BASIC'a, który używa 8 KB RAM), musimy ponownie załadować licznik pamięci ekranu za pomocą nowego rozkazu LMS. Po drugie, jeśli chcemy używać dowolnej procedury systemu operacyjnego (SO) obsługującej ekran, musimy dostosować się do konwencji używanych przez SO. Może to być dosyć trudne, gdy używamy zmodyfikowanego programu wyświetlania z BASIC'a. Jeśli zmieniliśmy standardowy program wyświetlania i próbujemy użyć instrukcji **PRINT** lub **PLOT**, system operacyjny wykona je tak, jakby program wyświetlania był niezmieniony. Spowoduje to prawdopodobnie uszkodzenie obrazu.

Istnieją trzy możliwe przyczyny zniknięcia obrazu podczas próby wyprowadzania informacji. Po pierwsze, BASIC może odmówić wykonania operacji na ekran, jeśli wybrany jest tryb graficzny, w którym taka operacja jest niemożliwa. System operacyjny przechowuje wartość aktualnego trybu graficznego w komórce o adresie 87 ($57). Możemy „przekonać" SO do współpracy przez wstawienie tam innej wartości. Pamiętajmy o wstawieniu liczby określającej tryb BASIC'u, nie ANTIC'u.

Drugi rodzaj uszkodzenia może wystąpić podczas mieszania linii obrazu wymagających różnych ilości bajtów pamięci ekranu. Pewne linie obrazu wymagają 40 bajtów na linię, inne — 20 bajtów, a niektóre tylko 10 bajtów na linię. Powiedzmy, że wstawiamy 20 bajtową linię obrazu do programu wyświetlania z 40-bajtową linią obrazu. Podczas wyprowadzania tekstu na ekran wszystko powyżej tej linii jest wyświetlane bez zmian, poniżej niej znaki są przesunięte o 20 pozycji w prawo. Jest tak, ponieważ system operacyjny przyjmuje, że każda linia wymaga 40 bajtów i odpowiednio pozycjonuje znaki. Lecz ANTIC, po napotkaniu takiej linii, wysyła tylko 20 bajtów. System operacyjny uważa, że powinna być linia 40-bajtowa. ANTIC interpretuje pozostałe 20 bajtów jako należące do następnej linii i tam je wyświetla. Powoduje to, że linia następna i dalsze będą przesunięte o 20 pozycji w prawo.

Jedynym sposobem uniknięcia tego problemu jest odstąpienie od używania instrukcji PRINT i PLOT do wyprowadzania informacji na ekran podczas stosowania własnego programu wyświetlania. Szybkim i prostym rozwiązaniem jest zorganizowanie ekranu w grupy linii, które zawie-

```
XS 185 ? :? "CZY CHCESZ MIEC WIDO
   CZNE OSIE X-Y";:INPUT A$:IF A$
   (1,1)="T" THEN W=1
TT 190 LST=1536
UD 195 GOSUB 30000
LE 200 POKE 559,0
HN 210 POKE 560,0
LH 220 POKE 561,6
RA 230 POKE 559,34
AZ 240 X=0:GOSUB 31000
ZM 250 POKE 87,0:POKE 752,1
EN 260 POSITION 13,0:? #6;"Wykres
    funkcji"
FR 265 POSITION 13,1:? #6;"------
   --------"
XU 268 X=1:GOSUB 31000:POKE 87,2:
   POSITION 0,0:? #6;"
              "
LH 270 ON N GOTO 1000,1100,1200,1
   300,1400,1500,1600,1700
BG 300 X=2:GOSUB 31000
WL 310 POKE 87,8:COLOR 1
SR 380 IF W<>1 THEN 410
KS 390 FOR I=0 TO 319 STEP 4:PLOT
    I,96:NEXT I
II 395 FOR I=0 TO 191 STEP 3:PLOT
    160,I:NEXT I
HJ 400 REM WYKRESLANIE
GI 410 FOR T=0 TO 360*U STEP KROK
JO 420 Q=T/57.3
YT 430 ON N GOTO 510,520,530,540,
   550,560,570,580
LL 500 REM ROWNANIA DLA R
FE 510 R=B*Q:GOTO 610
UV 520 R=A*(1+COS(Q)):GOTO 610
XL 530 R=A*(1-SIN(Q)):GOTO 610
YC 540 R=A*SIN(B*Q):GOTO 610
XB 550 R=A*COS(B*Q):GOTO 610
GJ 560 R=COS(A*SIN(B*Q)):GOTO 610
FH 570 R=SIN(A*COS(B*Q)):GOTO 610
ZW 580 R=A:GOTO 610
LC 600 REM WYKRESLANIE X,Y
UA 610 X=INT((R*COS(Q))*SK)
WJ 620 Y=INT((R*SIN(Q))*SK)
TR 630 IF ABS(X)>159 OR ABS(Y)>95
    THEN 670
IH 640 IF T=0 THEN PLOT 160+X,96-
   Y
ON 650 DRAWTO 160+X,96-Y
KO 660 NEXT T
IN 670 W=0
IH 680 X=3:GOSUB 31000:POKE 87,1
KC 690 POSITION 6,0:? #6;"nacisni
   j":POSITION 2,1:? #6;"DOWOLNY 
   KLAWISZ"
HJ 700 U=1:OPEN #1,4,0,"K:":GET #
   1,X:POSITION 0,0:PUT #6,125
YJ 710 POSITION 0,0:? #6;"czy kas
   owac ekran ?":GET #1,X:PUT #6,
   125
```

```
NL 720 IF X<>84 THEN GOTO 90
IO 730 X=2:GOSUB 31000:POKE 87,0:
   PUT #6,125:GOTO 90
QM 900 X=1:GOSUB 31000:POKE 87,2:
   POSITION 0,0:PUT #6,125:RETURN
XI 1000 POSITION 0,0:? #6;"
    R=";B;"*Q"
NR 1010 GOTO 300
ZL 1100 POSITION 0,0:? #6;"   R="
   ;A;"*(1+COS(Q))"
NT 1110 GOTO 300
JI 1200 POSITION 0,0:? #6;"   R="
   ;A;"*(1-SIN(Q))"
NV 1210 GOTO 300
EK 1300 POSITION 0,0:? #6;"   R="
   ;A;"*SIN(";B;"*Q)"
NX 1310 GOTO 300
YE 1400 POSITION 0,0:? #6;"   R="
   ;A;"*COS(";B;"*Q)"
NZ 1410 GOTO 300
AL 1500 POSITION 0,0:? #6;" R=COS
   (";A;"*SIN(";B;"*Q))"
OB 1510 GOTO 300
YP 1600 POSITION 0,0:? #6;" R=SIN
   (";A;"*COS(";B;"*Q))"
OD 1610 GOTO 300
KO 1700 POSITION 0,0:? #6;"
    R=";A
OF 1710 GOTO 300
DH 20000 REM ******************
IW 20001 REM *  Podprogram     *
DS 20002 REM * uklada program  *
OC 20003 REM *  wyswietlania   *
EB 20004 REM ******************
MK 30000 ON (START=1) GOTO 30380:
   SEG=1:DIM S$(12000):S$(1)=CHR$
   (0):S$(12000)=S$:S$(2)=S$:STAR
   T=1:I=1
IA 30010 LOK=LST
NX 30020 GEKR(0)=ADR(S$)
UX 30030 GRAN=INT((LST/1024+1)*10
   24)
UX 30040 GRAN2=INT(GEKR(0)/4096+1
   )*4096
BY 30050 REM
CC 30060 REM
CG 30070 REM
PM 30080 GEKR(SEG)=GEKR(SEG-1)
NK 30100 TRYB=NRGR(1+2*(I-1)):REP
   EAT=NRGR(2*I):I=I+1
YU 30110 IF TRYB<0 THEN OP=65:ADD
   R=LST:GOSUB 30330:RETURN
CM 30120 PRZYR=40
UM 30130 IF TRYB>=6 AND TRYB<=12
   THEN PRZYR=20
FM 30140 IF TRYB=8 OR TRYB=9 THEN
    PRZYR=10
WV 30150 FOR X=1 TO REPEAT
VO 30160 IF LOK<>GRAN-3 THEN 3020
   0
```

rają całkowite wielokrotności standardowych wymagań bajtów. To znaczy, nie wstawiamy 20-bajtowej linii do 40--bajtowego obrazu; zamiast tego wstawiamy dwie linie 20--bajtowe lub jedną 20-bajtową i dwie 10-bajtowe. Dopóki zachowujemy odpowiednie całkowite wielokrotności, unikamy przesuwania poziomego.

Rozwiązanie to podkreśla trzeci problem związany z własnymi programami wyświetlania: przesunięcie pionowe. System operacyjny pozycjonuje dane ekranu w pionie przez obliczenie liczby bajtów do przeskoczenia w dół od góry ekranu. W standardowej, 40-bajtowej linii obrazu BASIC może wstawić znaki w 10 linii po przeskoczeniu 360 bajtów. Jeśli mamy wstawić cztery linie 10-bajtowe, BASIC zakończy je trzy linie dalej w dół ekranu niż moglibyśmy oczekiwać. Ponadto różne linie obrazu zużywają różne ilości linii ekranu, zatem pozycja na ekranie nie będzie taka, jak oczekiwaliśmy, jeśli nie weźmiemy pod uwagę kosztów linii ekranu.

Jak możemy zobaczyć, mieszane tryby obrazu mogą być trudne do użycia. Często musimy oszukiwać system operacyjny, aby otrzymać żądane obrazy. W celu wydrukowania lub wykreślenia w wybranym oknie, wstawiamy **(POKE)** numer trybu BASIC'u tego okna do komórki o adresie 87, następnie wstawiamy adres lewego, górnego punktu elementarnego tego okna do komórek 88 i 89 ($58 i $59). W trybach znakowych wykonujemy **POSITION 0,0** w celu ustawienia lewego górnego rogu okna. W trybach kreślenia wszystkie instrukcje **PLOT** i **DRAWTO** będą wykonywane względem lewego górnego rogu okna jako początku układu współrzędnych.

Własny program wyświetlania może być używany do wytworzenia interesujących obrazów. Jego najbardziej oczywistym zastosowaniem jest mieszanie tekstu i grafiki. Na przykład, możemy przygotować ekran z wyraźnym tytułem w trybie 2 BASIC'a, średniej wielkości podtytułem w trybie 1 i małym napisem w trybie 0. Następnie możemy przejść do obrazu w trybie 8 BASIC'a w środku, z pewnym dłuższym tekstem u dołu ekranu. Dobry przykład tej techniki dostarczany jest przez obraz w programie Atari States and Capitals.

Zastosowania programów wyświetlania

Jednym z prostych zastosowań zmodyfikowanego programu wyświetlania jest pionowe odsunięcie linii na ekranie przez wstawienie pustych linii. Dzięki temu możemy uwypuklić krytyczne komunikaty i zwiększyć czytelność pewnych obrazów.

Innym ważnym zastosowaniem manipulacji programem wyświetlania jest umożliwienie wykorzystania własności niedostępnych z BASIC'a. Istnieje tryb tekstowy dostarczany przez ANTIC, którego nie obsługuje BASIC. Tylko manipulowanie programem wyświetlania umożliwia nam dostęp do tego trybu. Istnieją także przerwania programu wyświetlania i możliwości płynnego przesuwania obrazu, które są dostępne tylko po zmodyfikowaniu programu wyświetlania.

Manipulowanie rozkazem LMS i jego argumentami oferuje wiele możliwości twórczemu programiście. Na przykład, przez zmianę LMS w czasie powrotu pionowego strumienia elektronów, programista może zmieniać obrazy wyświetlane na ekranie. Można to robić z małą szybkością, w celu wymiany wcześniej przygotowanych obrazów, bez konieczności ponownego kreślenia każdego z nich. Dane obrazu mogą przebywać w pamięci nawet wtedy, gdy nie są używane, lecz mogą być dostępne w każdej chwili. Technika ta może być także używana do animacji. Przez wymianę ciągu obrazów możemy uzyskać cykliczną animację. Program, który to będzie realizował, może manipulować tylko dwoma bajtami adresu do wyświetlania wielu tysięcy bajtów RAM.

Możliwe jest także nakładanie obrazów przez ich wymianę z dużą szybkością. Oko ludzkie ma czas rozdzielczości

```
QN 30170 OP=1:ADDR=GRAN:GOSUB 303
   30
TT 30180 LOK=LOK+1:GRAN=GRAN+1023
YV 30190 GOTO 30200
IV 30200 POKE LOK,TRYB
SR 30220 IF (SEG<>1 OR X<>1) AND
   (GRAN2-GEKR(SEG)>=PRZYR) THEN
   30280
SL 30240 OP=TRYB+64
AO 30250 ADDR=GEKR(SEG)
FT 30260 IF GRAN2-GEKR(SEG)<PRZYR
    THEN GRAN2=GRAN2+4096:ADDR=AD
   DR+PRZYR
RP 30270 GOSUB 30330
CK 30280 LOK=LOK+1
TR 30290 GEKR(SEG)=GEKR(SEG)+PRZY
   R
MS 30300 NEXT X
HH 30305 IF TRYB=15 THEN GEKR(SEG
   )=GEKR(SEG)+PRZYR
VF 30310 SEG=SEG+1
BN 30320 GOTO 30080
ZE 30330 POKE LOK,OP
BX 30340 LOK=LOK+1
UW 30350 POKE LOK,ADDR-(INT(ADDR/
   256)*256)
CF 30360 LOK=LOK+1
PW 30370 POKE LOK,INT(ADDR/256)
EO 30380 RETURN
EJ 30900 REM ******************
JX 30910 REM *   Podprogram    *
TY 30920 REM *  ustala adres   *
WZ 30930 REM * pamieci ekranu  *
EZ 30940 REM ******************
PT 31000 POKE 88,GEKR(X)-(INT(GEK
   R(X)/256)*256)
RB 31010 POKE 89,INT(GEKR(X)/256)
   :RETURN
```

około 1/16 sekundy, zatem program może krążyć między trzema obrazami, każdy co 1/50 s, tak że każdy powtarza się co 1/16 sekundy. W ten sposób jednocześnie może wystąpić na ekranie do trzech obrazów. Oczywiście istnieją pewne wady tej metody. Pierwsza, oddzielne obrazy mogą zajmować sporo RAM. Druga, każdy obraz będzie rozmyty, ponieważ pokazywany jest tylko jedną trzecią czasu. Oznacza to, że tło wszystkich obrazów powinno być czarne, a każdy obraz — jasny. Ponadto podczas używania tej techniki wystąpią pewne nieprzyjemne migotania ekranu. Opisana technika może być także używana do zwiększenia rozdzielczości kolorów i jaskrawości. Dzięki wymianie trzech wersji tego samego obrazu, gdy każda wersja akcentuje jeden kolor lub zakres jaskrawości, dostępny jest większy zakres kolorów i jaskrawości.

Automatyczne tworzenie programu wyświetlania

Dla tych czytelników, którym udało się dobrnąć do końca poprzedniego artykułu oraz nie przestraszyli się trudności, jakie trzeba pokonać podczas tworzenia własnego programu wyświetlania, mamy niespodziankę. Pisaliśmy wcześniej, że na początku musimy zaplanować obraz za pomocą kartki papieru, opisaliśmy kolejno, krok po kroku, jakie czynności należy wykonać, aby uzyskać własny program wyświetlania. Omówiliśmy trudności, jakie należy pokonać, aby uzyskać żądane efekty na ekranie, przedstawiliśmy możliwe błędy i ich skutki. A teraz prezentujemy program, który uwalnia nas od troszczenia się o wszelkie szczegóły tworzenia programu wyświetlania i wykonuje za nas wszystkie obliczenia. Musimy, oczywiście, dostarczyć niezbędnych danych, do których należą: liczba segmentów, na jakie chcemy podzielić ekran, numery trybów ANTIC'a w poszczególnych segmentach oraz liczbę linii obrazu dla każdego segmentu. Informacje te podajemy w liniach 8 (ilość segmentów) oraz 10—43 (numer trybu oraz ilość linii obrazu dla każdego segmentu) programu nr 1. Program wyświetlania umieszczony będzie na stronie 6 pamięci RAM, od adresu 1536 (linia 140).

Linie 30000—30380 zawierają procedurę tworzenia programu wyświetlania na podstawie danych umieszczonych w tablicy NRGR, która w każdym wierszu zawiera numer trybu i liczbę linii obrazu. Procedura ta uwzględnia ograniczenia maksymalnej długości programu wyświetlania i maksymalnej ilości danych ekranu. Celowo nadaliśmy jej tak wysokie numery wierszy, aby każdy z zainteresowanych mógł go wykorzystać we własnych programach.

Podprogram umieszczony w liniach 30900—31010 aktualnie nie jest używany, lecz ułatwia rozwiązanie problemu adresowania ekranu dla poszczególnych segmentów. Jako dane wejściowe wykorzystuje on numer segmentu (zmienna X) oraz tablicę GEKR, utworzoną przez podprogram nr 1, która zawiera adresy początku pamięci ekranu dla każdego segmentu.

Program nr 2 demonstruje wykorzystanie omówionych podprogramów. Ma już określone następujące dane: liczba segmentów (zmienna N) oraz numery trybów i liczba linii obrazu dla każdego segmentu (linia 42). Przygotowany został w taki sposób, aby uniknąć dwukrotnego wprowadzania podprogramów nr 1 i 2 z poprzedniego programu. Należy po prostu program numer 2 dołączyć do programu nr 1 (za pomocą ENTER "C") w celu otrzymania gotowego programu. Utworzony przez niego program wyświetlania generuje obraz zawierający: 2 linie w trybie 0, jedną linię w trybie 2, 144 linie w trybie 8 oraz dwie linie w trybie 1 (są to tryby BASIC'u).

W liniach 200—300 następuje przekazanie systemowi operacyjnemu adresu programu wyświetlania. Linie 240—265 umieszczają na ekranie napisy w trybie 0, linie 270—290 umieszczają napis w trybie 2, linie 300—360 powodują wyświetlenie rysunku w trybie 8, a linie 370—395 umieszczają napisy w trybie 1.

Widoczne jest, że umieszczenie tekstu lub grafiki w wybranym segmencie wymaga: podania numeru segmentu, wywołania podprogramu nr 2, wstawienia do komórki o adresie 87 numeru trybu BASIC'a używanego w tym segmencie oraz wyprowadzenia informacji właściwych dla danego trybu. W wybranym segmencie adresowanie linii obrazu odbywa się względem lewego górnego rogu segmentu, który aktualnie jest początkiem układu współrzędnych tego segmentu.

Program nr 3 przygotowany został przez nas w celu zademonstrowania, jakich możliwości dostarcza manipulowanie programem wyświetlania. Najciekawszą z nich jest uzyskanie obrazu o wysokości 240 linii ekranu. Dzięki temu uzyskaliśmy dodatkowe 48 linii na umieszczenie różnorodnych napisów przy niezmienionej liczbie linii przeznaczonych na rysunek (192). Przyznajemy, jest to nieco rozbieżne z informacjami zawartymi w poprzednim artykule, ale tamte dane pochodzą z wszelkiego rodzaju opisów i instrukcji, a ten efekt wynikł z naszych poszukiwań i prób. Wymagało to wprowadzenia pewnych modyfikacji do procedury tworzenia programu wyświetlania, dlatego zamieszczamy pełny wydruk gotowego programu. Innym ciekawym efektem jest wykorzystanie dwóch obrazów: jeden z nich posiada standardowy program wyświetlania, a drugi — przygotowany przez nas. Dodatkową ciekawostką jest fakt, że dane do wyświetlania drugiego obrazu umieszczone są

wewnątrz programu, w zmiennej tekstowej S, a nie jak zwykle w obszarze pamięci poza programem.

Oprócz tego zadaniem programu nr 3 jest wyznaczenie i wyświetlenie wykresów takich jak: spirala, kardioida, rozeta, itp. Na początku prezentuje menu i prosi o wybranie funkcji, a następnie o jej parametry, przyrost i skalę. Przyrost jest ilością stopni (krokiem), które komputer używa do zwiększenia kąta T od 0 do 360 stopni. Musimy określić, czy preferujemy szybkość działania, czy dokładność. Mały przyrost, np.: 0,1, będzie kreślił dokładny wykres bardzo wolno. Większy przyrost, np.: 5.0 wykreśli szybciej lecz mniej dokładnie. Dobrym kompromisem jest 1.0. Skala umożliwia dobranie wielkości rysunku na ekranie. Jej wartość w zakresie 10 do 100 powinna być wystarczająca dla większości funkcji. Linie 175—180 sprawdzają, czy nie wprowadziliśmy wartości zerowej, która mogłaby przerwać działanie programu i pojawienie się komunikatu błędu. Linia 182 to pytanie, czy mają być wyświetlone osie x—y układu współrzędnych, a linie 390—395 kreślą je. Linie 400—690 zawierają procedury obliczania i kreślenia x i y. W linii 410 występuje zmienna U dla funkcji nr 1, która określa ilość obrotów spirali, jej wartość może być zmieniona w linii 170. Linia 430 kieruje program do obliczania wartości funkcji wybranej na początku. Linie 610—620 obliczają współrzędne x, y. Linia 630 sprawdza, czy nie przekroczyliśmy granic ekranu. W przypadku przekroczenia zatrzymuje kreślenie i nie dopuszcza do wystąpienia komunikatu błędu. W liniach 680—690 program czeka na naciśnięcie klawisza w celu przejścia do kreślenia innej funkcji i ewentualnego oczyszczenia ekranu. Jeśli wartości funkcji są zbyt duże do kreślenia, od razu pojawia się propozycja naciśnięcia klawisza (należy zmniejszyć wartość skali).

Ze względu na to, że nie każde wartości parametrów dają interesujące efekty na ekranie, proponujemy wykorzystać następujące wartości dla poszczególnych funkcji

1)	A=1,	B=1,	KROK=60,	SKALA=4
	A=1,	B=1,	KROK=45,	SKALA=5
	A=1,	B=1,	KROK=120,	SKALA=5
2)	A=2,	B=1,	SKALA=20	
3)	A=2,	B=1,	SKALA=20	
4)	A=2,	B=3,	SKALA=80	
6)	A=2,	B=6,	SKALA=90	
	A=2,	B=2,	SKALA=90	
	A=3,	B=1,	SKALA=90	
	A=1,	B=100,	SKALA=90	
7)	A=1,	B=1,	SKALA=90	
	A=4,	B=2,	SKALA=90	

Mamy nadzieję, że przedstawione propozycje zachęcą dociekliwych czytelników do dalszych poszukiwań interesujących efektów graficznych.

Do wprowadzania programów do pamięci komputera radzimy wykorzystać Edytor BASIC'u zamieszczony w nr. 4 „IKS-a" z bieżącego roku. Pamiętajmy: dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną danej linii, dlatego nie wprowadzamy ich do komputera.

Programy należy przechować w pamięci zewnętrznej (taśmowej lub dyskowej) przed ich uruchomieniem albo za pomocą funkcji „L" Edytora BASICU'u, albo za pomocą instrukcji LIST, w przypadku wprowadzania programu w sposób konwencjonalny

NOWOŚCI

Pomimo wprowadzenia na rynek rewelacyjnych 16-bitowych komputerów domowych serii ST, Atari dba również o posiadaczy sprzętu ośmiobitowego (niesłusznie uważanego u nas za przestarzały). Jednym z przejawów dbałości jest zestaw RAMBO XL, umożliwiający rozszerzenie pamięci RAM komputera 800 XL do 256 KB i uczynienia go zgodnym ze 130 XE. Dzięki temu możliwe jest efektywne wykorzystanie systemu operacyjnego DOS 2.5 ze standardowym RAM-dyskiem o pojemności 64 KB oraz najnowszej wersji języka BASIC XE. Nowy handler RD.COM dostarczany ze Sparta DOS Construction Set, udostępnia RAM-dysk o pojemności 192 KB. Jest to pamięć wystarczająca do skopiowania dyskietki o podwójnej gęstości zapisu w jednym przebiegu!

Powyższe rozszerzone możliwości „poczciwej 800" dostępne są również rodzimym posiadaczom tych komputerów i w dodatku za złotówki. KAREN, firma zajmująca się naprawami sprzętu ATARI, oprócz montowania BASIC'u XL (za około 26 tysięcy złotych), oferuje również rozszerzenie pamięci 800 XL do 256 KB za około 70 000 złotych oraz BASIC XE (najnowszy produkt Optimized System Software) za około 30 000 złotych.

Tomasz MROWIEC, Ludwik PIELA

ATARI

O zerowaniu zmiennych

W podręcznikach opisujących zasady programowania w języku Atari-Basic twierdzi się, że „po zatrzymaniu programu klawiszem RESET lub zakończeniu działania programu, w którym użyto instrukcji END, wszystkie zmienne są wyzerowane, tzn. zmienne liczbowe mają wartość 0, a tekstowe zawierają łańcuch spacji".

Zobaczmy, jak to jest w rzeczywistości? Posłuży nam do tego krótki program:

```
10 DIM A$(3)
20 INPUT A$,B,C
30 PRINT"A$=": A$: PRINT "B=", B: PRINT "C=": C
40 D = D + 1
50 PRINT"D=": D
60 END
```

Po uruchomieniu programu instrukcją RUN i wpisaniu wartości zmiennych

A$ = "ALA"
B = 1
C = 2

otrzymujemy na ekranie wynik działania programu w postaci

A$ = ALA
B = 1
C = 2
D = 1

Jeżeli teraz uruchomimy program instrukcją GOTO 30, otrzymamy ten sam wynik, przy czym D = 2. Po naciśnięciu klawisza RESET i uruchomieniu programu instrukcją GOTO 30, otrzymamy znów wynik (3) ale przy D = 3.

Przeprowadzone badania wskazują, że ani instrukcja END, ani klawisz RESET nie zerują zmiennych! Jaka zatem instrukcja powoduje ich zerowanie? Aby odpowiedzieć na to pytanie, wyeliminujemy z programu (1) instrukcję END. Po uruchomieniu programu instrukcją RUN i wprowadzeniu wartości zmiennych (2) otrzymujemy wynik (3). Jeżeli teraz wyeliminujemy linię 20 i uruchomimy program instrukcją RUN, otrzymamy wynik

A$ =
B = 0
C = 0
D = 1

Z powyższego wynika niezbicie, że to właśnie instrukcja **RUN** a nie **END** czy **RESET** powoduje wyzerowanie zmiennych. Nieuwzględnienie tego faktu wszędzie tam,

gdzie wielokrotnie uruchamiamy program (podprogram) instrukcją inną niż RUN, może prowadzić do powstania różnego rodzaju błędów. Można ich uniknąć nadając zmiennym pożądane wartości (w tym również zero) instrukcją podstawienia przed rozpoczęciem podprogramu czy pętli

Oczywiście zmienne zeruje również instrukcja CLR. Używając jej należy jednak pamiętać, że zeruje ona wszystkie zmienne programu (co nie zawsze nam odpowiada), a po jej użyciu należy ponownie zadeklarować zmienne indeksowane i tekstowe.

Andrzej GRZESIAK

Pogwarki o CPC 464

Pogawędka druga — firmware, czyli zdolności wrodzone

Dawno, dawno temu, gdy nie było jeszcze komputerów osobistych, a z dużymi komputerami moje kontakty ograniczały się do przekazywania kart dziurkowanych i odbierania wyników, elektroniczną maszynę cyfrową traktowałem jak komplet bezdusznych, metalowych mebli, migocących kolorowymi lampkami i obracających bezustannie krążki taśm magnetycznych. Całkiem odmienne wrażenie wywarł na mnie pierwszy, „poznany" mikrokomputer. Początkowo do złudzenia przypominał wyrośnięty kalkulator, ale gdy tylko został uruchomiony, złudzenie prysło. Zupełnie, jakby zbudzona została żywa istota. Każde polecenie, każda instrukcja powodowała szybką reakcję, która przy szczątkowej nawet znajomości języka angielskiego mogła kojarzyć się z prymitywnym dialogiem. Oczywiście przyczyną takiego zachowania, martwego skądinąd, narzędzia jest program. Wiadomo również, że w każdym „szanującym się" mikrokomputerze aktualnie realizowany program (lub jego część) musi znajdować się w pamięci wewnętrznej maszyny. Zatem, aby natychmiast po włączeniu urządzenia wyświetlony został powitalny napis:

Amstrad 64K Microcomputer (v1)
© 1984 Amstrad Consumer Electronics plc and Locomotive Software Ltd.

a następnie zgłosił się interpreter języka BASIC, producent musiał umieścić w pamięci odpowiednie zestawy instrukcji. Co ważne, nie mogą one ulec wyzerowaniu przy wyłączeniu komputera, ani tym bardziej przypadkowemu uszkodzeniu. Pamięć maszyny podzielono więc na część przeznaczoną na programy użytkowników, zwaną RAM (Random Access Memory — pamięć o dostępie przypadkowym) i część, którą zarezerwował dla siebie producent, czyli ROM (Read Only Memory — pamięć tylko do odczytu). Wbudowanie niewymazywalnej pamięci ROM jest istotną cechą mikrokomputerów osobistych, ułatwiającą ich eksploatację użytkownikowi bez przygotowania informatycznego. Nie oznacza to jednak, że wszystkie mikrokomputery muszą mieć ROM. Gdy są go pozbawione, jedną z pierwszych operacji po włączeniu zasilania jest wczytanie z pamięci zewnętrznej (dysku, taśmy, a w starych maszynach również z dziurkowanej taśmy papierowej) odpowiedniego programu zarządzającego, zwanego systemem operacyjnym. Dopiero wówczas można wprowadzić program użytkowy. Analogicznie postępujemy chcąc wczytać CP/M 2.2, wtedy gdy do naszego AMSTRADa jest przyłączona stacja dysków. Różnica polega na tym, że CPC 464 jest wyposażony w ROM i w dodatku z bardzo interesującą zawartością, a zatem CP/M jest systemem alternatywnym, pozwalającym zwiększyć możliwości komputera. O tym, jak istotną rolę odgrywa umieszczone w pamięci stałej oprogramowanie, niech świadczy fakt, że wszyscy, decydujący się na samodzielne budowanie mikrokomputerów, z tym elementem mają zazwyczaj najwięcej kłopotów. Najczęściej radzą sobie, kopiując produkt znanej firmy. W ten sposób unikają możliwości popełnienia wielu błędów, często niezwykle trudnych do wykrycia oraz zapewniają swojemu dziełu bibliotekę programów użytkowych. Warto w tym miejscu zaznaczyć, że różnice w programach zarządzających są jedną z ważnych przyczyn, iż np. ZX SPECTRUM i AMSTRAD CPC 464, pomimo zastosowania w nich mikroprocesorów tego samego typu, nie są wymienne programowo.

Przeglądając prospekty mikrokomputerów łatwo zauważyć, że wielkość pamięci stałej jest różna dla różnych typów maszyn. Jednak rozumowanie, że większa pojemność ROM, to większy program, a co za tym idzie większe możliwości komputera oraz komfort pracy może być złudne. Zbyt wiele zależy tu od przyjętej koncepcji i zawartości oprogramowania, a co bardzo ważne — również od jego poprawności.

Czym zatem producent wypełnia stałą pamięć komputera? W przypadku AMSTRADa CPC 464 w 32 Kbajtach ROMu firma Locomotive Software Ltd. wpisała interpreter języka BASIC i tak zwany **FIRMWARE**, to znaczy zbiór niskopoziomowych, czyli napisanych w kodzie maszynowym podprogramów, które sterują pracą urządzeń zewnętrznych, synchronizują i nadzorują pracę elementów komputera. Podprogramy te mogą być w pełni wykorzystane przez programistów piszących w asemblerze, a zdecydowana większość z nich jest dostępna także w BASICu i innych językach.

FIRMWARE został podzielony na pakiety dotyczące poszczególnych elementów systemu. Tak na przykład pakiet o wdzięcznej nazwie **KEY MANAGER** (co nieodparcie kojarzy mi się z „dyrektorem od klawiszy") jest przeznaczony do obsługi klawiatury oraz joysticków i realizuje trzy poziomy operacji. Najniższy poziom, to badanie „czy?" i „które?" klawisze zostały wciśnięte, poziom średni — przyporządkowanie tym klawiszom odpowiednich wartości liczbowych, a poziom najwyższy to zamiana wartości liczbowych na właściwe znaki. KEY MANAGER zawiera także procedury obsługi przerwań realizowanych przez wciśnięcie klawiszy ESC lub SHIFT, CTRL i ESC. Kolejny pakiet — **TEXT VDU** jest programem pozwalającym zapisywać na ekranie monitora znaki tekstu oraz odczytywać je. Kontroluje osiem różnych „strumieni", z których każdemu może odpowiadać wydzielony obszar ekranu, zwany oknem tekstowym (window). TEXT VDU traktuje ustalone znaki jako kody kontrolne, co umożliwia różnorodne efekty jak ruch kursora lub zmiana koloru znaków. Dostęp do pojedynczych punktów graficznych ekranu, to znaczy ich zapalanie lub określanie ich aktualnego stanu oraz rysowanie linii umożliwia pakiet zwany GRAPHICS VDU. W jego skład wchodzą między innymi podprogramy umożliwiające posługiwanie się układem współrzędnych, określające aktualną pozycję kursora graficznego, określające kolory tzw. pióra i tła albo inaczej „papieru". Układy elektroniczne związane bezpośrednio z tworzeniem obrazu oraz pakiety TEXT VDU i GRAPHICS VDU dopasowuje do siebie pakiet pośredniczący — **SCREEN PACK**. Zawarte w nim procedury pozwalają wybrać tryb graficzny, otrzymać „mieniące się" barwy, uzyskać zamiast kolorów odpowiadające im odcienie szarości w razie posługiwania się monitorem monochromatycznym. Najważniejsza funkcja tego programu polega na zamianie współrzędnych użytecznych w TEXT VDU i GRAPHICS VDU na adresy w pamięci ekranu.

Również układ elektroniczny generujący dźwięk, czyli **SOUND CHIP**, ma swojego „dyrektora" w pamięci ROM. SOUND MANAGER, bo o nim mowa, pozwala zmieniać wysokość i czas trwania dźwięku, kształtować jego obwiednię, a tym samym brzmienie oraz synchronizować ze sobą dźwięki wytwarzane w trzech kanałach. W AMSTRADzie CPC 464 sterowanie dźwiękiem realizowane jest nieomal całkowicie programowo.

Problem, który w tym mikrokomputerze został rozwiązany chyba najlepiej ze wszystkich znanych mi urządzeń tej klasy, to zapisywanie zbiorów (w tym i programów) na taśmie kasetowej, oraz wczytywanie ich do pamięci komputera. Zbiory zapisywane są w oddzielonych od siebie blokach zawierających 2K (2048) bajtów lub mniej, w wypadku gdy jest to ostatni blok w zbiorze. Zbiory i bloki mają odpowiednie nagłówki, czyli informacje umieszczone na początku danych oraz zakończenia, które w pełni je charakteryzują. Takie rozwiązanie znacznie ułatwia kopiowanie i wczytywanie danych, a także organizację bibliotek programowych, gdyż w razie wystąpienia błędów czytania, wystarczy powtórzyć wprowadzanie zbioru od bloku, w którym błąd się pojawił. Poza tym poszczególne bloki zbioru mogą być kopiowane niezależnie jeden od drugiego. Sterowanie zapisem i odczytem informacji oraz mechanizmem magnetofonu kasetowego, wbudowanego w komputer odbywa się w pełni programowo i jest realizowane przez pakiet FIRMWARE nazywany **CASSETTE MANAGER**.

Innym pakietem odpowiedzialnym za współpracę komputera z urządzeniami zewnętrznymi jest **MACHINE PACK**. Powoduje on, że między innymi układy interfejsu równoległego (PPI — port A), odpowiedzialnego za tryb wyprowadzania danych, a w tym za współpracę z generatorem dźwięku oraz układy sterujące monitorem ekranowym (Cathode Ray Tube Controller) i układy wyjścia **CENTRONICS**, służącego do podłączenia drukarki, przyjmują wymagane przez instrukcje programu stany fizyczne. A zatem MACHINE PACK synchronizuje pracę urządzeń zewnętrznych i podporządkowuje ją aktualnie realizowanemu programowi. Pakiet ten zawiera ponadto procedury wprowadzania i uruchamiania programu użytkowego.

Jądrem systemu operacyjnego, umieszczonego przez producenta w pamięci stałej, jest **KERNEL**, który koordynuje realizację programów, zarządza wykorzystaniem pamięci i wyborem przyłączonych układów ROM, a także uczestniczy w obsłudze przerwań i zdarzeń. Istnieje jeszcze jeden element FIRMWAREu — **JUMPER**, tworzący w chwili włączenia komputera, w pamięci **RAM** — **JUMPBLOCK**, czyli blok instrukcji skokowych umieszczonych pod określonymi, stałymi adresami. Procedury, które się tam znajdują, zapewniają możliwość wywołania rozmaitych podprogramów FIRMWAREu, pomimo że związane z tym parametry (np. adresy lub wartości pewnych zmiennych systemowych) mogą się zmieniać w trakcie realizacji programu. Wykorzystanie pośrednika, jakim jest JUMPBLOCK, powoduje, że zmiany te są niezauważalne dla użytkownika. Elementy FIRMWAREu są dla niego dostępne, podobnie jak inne podprogramy wyższego poziomu. Zatem, aby tworzyć bardziej zaawansowane „dzieła", w których konieczne jest sterowanie HARDWAREm, nie musimy poznawać dogłębnie zasad działania poszczególnych urządzeń i sposobu ich kontrolowania przez system operacyjny. W wielu wypadkach wystarczy znajomość tego pakietu. Oczywiście, ponieważ JUMPBLOCK znajduje się w pamięci RAM, bardziej zaawansowany użytkownik może dokonywać w nim zmian i w konsekwencji np. zastąpić pewne firmowe procedury własnymi. W tym celu niekoniecznie zresztą trzeba się dobierać do tego pakietu, wystarczy dobra znajomość asemblera i organizacji pamięci. Tak, tylko wykorzystanie JUMPBLOCKa pozwoli nam w dużej mierze uniknąć nieświadomego bałaganu, jaki może powstać przy takich operacjach.

A zatem, jest sobie JUMPBLOCK, a nawet cztery bloki instrukcji skokowych. Pierwszy i największy, zwany głównym... ale o tym w następnej pogawędce.

(ROMan)

Od zabawy do nauki

Rozmowa z kpt. Waldemarem Pająkiem, kierownikiem Ośrodka Obliczeniowego w Wyższej Oficerskiej Szkole Samochodowej im. gen. Aleksandra Waszkiewicza w Pile.

— Oprócz funkcji zawodowych pełni Pan także funkcję opiekuna uczelnianego Klubu Mikrokomputerowego „Aneks"...

— Owszem, od początków jego powstania, drugi rok. To chyba jeden z pierwszych klubów mikrokomputerowych w kraju, a na pewno w uczelniach wojskowych.

— Pytanie chyba zasadne w tym miejscu: jakie macie komputery?

— Nie jest źle, chociaż mogłoby być lepiej i zanosi się na to, że będzie. Zaczynaliśmy od sprzętu wypożyczonego. Wprawdzie kiedyś Rada Młodzieżowa Wojska Polskiego obiecywała, że wyposaży nas w mikrokomputery, ale tej obietnicy do tej pory nie dotrzymała. „Aneks" ma obecnie 14 mikrokomputerów. Siedem Spectrum należy do klubu, natomiast tyle samo Amstradów, przeznaczonych głównie do celów dydaktycznych, wypożyczyła nam uczelnia. W tym roku mamy nadzieję zwiększyć nasz stan posiadania o kolejnych kilka Amstradów. Są one bardziej przydatne dla naszych celów.

— A właściwie, jakie są to potrzeby?

— Klub Mikrokomputerowy powstał przy Ośrodku Obliczeniowym i jego pracownicy są równocześnie nauczycielami i instruktorami. Właśnie oni przy pomocy podchorążych z Naukowego Koła Informatyki uczą języka, programowania i obsługi komputera. Przede wszystkim opieramy się na programach przydatnych do studiów i odpowiadających profilowi szkoły. Obejmują one przykładowo charakterystyki poszczególnych elementów i podzespołów samochodu. W ten sposób ucząc się czegoś nowego można lepiej zrozumieć rozpatrywane zagadnienia i znacznie wzbogacić wiedzę.

Nie koniec na tym. Pracujemy nad różnymi programami użytkowymi, które przyczynią się do poprawy administrowania i dowodzenia szkołą. Na przykład, kpr. pchor. SPR Wojciech Karcz, zwolennik komputeryzacji, buduje program pomocny przy typowaniu na praktyki słuchaczy tutejszej Szkoły Podchorążych Rezerwy. Ofertę klubu dopełnia działalność typowo relaksowa, a więc wszelkiego rodzaju gry komputerowe.

— Kto może być członkiem „Aneksu"?

— Liczba miejsc jest ograniczona bazą, którą dysponujemy, czyli ilością sprzętu i lokalem. W Ośrodku Obliczeniowym jest ciasno, ale wkrótce mamy dostać pomieszczenia w remontowanym Klubie Podchorążych. Regułą jest, że należy ten, kto chce, ale też prawdą jest, że kto pierwszy, ten lepszy. W ubiegłym roku „Aneks" miał 35 członków, wyłącznie podchorążych wyższej szkoły oficerskiej. Czasami z komputerów korzystają podchorążowie SPR. Prócz tego raz w tygodniu prowadzimy zajęcia dla dzieci kadry zawodowej uczelni. Cieszą się one dużym wzięciem, przychodzą i pociechy, i ich rodzice.

— Z jakich korzystacie źródeł, z jakiej literatury?

— Ze wszystkiego, co zdobędziemy. Mamy swoją biblioteczkę, do której członkowie i sympatycy „Aneksu" przynoszą każdą pozycję, jaką uda im się zdobyć. Prenumerujemy czasopisma fachowe: „Iksa", „Bajtka" i „Komputer". Przegrywamy ciekawsze programy.

— A plany?

— Rozwinąć działalność, zdobyć jeszcze lepszy sprzęt, zainteresować komputeryzacją kogo się da. No i pogłębiać swoje umiejętności, aby przyczyniły się one do opanowania tajników wiedzy wymaganej od specjalistów samochodziarzy.

Rozmawiał: **J. RAJCH**

To urządzenie pozwala na szkolenie nawigatorów w warunkach symulowanych sytuacji powietrznych.





W tej kabinie-trenażera śmigłowca piloci doskon[...]lą w zdobywaniu umiejętności działania na wypadek n[...] przewidzianych zdarzeń. Przy użyciu sprzętu bojowe[...] byłoby to niemożliwe do osiągnięcia.

Starty i lądowania bez samolotów

Marek SIENIAWSKI

Na wskaźniku stacji radiolokacyjnej zaczyna pojawiać się coraz więcej celów powietrznych. Do odparcia ataku kierowane są siły obrony powietrznej kraju. Do akcji wkraczają samoloty bojowe, a więc piloci, kierownicy lotów, nawigatorzy. Uruchomiony zostaje cały system obrony. Alarm bojowy! Setki celów przeciwnika, a także własnych środków powietrznych znajdują się w akcji. W takiej sytuacji ludzie odpowiedzialni za kontrolę lotów jeszcze nigdy się nie znaleźli. Została ona tylko wymyślona. I byłoby najlepiej, gdyby do niej nie doszło. Gdyby jednak... Czy sobie poradzą?

Aby sprawdzić zachowanie się obsługi naziemnej w warunkach ekstremalnych, należałoby je realnie stworzyć. Innymi słowy w powietrzu umieścić odpowiednio oznakowane samoloty przeciwnika i własne. Taki sprawdzian jest jednak bardzo trudny do przeprowadzenia (nawet jednorazowo) i bardzo kosztowny. A jak zaplanować strącenia samolotów? Choćby z tych powodów imitacja jest trudna.

Każdy samolot, który znajduje się w powietrzu, **jest ciągle obserwowany i kierowany z ziemi.** Tak jest w lotnictwie cywilnym i wojskowym. Zajmijmy się wojskowym, wyposażonym w statki powietrzne, zdolne do działania w krótkim czasie na duże odległości, z dużą prędkością (ponad trzykrotnie większą od dźwięku — około 1 km na sekundę), uzbrojone w precyzyjne rakiety różnego typu. I jeszcze jedno. Lotnictwo wojskowe charakteryzuje się bardzo krótkimi czasami gotowości do masowego użycia. A to oznacza bardzo dużą odpowiedzialność obsługi naziemnej za podejmowane decyzje.

Jak szkolić ludzi do tej obsługi? Wygląda to mniej więcej tak, że po przeszkoleniu w uczelni wojskowej, nawigatorzy odbywają praktyki na stanowiskach dowodzenia. Okazuje się, iż najlepszą metodą nauczania jest praca indywidualna mistrza z uczniem. Trwa to dość długo, zanim „czeladnik" uzyska uprawnienia, zanim praktycznie będzie zdolny samodzielnie kierować lotem maszyny bojowej. Takie szkolenie, poza tym że jest bardzo pracochłonne, jest również kosztowne.

Obecnie technika komputerowa pozwala już na stworzenie urządzeń, w których można **imitować dowolne sytuacje powietrzne.** Ale sam pomysł nie wystarczy. Idea skonstruowania takiego imitatora była jednak na tyle kusząca, że zajęli się nią specjaliści z Instytutu Technicznego Wojsk Lotniczych.

Jak bardzo to trudne zadanie? Przyjrzyjmy się pracy ludzi z obsługi naziemnej lotnictwa wojskowego. Wróćmy zatem do sytuacji skrajnej, gdy ogłoszono alarm bojowy. Złożoność jej widać na ekranach stacji radiolokacyjnych.

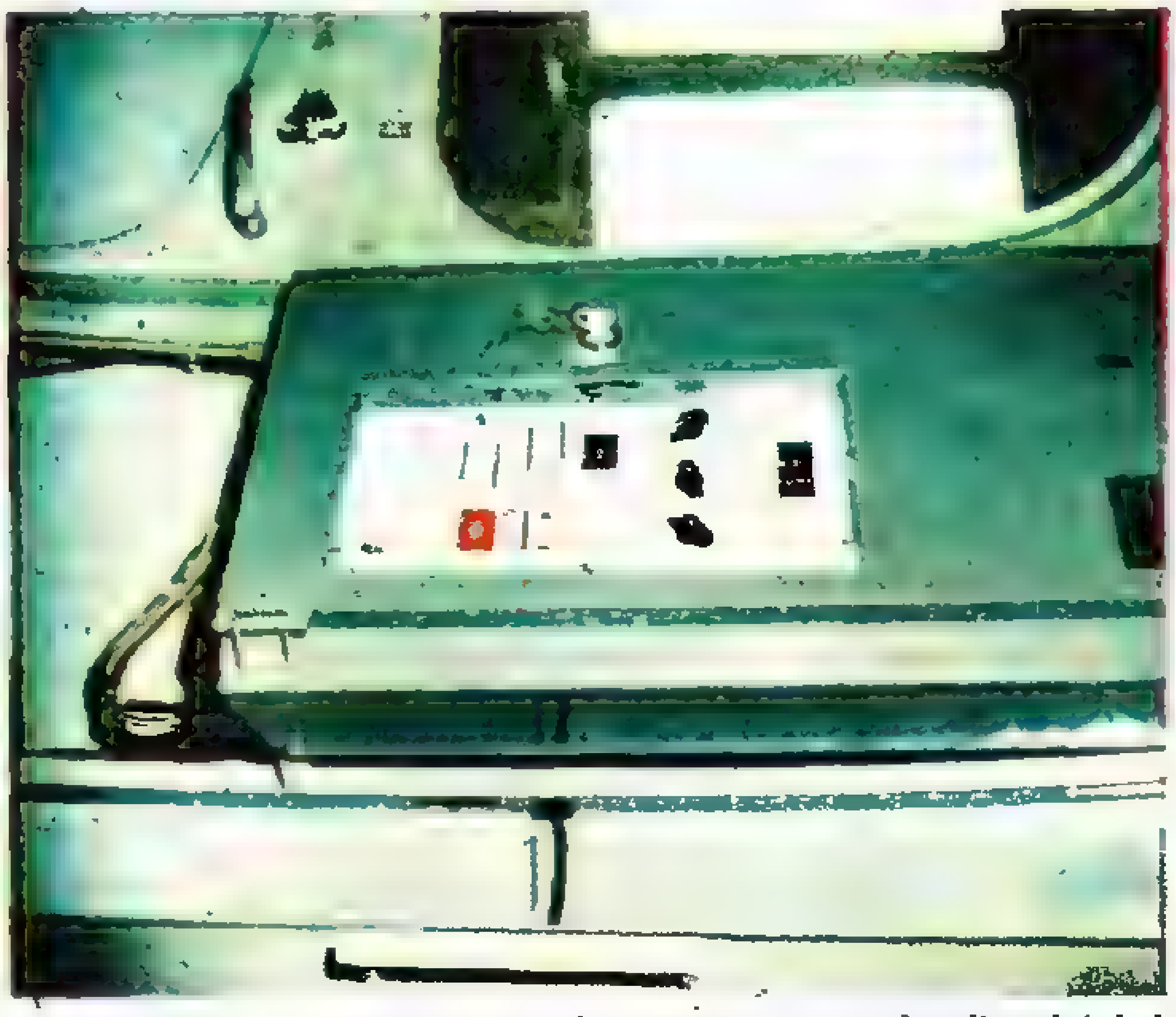
Kierownik lotów zarządza start samolotów dyżurnych. Gdy znajdują się one w powietrzu, kieruje nimi nawigator-operator. Prowadzi maszyny tylko w obrębie kilkudziesięciokilometrowego lotniska. Ma za zadanie **nie dopuścić do zagrożenia,** gdy wzrośnie liczba samolotów. Dla współczesnego myśliwca, tylko przy prędkości równej prędkości dźwięku, jedna minuta lotu oznacza przebycie około dwudziestu kilometrów.

Za strefą lotniska następuje przekazanie maszyny nawigatorowi naprowadzania. Z ziemi kieruje on „swój" samolot w rejon działania obcego celu. Kiedy pilot przechwyci cel — czyli widzi go na radarze pokładowym, wówczas samodzielnie wykonuje zadanie. Odpala rakiety. Znajduje się wtedy setki kilometrów od macierzystego lotniska.

Wraca. I znowu zostaje po kolei przekazywany „z rąk do rąk" osobom funkcyjnym na ziemi. Sytuacja spiętrza się, gdy w tym samym czasie w rejonie lotniska znajduje się dużo samolotów. Trzeba określić, który z nich ma **lądować, i w jakiej kolejności.** Maszyny są więc poukładane w przestrzeni powietrznej jakby w szufladkach. Nie mogą się tam znajdować długo. Mają za mały zapas paliwa. A krążenie wokół lotniska jest kosztowne, niepotrzebne i ryzykowne.

Decyzję kierownika lotów o lądowaniu wykonuje kierownik systemu lądowania. Doprowadza samolot precyzyjnie do pasa startowego. Najtrudniejsze są **sytuacje awaryjne,** np. gdy samolot jest uszkodzony i musi lądować w pierwszej kolejności. Wtedy





Instruktor może zaprogramować najbardziej skomplikowaną sytuację, aby w ten sposób określić przydatność ludzi do kierowania ruchem samolotów.



łą się
k nie-
wego



Kabina trenażera nie różni się niczym od prawdziwego sprzętu latającego.

wszystkie inne trzeba szybko poszufladkować i wytyczyć pustą drogę temu jednemu. Tego manewru, choćby ze względu na bezpieczeństwo lotów, nie uda się nigdy przeprowadzić dla celów szkoleniowych. A wiadomo, że ludzie muszą być do tego przygotowani. Potrzebny jest więc odpowiedni trenażer.

Te i wiele innych barier udało się pokonać naukowcom z ITWL. Zespół kierowany przez płk. mgr. inż. Janusza Kucfira i płk. dr inż. Henryka Gajewskiego opracował całą rodzinę imitatorów do kierowania lotami samolotów przy użyciu komputera.

Ale przecież komputer nie rozumie ludzkiej mowy, może ktoś zarzucić. Tymczasem trzeba zagwarantować naturalne warunki pracy kierownikowi lotów, nawigatorowi i pilotom. Funkcję pośrednika między ludźmi i komputerem zupełnie dobrze pełni operator lub operatorzy, gdy występuje więcej ćwiczących lub kontrolowanych.

Tak więc sposób kierowania samolotami przy użyciu imitatora IKS (zbieżność z tytułem miesięcznika jest przypadkowa), nie różni się niczym od kierowania lotami realnymi. Szczególną cechą urządzenia jest **modułowość w sensie technicznym i programowym.** Można szkolić specjalistów indywidualnie, w różnych grupach, włącznie z pilotami i dowolnie dobranymi zespołami. Maksymalnym wykorzystaniem wszystkich możliwości IKS-a (co już zrealizowano) jest prowadzenie walki powietrznej przez całą jednostkę lotniczą. Stwarzano różne skomplikowane sytuacje. W trakcie ćwiczenia ingerowano w jego przebieg, co świadczy o tym, że **imitator nie uczy schematów działania,** jest elastyczny w użyciu. Można zmienić liczbę celów powietrznych, imitować zestrzelenia, awarie, wypadki, katastrofy, a zatem sprawdzać zachowanie się wszystkich uczestników w skrajnie trudnych warunkach.

Obecnie więc bez startów, lotów i lądowań samolotów, bez narażania zdrowia ludzi, sprzętu na straty, a także wystawiania nerwów na poważne próby, wszystko może się odbywać w pomieszczeniach imitatora. Różnorodne sytuacje planuje dowódca lub instruktor. Włącznie z zakłóceniami radiowymi, atmosferycznymi, warunkami krytycznymi. Czyli takimi, jakie mogą realnie zaistnieć. Obsługa naziemna jest już do nich przygotowana.

Próby, w których uczestniczyli mistrzowie w specjalnościach kierowania lotami, ujawniły, że nie zawsze potrafili oni wyjść obronną ręką z trudnych zadań. Bo w takich warunkach dość często się nie znajdowali.

Jak wynika ze statystyk wypadków lotniczych, aż połowa z nich spowodowana jest winą pilotów i ludzi z obsług naziemnych, czyli innymi słowy wynika z niedoszkolenia. Już te fakty wskazują, że zastosowanie IKS-a uzupełnia tę lukę znakomicie.

Jak wielka jest to luka, świadczą też opinie specjalistów lotnictwa wojskowego w całym świecie. Twierdzą oni, że możliwości super nowoczesnych statków powietrznych w zestawieniu z wyszkoleniem obsług naziemnych, zgrania z pilotami, są niewykorzystane w dość dużym procencie.

Imitatory kierowania samolotami stwarzają więc ogromne **możliwości zgrywania systemów obrony powietrznej,** prowadzenia walki i wykorzystania do końca wszystkich walorów sprzętu bojowego oraz umiejętności ludzi.

W państwach socjalistycznych jest to pierwszy tego typu imitator o tak uniwersalnym zastosowaniu i dużych walorach użytkowych. Nic dziwnego, że zainteresowanie nim naszych sąsiadów i sojuszników jest bardzo duże. Ale nie tylko do celów lotnictwa wojskowego IKS może być dostosowany. Dzięki swojej modułowej budowie może równie dobrze wykonywać zadania w szkoleniu kontrolerów ruchu powietrznego na każdym lotnisku. Jeszcze jedną właściwością jest bowiem dostosowanie komputera do warunków każdego lotniska i obszaru dowolnego państwa.

Względy ekonomiczne (koszty paliwa, zużycia sprzętu, pracy wielu urządzeń lotniskowych), a także bezpieczeństwa lotów i tempa, w jakim można przygotować oraz kontrolować dokładność pracy obsługi naziemnej, zgrania z pilotami, są niewątpliwe.

Marek SIENIAWSKI

Sztuki i sztuczki (9)

Zmienne systemowe ROM-SPECTRUM

Wskaźniki stanu

FRAMES 23672..4
FLAGS 23611
FLAGS2 23658
TYFLAG 23612
FLAG-X 23665
Seed 23670/1
ULA (Uncommited Logic Array) jest modułem odpowiedzialnym między innymi za przekazanie pamięci ekranu z RAM na monitor. Od momentu włączenia komputer wysyła informacje w ustalonym formacie 256x192 punkty co 20 ms. Zrozumiałe, że ULA przerywa wtedy wszelkie operacje wykonywane przez mikroprocesor. Jednocześnie wymusza na nim wykonanie procedury o adresie 56 (RST 8), o której wspomniałem w odcinku 1. Anglicy nazywają to zjawisko ogłupieniem procesora. Oprócz przeszukiwania klawiatury

```
10 DEF FN z()=PEEK 23672+256*PEEK 23673
+65536*PEEK 23674
20 PRINT FN z()
```

dokonuje się wtedy zliczanie przerwań. Z bajtów FRAMES możemy dowiedzieć się, ile razy od momentu włączenia komputera ULA „ogłupiła" mikroprocesor.

Ponieważ dokładność taktowania zegara jest wystarczająco duża (błąd rzędu 0.01%) można założyć, że w jednej sekundzie przerwań będzie 50. Stąd prosta droga, aby liczba przerwań określała czas w sekundach od inicjalizacji systemu.

20 PRINT FN z ()/50; „s".

Nasz sztuczny „mechanizm zegarowy" jest następujący: po 255 przerwaniach (5. 1s.) zliczanych w 23672 następuje zerowanie tego bajtu i inkrementacja bajtu 23673. Po 65535 przerwaniach nastąpi wyzerowanie obu i inkrementacja kolejnego bajtu. „Zegar" wróci do pozycji wyjściowej po 2124-1 (czyli 16777215) przerwaniach. Chcąc wyrazić to w jednostkach czasu, otrzymamy 3 dni, 21 godzin, 12 minut, 24.32 sekundy. Mając taki argument w ręku, można praktycznie do każdego programu wprowadzić ten „czas". Należy jednak pamiętać, że niektóre procedury wyłączają przerwania i czas „stoi w miejscu". Są to wszystkie komendy współpracujące z urządzeniami zewnętrznymi, a więc **LOAD, SAVE, MERGE, VERIFY, COPY** i **BEEP**.

Wykorzystamy zegar do sprawdzenia czasu reakcji wzrokowej i manualnej:

```
 10 DIM t(10)
 20 DEF FN z()=PEEK 23672+256*PEEK 23673
+65536*PEEK 23674
 30 LET b=1: LET s=0: CLS
 40 PRINT "program sprawdza twój refleks"
 50 PRINT "        s start"
 60 IF PEEK 23556<>83 THEN GOTO 60
 70 CLS: PRINT AT 10,0;
 80 FOR a=0 TO 9
 90 PRINT PAPER 4;a;PAPER 7;"  ";: NEXT a
100 RANDOMIZE INT (RND*65536)
110 LET r= INT (RND*10)
120 PRINT AT 1,0;"wciskaj odpowiedni klaw
isz numeryczny"
130 POKE 23674,0: POKE 23673,0: POKE 2367
2,0
140 PRINT AT 10,r*3: FLASH 1;r
150 IF INKEY$<>STR$ r AND FN z()<=5 THEN
GOTO 150
160 LET t(b)=FN z()/50
170 IF t(b)>5 THEN LET t(b)=5
180 LET b=b+1: IF b<11 THEN GOTO 70
190 FOR a=1 TO 10: LET s=s+t(a): NEXT a
200 LET b=s/10: CLS
210 PRINT "sredni czas reakcji """"     ";
b;" s."
220 REM histogram
230 DRAW 150,0
240 FOR a=1 TO 10: PLOT 10*a,0
250 FOR b=0 TO 30*t(a): PLOT 10*a,b
260 DRAW 5,2: NEXT b: NEXT a
270 PLOT 10,30*s: DRAW OVER 1;150,0
```

Flagi systemowe przechowują bardzo istotne informacje o stanie, w jakim znajduje się system operacyjny. Każda flaga zajmuje 1 bit, a więc może wynosić 0 lub 1. Poniżej przedstawione są znaczenia poszczególnych flag:

FLAGS	stan 0	stan 1
bit 0	drukuj spację przed znakiem	wydruk spacji zabroniony
bit 1	wyjście na ekran (kanał S)	wyjście na drukarkę (kanał P)
bit 2	drukuj kursor K	drukuj kursor L
bit 3	tryb K	tryb L
bit 5	nie wykryto klawisza	naciśnięto klawisz
bit 6	wynik ostatniej operacji jest łańcuchem tekstowym	wynik jest liczbą
bit 7	sprawdzanie syntaktyki linii	egzekucja linii

FLAGS 2	stan 0	stan 1
bit 0	czyszczenie ekranu nie jest konieczne	ekran główny jest czysty
bit 1	bufor drukarki jest pusty	bufor drukarki powinien być wyzerowany
bit 2	dwukropek rozdziela wyrażenia w linii	dwukropek jest znakiem w łańcuchu
bit 3	tryb L (low letters)	tryb C (capitals)
bit 4	aktualnym kanałem nie jest K	aktualnym kanałem jest K

TVFLAG	stan 0	stan 1
bit 0	wydruk na ekranie głównym	wydruk na ekranie edycyjnym
bit 3	tryb pracy nie może się zmienić (INPUT)	tryb pracy może się zmienić (edycja)
bit 4	zwykły listing (LIST)	listing automatyczny
bit 5	ekran edycyjny nie wymaga czyszczenia po naciśnięciu klawisza	wyczyścić ekran edycyjny po naciśnięciu klawisza

FLAG-X	stan 0	stan 1
bit 0	operacja wycinania znaków (slicing)	operacja na całym stringu
bit 1	zmienna już istnieje	nowa zmienna
bit 5	tryb edycji (EDIT)	tryb wprowadzania (Input)
bit 6	INPUT tekstowy	INPUT NUMERYCZNY
bit 7	INPUT bez frazy LINE	INPUT LINE

Zastosowanie flag w BASIC'u jest niewielkie i w zasadzie nie spotyka się ich w programach. Najczęściej wykorzystuje się zmianę trybu wydruku liter na ekranie.

POKE 23658,8

ustawia tryb C (duże litery), a

POKE 23658,0

powoduje wydruk liter małych w trybie L.

Zatrzymanie egzekucji programu w dowolnym miejscu z raportem OK odbędzie się po instrukcji:

POKE 23611, PEEK 23611-128

Częściej używa się flag systemowych przy programowaniu w języku wewnętrznym, lecz należy to robić ze szczególną ostrożnością.

Na koniec zmienna **SEED**, która znalazła się w tej grupie dodatkowo. Jeśli używamy funkcji **RND**, która generuje liczby pseudolosowe, to jest ona obliczana właśnie na podstawie zawartości **SEED**. Wniosek, że umieszczenie tam jakiejś liczby z zakresu 1..65535 spowoduje narzucenie zawsze stałego ciągu wartości funkcji **RND**, jest więc prawidłowy. Można taki proces zainicjować instrukcją **RANDOMIZE n**, której jedynym efektem jest przeniesienie **n** do **SEED**. Stąd sekwencja rozkazów:

```
10 RANDOMIZE 70
20 PRINT INT (RND*6)+1
30 GOTO 10.
```

spowoduje zawsze wygenerowanie liczby 1.

Zmienną SEED wykorzystuje się często do konwersji liczby 16-bitowej na postać dwubajtową np.:

```
10 INPUT n
20 IF n<> INT n THEN GOTO 10
30 IF n<0 OR n>65535 THEN GOTO 10
40 RANDOMIZE n
50 PRINT n;"(";PEEK 23670;",";PEEK
   23671;")"
```

Ostatnia część cyklu będzie poświęcona kalkulatorowi i arytmetyce zmiennoprzecinkowej w Spectrum..

Krzysztof MAMCARZ

Komputer w medycynie

Tomograf jądrowego rezonansu magnetycznego

W ostatnim dziesięcioleciu diagnostyka medyczna wzbogaciła się o liczną rodzinę urządzeń zwanych tomografami, umożliwiającymi za pomocą różnych metod fizyki, bezinwazyjne — a więc nie wymagające użycia skalpela, badanie dowolnego narządu wewnętrznego pacjenta. Poznaliśmy już bliżej tomograf komputerowy, w którym czynnikiem penetrującym organizm pacjenta jest promieniowanie Roentgena. Charakter tego promieniowania niesie za sobą pewne ograniczenia w wykorzystaniu tomografu komputerowego ze względu na szkodliwość częstych prześwietleń. Urządzenie to doskonale nadaje się do obrazowania na monitorze telewizyjnym trójwymiarowych kształtów badanego narządu, głównie części kostnych, ale nie dostarcza wielu informacji o wewnętrznej budowie oglądanych struktur. Tomografia komputerowa jako pierwsza i względnie tania metoda diagnostyki medycznej, wykorzystująca elementy wiedzy medycznej, fizyki i informatyki, jest jednak powoli wypierana przez inną, bardziej precyzyjną metodę tomografii jądrowego rezonansu magnetycznego.

Nowa metoda obrazowania wewnętrznych struktur organizmu na monitorze telewizyjnym wkroczyła przebojem na scenę medyczną, dokonując przełomu w technice wykrywania chorób nowotworowych. Tomograf jądrowego rezonansu magnetycznego, podobnie jak rentgenowski tomograf komputerowy, pozwala obserwować zmiany nowotworowe w organiźmie, czyli narastanie guzów niszczących tkanki i narządy; jest jednak jedynym tego rodzaju urządzeniem wskazującym również miejsca zagrożone możliwością powstania raka.

Aby zrozumieć zasadę działania tego rewelacyjnego urządzenia, musimy przypomnieć sobie kilka znanych z lekcji fizyki wiadomości o budowie atomu. Atom składa się z jądra atomowego, zbudowanego z protonów i neutronów oraz z elektronów krążących w dość dużej odległości wokół jądra. W interesującym nas zjawisku ograniczmy się do pamiętania o jądrach atomowych, które możemy wyobrazić sobie jako bączki wirujące bezładnie w przestrzeni. Okazuje się, że w przypadku atomów wodoru, umieszczonych w polu działania dostatecznie dużego magnesu, jądra atomowe tych pierwiastków, a więc nasze bączki, zaczynają ustawiać swoje osie obrotu równolegle do kierunku linii pola magnetycznego. Za pomocą dodatkowego, pulsującego pola elektromagnetycznego możemy tak rozhuśtać wirujące bączki, że zmieni się ich prędkość obrotowa i położenie osi obrotu. O jądrach znajdujących się w takim właśnie, wzbudzonym stanie mówimy, że zostały wprowadzone w rezonans, a całe zjawisko nazywa się jądrowym rezonansem magnetycznym. Po ustaniu pulsującego pola magnetycznego jądra atomów wracają do poprzedniego stanu, zwracając przyjętą energię w postaci fal radiowych — czyli wzbudzone jądra atomowe głośno sygnalizują powrót do normalności. Czas trwania tego zjawiska, zwany czasem relaksacji jądrowego rezonansu magnetycznego, wskazuje czy narząd, którego atomy wodoru wprowadziliśmy w stan rezonansu magnetycznego, jest zdrowy, czy odbiega od normy na skutek zmian nowotworowych. Czas relaksacji u osób chorych lub zagrożonych nowotworem jest dłuższy niż u ludzi zdrowych. Tyle należałoby powiedzieć, aby zrozumieć teoretyczną zasadę pracy tomografu jądrowego rezonansu magnetycznego, możemy zatem przejść do samego urządzenia.

Jest bardzo kosztowne. W jego skład wchodzi potężny elektromagnes, generator pulsującego pola elektromagnetycznego oraz komputer, sterujący urządzeniem i przechowujący zbierane parametry fizyczne badanego narządu. Pacjent umieszczany jest w specjalnej kapsule, wewnątrz której wytwarzane jest pole magnetyczne 60 000 razy przekraczające ziemskie pola magnetyczne. Pole to porządkuje jądra atomowe wodoru, ustawiając ich osie obrotu w kierunku linii pola magnetycznego. Następnie do pracy wkracza generator pulsującego pola elektromagnetycznego, wprowadzając kolejno w rezonans magnetyczny jądra atomowe, leżące wzdłuż osi ustalonego przez komputer przestrzennego układu współrzędnych. Dla stworzenia grafiki komputerowej wybranego narządu wewnętrznego obszar zawierający ten narząd podzielony jest w pamię-

ci komputera na cegiełki porównywalne rozmiarami z cząsteczkami chemicznymi. W pamięci komputera zarejestrowane zostają czasy relaksacji jąder atomów każdej cegiełki budującej badany obszar wraz z informacją o liczbie protonów i właściwościach magnetycznych każdej z nich. Dane te podlegają obróbce komputerowej, a jej rezultatem jest kolorowy, przestrzenny obraz wideo badanego narządu. Próbkę możliwości tej metody diagnostycznej daje zamieszczone powyżej zdjęcie przedstawiające grafikę komputerową głowy zdrowego człowieka. Na zdjęciu tym widoczne są wszystkie miękkie części głowy, takie jak mózg czy jama ustna. O ich kolorze i stopniu jego jaskrawości decyduje budowa i magnetyczne właściwości obrazowanych elementów anatomicznych.

Tomograf jądrowego rezonansu magnetycznego, dzięki swojej nieszkodliwej dla zdrowia pacjenta metodzie działania, staje się w rękach lekarzy narzędziem niezwykle pożytecznym i mimo swojej ceny, stosowanym w świecie na coraz większą skalę. Długą listę jego możliwości otwiera sztandarowy przykład diagnostyki chorób nowotworowych, nie należy jednak zapominać, że lista ta jest dużo dłuższa. Rozpoznawanie stanów przedzawałowych, alergii czy wczesnych stadiów cukrzycy — to tylko kilka z jego dalszych zastosowań. Dlatego też dobrze się stało, że kilka naszych rodzimych ośrodków medycznych podjęło intensywne starania, mające na celu zakup i zainstalowanie w Polsce pierwszych egzemplarzy tego niezwykle potrzebnego urządzenia.

(A.T.)

Elementy systemów mikrokomputerowych (5)

Intel 8080

Jacek WOJTALA

Mikroprocesor INTEL 8080 jest przedstawicielem linii rozwojowej, której kolejnymi konstrukcjami są mikroprocesory: firmy INTEL 8008, 8080, 8085; firmy ZILOG Z80 i Z8000 oraz rodzina firmy INTEL i APX 86, 88 czyli 8086, 8088, 80188, 80186, 80286, 80386.

INTEL 8080 jest układem wykonanym w technologii NMOS. Technologia ta umożliwiła zwiększenie liczby elementów umieszczonych na półprzewodnikowej strukturze układu oraz zdecydowany wzrost szybkości działania mikroprocesora, w stosunku do układów wykonanych we wcześniej wykorzystywanej technologii PMOS (np. INTEL 8008). Mikroprocesor, wprowadzony na rynek w 1974 roku należy do najpopularniejszych mikroprocesorów 8-bitowych.

ARCHITEKTURA MIKROPROCESORA

Mikroprocesor 8080, przedstawiony na rys. 10, zawiera m.in. jednostkę arytmetyczno-logiczną ALU, zespół rejestrów z układami adresowania, 8-bitowy rejestr rozkazów IP oraz wewnętrzną magistralę danych.

8-bitowa **ALU** przeznaczona jest do wykonywania operacji arytmetycznych, logicznych, przesunięć. W jej skład wchodzą: blok do prowadzenia operacji arytmetycznych i logicznych oraz rejestry — 8-bitowy **akumulator A,** 8-bitowy **rejestr buforowy,** 8-bitowy **rejestr pomocniczy (przejściowy),** 5-bitowy **rejestr wskaźników F.**

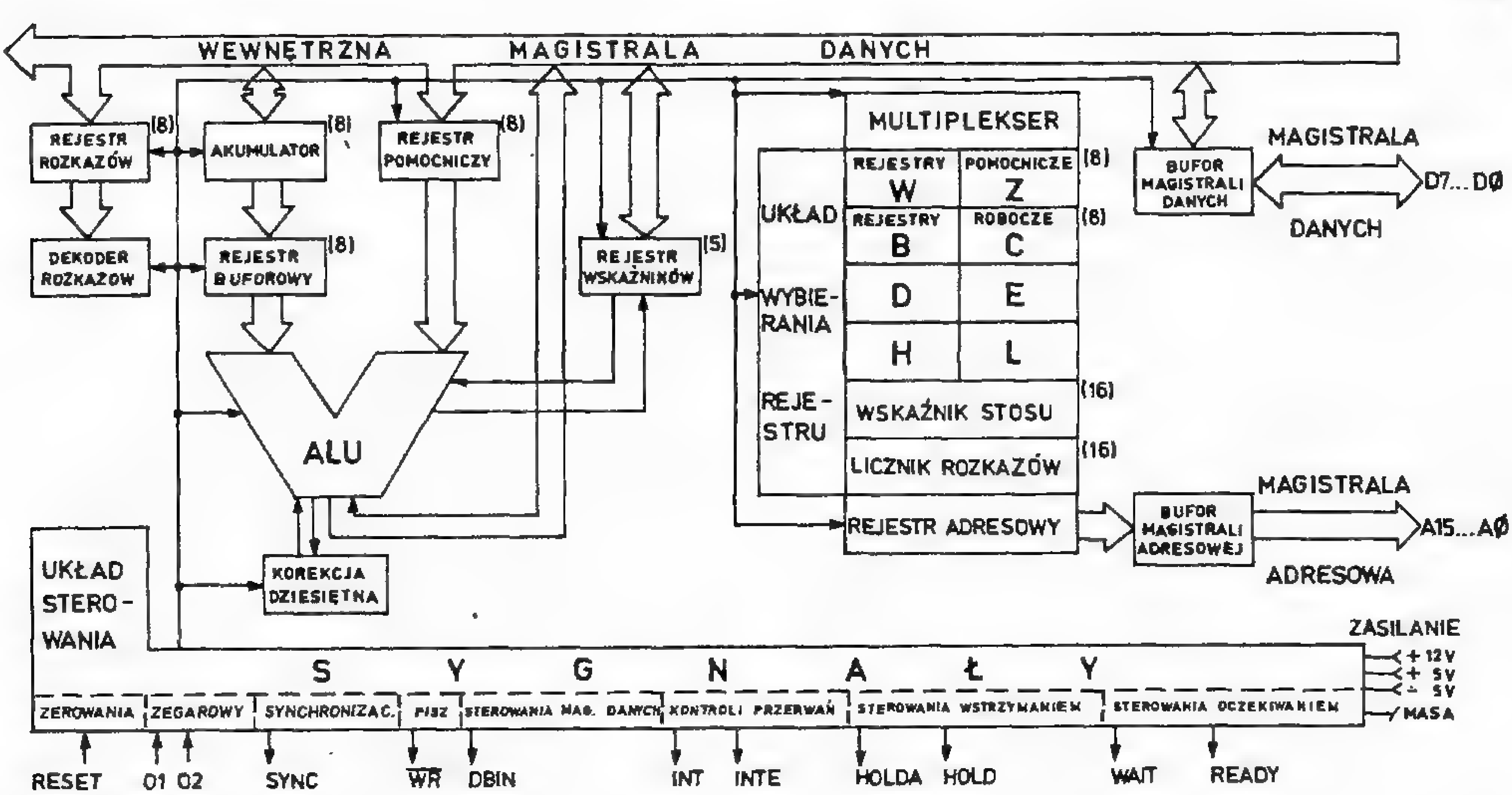
Zespół rejestrów z układami adresowania składa się z: 16-bitowego **licznika rozkazów PC;** 16-bitowego **wskaźnika stosu SP;** sześciu 8-bitowych **rejestrów roboczych,** oznaczonych literami B, C, D, E, H, L, tworzących trzy pary rejestrów oraz z dwóch rejestrów W, Z, niedostępnych programowo. Para rejestrów może tworzyć 16-bitowy **licznik danych DC,** adresujący komórki pamięci w przestrzeni adresowej 0,..., 65535 (2^{16}).

DZIAŁANIE MIKROPROCESORA

Każdy cykl rozkazów mikroprocesora 8080, czyli czas konieczny na pobranie i wykonanie rozkazu, zawiera od 1 do 5 cykli maszynowych. Cykl maszynowy trwa od 3 do 5 taktów zegarowych. Pobranie rozkazu oznacza przesłanie go z pamięci do rejestru rozkazów. Wykonanie rozkazu oznacza zdekodowanie go i przetłumaczenie na odpowiednie mikrorozkazy.

W zależności od tzw. **słowa statusu** (status word) w mikroprocesorze 8080 wyróżnia się 10 następujących cykli maszynowych: pobranie rozkazu, odczyt z pamięci, zapis w pamięci, odczyt ze stosu, zapis na stos, odczyt z układu wejścia-wyjścia, zapis w układzie wejścia-wyjścia, potwierdzenie przerwania, potwierdzenie rozkazu HLT, potwierdzenie przerwania podczas stanu HALT.

Uproszczoną zasadę działania mikroprocesora 8080 można przedstawić następująco. Wykonanie każdego z rozkazów rozpoczyna się od pobrania kodu rozkazu i przekazania go do rejestru rozkazów, co następuje w pierwszym cyklu maszynowym.



Intel 8080

Licznik rozkazów zwiększany jest o 1, w układzie sterowania dekodowany jest kod rozkazu i mikroprocesor decyduje, czy konieczne jest pobranie następnych bajtów. Jeśli nie jest konieczne pobieranie następnych bajtów, wówczas rozpoczyna się cykl, w którym następuje wykonanie rozkazu.

ZESPÓŁ JEDNOSTKI CENTRALNEJ

W skład systemu 8080 wchodzą: mikroprocesor 8080, **generator sygnałów zegarowych** 8224 i **kontroler systemu** 8228.

Układ 8224 przeznaczony jest do generowania: sygnałów zegarowych, sterujących jednostką centralną; i sygnału sterującego kontrolerem systemu oraz do synchronizacji pracy jednostki centralnej.

Układ 8228 pełni funkcję kontrolera systemu i dwukierunkowego bufora do magistrali danych. Wytwarza sygnały kontrolne niezbędne do bezpośredniej współpracy jednostki centralnej z pamięciami i układami obsługującymi urządzenia wejścia-wyjścia. Zespół jednostki centralnej przedstawiono na rys. 11.

UKŁADY WSPÓŁPRACUJĄCE

System 8080 wymaga stosowania układów pomocniczych, poniżej przedstawiono niektóre z nich.

8205 dekoder binarny 1 z 8. Wykorzystywany jest do dekodowania struktur pamięci i układów wejścia-wyjścia. Jedno z ośmiu wyjść przechodzi w stan zera logicznego, gdy zostaną odpowiednio ustawione 3 wejścia zezwalające.

8212 8-bitowy rejestr buforowy. Wykorzystywany jest do wprowadzania i wyprowadzania informacji z systemu oraz do zsynchronizowania przepływu informacji wewnątrz systemu.

8214 kontroler priorytetu przerwań. Wykorzystywany do przejmowania maksymalnie 8 sygnałów żądania przerwania i wybrania sygnału o najwyższym priorytecie.

8216 4-bitowy dwukierunkowy wzmacniacz. Wykorzystywany w celu zapewnienia właściwej współpracy układów MOS i TTL, tzn. do wzmacniania sygnałów przesyłanych magistralą danych i magistralą adresową.

8251 programowany szeregowy układ wejścia-wyjścia (USART). Wykorzystywany do transmisji szeregowej synchronicznej i asynchronicznej między systemem a innymi urządzeniami.

8255 programowany równoległy układ wejścia-wyjścia. Wykorzystany może być jako dwa 8-bitowe porty wejścia-wyjścia A i B, 8-bitowy port wejścia-wyjścia C w dwóch 4-bitowych częściach i 8-bitowy dwukierunkowy bufor magistrali danych.

8253 układ programowanych liczników. Zawiera trzy niezależne układy czasowe, każdy z liczników programowany jest indywidualnie.

8257 układ bezpośredniego dostępu do pamięci DMA. Jest specjalizowanym procesorem, sterującym bezpośrednim przesyłaniem danych między pamięcią a urządzeniami zewnętrznymi, z pominięciem jednostki centralnej.

8259 programowany układ sterowania przerwań. Wykorzystywany do sterowania 8 poziomami przerwań. Umożliwia rozbudowę systemu przerwań do 64 poziomów oraz ma wiele dodatkowych możliwości w stosunku do 8214.

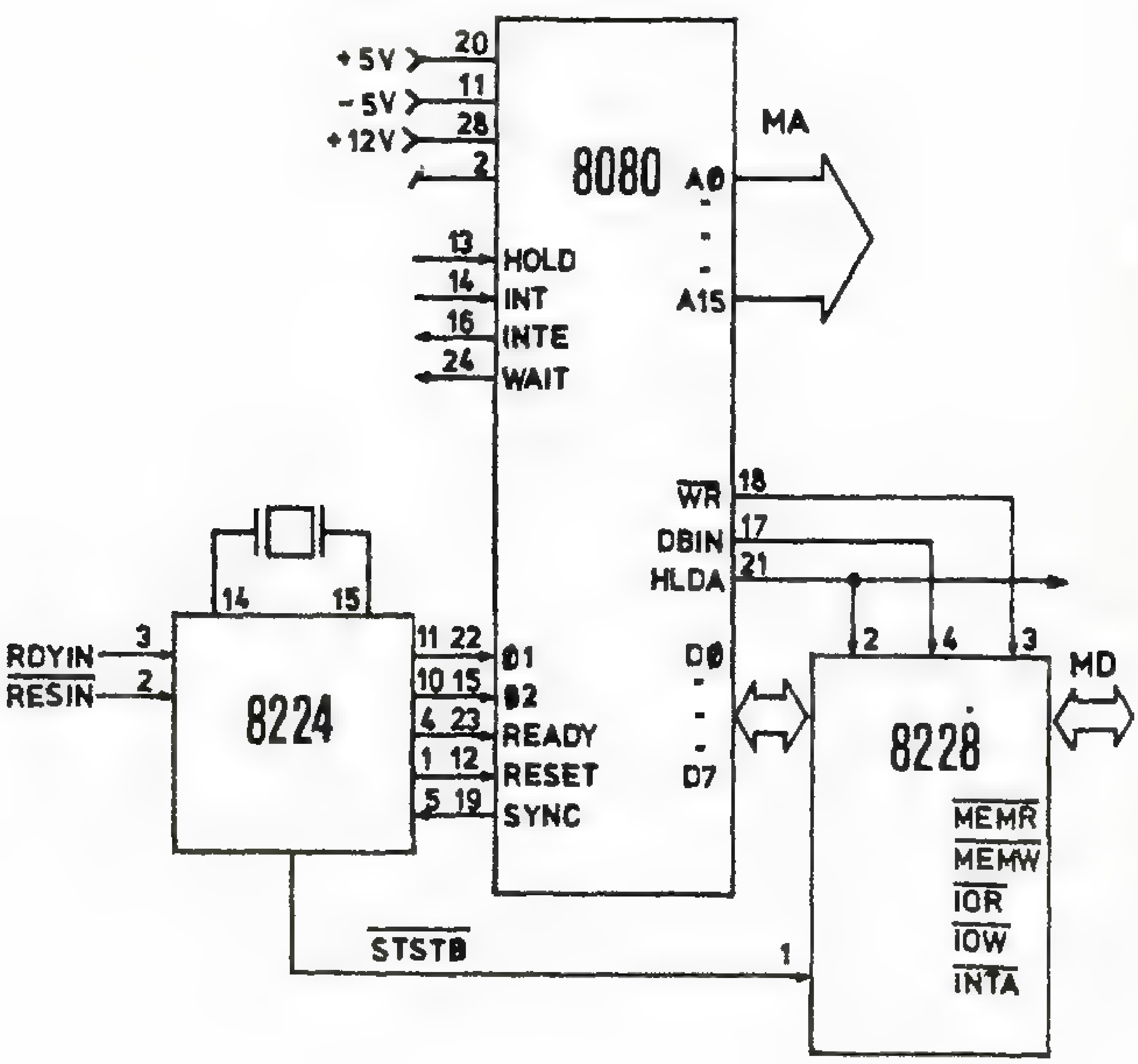
8271 kontroler stacji dysków elastycznych

8275 kontroler monitora ekranowego

8279 programowany układ współpracy z klawiaturą oraz wskaźnikami cyfrowymi.

W systemie mogą być wykorzystywane m.in. następujące typy pamięci RAM:
- statyczne
  2101 o pojemności 256×4 bity;
  2102 o pojemności 1024×4 bity;
- dynamiczne
  2107 o pojemności 4096×1 bit;
  2116 o pojemności 16384×1 bit.

Jako pamięci stałe wykorzystuje się najczęściej:
- pamięci PROM
  3601 o pojemności 256×4 bity;
  3604 o pojemności 512×8 bitów,
- pamięci EPROM
  2708 o pojemności 1024×8 bitów,
  2716 o pojemności 2048×8 bitów.



PRZERWANIA

W wielu systemach mikroprocesorowych musi istnieć możliwość zawieszenia wykonywania programu i zareagowania mikroprocesora na sygnał przychodzący z zewnątrz. Mikroprocesor 8080 umożliwia włączenie lub wyłączenie **układu przerwań** (systemu przerwań). Po włączeniu tego układu do kontrolera 8228, mikroprocesor, kończąc każdy cykl rozkazowy, bada stan linii zgłoszenia przerwania. Jeśli przerwanie nastąpiło, mikroprocesor zawiesza wykonywanie programu i wykonywany jest skok do podprogramu, tzn. programu obsługi przerwania. Po zakończeniu obsługi przerwania następuje powrót do programu.

LISTA ROZKAZÓW

Mikroprocesor 8080 charakteryzuje się listą 78 rozkazów, które można podzielić na pięć grup:
- **rozkazy przesyłania informacji;**
- **rozkazy arytmetyczne;**
- **rozkazy logiczne;**
- **rozkazy skoków, wywoływania podprogramów;**
- **rozkazy sterujące, obsługi wejścia-wyjścia, stosu.**

Lista rozkazów mikroprocesora rozpocznie następny odcinek cyklu.

Jacek WOJTAŁA

Tajny zapis

Janusz MILLER

Program wywołuje etykieta „X" (RU ˈ0). Podajemy *liczbę liter* tajnego zaı u.

W tym momencie rusza generat. czb losowych, który sam — bez n: zej wiedzy i woli — kolejno typuje ty ter, ile podaliśmy.

Tak powstaje *tajny zapis*, który z r :uły nie ma żadnej sensownej treści rzypomina kryptogram szyfru.

Tajny zapis odczytujemy następują o

— najpierw typujemy literę z zakr ı od A do Z.

— przypuśćmy, że wytypowaliśmy li- ·erę „M".

— teraz komputer udziela nam vskazówki, iż tajna litera jest *„przed li- erą M" lub „za literą M"*.

— tak kolejno zawężamy pole po- zukiwań.

Gdy (po którymś pytaniu-wskazów- e) odgadniemy, to wtedy pojawia się meldunek *„zgadłeś 1 literę"*

W ten sposób, kolejno typując, od- :adujemy wszystkie litery tajnego do- umentu, co potwierdza meldunek. *"ZYFR ="* i tu jest wymieniony ciąg *ajnego zapisu"*.

Nasz program pozwala na zadanie 0 pytań, aby odgadnąć jedną literę.

Uwaga, nie znamy „metody na szyfr". ·roponujemy zaś *metodę podziału*. Oto na:

— całość (od A do Z) podziel na po- wę i wytypuj literę krańcową podziału,

— wskazaną przez komputer „po- wkę" znów podziel na połowę..

Teoria mówi iż *wystarczy tylko 5 py- ń*.

Czy to prawda?

SHARP

JANUSZ MILLER
SHARP 1500-PC
TAJNY ZAPIS

```
 10:"X":RANDOM :
    CLEAR :CLS :
    USING :WAIT 0
 20:INPUT "ILE LIT
    ER SZYFRU N=";
    N
 30:IF N<1OR N>16
    THEN BEEP 10,2
    5,50:PRINT "1<
    N<16":GOTO 20
 40:FOR K=1TO N
 50:A=(RND 26)+64:
    P$=P$+CHR$ A
 60:NEXT K:M=N:Y$=
    P$:Y=ASC Y$:A$
    ="OD A DO Z":
    PRINT A$;
 70:IF Q>10GOTO 18
    0

 80:INPUT ".TYPUJE
    :";X$
 90:CLS :X=ASC X$
100:IF X<YLET D$="
    ZA LITERA "
110:IF X>YLET D$="
    PRZED LITERA "
120:IF X=YLET L=L+
    1:GOTO 140
130:PRINT D$;X$;:Q
    =Q+1:GOTO 70
140:PRINT "ZGADLES
    ";L;" LITERE":
    BEEP 3,100,500
150:Z$=Z$+X$:M=M-1
    :IF Z$=P$GOTO
    170
160:Y$=RIGHT$ (Y$,
    M):Y=ASC Y$:Q=
    0:PRINT A$;:
    GOTO 80
170:WAIT :PRINT "S
     Z Y F R =";Z$
    :GOTO 190
180:CLS :WAIT 0:
    PRINT "A . L  A
       R  M":BEEP 7
    ,65,1000
190:INPUT "OD NOWA
     ? T=1 N=0 ";D
200:IF D=1GOTO 10
210:END
```

ZX SPECTRUM

```
  9 REM tajny zapis
 10 LET q=1: DIM p$(16)
 12 DIM z$(16): DIM y$(16)
 15 LET l=0
 20 INPUT "podaj liczbe liter szyfru n =";n
 30 IF n<1 OR n>16 THEN  GO TO 20
 40 FOR k=1 TO n
 42 LET a=INT (RND*26)+64
 44 LET p$(k)=CHR$ a
 46 NEXT k
 60 LET y$(1 TO 16)=p$(1 TO 16)
 61 LET k=1
 62 LET y=CODE y$(k)
 64 LET a$=" od A do Z": PRINT a$

 70 IF q>10 THEN  GO TO 180
 80 INPUT "Typuje ";x$
 90 LET x=CODE x$
100 IF x<y THEN  LET d$="za litera "
102 FOR n=1 TO 5: BEEP .1,7-n: NEXT n
110 IF x>y THEN  LET d$="przed litera "
112 FOR n=1 TO 5: BEEP .1,n: NEXT n
120 IF x=y THEN  LET l=l+1: GO TO 140
130 PRINT d$;x$: LET q=q+1: GO TO 70
140 PRINT "zgadles ";l;" litere": LET k=k+1
141 FOR n=1 TO 7: BEEP .2,n: NEXT n
150 LET z$(k-1)=x$
155 IF z$(1 TO 16)=p$(1 TO 16) THEN  GO TO 170
160 LET y=CODE y$(k): LET q=0
161 PRINT a$: GO TO 80
170 PRINT " szyfr = ";z$(1 TO 16): GO TO 190
180 PRINT " alarm "
185 BEEP 2,1: BEEP 2,2
190 INPUT "od nowa (tak/nie) ";w$
192 IF w$="tak" THEN  CLS : GO TO 10
210 STOP
```

PROGRAM 17

ATARI

Program kopiujący

Program służy do kopiowania gier i innych programów o pojemności do 36 kB, nagranych na kasecie.

Po uruchomieniu (RUN) naciskamy O (odczyt). Gdy usłyszymy pojedynczy sygnał dźwiękowy, przygotowujemy kasetę w magnetofonie oraz wciskamy RETURN. Jeśli kopiowany program składa się z kilku części, można wgrywać je kolejno, ale należy uważać, by nie przekroczyć zawartości wolnej pamięci. Po wgraniu programu otrzymuje się na ekranie wartość: pojemność kopiowanego programu oraz pozostałość wolnej pamięci RAM. Chcąc zapisać program na innej kasecie, należy przygotować magnetofon do nagrywania, nacisnąć Z (zapis), a następnie numer kopiowanej części.

Gdy wystąpi błąd w czasie wprowadzania, program zostaje zatrzymany, a na ekranie otrzymuje się potwierdzenie o wystąpieniu błędu.

Stanisław JANKOWSKI

```
ZI 100 DIM C$(7),D(9):GRAPHICS 0:PO
   KE 709,14:POKE 710,0:POKE 752,1
UD 110 C$(1,1)=CHR$(104):C$(2,2)=CH
   R$(104):C$(3,3)=CHR$(104):C$(4,4
   )=CHR$(170):C$(5,5)=CHR$(76)
GE 111 C$(6,6)=CHR$(86):C$(7,7)=CHR
   $(228):C=ADR(C$):D(0)=3994
EU 120 AD=868:CO=866:LE=872:D=39994
   :M=0:FOR I=1 TO 9:D(I)=0:NEXT I
ZQ 130 POSITION 1,0:? " ***  Kopiow
   anie programow i gier  *** ":POS
   ITION 21,23:? "wolne        bajto
   w";
LU 140 POSITION 27,23:? D-D(M);:OPE
   N #1,4,0,"K"
HF 160 GET #1,L
MP 170 IF L=79 THEN FLAG=0:M=M+1:PO
   SITION 2,M*2+2:? M:POSITION 5,M*
   2+2:? ": ... ";:POSITION 17,2:?
   "Odczyt":GOTO 200
NE 180 IF L=90 THEN 500
AU 181 IF L=77 THEN POKE 54018,52
LZ 182 IF L=32 THEN POKE 54018,0
HE 184 IF L=75 THEN RUN
OM 190 GOTO 160
IQ 200 OPEN #2,4+FLAG,128,"C:":POSI
   TION 7+2*FLAG,2+2*M:? "   "
AQ 220 L=D(M-1)+1:H=INT(L/256):L=L-
   H*256:POKE AD,L:POKE AD+1,H
JE 230 L=(FLAG=0)*D+(FLAG=4)*D(M)-D
   (M-1):H=INT(L/256):L=L-H*256:POK
   E LE,L:POKE LE+1,H
VM 240 POKE CO,7+FLAG
UL 250 L=USR(C,32):E=PEEK(867)
LW 260 CLOSE #2
ES 270 IF FLAG=4 THEN POSITION 17,2
   *M+2:? " ":M=N:POSITION 17,2:? "
        ":GOTO 160
KZ 280 IF E<>136 THEN POSITION 2,2+
   2*M:? "            ":POSITION 17,2:
   ? "Pomylka !":M=M-1:GOTO 160
JK 290 D(M)=D(M-1)+PEEK(LE)+256*PEE
   K(LE+1)
ZX 300 POSITION 7,2+2*M:? D(M)-D(M-
   1):POSITION 27,23:? "      ";CHR
   $(30);CHR$(30);CHR$(30);CHR$(30)
   ;CHR$(30);
EC 301 ? CHR$(30);D-D(M);:GOTO 160
EN 500 POSITION 17,2:? "Zapis "
HU 505 GET #1,L:IF L<49 OR L>48+M T
   HEN POSITION 17,2:? "       ":GOT
   O 160
PC 510 N=M:M=L-48:FLAG=4:POSITION 1
   7,2+M*2:? "*":GOTO 200
```

Test stacji dysków

Program DRIVETEST umożliwia sprawdzenie prędkości obrotowej stacji dysków ATARI 1050 oraz sygnalizuje ewentualne odchylenie tej prędkości od wartości uznawanej za poprawną. Pomiar dokonywany jest przy wykorzystaniu procedury w kodzie maszynowym (linie DATA). Procedura ta po wywołaniu z programu w języku BASIC zeruje na początku dwa najmniej znaczące bajty zegara systemowego (bajty o adresie $13 i $14), a następnie odczytuje z dyskietki sto razy sektor o numerze jeden. W czasie trwania odczytu zegar systemowy jest zwiększany o jeden co 1/50 sekundy. Po zakończeniu odczytu zawartość bajtów $13 i $14 (jest to czas trwania odczytu stu sektorów o tym samym numerze, inaczej stu obrotów, wyrażonych liczbą impul-

sów generowanych co 1/50 sekundy) przekazywana jest do programu wywołującego procedurę, gdzie następuje obliczenie liczby obrotów stacji na minutę. Jeżeli obliczona liczba obrotów mieści się w przedziale 284<LO<292, to wyświetlany jest komunikat „DRIVE OK". Jeżeli nie, to w zależności od obliczonej wartości program informuje o „obrotach zbyt wolnych" lub „zbyt szybkich".

H. KRASUSKI

```
WO 1 REM ****************************
   ****
HG 2 REM DRIVE TEST
DU 3 REM (C) 1987 H. KRASUSKI
WR 4 REM ****************************
   ****
PC 10 DIM SZYB$(66)
DX 15 GOSUB 200
XM 20 ? CHR$(125)
DK 30 ? "TEST SZYBKOSCI OBROTOW STA
   CJI"
HB 40 ? "          DYSKOW"
QL 41 ? "   (C) 1987 H.KRASUSKI"
FU 50 ? :? :?
QX 60 ? "Podaj numer drive'u (1-2)
   ";:INPUT ND
YQ 70 ? "Prosze chwile poczekac (ok
   . 1 min)..."
XF 100 X=USR(ADR(SZYB$),ND)
BC 110 SZYB=X/3600:SZYB=INT((100/SZ
   YB+0.5)*5/6)
CH 120 ? "Drive ";ND;"  ";SZYB;"  o
   br./min"
BN 130 IF SZYB<292 AND SZYB>284 THE
   N ? "drive OK"
AC 140 IF SZYB<=284 THEN ? "OBROTY
   ZBYT WOLNE"
BZ 150 IF SZYB>=292 THEN ? "OBROTY
   ZBYT SZYBKIE"
DC 160 END
HP 200 RESTORE 250:FOR I=1 TO 66
AU 210 READ A:SZYB$(I)=CHR$(A)
FW 220 NEXT I
ZE 230 RETURN
JM 250 DATA 104,104,104,141,1,3,169
   ,1
WP 251 DATA 141,10,3,169,0,141,11,3
   ,169,0
ZW 252 DATA 141,4,3,169,4,141,5,3,1
   69,82
UU 253 DATA 141,2,3,169,5,32,83,228
   ,198
ZZ 254 DATA 203,208,249,169,100,133
   ,203
PZ 255 DATA 169,0,133,19,133,20,32,
   83,228
DD 256 DATA 198,203,208,249,165,20,
   164,19
FB 257 DATA 133,212,132,213,96
```

PROGRAM 18

Golf

■ ■ ■

Program jest formą zabawy, w której siłę uderzenia piłeczki golfowej należy potraktować jako liczbę. Wprowadza się ją w L. 65 w wartościach od 1 do 20. Uderzenie piłeczki jest sygnalizowane dźwiękiem L. 68 ÷ L. 72, a trafienie do dołka krótką melodią L. 98 do L. 103 oraz L. 500 ÷ L. 515. Aby ukończyć grę, należy pięciokrotnie trafić do dołka. Nie jest to proste, gdyż w zależności od podłoża, po jakim porusza się piłeczka oraz odległości do dołka, należy użyć różnej siły. Po trafieniu do ostatniego dołka, podawana jest suma uderzeń w całej grze.

Fragmenty grafiki w kolejnych wierszach umiejscawia się najczęściej za pomocą instrukcji POSITION. Można (jeśli to ułatwi) wykonać ją w inny sposób, a mianowicie przez zmianę położenia kursora. Dokonuje się tego przez naciśnięcie klawisza ESC, a następnie jednocześnie CONTROL i klawisza ze strzałką skierowaną w stronę, w którą ma być przesunięty kursor. Przesuwane są wszystkie znaki wprowadzone za strzałką do zamykającego je cudzysłowu. W niektórych sytuacjach (np. aby uzyskać czytelny wydruk) można znaki przyporządkować liczbie odpowiadającej im wg kodu ATASCII za pomocą instrukcji CHR$. Ilustracją obu przykładów jest L. 300.

Janusz JANIEC

■ ■ ■

```
8 OPEN #1,4,0,"K:"
9 CC=100
10 C=0
15 Z=0
20 E=INT(RND(1)*120+200)
25 Z=Z+1
27 ? CHR$(125)
30 POSITION 6,3:? "ODLEGLOSC DO
DOLKA Nr ";Z;":";E
35 C=C+1
40 IF E<10 THEN 300
45 FOR I=1 TO E/10
50 POSITION I,14:? "-"
55 NEXT I
60 POSITION 4,5:? "WYBIERZ LICZB
E W ZAKRESIE (1-20)"
65 POSITION 32,14:? ;:INPUT S
```

```
65 IF S<1 OR S>20 THEN 65
68 FOR H=15 TO 0 STEP -1
69 SOUND 0,250,10,H
70 FOR W=1 TO 6
71 NEXT W
72 NEXT H
75 ? CHR$(125)
80 E=E-INT(RND(0)*5*S+S)
85 E=ABS(E)
90 IF E>1 THEN 30
94 GRAPHICS 2+16
95 POSITION 5,4:? #6;"TRAFILES"
97 RESTORE
98 READ H1,H2,T
99 SOUND 0,H1,10,15
100 SOUND 1,H2,10,15
101 FOR CZ=1 TO T
102 NEXT CZ
103 IF T<>2 THEN 98
104 GRAPHICS 0
105 IF Z<5 THEN 20
110 POSITION 3,7:? "ZAKONCZYLES
Z ";C;" UDERZENIAMI PILKI."
115 IF C<CC THEN CC=C
120 POSITION 5,5:? "NAJLEPSZY WY
NIK :";CC;" UDERZEN"
150 POSITION 10,16:? "Czy grasz
jeszcze raz ?"
155 POSITION 22,18:? CHR$(212);"
ak/";CHR$(206);"ie"
160 GET #1,P
165 IF P=84 THEN 10
170 IF P<>84 THEN ? CHR$(125)
175 END
300 POSITION 14,8:? CHR$(2);CHR$
(160);CHR$(160);CHR$(136);CHR$(2
9);CHR$(30);CHR$(30);CHR$(30);CH
R$(30);
301 ? CHR$(2);CHR$(32);CHR$(32);
CHR$(7);CHR$(29);CHR$(30);CHR$(3
0);CHR$(30);CHR$(30);
302 ? CHR$(2);CHR$(13);CHR$(13);
CHR$(13);CHR$(29);CHR$(30);CHR$(
30);CHR$(30);CHR$(30);
303 ? CHR$(2);CHR$(29);CHR$(30);
CHR$(2);CHR$(29);CHR$(30);CHR$(2
);CHR$(29);CHR$(30);
304 ? CHR$(2);CHR$(29);CHR$(31);
CHR$(13)
306 POSITION 13,14:? CHR$(13);CH
R$(13);CHR$(7);CHR$(2);CHR$(13);
CHR$(13);CHR$(13);CHR$(13);CHR$(
13);
307 ? CHR$(13);CHR$(13);CHR$(13)
;CHR$(13)
310 POSITION E+16,13:? CHR$(20)
315 GOSUB 400+Z
320 GOTO 60
401 POSITION 15,8:? CHR$(148):RE
TURN
402 POSITION 15,8:? CHR$(144):RE
TURN
403 POSITION 15,8:? CHR$(224):RE
TURN
404 POSITION 15,8:? CHR$(128):RE
TURN
405 POSITION 15,8:? CHR$(251):RE
TURN
500 DATA 60,45,150,72,53,80
505 DATA 96,72,40,81,60,40
510 DATA 60,45,80,0,0,10,60,45,8
0
515 DATA 72,53,150,0,0,2
```

PROGRAM 19

Pojedynek

■ ■ ■

Jest to prosta, kolorowa gra zręcznościowa dla dwóch osób. Można w nią grać, używając klawiatury bądź joysticka. Gra polega na trafieniu poruszającego się przeciwnika.

Sposób poruszania się oraz oddawania strzałów znajdziecie w wyświetlonym na wstępie opisie gry.

■ ■ ■

AMSTRAD

```
10 REM P O J E D Y N E K
20 DEFINT a-z
30 MODE 0
40 GOSUB 800
50 GOSUB 1140
60 GOSUB 190
70 GOSUB 1270
80 GOSUB 1140
90 GOSUB 1050
100 REM start
110 IF koniec THEN GOTO 80
120 GOSUB 180
130 FRAME:IF zmie1 THEN GOSUB 480
    ELSE FRAME:FRAME
140 FRAME:IF zmie2 THEN GOSUB 510
    ELSE FRAME:FRAME
```

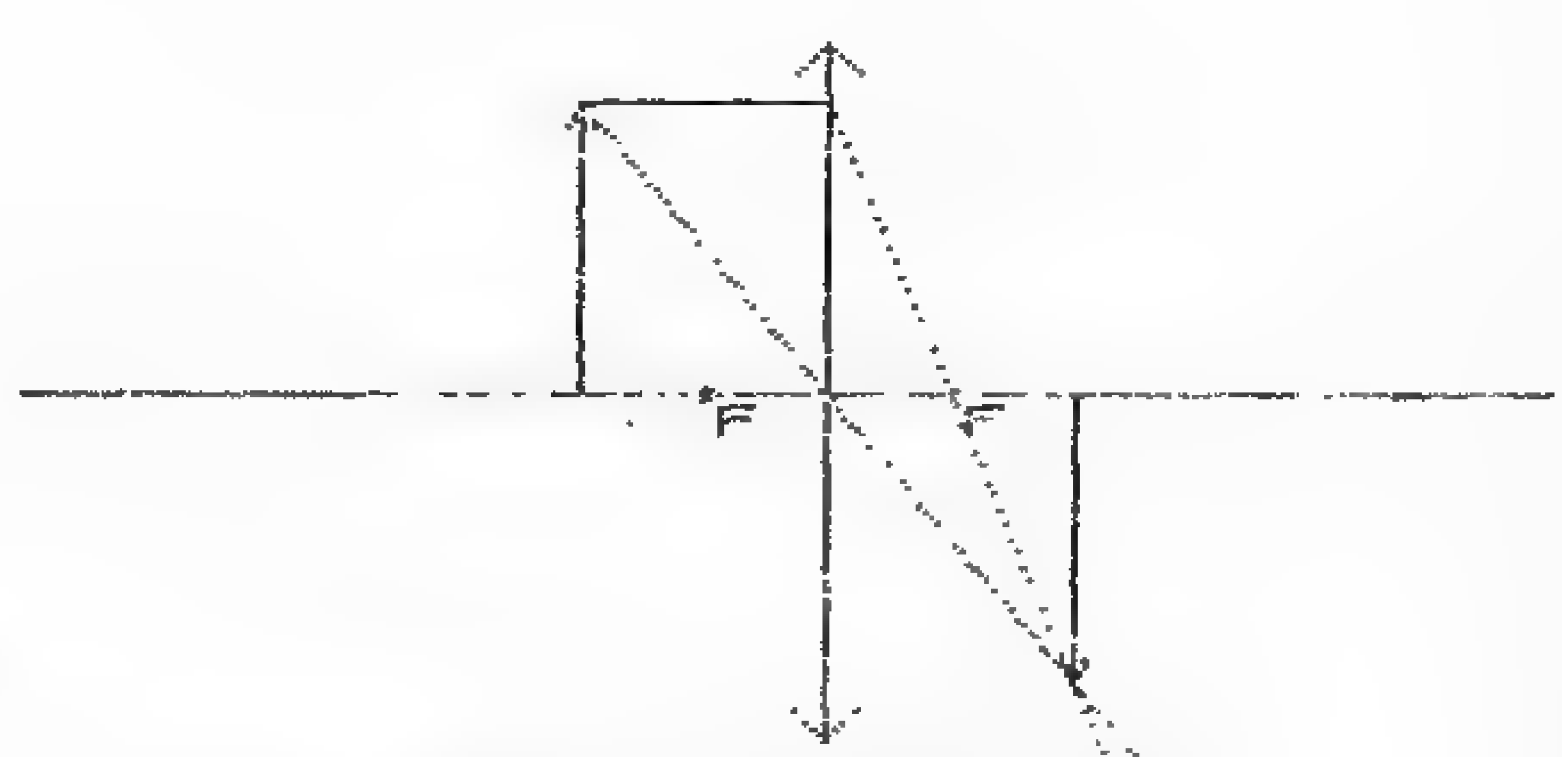
```
150 IF p1sa=-1 THEN GOSUB 540
160 IF p2sa=-1 THEN GOSUB 570
170 GOTO 100
180 IF j THEN 330 ELSE 410
190 CLS:PEN 6
200 PRINT:PRINT" nacisnij klawisze"
210 PRINT:PRINT:PRINT
220 PRINT " joystick   = 'J'"
230 PRINT:PRINT
240 PRINT " klawiatura = 'K'"
250 PRINT:PRINT:PRINT
260 PRINT " oraz [RETURN]"
270 LOCATE 4,10
280 LOCATE 12,10:IF j THEN PRINT"*":
    ELSE PRINT " "
290 LOCATE 12,11:IF j THEN PRINT" ":
    ELSE PRINT "*"
300 IF NOT(INKEY(45)) THEN j=-1
310 IF NOT(INKEY(37)) THEN j=0
320 IF NOT(INKEY(18)) THEN RETURN
    ELSE 280
330 p1=JOY(0):p2=JOY(1)
340 zmie1=(p1 AND 1)*-1+(p1 AND 2)*0.5
350 zmie2=(p2 AND 1)*-1+(p2 AND 2)*0.5
360 IF p1 AND 16 THEN p1sa=p1sa-1:
    IF p1sa=-1 THEN AFTER 15 GOSUB 600
370 IF p2 AND 16 THEN p2sa=p2sa-1:
    IF p2sa=-1 THEN AFTER 15 GOSUB 600
380 IF p1sa THEN zmie1=0
390 IF p2sa THEN zmie2=0
400 RETURN
410 zmie2=((INKEY(4)=0)*1)+
    ((INKEY(5)=0)*-1)
420 zmie1=((INKEY(69)=0)*1)+
    ((INKEY(71)=0)*-1)
430 IF INKEY(63)=0 THEN p1sa=p1sa-1:
    IF p1sa=-1 THEN AFTER 15 GOSUB 600
440 IF INKEY(10)=0 THEN p2sa=p2sa-1:
    IF p2sa=-1 THEN AFTER 15 GOSUB 600
450 IF p1sa THEN zmie1=0
460 IF p2sa THEN zmie2=0
470 RETURN
480 pt=p1wp+zmie1:IF pt>25 OR pt<6
    THEN RETURN ELSE p1wp=pt
490 zmie1=0
500 PEN 1:LOCATE 3,p1wp:CLS #3:PRINT
    CHR$(209);:RETURN
510 pt=p2wp+zmie2:IF pt>25 OR pt<6
    THEN RETURN ELSE p2wp=pt
520 zmie2=0
530 PEN 2:LOCATE 18,p2wp:CLS #5:PRINT
    CHR$(211);:RETURN
540 PAPER #4,4:WINDOW #4,4,17,p1wp,p1wp:
    CLS#4:FRAME:FRAME
550 PAPER #4,0:CLS #4
560 GOTO 480
570 PAPER #6,5:WINDOW #6,4,17,p2wp,p2wp:
    CLS#6:FRAME:FRAME
580 PAPER #6,0:CLS#6
590 GOTO 510
600 pwpe=(p1wp=p2wp)
610 IF p1sa AND NOT(p2sa) AND pwpe THEN
   p1sc=p1sc+1:SOUND 132,120,10,0,1,0:
   PRINT#1,a$(p1sc);:IF p1sc=9 THEN 680
620 IF p2sa AND NOT(p1sa) AND pwpe THEN
   p2sc=p2sc+1:SOUND 132,100,10,0,1,0:
   PRINT#2,a$(p2sc);:IF p2sc=9 THEN 680
630 IF p1sa THEN SOUND 132,40,70,0,1,1
640 IF p2sa THEN SOUND 132,56,70,0,1,1
650 p1sa=0
660 p2sa=0
670 RETURN
680 PEN 6
690 LOCATE 6,10:PRINT"KONIEC GRY"
700 IF p1sc=9 THEN INK 1,2,20
710 INK 2,0 ELSE INK 2,6,17:INK 1,0
720 SOUND 129,1000,0,12,3
730 SOUND 130,900,0,12,3
740 WHILE INKEY$<>"":WEND
750 t!=TIME:WHILE t!+2000>TIME:WEND
760 WHILE INKEY$="":WEND
770 CLS
780 koniec=-1
790 RETURN
800 a$(0)="111101101101111"
810 a$(1)="001001001001001"
820 a$(2)="111001111100111"
830 a$(3)="111001111001111"
840 a$(4)="100100101111001"
850 a$(5)="111100111001111"
860 a$(6)="111100111101111"
870 a$(7)="111001001010010"
880 a$(8)="111101111101111"
890 a$(9)="111101111001001"
900 FOR n=0 TO 9
910 dlug=LEN(a$(n))
920 FOR n2=1 TO dlug
930 IF MID$(a$(n),n2,1)="1"THEN
    MID$(a$(n),n2,1)=CHR$(143)ELSE
    MID$(a$(n),n2,1)=CHR$(32)
940 NEXT n2,n
950 b$="P O J E D Y N E K"
960 c$=CHR$(32)+"      "+CHR$(164)+
   "  CJS"
970 ENV 1,=9,2000:ENT -1,6,3,1
980 ENV 2,127,0,0,127,0,0,127,0,0,
    127,0,0,127,0,0
990 ENV 3,=9,9000
1000 BORDER 0
1010 INK 0,12:PEN #4,1:PEN #6,2
1020 PEN #1,1:PEN #2,2:PAPER #1,3
1030 PAPER #2,3:PEN #0,6
1040 RETURN
1050 INK 0,12:INK 1,2:INK 2,6:INK 3,13:
     INK 4,20:INK 5,17:INK 6,20
1060 WINDOW #3,3,3,6,25:WINDOW #5,18,18,
     6,25
1070 WINDOW #1,3,5,1,5:WINDOW #2,16,18,
     1,5:WINDOW #7,1,20,1,5:PAPER #7,3
1080 CLS:CLS#7:PRINT#1,a$(0);:PRINT#2,
     a$(0);:p1sc=0:p2sc=0:p1wp=5:p2wp=24:
     zmie1=1:zmie2=1
1090 GOSUB 490:GOSUB 510
1100 SOUND 1,1000,0,12,2
1110 SOUND 2,900,0,12,2
1120 p1sa=0:p2sa=0:koniec=0
1130 RETURN
1140 CLS
1150 PEN 7
1160 FOR n=1 TO LEN(b$)
1170 LOCATE 2+n,10
1180 FOR n2=LEN(b$) TO n STEP -1
1190 PRINT MID$(b$,n2,1)
1200 LOCATE 2+n,10
```

```
1210 SOUND 135,20*n2,5,12,2,1
1220 NEXT n2,n
1230 SOUND 135,100,0,13,3,1,20
1240 PEN 6:PRINT:PRINT:PRINT:PRINT c$
1250 FOR n=1 TO 5000:NEXT
1260 RETURN
1270 IF j THEN RETURN
1280 CLS
1290 LOCATE 6,4
1300 PRINT"STEROWANIE"
1310 PRINT:PRINT
1320 PRINT" GRACZ 1     GRACZ 2
1330 PRINT:PRINT:PRINT
1340 PRINT"    A     gora     6"
1350 PRINT:PRINT
1360 PRINT"    Z     dol      3"
1370 PRINT:PRINT
1380 PRINT"    X     ognia    7"
1390 t!=TIME:WHILE t!+4000>TIME:WEND
1400 LOCATE 7,5
1410 RETURN
```

PROGRAM 20 ZX SPECTRUM

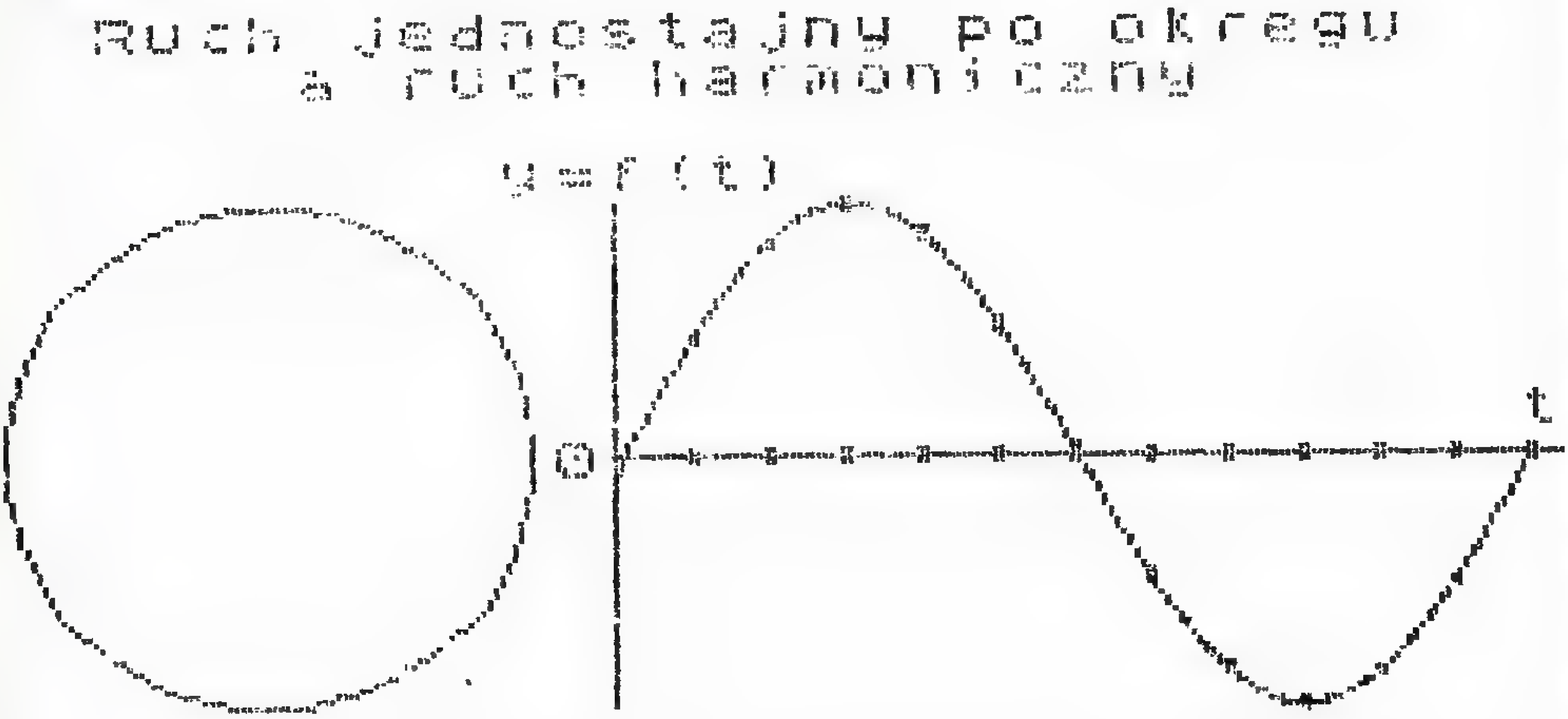
```
  5 REM OBRAZY W SOCZEWCE SKUPIAJACEJ I ROZPRASZAJACEJ, autor JAN GOLLA
 10 INPUT "Podaj ogniskowa soczewki (dla soczewki rozpraszajacej przyjmij f<0)f
="; f: INPUT "Podaj wysokosc przedmiotu (przyjmij h<60) h=";h: INPUT "Podaj odleg
losc przedmiotu od soczewki (przyjmij takie wartosci,by rysunek zmiescil sie na
ekranie)x=";x
 15 LET y=f*x/(x-f): LET h1=f*h/(x-f): REM y-odleglosc obrazu od soczewki,h1-wy
sokosc obrazu
 20 PRINT "f=";f;",";"h=";h;",";"x=";x;",";"y=";INT (y*10)/10;",";"h1=";INT (h1
*10)/10
 30 PLOT 0,80: DRAW 255,0
 40 PLOT 130,20: DRAW 0,120: CIRCLE 130-f,80,1: CIRCLE 130+f,80,1
 42 IF f<0 THEN  GO TO 250
 43 PLOT 125,135: DRAW 5,5: DRAW 5,-5: PLOT 125,25: DRAW 5,-5: DRAW 5,5
 45 PRINT AT 12,32/255*(130-f);"F": PRINT AT 12,32/255*(130+f);"F"
 50 PLOT 130-X,80: DRAW 0,h
 55 PLOT 128-x,76+h: DRAW 2,4: DRAW 2,-4
 60 PAUSE 100
 70 FOR u=130-x TO 130
 75 LET v1=80+h+h/x*(130-x-u)
 80 PLOT u,80+h: PLOT u,v1
 90 NEXT u
100 FOR u=130 TO 250
102 LET v1=80+h+h/x*(130-x-u): LET v2=80+h+h/f*(130-u)
105 IF v1<=0 THEN  GO TO 140
106 IF v2<=0 THEN  GO TO 125
107 IF v2>=160 THEN  GO TO 140
110 PLOT u,v1: PLOT u,v2
120 NEXT u
121 IF f<0 OR x<f THEN  GO TO 140
125 PLOT 130+y,80: DRAW 0,-h1
130 PLOT 128+y,84-h1: DRAW 2,-4: DRAW 2,4
135 STOP
140 FOR u=130-x TO 130+y STEP -3
145 LET v1=80+h+h/x*(130-x-u)
150 PLOT u,v1
160 NEXT u
170 FOR u=130 TO 130+y STEP -3
175 LET v2=80+h+h/f*(130-u)
180 PLOT u,v2
190 NEXT u
200 FOR t=80 TO 80-h1 STEP 3
201 PLOT 130+y,t
202 NEXT t
210 PLOT 128+y,76-h1: DRAW 2,4: DRAW 2,-4
240 STOP
250 PLOT 125,145: DRAW 5,-5: DRAW 5,5: PLOT 125,15: DRAW 5,5: DRAW 5,-5
260 GO TO 45
```

f=20,h=50,x=40,y=40,h1=50



PROGRAM 21 — ZX SPECTRUM

```
  5 REM RUCH JEDNOSTAJNY PO OKREGU A RUCH HARMONICZNY     autor JAN GOLLA
 10 PRINT AT 1,3;"Ruch jednostajny po okregu": PRINT AT 2,7;"a ruch harmoniczny"
 15 PAUSE 150
 20 DIM x(300): DIM y(300)
 30 CIRCLE 51,92,42
 40 FOR n=0 TO 11
 50 PRINT AT 10-6*COS (-n/6*PI+1/2*PI),6+6*SIN (-n/6*PI+1/2*PI);n
 60 PAUSE 100
 61 FOR n=106 TO 255
 62 PLOT n,92
 63 NEXT n
 70 PLOT 106,50: DRAW 0,84
 80 FOR n=118 TO 255
 90 CIRCLE n,92,1
100 LET n=n+11
110 NEXT n
111 PRINT AT 4,11;"y=f(t)"
112 PRINT AT 9,31;"t"
115 PAUSE 100
120 FOR t=0 TO 11
130 LET a=-t/6*PI+1/2*PI: REM a-kat promienia wodzacego w radianach
140 LET sx=42*SIN a: LET sy=42*COS a
150 PLOT 51,92: DRAW  OVER 1;sx,sy: REM wydruk promienia wodzacego
160 LET u=12*t+106
170 LET x(u)=u: LET y(u)=92+42*SIN (0.0439*u+33)
180 CIRCLE x(u),y(u),1
185 PLOT 51,92: DRAW  OVER 1;sx,sy: REM kasowanie promienia wodzacego
190 PAUSE 50
200 NEXT t
210 FOR t=106 TO 250 STEP 0.5
690 LET x(t)=t: LET y(t)=92+42*SIN (0.0439*t+33)
700 PLOT x(t),y(t)
710 NEXT t
```



Liga Myślących

Zadanie 1

W czasie eliminacji do konkursu należało odpowiedzieć na 30 pytań. Za każdą poprawną odpowiedź przyznawano 7 punktów, a za błędną odpowiedź lub jej brak odejmowano 12 punktów. Na ile pytań odpowiedział prawidłowo uczestnik eliminacji, uzyskując 77 punktów?

Zadanie 2

Gdyby Aleksander Wielki umarł o 5 lat wcześniej, panowałby 1/4 swego życia, gdyby zaś żył o 9 lat dłużej panowałby połowę swego życia. Ile lat żył, a ile panował Aleksander Wielki?

Zadanie 3

Statek ładowano za pomocą 3 dźwigów o tej samej mocy. Po jednej godzinie pracy uruchomiono dodatkowo 3 jednakowe dźwigi o większej mocy i ukończono załadunek statku po 2 godzinach wspólnej pracy wszystkich dźwigów. Gdyby uruchomiono wszystkie dźwigi jednocześnie, to załadunek statku trwałby tylko 2 godziny 24 minuty. Należy obliczyć, w ciągu ilu godzin załadowałby statek jeden dźwig o mniejszej mocy, a w ciągu ilu godzin załadowałby statek jeden dźwig o większej mocy?

Zadanie 4

Dwa samochody wyruszyły jednocześnie z miejscowości A do B. Oba samochody jechały tą samą trasą, ale z różnymi, chociaż stałymi, średnimi prędkościami, które wyrażają się liczbami naturalnymi. Różnica prędkości samochodów jest liczbą pierwszą. Odległość między A i B wynosi 100 km. Po dwóch godzinach jazdy odległość wolniejszego samochodu, od miejscowości A była pięć razy większa od odległości szybszego samochodu, od miejscowości B.
Z jaką średnią prędkością jechały oba samochody?

Rozwiązania zadań prosimy przysyłać pod adresem redakcji do końca września br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody — niespodzianki.

Ogłoszenia

REGULAMIN

III konkursu na program gry edukacyjnej

1. Cel konkursu.

Biorąc pod uwagę że:
— głównym problemem rozwojowym w Polsce i na świecie jest bariera edukacyjna,
— jednym z najważniejszych wskaźników rozwoju w roku 2000 będzie ilość i jakość programów edukacyjnych,
Stowarzyszenie Mikrokomputerowe „Abakus" proponuje wszystkim zainteresowanym współudział w tworzeniu takich programów.

Celem konkursu jest tworzenie polskiej kultury mikroinformatycznej, pozyskanie sympatyków i członków Stowarzyszenia na terenie Polski, wzbudzenie szerokiego, społecznego zainteresowania zastosowaniem mikrokomputerów do edukacji oraz wzbogacenie klubowej biblioteki programów.

2. Czym jest gra edukacyjna?

Jest to program na mikrokomputer, który bawi i uczy. Ważniejsza jest jednak w naszym konkursie strona zabawowa gry edukacyjnej, gdyż uważamy, że motywacja ma podstawowe znaczenie w procesie nauczania.
Z drugiej strony kontakt z komputerem powinien być również maksymalnie pożyteczny i kształcący, tzn.: doskonalić umysł człowieka, nauczyć go czegoś nowego, umożliwiać człowiekowi rozwój i doskonalenie się etc.

3. Organizatorzy.

Organizatorem konkursu jest Stowarzyszenie Mikrokomputerowe „Abakus" przy współpracy z Centrum Komputerowym „System" oraz Redakcja miesięcznika „IKS"— dodatku „Żołnierza Wolności". Patronat nad konkursem objęło Biuro ds. Młodzieży Urzędu Rady Ministrów.

4. Uczestnicy.

Uczestnikiem konkursu może być każda osoba fizyczna lub prawna oraz ich zespoły.

5. Wymagania stawiane przesyłanym pracom.

Program gry edukacyjnej może być przeznaczony dla osób w dowolnym wieku i materiał nauczania może być dowolny. Powinien być do niego dołączony opis zawierający krótką charakterystykę, instrukcje ładowania, uruchamiania i możliwości wykorzystania.

Program powinien być przeznaczony na jeden z następujących mikrokomputerów: SPECTRUM 48, TIMEX 2048, ATARI 800 XL, ATARI 130XE, SPECTRAVIDEO 738, COMMODORE 64, AMSTRAD 6128, AMSTRAD 1512 i IBM XT. Prace (programy i opis) należy nadsyłać do 15 września 1987 r. na adres Stowarzyszenia Mikrokomputerowego „Abakus", Warszawa, ul. Senatorska 12

Prezentowany program pozwala łączyć przyjemne z pożytecznym. Grający ma okazję dobrze poznać klawiaturę MERITUM, a przy okazji również się zabawić. Część akcyjna gry została napisana w BASICU, a dźwięki wywoływane instrukcjami GET, PUT, OPEN w języku wewnętrznym mikroprocesora Z-80.

Ulepszenie gry nową grafiką i dźwiękami pozostawiam Czytelnikom.

ADAM CYGAN
Uczeń Zasadniczej Szkoły Chemiczno-Mechanicznej w Gliwicach

Giełda pomysłów

MERITUM

```
1 ' GRA ZRECZNOSCIOWA "REFLEKS" . AUTOR ADAM CYGAN.
 5 ' GRA POLEGAJACA NA NISZCZENIU BOMB SYMBOLIZOWANYCH
6 ' ZNAKAMI ZA POMOCA ODPOWIEDNICH KLAWISZY.

10 ' GDY ZNISZCZONE ZOSTANA REAKTORY PRZEGRYWASZ
20 CLEAR 1000:CLS:GOSUB 1100:GOSUB 800
25 FOR A=980 TO 992 STEP 4:C=C+1:K(C)=A:NEXT
30 FOR D=1 TO 4:POKE 15360+K(D),127:NEXT
35 V=V+1:L=0
40 FOR M=148 TO 160 STEP 4:POKE 15360+M,128:NEXT
45 FOR C=1 TO 4:GOSUB 500:NEXT
70 DW=0:FOR R=212 TO 224 STEP 4:DW=DW+1:V(DW)=R+15360:NEXT R
80 FOR E=1 TO 4
85 IF T(E)=0 POKE V(E)+S(E)*64,A(E):POKE V(E)+S(E)*64-64,32
87 NEXT
88 C$="":C$=CHR$(PEEK(16537))
90 FOR C=1 TO 4
100 IF C$=A$(C) THEN A$(C)="":GOSUB 500
110 NEXT
130 FOR C=1 TO 4
140 IF PEEK(V(C)+(S(C)+1)*64)<>32 GOSUB 400
150 NEXT
160 FOR F=1 TO 4:S(F)=S(F)+1:NEXT
165 FOR A=1 TO AC(V):NEXT
170 GOTO 80
400 IF PEEK(V(C)+(S(C)+1)*64)=127 GOTO 450
410 IF PEEK(V(C)+(S(C)+1)*64)=128 THEN S(C)=0:RETURN
420 POKE V(C)+(S(C)+1)*64,32:GOSUB 500
430 RETURN
450 GET:PUT:OPEN:OPEN:PUT:GET
460 POKE V(C)+(S(C)+1)*64,128:T1=T1+1:T(C)=1
470 IF T1=4 GOTO 1300
475 S(C)=0:GOSUB 500:RETURN
480 GOSUB 500:RETURN
500 IF L=W(V) GOTO 900
510 RANDOM:A(C)=RND(RL)+MM:A$(C)=CHR$(A(C)):L=L+1
518 POKE V(C)+S(C)*64,32:S(C)=0:PUT:RETURN
800 FOR D=1 TO 10:READ Q,P:FOR X=Q TO P
810 POKE 15360+X,191:NEXT X,D
820 FOR D=1 TO 16:READ Q,P,R:FOR X=Q TO P
830 SET(X,R):NEXT X,D
840 FOR D=0 TO 61:READ B:POKE 32000+D,B:NEXT
845 POKE 16762,0:POKE 16763,125:POKE 16732,11:POKE 16733,125
850 POKE 16768,22:POKE 16769,125:POKE 16771,43:POKE 16772,125
855 RETURN
900 IF V=5 GOTO 1200
910 RESTORE:FOR D=1 TO 20:READ M:NEXT
920 FOR D=0 TO 2:PRINT @D*64,STRING$(64,32);:NEXT
930 FOR D=1 TO 16:READ Q,P,R:FOR X=Q TO P
940 SET(X,R):NEXT X,D:GOTO 35
1100 INPUT"PODAJ STOPIEN TRUDNOSCI (1,2 LUB 3)";AD
1110 IF AD<1 OR AD>3 GOTO1100
1120 CLS:AD=INT(AD):ON AD GOTO 1130,1140,1150
1130 FOR K=1 TO 4:W(K)=30+K*10:AC(K)=250-K*40:NEXT
1135 RL=26:MM=64:RETURN
1140 FOR K=1 TO 4:W(K)=40+K*15:AC(K)=200-K*50:NEXT
1145 RL=34:MM=57:RETURN
1150 FOR K=1 TO 4:W(K)=50+K*20:AC(K)=0:NEXT
1155 RL=59:MM=32:RETURN
1200 PRINT @338,"ZDOLALES SIE OBRONIC.BRAWO!":END
1300 PRINT @333,"WOBEC BRAKU ENERGII MUSISZ SIE PODDAC.":END
2000 DATA 770,785,798,819,832,837,848,867,876,889,896
2010 DATA 919,924,939,948,957,960,975,994,1023
2020 DATA 39,42,8,47,50,8,55,58,8,63,66,8,38,71,7,38,73
2025 DATA 6,38,71,5,41,50,4,55,57,4,64,64,4
2030 DATA 41,44,3,52,54,3,58,63,3,40,44,2,40,43,1,40,42,0
2040 DATA62,255,71,211,254,16,254,61,32,248,201,62,0,71
2050 DATA 211,254,16,254,60,32,248,201,6,255,14,253,71,211
2060 DATA 254,16,254,152,71,211,254,16,254,109,254,128,32
2070 DATA 240,201,229,14,225,33,153,64,70,211,254,61,16
2080 DATA 254,211,254,13,32,245,225,201
```

W naszym komputerlandzie

Kierowanie firmą, w której pracował Spektruś, to proces niezwykle skomplikowany, zaistniała potrzeba powołania brakujących ciał. Spektrusiowi powierzono rolę doradcy naczelnego robota. Wprawdzie niechętnie brał on na siebie ten ciężar, nie był jednak jednostką aspołeczną. Zresztą, gdy ogarnął gospodarstwo wzrokiem, doszedł do wniosku, że faktycznie można wiele zrobić.

Gdy doradził, by poprawić robotom warunki pracy — drgnęło — na razie leciutko. Doradził więc, by podziałać motywacją. Naczelny był zdziwiony, że pieniądze mają dla sprawy aż takie znaczenie. — Popatrzcie, kolego, jak drgnęło! — mówił.

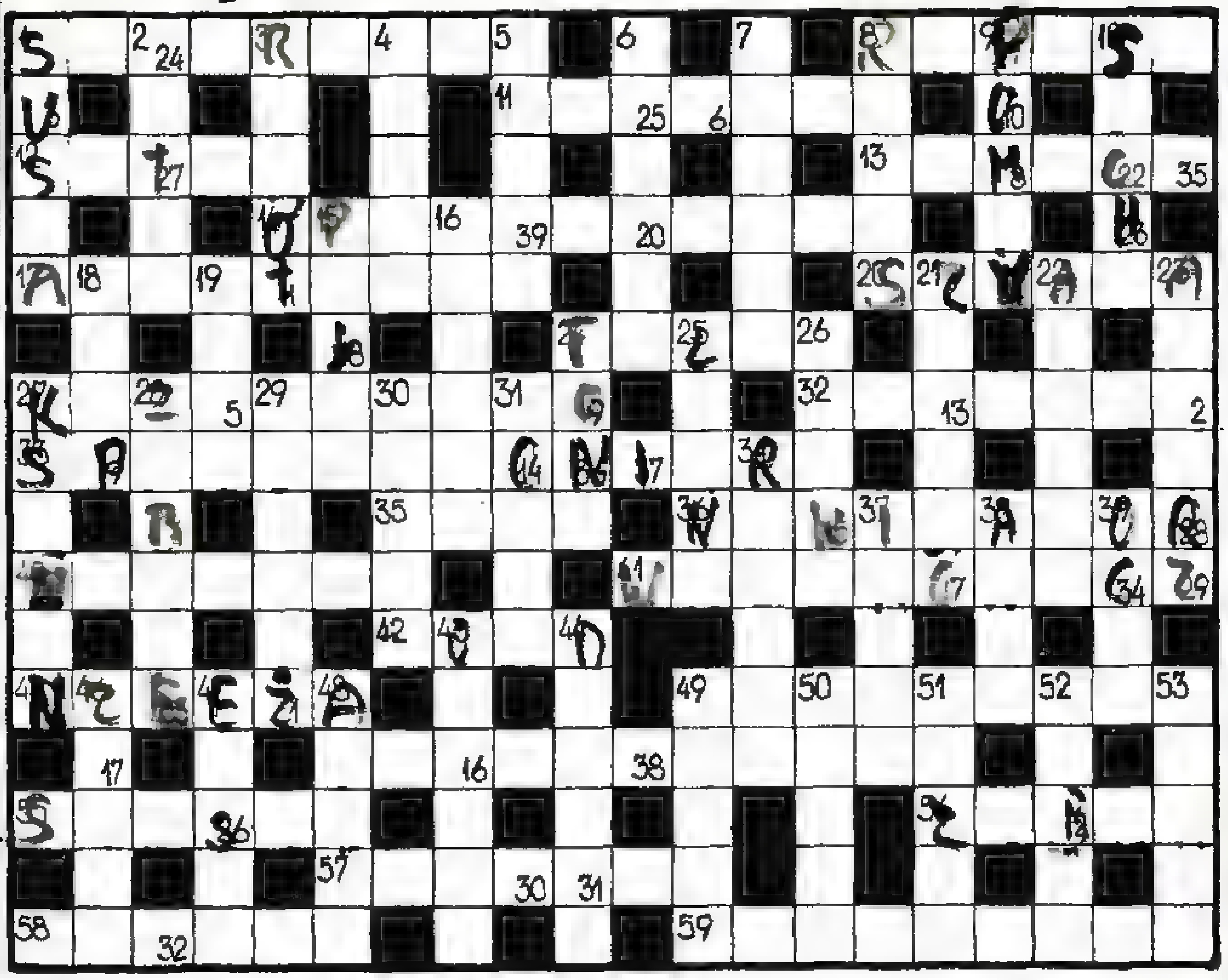
Krzywa na tablicy w gabinecie szefa wznosiła się niemal pionowo, gdy zabrakło surowców: taśm i papieru. Spektruś rozgrzany powodzeniem poradził, by programy drukować na odpadach.

Najtrudniejszy moment jednak miał dopiero nastąpić. Roboty bowiem zaproponowały praktycznie wszystko i gwałtownie spadło zapotrzebowanie na ich pracę. Po jakimś czasie bank poinformował, że firma jest goła.

Nie było wyjścia. Decydując się na gest szaleńca, Spektruś doradził ogłoszenie bankructwa. Dopiero zaczął się ruch w interesie. W największej komputerlandzkiej gazecie ukazał się wywiad z naczelnym, zaś w lokalnej — ze Spektrusiem. Mówili o twardych regułach gry ekonomicznej. Stali się bohaterami. Gazety napisały, że firmie trzeba pomóc, bo bez niej życie byłoby nie do zniesienia. Znalazły się dotacje, były też gratulacje i odznaczenia. Naczelnego przeniesiono do jednostki nadrzędnej, zaś zahartowany w bojach Spektruś awansował do rangi doradcy generalnego.

Podglądał:
Eugeniusz MLECZAK

Krzyżówka nr 7



POZIOMO: 1) metal (np. złoto) znajdywany w ziemi, w stanie rodzimym, 8) dyscyplina sportu dla atlety, 11) cudowny napój poszukiwany przez alchemików, mający przedłużać życie, 12) „mózg armii", 13) tytuł znanego filmu Jerzego Passendorfera, 14) niewielki utwór epicki pisany prozą, 17) kraina historyczna w płd.-zach. Francji (dorzecze Garonny), 20) do zmywania naczyń kuchennych, 24) kupon, kwit upoważniający do zakupu towarów reglamentowanych, 27) zakon krzyżacki, Krzyżacy, 32) słynne wzgórze w centrum Aten z muzealnym zespołem ruin, 33) żołnierz wojsk powietrznodesantowych, 35) wskazówka, 36) członek załogi statku wodnego wyznaczający jego kurs, v0) roślina z rodziny pierwiosnkowatych o kwiatach białych lub różowych, rośnie na skałach w wyższych rejonach górskich, 41) uczeń przebywający samowolnie poza szkołą podczas zajęć szkolnych, 42) zabójstwo, 45) podłużne naczynie drewniane, używane do wyrabiania ciasta, 49) rodzaj opaski, 54) człowiek przebywający gdzieś stale albo czasowo, 55) metoda, 56) pręgowany padlinożerny ssak drapieżny, 57) alkoholowy napój japoński z ryżu; sake, 58) akt prawny ustalany przez Sejm, 59) robotnik pracujący w tkalni przy krosnach.

PIONOWO: 1) długi okres bez deszczu, 2) drewienko do mieszania substancji sypkich i płynów, 3) automat zastępujący pracę człowieka, 4) dzieło ludwisarza, 5) kraj afrykański z Nairobi, 6) erudycja, 7) rzeka we Włoszech (w Jugosławii — Socza), 8) ścięta część pędu drzewa liściastego, 9) modne niegdyś sportowe spodnie męskie o krótkich nogawkach zapinanych pod kolanami, 10) dawne prymitywne narzędzie orne, 15) królewska rezydencja, 16) sztuczne włókno celulozowe, 18) sazan lub lustrzeń, 19) bardzo twardy metal szlachetny, 21) ekstrakt ze zmielonej kawy naturalnej palonej, 22) płn.-amer. gatunek jelenia, 23) torowiec albo szosowiec, 24) 1000 kg, 25) tropikalne pnącze, 26) miano, 27) gaz szlachetny używany do wypełniania żarówek dużej mocy, 28) Górnik ..... — aktualny mistrz kraju w piłce nożnej, 29) żołnierz dopuszczający się gwałtów, grabieży, 30) dawna świątynia pogańska, 31) ciężki ... — czyli deuter, 34) fason płaszcza lub sukni, 37) rozkaz sułtana do wezyra (anagram wyrazu DIERA), 38) członek kocz. ludu tureckiego lub mongolskiego; od VI — IX wieku osiadłego nad Dunajem, 39) haceł, 43) gospoda, karczma, 44) palisander, 46) sytuacja, położenie bez wyjścia, 47) przelanie praw jednej osoby na drugą, 48) aromatyczna wydzielina z kaszalota, 49) tymczasowy budynek, 50) firmament, 51) archit. dolna część głowicy doryckiej i toskańskiej, 52) np. „Aida" — G. Verdiego, 53) paryski opryszek.

Po prawidłowym rozwiązaniu krzyżówki litery z kratek ponumerowanych dodatkowo w prawym dolnym rogu, czytane w kolejności odpowiadających im liczb od 1 do 39, dadzą hasło, które wystarczy nadesłać jako rozwiązanie zadania, pod adresem redakcji do końca września 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

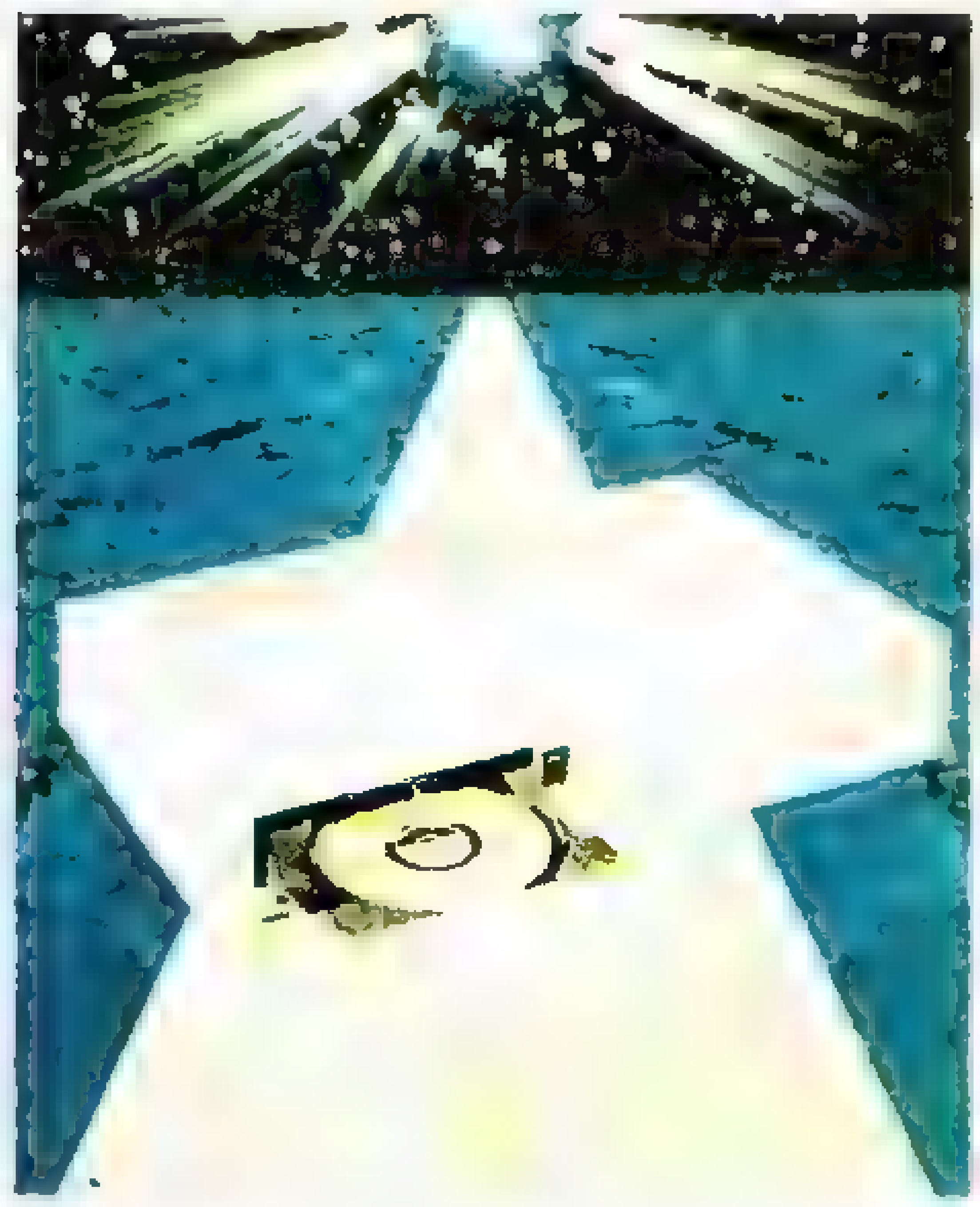
Mikrociekawostki

Laserowa pamięć ROM

HITACHI oferuje lśniący dysk o średnicy 12 centymetrów. Pamięta on 552 megabajty, czyli 270 000 stron, które można przesłać w średniej kopercie.

A oto kolejne zalety: precyzyjny odczyt laserowy z prędkością 176 000 znaków na sekundę oraz ... praktyczna niezniszczalność.

Jedną wadą jest ROM: raz zapisany dysk można TYLKO ODCZYTYWAĆ (czy to na pewno wada?).





— Kto go tak zaprogramował, żeby biegał za nami po całym biurze?

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 9043. Nr. ind. 361682. K-87

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 9/1987 ISSN 0860-27[illegible]



Uwaga! Konkurs:
„45 lat LWP"
Atari dla zwycięzcy
— str. 2

Uczniowie rozpoczęli kolejny rok trudnej pracy. Dla wielu jest to również kolejny rok nauki z komputerem. Niestety, w niewielu przypadkach mikrokomputer jest w szkole rzeczywistą pomocą — najczęściej jedynie w pracowni informatycznej można spotkać to urządzenie. W najlepszej sytuacji są ci, którzy mają go w domu, ale czy takich jest większość?

Niestety, z informatyki uczyniono jedynie przedmiot nauczania i to jak się okazuje, dla wielu bardzo trudny, jakby zapominając zupełnie, że nie chodzi o wychowanie pokolenia informatyków, a o wykształcenie ludzi potrafiących ten nowoczesny sprzęt wykorzystać. Dlaczego zatem mikrokomputer tak rzadko obecny jest na lekcjach fizyki, chemii czy matematyki? Dlaczego najczęściej ważniejsza jest zasada działania mikroprocesora niż umiejętność posługiwania się podstawowymi programami użytkowymi?

Ideałem byłaby sytuacja, kiedy każda specjalistyczna pracownia miałaby swój sprzęt informatyczny i odpowiednie oprogramowanie. Tymczasem brakuje wszystkiego. Nie mamy nadal komputera szkolnego, brak jest odpowiedniego oprogramowania, nie ma również zbyt wielu chętnych nauczycieli do poznawania nowych przecież możliwości.

Dla tych jednak, którzy nie rezygnują, prezentować będziemy na łamach „IKS-a" nasze propozycje programów. Są to programy, których w wielu przypadkach autorami są nauczyciele wykorzystujący je na co dzień w swojej pracy pedagogicznej. Każda nasza propozycja może być wzbogacona dodatkowymi rozwiązaniami — liczą się jednak pomysły i tych będziemy dostarczać.

Tymczasem w nowym roku szkolnym życzymy samych piątek i sukcesów

W NUMERZE:

Komputer w medycynie — str. 4
Niepowodzenia sztucznej inteligencji — str. 5
W szponach Atari (7) — str. 6—11
Emulator BASIC 1.1 dla CPC 464 (1) — str. 12 — 13
Kreska (BASIC CPC 464) — zakończenie — str. 12
Mikrokomputery IBM PC (4) — str. 13—14
Kompilator Hisoft-Pascal — str. 14—15
Komputer u kierowców — str. 17—19
Grafika na ekranie (1) — str. 20—22
Sztuki i sztuczki (10) — str. 23—25
Grafika C-64 (cz. IV) — str. 26—29
Liga Myślących — str. 31

Uwaga! Konkurs: „45 lat LWP"

Wojsko w komputerze

Czy historia, wydarzenia z przeszłości i współczesności dają się zapisać komputerowym kodem? Wiemy, że tak i dlatego proponujemy coś, czego jeszcze nie było!

45 lat temu, w trudnych wojennych czasach powstało ludowe Wojsko Polskie. Powstało by walczyć o wyzwolenie kraju, powstało by w pokojowych warunkach jak najlepiej służyć pomyślności ojczyzny. Dziesiątki wojennych wydarzeń, tysiące dobrych i złych ludzkich losów, wiele dni zapisanych pracą i służbą — przebogata mozaika tworzy 45-letnią historię wojska.

Książka, artykuł, film, opowieść — to źródła naszej wiedzy o niej. Chcemy je jednak wzbogacić. Niech także komputer uczy, zaciekawia i rozbudza historyczną pasję.

Dlatego ogłaszamy nasz nowy konkurs dla miłośników historii i... komputerów. Niech ten — tyleż zaskakujący co interesujący — mariaż zaowocuje dorobkiem, z którego skorzystamy potem wszyscy — w szkole, w domu i w klubie komputerowym.

Proponujemy:

1. Opracowanie scenariusza programu komputerowego, którego treścią będą tradycje i współczesność WP. Wyobraźnia plus wiedza niech wskażą co wybrać: czy epizod z bojowego szlaku, czy wybrany okres wojennych zmagań, czy też współczesny wątek związany z życiem lub działalnością wojska.

Dla autorów najlepszych scenariuszy przeznaczone są nagrody:

I miejsce — ATARI 130XE
II miejsce — 25 tys.
III miejsce — 20 tys.

oraz trzy wyróżnienia po 10 tys.

2. Opracowanie programu komputerowego na podstawie jednego z nagrodzonych scenariuszy lub na własnym, oryginalnym pomyśle.

W tej konkurencji programów komputerowych pierwszą nagrodą jest również mikrokomputer ATARI 130XE, zaś autorzy wyróżniających się programów zostaną uhonorowani sprzętem elektronicznym.

Regulamin konkursu

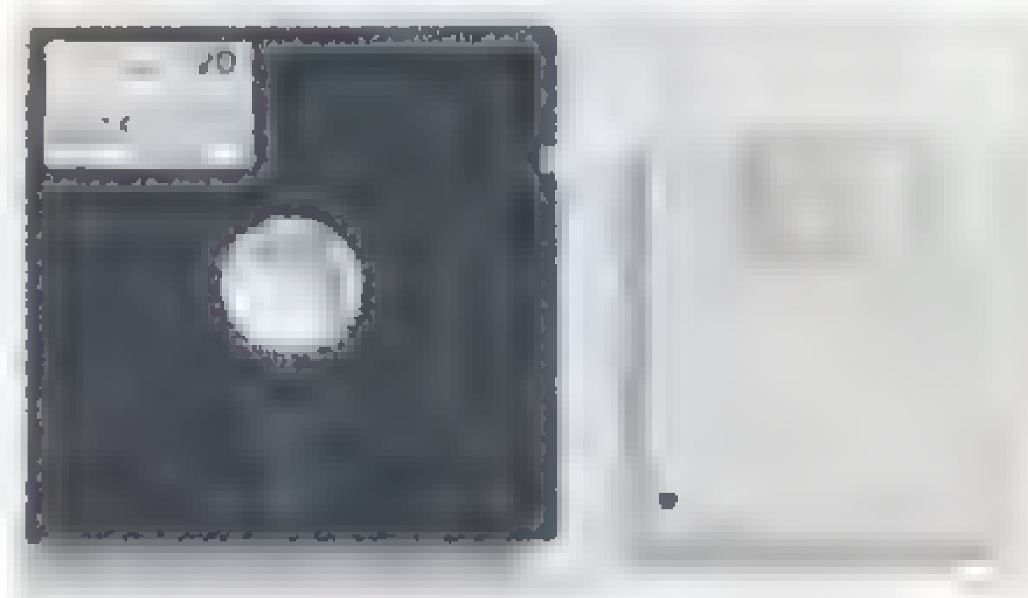
1) Zgłoszone do konkursu prace muszą być oryginalne
2) Termin nadsyłania scenariuszy programów (w trzech egzemplarzach ok. 5 stron maszynopisu) upływa 31.12.1987 roku. Ogłoszenie wyników tego etapu nastąpi w lutym na łamach „Żołnierza Wolności" i „IKS-a".
3) Autorzy programów, którzy podejmą się udziału w konkursie będą mogli otrzymać nagrodzone scenariusze w redakcji od 15 lutego 1988 roku.
4) Oceniane i klasyfikowane będą programy na Atari, Spectrum, Amstrad 464 i 6128 oraz Commodore 64.
5) Programy na kasetach magnetofonowych lub dyskietkach oraz scenariusze programów należy przesyłać pod adresem redakcji 00—950 Warszawa ul. Grzybowska 77.

M * I * K * R * O * S * E * R * W * I * S

Dyskietki 3 1/2 cala dla mikrokomputera

IBM PC/XT i IBM PC/AT

Uzytkownicy komputerów rodziny IBM PC mogą obecnie wykorzystywać programy zapisane na dyskietkach 3 1/2 cala. Umozliwia to większą elastyczność pracy z tymi komputerami.



Napędy dyskowe 3 1/2 cala mogą być dołączone do mikrokomputera IBM PC / XT lub IBM PC/AT, pracującego pod kontrolą systemu operacyjnego DOS 3.20. Transmisja zbiorów i programów uzytkowych między stacjami dysków 5 1/4 '' i dołączoną, zewnętrzną stacją dysków 3 1/2 '' jest tak łatwa, jak wykonanie kopii zastępczej dyskietki (BACKUP). Zastosowanie napędów 3 1/2 cala daje mozliwość łatwego przenoszenia oprogramowania, pomiędzy komputerami uzywającymi dyskietek 5 1/4 i 3 1/5 cala. Dyskietki 3 1/2 ''posiadają dwukrotnie większą pojemność, 720 KB pozwala na przechowywanie ponad 350 stron maszynopisu.

B u l l   M i c r a l   60



Mikrokomupter firmy **B U L L**, *jest kompatybilny z IBM PC/AT. Zastosowano w nim : 16-bitowy mikroprocesor 80286, koprocesor 80287, pracujące z f zegarową 6 / 8 MHz. Na płycie systemowej znajduje się RAM o pojemności 512 KB, którą mozna rozszerzyć maksymalnie do 3 640 KB. Pamięć masową stanowią : stacja dysków elastycznych o pojemności 1.2 MB i dysk stały 40 MB. Firma proponuje cztery typy klawiatury, w tym QWERTY. Karty graficzne umozliwiają : ● tryb alfanumeryczny 25 linii po 80 znaków ; ● tryb graficzny : z kartą monochromatyczną 720 puntków w 348 liniach ; średnią i wysoką rozdzielczość z kartą kolorową czyli 320 punktów w 200 liniach z paletą 16-tu kolorów lub dwoma paletami po 4 kolory. Micral60 współpracuje z dukarkami PRT1910, 1911, 1912. Systemy operacyjne komputera to DOS 3.10 lub CP/M. Jezyki programowania : BASIC (kompilator i interpreter), C, COBOL, FORTRAN, PASCAL.*

□□ ■ □□ M I K R O D E P E S Z E □□ ■ □□

Firma MOTOROLA zamierza w pierwszym kwartale tego roku uruchomić produkcję mikroprocesora typu RISC, który bedzie wykonywał 20 milionów instrukcji na sekundę. Nowy procesor oznaczony zostanie symbolem 78000. ■ □□ ■ □□ ■ *OEM Marketing wprowadziła na rynek Gorącą Kartę, która kontroluje temperaturę wewnątrz komputera. Alarm termiczny włączany jest, gdy temperatura przekracza 50 stopni C. Jaki będzie następny produkt ? Zimna karta ?* ■ □□ ■ □□ ■ *Firma Maxell pracuje nad dyskami 5 1/4 cala o pojemności* **100 MB**. *Dyski o rekordowej pojemności mają pojawić się na rynku już w tym roku.*

opracowali: Jacek Wojtala i Mariusz Jarzebowski

Tomografia a układ krwionośny

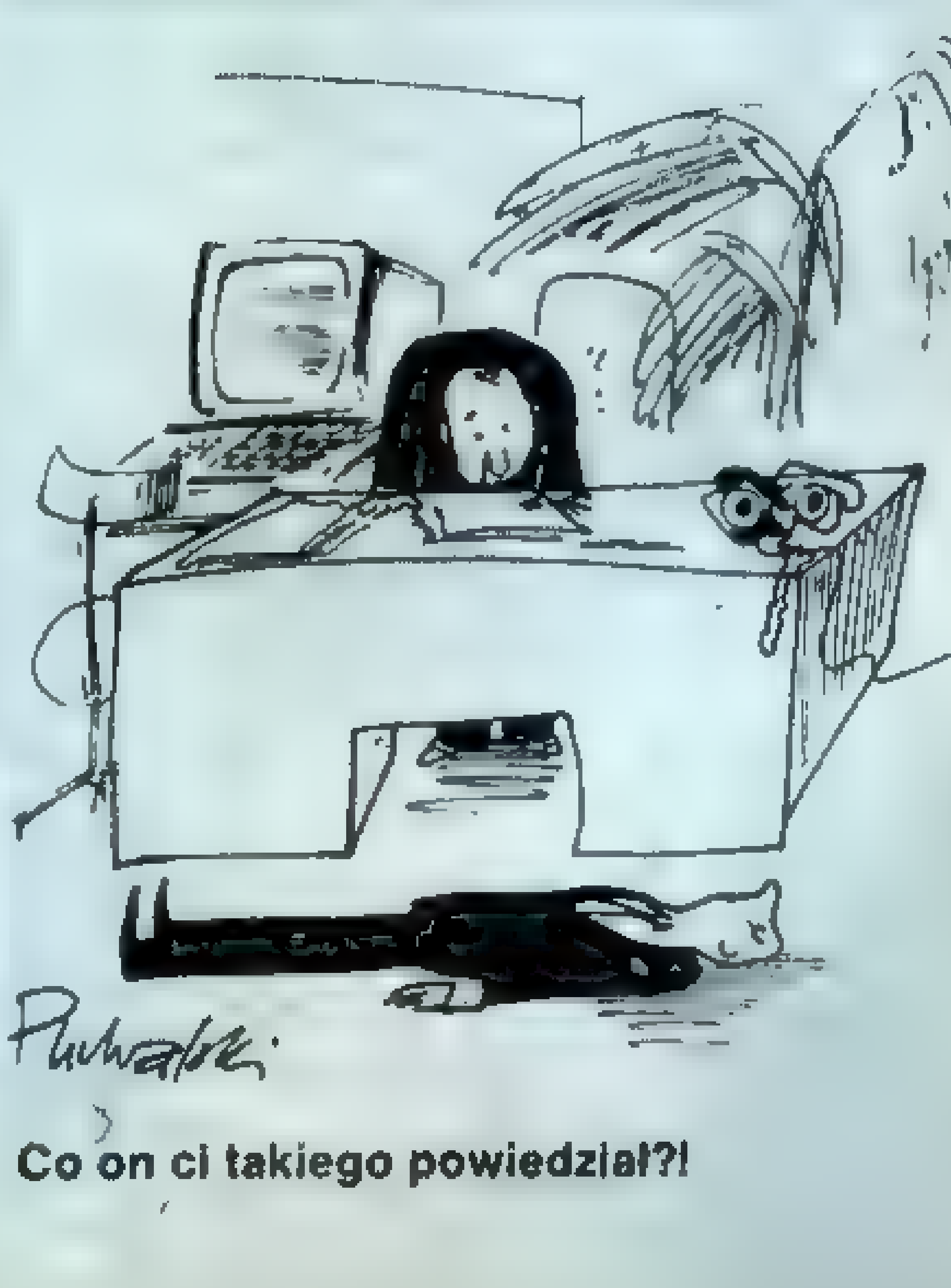
W latach siedemdziesiątych naszego wieku obraz diagnostyki medycznej uległ, za pomocą elektronicznej techniki obliczeniowej, jakościowej zmianie. W oparciu o istniejące już metody bezinwazyjnego badania organizmu ludzkiego np. rentgenografię, jak i o nowe rewelacyjne odkrycia fizyki nuklearnej np. zjawisko magnetycznego rezonansu jądrowego, zbudowano szereg urządzeń obrazujących na monitorze telewizyjnym wnętrze ciała ludzkiego. W poprzednich artykułach poznaliśmy już rentgenowski tomograf komputerowy oraz tomograf magnetycznego rezonansu jądrowego. Pierwszy z nich w oparciu o przetworzone przez komputer dane, uzyskane podczas serii prześwietleń pacjenta promieniami X, umożliwia uzyskanie obrazu wideo dowolnego narządu wewnętrznego pacjenta. Gdy obraz zewnętrzny badanego narządu, uzyskany techniką rentgenowskiej tomografii komputerowej nie odbiega od normy, można zastosować drugi z wymienionych tomografów dla upewnienia się czy komórki budujące dany narząd nie podlegają wczesnym zmianom chorobowym. Badania tomograficzne dają więc dużą gwarancję prawidłowej diagnozy lekarskiej, a także ułatwiają śledzenie postępu choroby w każdym, nawet bardzo wczesnym jej stadium rozwojowym. W przypadku chorób nowotworowych, gdy od szybkości rozpoznania zmian chorobowych zależy pozytywny wynik leczenia, jedynie urządzenia tomograficzne dają szansę stłumienia choroby w zarodku. Zarówno rentgenowski tomograf komputerowy, jak i tomograf magnetycznego rezonansu jądrowego są urządzeniami niesłychanie kosztownymi, a przez to wciąż unikalnymi w normalnej praktyce diagnostycznej.

Oto dwie powszechne dziś metody diagnostyki medycznej, których wspólną cechą jest to, **że są prostymi realizacjami prostych pomysłów**. Celem ich twórców nie było uzyskanie dokładnego, trójwymiarowego obrazu badanego narządu ani zbadanie właściwości struktur atomowych narząd ten budujących. Chcieli oni uzyskać maksimum informacji o systemie zasilającym każdą komórkę ciała ludzkiego, czyli układzie krwionośnym, odgrywającym kluczową rolę w prawidłowym funkcjonowaniu organizmu. Dla uzyskania czytelnego obrazu telewizyjnego naczyń krwionośnych w obu przypadkach użyto znaczników kontrastujących, podawanych dożylnie pacjentowi. Zadaniem ich jest — jak sama nazwa wskazuje, nadanie krwi własności kontrastujących, a więc innych od własności otoczenia. Pierwszy pomysł po ubraniu go w materialne kształty, uzyskał nazwę **tomografu emisji pozytronowej**. W tej metodzie jako znacznika kontrastującego, użyto substancji radioaktywnej, której rozchodzenie się w naczyniach krwionośnych może być śledzone czujnikami promieniowania gamma. Czujniki przekazują wyniki swojej pracy do pamięci komputera, który zamyka cykl badania, porządkując uzyskane dane i dając w efekcie kolorowy obraz wideo naczyń krwionośnych badanego narządu.

Dla dokładniejszej ilustracji tej techniki diagnostycznej opiszemy badanie serca przy użyciu tomografu emisji pozytronowej. Pacjentowi podaje się wówczas dożylnie znacznik radioaktywny w czasie połowicznego rozpadu 10 minut. Znaczy to, że przez pierwsze 10 minut znacznik straci połowę swej aktywności, przez następne 10 minut połowę pozostałej radioaktywności itd. Gwarantuje to pacjentowi bezpieczeństwo, a od lekarza natomiast wymaga szybkości i precyzji w działaniu. Najpierw serce badane jest w stanie normalnej pracy, po czym pobudzane jest do szybkich skurczów specjalnym, łagodnym narkotykiem dozowanym pacjentowi dożylnie. Komputer zapamiętujący informacje o stanie ukrwienia serca w czasie całego badania, jako rezultat obliczeń podaje dwa obrazy naczyń krwionośnych serca: w czasie normalnej pracy i w stanie jego pobudzenia. Porównanie tych obrazów pozwala na wykrycie zaburzeń pracy serca lub zatorów w naczyniach krwionośnych, co jest symptomem zbliżającego się zawału. Dobre efekty, daje użycie tomografu emisji pozytronowej do badania mózgu, umożliwiając wykrycie i w konsekwencji leczenie chorób psychicznych.

Drugi pomysł wizualizacji systemu krwionośnego człowieka zaowocował budową **tomografu odejmującego**. Tomograf ten, mówiąc w dużym uproszczeniu, to aparat rentgenowski sprzężony z komputerem. Badanie polega na wykonaniu serii prześwietleń rutynowych pacjenta, których wyniki rejestrowane są w pamięci komputera wraz z wynikami drugiej serii prześwietleń, lecz tym razem z podaną pacjentowi dożylnie substancją kontrastującą, pochłaniającą promieniowanie X. Komputer porównuje, uzyskane w ten sposób dwa zbiory danych, z których oczywiście drugi jest bogatszy o wielkości charakterystyczne dla układu krwionośnego, uwydatnionego przez zawarty we krwi kontrast. Właśnie dane różniące pierwszy zbiór od drugiego dotyczą układu krwionośnego i są materiałem obrabianym przez komputer, który daje na wyjściu graficzne przedstawienie naczyń krwionośnych badanego narządu. Tomograf odejmujący utorował drogę nowoczesnej technice usuwania zatorów żylnych oraz tamowania krwotoków wewnętrznych. Zabiegi takie, wykonywane dotychczas operacyjnie przez chirurgów, przeprowadzane są obecnie w sposób bezbolesny i nie wymagają długiej rekonwalescencji — przeprowadza się je za pomocą sondy wpuszczonej do tętnicy lub żyły wprawną ręką lekarza. Na ekranie tomografu śledzony jest komputerowy obraz sondy przesuwającej się wzdłuż naczyń krwionośnych do miejsca zatoru lub krwotoku. Następnie — przez sondę podawane są środki rozpuszczające zator lub hamujące przepływ krwi. Zabiegi takie trwają około 90 minut i zwykle kończą się pełnym sukcesem. Fotografia zamieszczona powyżej przedstawia jeden z bardziej wyspecjalizowanych tomografów odejmujących, zapewniający obraz badanego narządu pod dwoma kątami widzenia. Na 4 monitorach umieszczonych przy konsoli tomografu prezentowane są równocześnie: naturalny obraz serca, dwa obrazy opracowane przez komputer oraz dokładna historia choroby. Wyniki te uzyskujemy za naciśnięciem jednego guzika, a lekarzom pozostaje już tylko... leczenie

AT



Co on ci takiego powiedział?!

Niepowodzenia sztucznej inteligencji

W. GOGOŁEK

Od dawna, a ostatnio coraz częściej, informatyczna literatura profesjonalna zwraca uwagę na zagadnienia związane ze sztuczną inteligencją (AI — Artificial Inteligence). Rangę tego problemu podkreślają oczekiwania olbrzymiego postępu wdrożeń tego typu systemów, zarówno przez profesjonalistów, jak i innych użytkowników mikro. Jednak rzeczywistość w dużej części nie dorównuje planom. Przykładem tego jest „interfejs naturalnego języka" do Lotusa 1—2—3. Idzie o to, by móc sterować pracą programu, przekazywać do niego polecenia i pytania stosując codzienny język „mówiony". Omawiany interfejs umożliwia wykorzystanie wszystkich zalet Lotusa — poprzez konwersację. Producent — GNP Development Hal — dopiero po roku od ogłoszenia gotowości dystrybucji „interfejsu" rzeczywiście rozpoczął jego sprzedaż. Opóźnienie to wynikało głównie ze względów na różnice pomiędzy proponowanymi możliwościami programu, a oczekiwaniami użytkowników w zakładach pracy, uczelniach i innych instytucjach. Okazało się bowiem, że pierwotna uniwersalność programu musiała być zastąpiona oprogramowaniem o ściśle określonej specjalizacji. Praktyka jeszcze raz okazała się bezwzględnym sędzią założeń teoretycznych, dowiodła, że zwrot „sztuczna inteligencja" sam siebie „oskarża" — obiecuje bowiem coś magicznego, co faktycznie nie jest możliwe do osiągnięcia. Wydaje się to typowym „problemem armat" — inteligencja nie jest dostatecznie poznana, rozumiana, by ją tworzyć.

Użytkownicy komputerów w zasadzie nie oczekują tego, co rzeczywiście określone jest jako sztuczna inteligencja. Tak jak tłumaczą ją w takich dyscyplinach nauki jak: psychologia, lingwistyka, a także informatyka. Praktycznie AI można opisać jako nieosiągalną zdolność urządzeń elektronicznych.

Zazwyczaj dostęp do sprzętu o możliwościach, które wcześniej oceniane były jako sztuczna inteligencja, zmienia naszą ocenę zdolności tego sprzętu — staje się on nowym „normalnym" produktem. Analizując to zagadnienie przytaczany jest przykład samochodu, który „rozmawia" z kierowcą. Zgodnie z tradycyjnymi wyobrażeniami było to uznawane jako „sztuczna inteligencja". Okazało się jednak, że w tym, jak i wielu innych przypadkach, liczy się nazwa, określenie, a nic ponadto — samochód pozostał samochodem, tyle że z jeszcze jednym urządzeniem stanowiącym o komforcie jazdy.

Zazwyczaj sztuczna inteligencja stanowi uzupełnienie (otoczenie przestrzeni, „muszlę") systemów typu ekspert — **„ekspert-systemów"**. Mamy z nią do czynienia pod postacią interfejsu komunikacji z użytkownikiem w naturalnym języku. Użytkownik może sterować programami używając komend wyrażanych codziennym językiem angielskim; pisząc polecenia na klawiaturze, także wypowiadając je do mikrofonu sprzężonego z komputerem. Coraz częściej pojawiają się także systemy wzbogacone o „komputerowe widzenie" — „okiem" jest odpowiednio sprzężona z maszyną kamera.

Większość prac nad produktami typu AI dla personalnych komputerów dotyczy ekspert-systemów oraz naturalnego języka.

W przyszłym roku spodziewane są na światowym rynku systemy komputerowe, które z powodzeniem będą zastępowały sekretarki — na razie w kwestii pisania pism dyktowanych przez szefa... Natomiast wspomniane „widzenie maszynowe" obecnie praktycznie wykorzystywane jest masowo tylko w przemyśle m.in. samochodowym.

Ekspert-system jest to program komputerowy, który prezentuje wiedzę eksperta np.: lekarza, inżyniera, tłumacza. Okazuje się, że inwestowanie w tego typu oprogramowanie gwarantuje aktualnie najszybszy zwrot kosztów ponoszonych na badania nad sztuczną inteligencją.

Gromadzenie wiedzy w bazach danych, dostępnych ekspert-systemom, manipulowanie nią, diagnozowanie problemów wraz z sugerowaniem czynności prowadzących do ich rozwiązania, szacowanie ryzyka określonego przedsięwzięcia jest o wiele łatwiejsze niż stworzenie programu rozumiejącego naturalny, mówiony język.

Przykładem wyspecjalizowanego ekspert-systemu jest „Analizator zbożowego rynku", łączy on w sobie naturalny język oraz „wiedzę" niezbędną do wiarygodnej pomocy (rady) farmerom poszukującym najlepszej rynkowej strategii dla ich zboża (produkcji i handlu) — bazując na spodziewanych cenach i innych parametrach rynku.

Podobne systemy budowane są dla potrzeb przetwarzania tekstów, baz danych, a także... produkcji i handlu winem.

Postęp w tworzeniu programów rozpoznających naturalny język jest znacznie trudniejszy niż postęp budowy ekspert-systemów. Produkty bowiem „naturalnego języka" nie funkcjonują samodzielnie. Mają na celu ułatwić używanie ściśle określonego programu i w rzeczywistości są tylko jego uzupełnieniem (otoczeniem, muszlą). Od czasu, gdy każdy pakiet programowy musi mieć swego rodzaju interfejs dla użytkownika — język naturalny jest oczywistym wyborem. Należy przy tym pamiętać, że technika „naturalnego języka" ma o wiele dłuższą historię niż ekspert-systemy.

Pierwszym produktem wykorzystującym naturalny język była wprowadzona w połowie 1983 roku baza danych R.base 460, następnie także baza danych, tym razem firmy Q&A. Wydawanie komend sterujących pracą tych programów odbywa się za pośrednictwem zdań wypowiadanych w języku naturalnym (angielskim). Użytkownicy zmieniają, wprowadzają i wyszukują dane „rozmawiając" z maszyną. Program ten często nazywany jest „inteligentnym asystentem" — szybko „rozumie" wielokrotne warunki. Na przykład: „podaj mi nazwiska osób, których pensja łącznie z innymi dochodami jest większa od 10 000$".

Niektóre programy języków naturalnych, zamiast dokonywać analizy gramatycznej rozpoznawanego zdania, wykorzystują analizę częstotliwościową występowania liter w słowach, funkcjonując w efekcie, jak gdyby „rozumiały" angielski. Pierwszym przykładem tego typu programu był MindReader. Następnie pod koniec ubiegłego roku pojawił się Write Now, który poza typowymi zdolnościami przetwarzania tekstów, pomaga użytkownikowi uniknąć robienia błędów ortograficznych. Podczas pisania programu sprawdza każdy znak słowa (porównując z pamiętanym słownikiem ortograficznym). Gdy napiszesz błędną literę — program wyświetla ją w negatywie, „buczy" i sugeruje poprawkę.

Prawdopodobnie najbardziej wyróżniającym się przykładem „wbudowanej" techniki języka naturalnego jest „Hal". Jest to narzędzie wspomagające użytkowanie znanego od dawna programu Lotus 1-2-3. „Język naturalny" znacznie rozszerzył możliwości Lotusa. Użytkownik może np.: wydać polecenie w postaci: „wyświetl nazwy i liczby produktów typu X", wówczas Hal „samodzielnie" odnajdzie poszukiwane informacje i wyświetli lub wydrukuje na drukarce. Można wydawać także bardziej

złożone polecenia — np.: „wymień nazwy artykułów droższych od 100 $". Producent Lotusa zdając sobie sprawę z odpowiedzialności stosowania zwrotu „sztuczna inteligencja" — nie stosuje go w żadnym kontekście przy nazwie Hal.

Przykładem ekspert-systemu jest Paradox. Posiada on tę samą moc obliczeniową jak programowana baza danych typu dBase lub R:base. Zasadniczą techniką w Paradox'ie jest sposób pytania o wiedzę w nim zgromadzoną — poprzez przykłady. Także jego „ekspertowa wiedza" gromadzona jest w podobny sposób — użytkownik podaje przykłady określonych sytuacji, które są odpowiednio analizowane i zapisywane do bazy

Jedną z istotnych przyczyn stosunkowo wolnego „zdobywania" przez ekspert-systemy rynku jest kosztowne przygotowanie ich użytkowników (idzie tu o nauczenie zmian, dostosowania struktury systemu do specyficznych zastosowań). Na przykład systemy M.1, S.1 firmy Teknowledge kosztują po 5000 $, a nauka ich wykorzystania aż 2500 $!

Ekspert-systemy realizują polecenia wydawane zgodnie z określonymi zasadami (od 40 do 500 zasad). Wyrażają one w ogólności zdania typu: czy (wówczas in/then). Warto w tym miejscu wspomnieć plany sprzed 10 lat, kiedy to zakładano, iż zasad tych będą tysiące...

W praktyce stosuje się obecnie względnie małe systemy, które szybko „się spłacają". Niewielkie ekspert-systemy są bardziej ekonomiczne dla obu stron — producenta i użytkownika Dowodzą tego doświadczenia ze sprzedaży i eksploatacji tańszych, ale bardziej sprawdzonych („dopasowanych" do konkretnego zastosowania) narzędzi. Przykładami są wyprodukowane w ostatnich 10 latach systemy: 1 st Class, Insight 2 Plus, Personal Consultant Easy, the Arity Expert System Development Package i Wisdom XS. Systemy te mają moc podobną do narzędzi sprzed 10 lat, ale kosztują $^{1}/_{10}$ ówczesnej ceny i generalnie są łatwiejsze w wykorzystaniu.

Przykładem „życzliwego", o dużej mocy, ekspert-systemu jest Personal Consultant Series (produkt Texas Instruments), a także o dwa lata młodsze jego klony". Dołączone do nich poprawki to między innymi „okna" (fragmenty ekranu informujące o stanie i realizacji programu) oraz znacznie dokładniejsza dokumentacja.

Nowy Personal Consultant Series reprezentuje wiedzę uporządkowaną w struktury. Termin ten został wprowadzony przez Marvin'a Minsky'ego — jednego z inicjatorów prac nad AI w latach pięćdziesiątych. Struktury dzielą wiedzę na części, które są następnie zapamiętane tak, że opisują tę wiedzę w hierarchicznych relacjach. Dla przykładu, silnik samochodu powinien być widziany jako struktura, która jest rozłożona na elementy takie jak: system elektryczny, system chłodzenia (każdy subsystem jest także strukturą) itp

Uporządkowana w ten sposób wiedza stwarza większe możliwości programowego manipulowania na niej — przeglądanie, wybieranie, łączenie itp.

Podsumowując należy podkreślić, iż nie osiągnięto zamierzonego postępu w produktach sztucznej inteligencji. Skoncentrowano się natomiast na systemach specjalistycznych i uzyskano na tym polu pewne sukcesy. Podobnie systemy „naturalnego języka" stają się coraz mniej uniwersalne — rozwijane są specjalistyczne oprogramowania przeznaczone dla określonych grup użytkowników

Wydaje się, że zjawisko to — zwolnienia — stanowić może symptom wyhamowania gwałtownego postępu oprogramowania profesjonalnego sprzętu mikrokomputerowego

W. GOGOŁEK

Opracowane na podstawie: H. Fersko Weiss, 1986 hits and misses AI UPDATE Personal Computing/December 1986.

W szponach Atari (7)

Igraszki z duchami

Ludwik PIELA, Tomasz MROWIEC

Kłopoty z animacją

Dowolny domowy system komputerowy byłby niewiele wart bez możliwości animacji. Czyż można wyobrazić sobie efektowną grę bez ruchu obiektów na ekranie?

Konwencjonalnym sposobem uzyskania animacji jest przesuwanie obrazu w obszarze pamięci ekranu. Odbywa się to w dwóch etapach. Po pierwsze, program musi wykasować stary obraz poprzez zapisanie wartości tła do RAM zawierającej aktualny obraz. Następnie trzeba zapisać dane obrazu w obszarze pamięci odpowiadającym nowej pozycji obrazu. Poprzez ciągłe powtarzanie tego procesu uzyskujemy przemieszczanie się obrazu po ekranie.

Technika ta stwarza dwa problemy. Pierwszy, jeśli animacja wykonywana jest w trybie graficznym z dużymi punktami elementarnymi, ruch nie będzie płynny; obraz będzie przeskakiwał przez ekran. Dla innych komputerów jedynym rozwiązaniem jest użycie trybu graficznego o wysokiej rozdzielczości. Drugi problem jest o wiele gorszy. Obraz na ekranie jest dwuwymiarowy, a obszar pamięci ekranu jest jednowymiarowy. Oznacza to, że obraz, który jest ciągły na ekranie, nie będzie ciągły w RAM.

Znaczenie tego stanie się oczywiste jeżeli spróbujemy napisać program do przesuwania takiego obrazu. Przyjrzyjmy się, jak rozproszone są w pamięci bajty tworzące obraz. W celu ich wykasowania program musi obliczyć ich adresy, co nie zawsze jest łatwe do zrobienia. Poza tym nawet w kodzie maszynowym obliczenia zajmą trochę czasu. Oznacza to, że ten typ pamięci (zwany „animacją pola gry") jest zbyt wolny dla wielu celów.

Rozwiązaniem tego problemu jest technika wykorzystująca „sprite'y" czyli „duchy". W nomenklaturze Atari nosi ona nazwę grafiki gracz-pocisk (ang. player-missile graphics). Gracze (players) i pociski (missiles) są specjalnymi obiektami graficznymi przeznaczonymi do szybkiego przemieszczania się po ekranie.

Udało się utworzyć obiekt graficzny, który jest jednowymiarowy zarówno na ekranie jak i w RAM. Taki obiekt (zwany graczem) występuje w pamięci jako tablica (ciąg bajtów) o długości 128 lub 256 bajtów. Pojawia się jako pionowy pasek rozciągający się od góry do dołu ekranu. Każdy bajt tablicy odwzorowywany jest w jednej lub dwóch poziomych liniach ekranu (wybór należy do nas). Obraz jest bezpośrednim odwzorowaniem bitów danych w tablicy. Jeśli bit jest równy 1; wtedy zapalony jest odpowiadający mu punkt elementarny; jeśli bit jest równy 0, punkt jest zgaszony.

Grafika gracz-pocisk jest niezależna od innych trybów graficznych Atari, które nazywane są grafikami pola gry (ang. playfield). Instrukcje PRINT, PLOT, DRAWTO realizują grafikę pola gry. Technikę gracz-pocisk należy traktować jako drugi obraz na ekranie. Granice tego obrazu przekraczają wielkość pola gry i nie są związane z aktualnym trybem graficznym. Ponadto obrazy te mogą występować jakby były przed lub za grafiką pola gry na ekranie.

Dodatkowo grafika gracz-pocisk zwiększa ilość kolorów na ekranie. Każdy obiekt typu gracz ma swój własny rejestr koloru, rejestry te są niezależne od używanych dla grafiki pola gry. Bez względu na używany tryb zawsze będą dostępne cztery dodatkowe rejestry koloru dla obiektów graczy. Dzięki temu w trybie 0 (BASIC'u) można uzyskać na ekranie pięć kolorów.

Początkowo nie jest łatwo zrozumieć grafikę gracz-pocisk (G-P), ponieważ Atari BASIC nie ma odpowiednich instruk-

cji. Jej organizacja i wykorzystanie są przez to bardziej absorbujące, ponieważ muszą być realizowane na poziomie języka maszynowego. Czyni to tworzenie programów znacznie trudniejszym.

Definiowanie gracza

Każdy obiekt typu gracz ma szerokość ośmiu bitów. W celu zdefiniowania obiektu musimy utworzyć jego odwzorowanie w pamięci. Najpierw kreślimy żądany rysunek na papierze milimetrowym. Następnie tłumaczymy go na kod binarny, wstawiając jedynki dla punktów zapalonych i zera dla zgaszonych. Odwzorowaniem gracza jest zatem ciąg liczb jednobajtowych, które wstawimy do tablicy grafiki G-P. Przed wpisaniem musimy je przetłumaczyć na liczby dziesiętne lub szesnastkowe.

Lepszym rozwiązaniem jest wykorzystanie programu nr 1, który uwalnia nas od żmudnych obliczeń i umożliwia kreślenie kształtu obiektu bezpośrednio na ekranie za pomocą manipulatora drążkowego. Zauważmy, że ramki ograniczają nas do definiowania ośmiu komórek obrazu w poziomie. Po zakończeniu projektowania wyglądu gracza naciśnijmy klawisz (RETURN) i obraz gracza pojawi się w swojej naturalnej wielkości. Jeśli chcemy dokonać dodatkowych zmian w kształcie obiektu, musimy nacisnąć klawisz spacji. Po ponownym pojawieniu się migającego kursora, znowu możemy użyć manipulatora do zmiany obrazu gracza. Poprzez naciśnięcie (RETURN), po obejrzeniu kształtu gracza naturalnej wielkości, zobaczymy na ekranie odwzorowanie binarne obiektu wraz z innym spojrzeniem na utworzony obraz.

Obiekt typu gracz może być zdefiniowany w 128 lub 256 bajtach. Gdy zdefiniowany jest w 128 bajtach, każdy jego bajt zajmuje dwie linie ekranu. Gracze zdefiniowani w 256 bajtach będą używać tylko jednej linii ekranu dla każdego bajtu obiektu.

Zauważmy, że obiekty graczy różnią się swoimi wielkościami na obrazie. Gracze zdefiniowani w 256 bajtach mają rozdzielczość pionową dwa razy większą niż obiekty 128-bajtowe i są na ekranie mniej „zblokowane". Powinniśmy zdecydować, czy potrzebujemy tej dodatkowej rozdzielczości. Ponieważ wszyscy gracze muszą być zdefiniowani z taką samą długością, decyzja ta może nam zaoszczędzić trochę pamięci. Gracze zdefiniowani w 128 bajtach nazywani są graczami o rozdzielczości dwuwierszowej, a gracze zdefiniowani w 256 bajtach są zwani graczami o rozdzielczości jednowierszowej.

Tablica grafiki gracz-pocisk

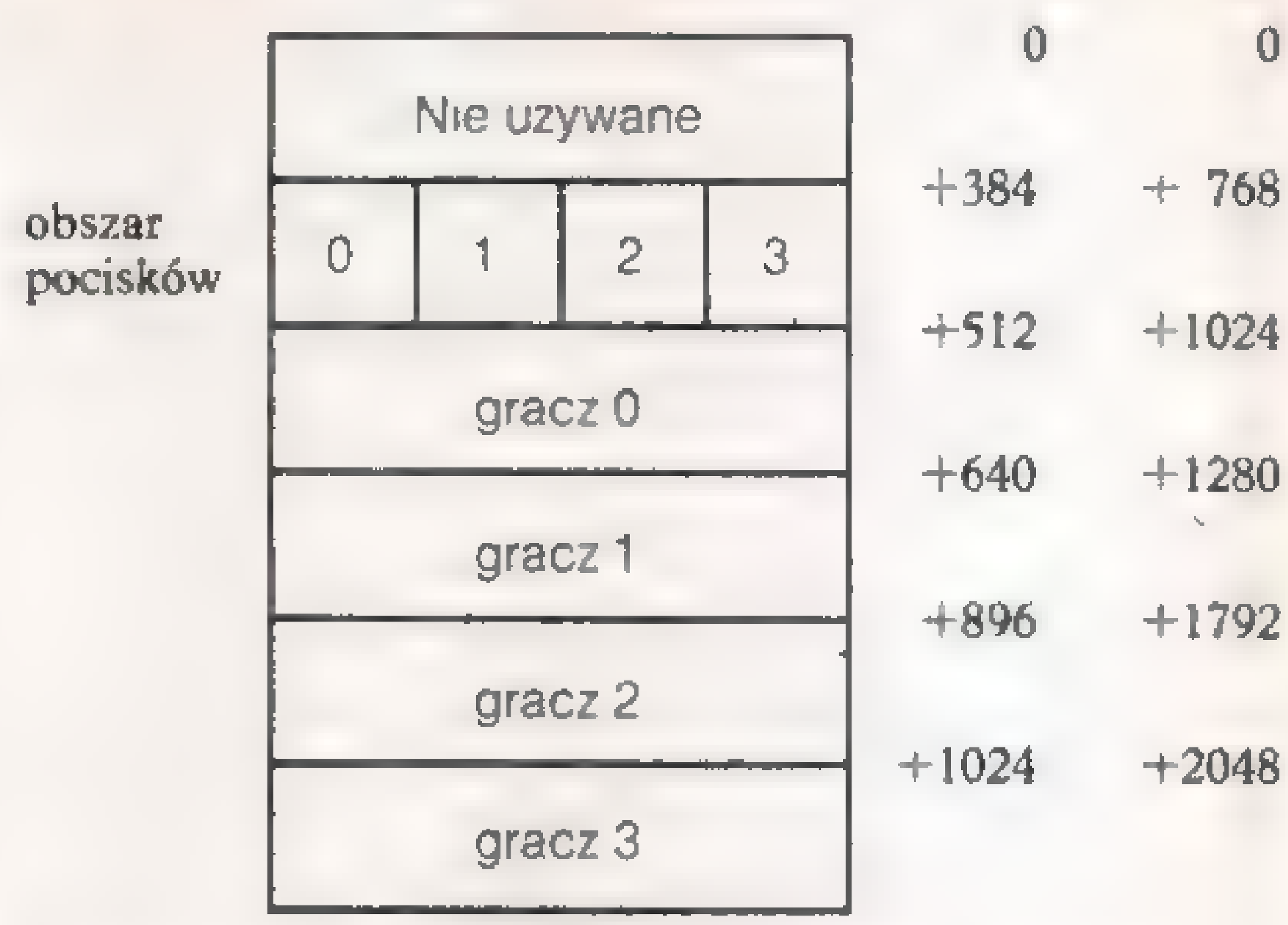
Tablica grafiki G-P musi rozpoczynać się od adresu podzielnego przez 1024, dla graczy o rozdzielczości dwuwierszowej, lub 2048 dla graczy o rozdzielczości jednowierszowej. Instrukcja PRINT PEEK (106)*256 wyświetli ostatni używany adres pamięci naszego komputera. W celu właściwego umieszczenia tablicy w pamięci, musi być znaleziony najbliższy adres na granicy 1024 lub 2048 bajtów.

Zawartość tablicy

Tablica grafiki G-P, bez względu na ilość zdefiniowanych graczy, zawsze będzie mieć wielkość 1024 lub 2048 bajtów dla umieszczenia czterech graczy i pocisków. Pierwsza część tablicy jest pusta; ten obszar dostępny jest dla innych celów, takich jak przechowywanie alternatywnych definicji graczy lub programów wyświetlania. Po tym pustym obszarze jest pięć innych, w których zdefiniowani są gracze i pociski.

Obszar definiowania pocisków będzie przechowywał cztery obiekty pocisków, każdy o szerokości dwóch bitów. Podobnie jak gracze, pociski także mogą być definiowane z rozdzielczością jedno- lub dwuwierszową.

Następne cztery obszary są jednakowej wielkości, każdy z nich przechowuje jeden obiekt gracza. Poniższy rysunek pokazuje przesunięcie od początku tablicy grafiki G-P dla każdego gracza i pocisku.



Następnym krokiem jest ponowna zmiana zawartości komórki 106, poprzez wpisanie adresu tablicy grafiki G-P. Ten krok jest konieczny, ponieważ system operacyjny będzie używał całej dostępnej pamięci, najwyższe dostępne adresy obszaru pamięci zawsze używane są do ustawienia grafiki pola gry. Ten konflikt użycia pamięci może zmienić grafikę pola gry, grafikę G-P, lub obie. W najgorszym przypadku może dojść do zablokowania komputera.

Obliczanie adresu początkowego

Komputer nie rozwiązuje automatycznie konfliktu pamięci, musimy to zrobić za niego. Chociaż wiemy już, że tablica grafiki G-P musi rezydować na granicy 1K lub 2K, obraz pola gry także ma ograniczenia. Umieszczenie tablicy grafiki G-P w najwyższej części pamięci spowoduje problemy adresowania dla grafiki pola gry. Na przykład, pewne obszary obrazu mogą być nie do użycia, lub instrukcje PLOT nie umieszczą punktów graficznych w oczekiwanych wierszach i kolumnach.

Jeśli pamięć ekranu grafiki pola gry przydzielona jest normalnie, możemy umieścić tablicę grafiki G-P tuż poniżej niej bez jakichkolwiek konfliktów pamięci. Jeśli jednak program zmienia tryby graficzne, wtedy możliwe jest powstanie innego konfliktu, który mógłby zniszczyć całkowicie tablicę grafiki G-P. Problem ten może wystąpić, jeśli program przełączy GRAPHICS 0 na GRAPHICS 7. W tym przypadku tablica może być całkowicie wykasowana. Należy zatem przyjrzeć się instrukcjom GRAPHICS w naszym programie; znaleźć tę, która przydziela najwięcej RAM i odpowiednio zaplanować umieszczenie tablicy. Dla obliczenia adresu początkowego grafiki G-P, wykonajmy następujące kroki:

1. Użyjmy funkcji PEEK dla określenia zawartości komórki 560.
2. Użyjmy funkcji PEEK dla określenia zawartości komórki 561 i pomnóżmy wynik przez 256.
3. Dodajmy wyniki kroków 1 i 2.
4. Podzielmy wynik kroku 3 przez 1024, jeśli używamy grafiki o rozdzielczości dwuwierszowej lub przez 2048, jeśli używamy grafiki o rozdzielczości jednowierszowej.
5. Odrzućmy resztę, odejmijmy 1 i pomnóżmy to przez 1024 (rozdzielczość dwuwierszowa) lub 2048 (rozdzielczość jednowierszowa).

Wynik kroku 5 jest adresem początkowym tablicy grafiki G-P. Programowo można to wykonać następująco:

```
1000 REM JEDN=1 oznacza rozdzielczość jednowierszową
1010 PMBASE=PEEK (560)+PEEK(561)*256
1020 IF JEDN THEN DIV=2048
1030 IF (NOT JEDN) THEN DIV=1024
1040 PMBASE=INT (PMBASE/DIV—1)*DIV
```

Ochrona tablicy grafiki G-P

Po określeniu adresu końcowego tablicy grafiki G-P, użyjmy instrukcji POKE do wstawienia dwubajtowej wartości do adresów 14 i 15 (najpierw młodszy bajt). System operacyjny interpretuje adres zawarty w tych komórkach jako bezwzględną dolną granicę przydziału pamięci grafiki pola gry. Zatem ustawienie tego adresu jest istotne dla ochrony tablicy grafiki G-P przed zniszczeniem kiedykolwiek realizowana jest nowa instrukcja GRAPHICS.

Umieszczenie graczy i pocisków w tablicy

Teraz możemy już wprowadzić bity graczy i pocisków do tablicy. Pierwszym krokiem jest oczyszczenie obszarów tablicy, które będą aktualnie przechowywać dane. Pierwszy obszar tablicy grafiki G-P nie jest używany. Nie ma potrzeby czyszczenia go, nie trzeba też czyścić obszarów tablicy, które nie będą zawierać aktywnych bitów.

Sterowanie wyświetlaniem

Istnieje kilka rejestrów sterujących, które jak wynika z ich nazwy, sterują bieżącym wyświetlaniem grafiki G-P:

- — rejestr bazowy,
- — rejestr sterujący grafiki i DMA,
- — rejestry szerokości,
- — rejestry kolorów,
- — rejestry pozycji poziomych,
- — rejestr sterowania priorytetami.

Pewne z tych rejestrów ustawiane są tylko raz, podczas inicjowania grafiki G-P, inne wymagają stałego ustawiania w zależności od tego jak nasz program będzie manipulował graczami i pociskami. Podręczniki Atari używają skróconych nazw tych rejestrów, które są umieszczone w nagłówkach.

Rejestr bazowy gracz-pocisk (PMBASE)

Komórki pamięci o adresach 54279 i 54280 (D407 i D408 szesnastkowo) będą zawierać adres początkowy tablicy grafiki gracz-pocisk. Ponieważ adres ma być na granicy 1K lub 2K, komórka 54279 zawsze musi zawierać 0. Znaczący jest tylko numer stronicy (starszy bajt adresu).

Rejestr sterujący grafiki (GRACTL)

Rejestr sterujący grafiki wraz z rejestrem DMACTL zezwala na bezpośredni dostęp do pamięci (DMA) dla grafiki

```
DL 4 REM *******************
MC 5 REM *                 *
DY 6 REM *  Prosram nr 1   *
ME 7 REM *                 *
DP 8 REM *******************
NO 9 REM
QJ 10 DIM KURSOR(2),GRACZ(23,7)
ND 20 GRAPHICS 3
RY 30 SETCOLOR 2,0,0:REM CZARNE O
   KNO TEKSTOWE
LR 40 ? "WLACZ MANIPULATOR DO GNI
   AZDA 1":? "NACISNIJ PRZYCISK"
BO 50 GOSUB 610:?
RH 60 PRINT "UZYJ PRZYCISKU DO KR
   ESLENIA LUB     KASOWANIA GR
   ACZA"
DA 70 PRINT "NACISNIJ <RETURN> PO
    ZAKONCZENIU       KRESLENIA.";
TH 80 GOSUB 670
JK 90 IF PEEK(764)=12 THEN GOTO 2
   90
WG 100 GOSUB 770
DR 110 IF (GORA AND (KURSOR(2)<=0
   )) OR (DOL AND (KURSOR(2))=23)
   ) THEN GOTO 90
GJ 120 IF (LEWO AND (KURSOR(1)<=0
   )) OR (PRAWO AND (KURSOR(1))=7
   )) THEN GOTO 90
XP 130 COLOR 1
ZU 140 IF GRACZ(KURSOR(2),KURSOR(
   1))=0 THEN COLOR 0
IT 150 PLOT KURSOR(1)+16,KURSOR(2
   )
SS 160 KURSOR(2)=KURSOR(2)-GORA
PV 170 KURSOR(2)=KURSOR(2)+DOL
GQ 180 KURSOR(1)=KURSOR(1)-LEWO
RP 190 KURSOR(1)=KURSOR(1)+PRAWO
XV 200 COLOR 2
IM 210 PLOT KURSOR(1)+16,KURSOR(2
   )
UF 220 X1=STRIG(0)
GE 230 IF X1=1 THEN 90
XA 240 GRACZ(KURSOR(2),KURSOR(1))
   =1-SGN(GRACZ(KURSOR(2),KURSOR(
   1)))
YQ 250 COLOR 3
ZZ 260 IF GRACZ(KURSOR(2),KURSOR(
   1))=0 THEN COLOR 0
IY 270 PLOT KURSOR(1)+16,KURSOR(2
   )
SV 280 GOTO 90
WT 290 GRAPHICS 7
KC 300 POKE 764,0
WD 310 X2=2:X3=0
EG 320 X1=76:Y1=20
UQ 330 GOSUB 910
QW 350 PRINT "<RETURN> - KONIEC":
   ? "<SPACE> - POWROT DO GRACZA"
BL 360 IF PEEK(764)=12 THEN GOTO
   440
LD 370 IF PEEK(764)<>33 THEN 360
GV 380 GRAPHICS 3+16
VQ 390 GOSUB 740
ZN 400 X1=16:Y1=0
VW 410 X2=1:X3=0
UP 420 GOSUB 910
SN 430 GOTO 90
SR 440 GRAPHICS 0
YQ 450 FOR Y=0 TO 23
FQ 460 POSITION 0,Y
AX 470 PRINT "BAJT ";Y;
YF 480 X1=0
UI 490 FOR X=0 TO 7
LR 500 X2=GRACZ(Y,7-X)
QK 510 IF X2=0 THEN 530
VC 520 X1=X1+INT((X2*2)^X+0.01)
LV 530 NEXT X
BK 540 POSITION 12,Y:PRINT X1;
MJ 550 NEXT Y
ZQ 560 X1=24:Y1=0
FR 570 X2=ASC("□"):X3=ASC(" ")
JZ 580 POKE 752,1
VE 590 GOSUB 910
QU 600 GOTO 590
YG 610 REM OCZEKIWANIE NA PRZYCIS
   K/GENERACJA DZWIEKU
YZ 620 IF STRIG(0)=1 THEN 620
RT 630 SOUND 0,50,10,4
MW 640 FOR OP=1 TO 10:NEXT OP
WA 650 SOUND 0,0,0,0
ZO 660 RETURN
XQ 669 REM USTAWIENIE ZMIENNYCH I
    EKRANU
RX 670 KURSOR(1)=0
SL 680 KURSOR(2)=0
WP 690 FOR R=0 TO 23
MM 700 FOR C=0 TO 7
WX 710 GRACZ(R,C)=0
DT 720 NEXT C
JP 730 NEXT R
YT 740 COLOR 3
NE 750 PLOT 15,0:DRAWTO 15,23:PLO
   T 24,0:DRAWTO 24,23
FL 760 GRAPHICS 3+48:RETURN
WX 770 REM ODCZYT MANIPULATORA
IL 780 S=STICK(0)
AN 790 GORA=0
ZM 800 DOL=0
EG 810 LEWO=0
KE 820 PRAWO=0
YK 830 IF S=15 THEN RETURN
OQ 840 IF S=14 THEN GORA=1
TT 850 IF S=7 THEN PRAWO=1
CY 860 IF S=13 THEN DOL=1
YX 870 IF S=11 THEN LEWO=1
UW 880 GOSUB 630
ZW 890 RETURN
QL 900 REM WYSWIETLANIE GRACZA
YN 910 FOR Y=0 TO 23
TZ 920 FOR X=0 TO 7
LM 930 COLOR X2
ES 940 IF PLAYER(Y,X)=0 THEN COLO
   R X3
CQ 950 PLOT X+X1,Y+Y1
MF 960 NEXT X
MR 970 NEXT Y
ZV 980 RETURN
```



```
DM 5 REM *******************
MD 6 REM *                 *
EU 7 REM *  Prosram nr 2   *
MF 8 REM *                 *
DQ 9 REM *******************
SG 50 GRAPHICS 0:SETCOLOR 2,0,0:P
   OKE 755,0
NA 60 PMB=PEEK(106)-16
SN 70 POKE 54279,PMB
AB 80 PMBASE=PMB*256
LU 90 FOR J=0 TO 255:POKE PMBASE+
   1024+J,0:NEXT J
YE 100 FOR J=0 TO 12:READ A:POKE
   PMBASE+1024+140+J,A:NEXT J
RH 110 POKE 559,62
QK 120 POKE 704,204
MV 130 POKE 712,34
HU 140 POKE 53248,130
VM 150 POKE 53277,3
TA 160 POSITION 6,18:? "UZYJ MANI
   PULATORA DO PRZESUWANIA
              GRACZA":?
XB 170 H=130:V=140
ZM 180 S=STICK(0):RESTORE
BR 190 IF S=11 THEN H=H-1
HL 200 IF S=7 THEN H=H+1
HE 210 POKE 53248,H
XH 220 V1=0
AR 230 IF S=14 THEN V=V-1:V1=1
YS 240 IF S=13 THEN V=V+1:V1=1
KW 250 IF V1=1 THEN FOR J=0 TO 12
   :READ A:POKE PMBASE+1024+V+J,A
   :NEXT J
VT 260 POSITION 6,21:? "POZYCJA P
   OZIOMA =    ";CHR$(254);CHR$(25
   4);CHR$(254);H
JL 270 POSITION 6,22:? "POZYCJA P
   IONOWA =    ";CHR$(254);CHR$(25
   4);CHR$(254);V
PH 280 GOTO 180
JS 290 DATA 0,40,146,214,254,238,
   254,238,186,170,56,16,0
```

gracz-pocisk. GRACTL umieszczony jest pod adresem 53277 (D01D szesnastkowo) i możemy wybrać zezwolenie tylko dla gracza (POKE 53277,2), tylko dla pocisku (POKE 53277,1) lub łącznie dla gracza i pocisku (POKE 53277,3).

Rejestr sterujący DMA (DMACTL)

Ustawienie rejestru sterującego DMA włącza lub wyłącza grafikę G-P. Jeśli rejestr GRACTL nie jest ustawiony na zezwolenie DMA, będziemy widzieć tylko grafikę pola gry. DMACTL i GRACTL muszą być ustawione jednocześnie w celu wyświetlania graczy i pocisków. DMA działa jak pasożyt na mikroprocesorze 6502, któremu kradnie cykle maszynowe. Jeśli chcemy wstrzymać wyświetlanie obiektów grafiki G-P, musimy wyzerować rejestr DMACTL. Przyspieszy to trochę działanie mikroprocesora.

Użyjmy instrukcji POKE do wstawienia jednej z poniższych wartości do komórki o adresie 559 (22F szesnastkowo) w celu ustawienia rejestru DMACTL.

Wartość	Wynik działania
4	włącza DMA tylko dla pocisku
8	włącza DMA tylko dla gracza
12	włącza DMA dla gracza i pocisku
+16	rozdzielczość jednowierszowa (domyślnie dwuwierszowa)

Rejestry szerokości graczy (SIZEP0—SIZEP3)

Cztery rejestry ośmiobitowe, o adresach 53256 do 53259 (D008 do D00B szesnastkowo), sterują poziomą wielkością czterech graczy. Poprzez zmianę zawartości tych komórek możemy zwiększyć dwukrotnie lub czterokrotnie szerokości obiektów (lecz nie ich wysokości). Jeśli wielkość gracza nie będzie zmieniana w programie, pozostaje normalnej wielkości.

Adres 53256 steruje szerokością pierwszego gracza, adres 53257 — drugiego, i tak dalej. Podczas pisania programów do przesuwania poziomego obiektów ustawienie rejestru szerokości może zmienić zawartość rejestru pozycji poziomej gracza. W celu przełączenia gracza na podwójną szerokość musimy ustawić jego rejestr szerokości na 1; dla poczwórnej szerokości — na 3. Wartości 0 i 2 określają normalną szerokość. Przykładem może być instrukcja ustawienia trzeciego gracza na podwójną szerokość:

POKE 53258,1.

Rejestr szerokości pocisku (SIZEM)

Jeden rejestr, o adresie 53260 (D00C szesnastkowo), steruje wielkością wszystkich pocisków. Takie same wartości, jak pokazane powyżej dla szerokości gracza, stosują się do rejestru szerokości pocisku: 0 lub 2 dla normalnej szerokości, 1 dla podwójnej i 3 dla poczwórnej.

Rejestry kolorów gracz-pocisk (COLPM0—COLPM3)

Cztery rejestry kolorów gracz-pocisk, o długości 1 bajta każdy, rozpoczynają się od adresu 704 (2C0 szesnastkowo) dla pierwszego gracza i kończą w 707 (2C3 szesnastkowo) dla czwartego. Zarówno gracz, jak i skojarzony z nim pocisk, mają ten sam kolor.

Rejestry pozycji poziomej graczy (HPOSP0—HPOSP3)

Rejestry pozycji poziomej graczy używane są do przemieszczania obiektów wzdłuż osi poziomej. Poprzez prostą zmianę zawartości rejestru instrukcją POKE, możemy przesunąć gracza do nowej pozycji poziomej. W zależności od ustawienia rejestru szerokości możemy ustawić gracza przy lewym brzegu ekranu, następnie ustawić nową wartość pozycji poziomej, która spowoduje, że obiekt ponownie pojawi się gdzieś na ekranie. Minimalną wartością każdego rejestru pozycji jest 0, maksymalną — 227. W zależności od wielkości gracza, określonej w DMACTL, ustawienia tych rejestrów będą zmieniać się od 40, jako widocznej pozycji najbardziej z lewej, do 190, jako pozycji najbardziej z prawej.

Rejestry te mogą być używane tylko w trybie zapisu, to znaczy nie będziemy mogli użyć instrukcji PEEK dla określenia pozycji gracza. Zatem nasz program musi mieć zmienne, które przechowują aktualne pozycje poziome graczy i pocisków na ekranie. Rejestr pozycji poziomej gracza 0 mieści się pod adresem 53248 (D000 szesnastkowo); gracza 1 — pod 53249; gracza 2 — pod 53250 i gracza 3 — pod 53251.

Rejestry pozycji poziomej pocisków (HPOSM0—HPOSM3)

Cztery rejestry, od adresu 53252 (D004 szesnastkowo), przechowują wartości używane do zmiany pozycji pocisków na osi poziomej.

Manipulowanie duchami

Po tym, nieco przydługim wprowadzeniu, niezbędnym z powodu trudności ze zdobyciem odpowiedniej literatury (a także jej giełdowej ceny), przejdziemy do omówienia praktycznego wykorzystania opisanej techniki.

Jak wspomnieliśmy na wstępie, podstawowym przeznaczeniem grafiki gracz-pocisk jest uzyskanie efektu animacji poprzez przemieszczanie obiektów na ekranie. Ruch obiektów rozłożymy na poziomy i pionowy, nie tyle ze względów formalnych, ile z powodu różnych sposobów jego realizacji.

Poziome przemieszczanie gracza jest bardzo proste. Po prostu zwiększamy lub zmniejszamy wartość przechowywaną w rejestrze pozycji poziomej odpowiadającym graczowi, którego chcemy przemieszczać. Jedyny problem z rejestrami pozycji poziomej polega na tym, że nie możemy pobrać ich zawartości dla określenia aktualnej pozycji poziomej gracza lub pocisku. Jeśli wprowadzimy:

POKE 53248,140:PRINT PEEK(53248)

moglibyśmy oczekiwać, że komputer wyprowadzi 140, wartość, którą zapamiętaliśmy w komórce 53248. Zamiast tego otrzymujemy liczbę 0. Co zatem stało się ze 140? Układ ANTIC pobrał 140 do swoich wewnętrznych rejestrów, odpowiednio zmienił pozycję poziomą gracza i wyzerował zawartość komórki 53248, zanim została wykonana instrukcja PRINT PEEK (53248). Jest to dosyć niewygodne, jeśli chcemy pamiętać kolejne współrzędne poziomego położenia gracza.

Program nr 2 demonstruje podstawową metodę przemieszczania graczy. Pierwsza sekcja, linie 50 do 150, zawiera standardowe procedury inicjowania. POKE 755,0 w linii 50 wyłącza kursor, zatem jeśli zatrzymamy program i chcemy z powrotem włączyć kursor, musimy nacisnąć (RESET) lub wprowadzić POKE 755,2.

Dla większości procedur animacji musimy przydzielić zmienną do przechowywania aktualnej pozycji paska gracza. Linia 170 używa zmiennej H do zapamiętania aktualnej wartości położenia poziomego gracza 0, który początkowo umieszczony jest w 130.

Linia 180 przydziela inną zmienną, S, do przechowywania wartości STICK(0) — stanu manipulatora. Możliwymi wartościami są: 11—lewo, 7—prawo, 14—w górę, 13—w dół i 15—środek.

Linie 190 i 200 sprawdzają czy S jest równe 11, czy 7 — manipulator wskazuje lewo lub prawo i wartość H zmienia się o jeden, zgodnie z kierunkiem ruchu gracza. Linia 210 wstawia wartość przechowywaną w H do rejestru pozycji poziomej gracza 0, w celu przesunięcia go o jedną pozycję w prawo lub w lewo. Każde przemieszczenie gracza równe jest szerokości punktu elementarnego.

U dołu ekranu wyświetlana jest aktualna wartość pozycji poziomej. Od wartości równej 48 gracz rozpoczyna opu-

szczanie niebieskiego obszaru i przechodzi przez tło, aż wyjdzie poza ekran, jeśli kontynuujemy ruch manipulatora w lewo. Dalsze zmniejszenie wartości poziomej nie daje żadnego efektu na ekranie.

Jeśli wartość H stanie się mniejsza od 0 lub większa od 255, otrzymamy komunikat błędu, gdy program próbuje wstawić wartość H do rejestru pozycji poziomej 53248 (linia 210). Wynika to z faktu, że nie możemy wstawić, za pomocą POKE, liczby nie mieszczącej się na pozycji jednego bajtu, do pojedynczej komórki pamięci. Dla uniknięcia tego problemu powszechnie stosuje się w takich programach procedury śledzenia błędów. Program nr 2 nie ma takiej procedury, dlatego możemy sprawdzić co zdarzy się, kiedy spróbujemy przesunąć gracza zbyt daleko.

Jak widać, uzyskanie ruchu poziomego jest bardzo łatwe. Kiedy przychodzi do ruchu pionowego, sprawa się komplikuje. Nie jest możliwe przesuwanie paska gracza w pionie. Zawsze będzie on rozciągał się od góry do dołu ekranu, bez względu na to, ile punktów elementarnych wewnątrz niego jest wyświetlanych.

Pionowe położenie obiektu na ekranie określa odległość, od początku obszaru gracza do miejsca, w które ładujemy dane. Dlatego w linii 100 dane o kształcie gracza wprowadzane są o 140 bajtów od początku obszaru danych gracza 0 dla upewnienia się, że obiekt wystąpi na ekranie. Jeśli zmienimy wartość 140 na 141, gracz pojawi się o jedną linię ekranu niżej. Wartość 139 spowoduje wyświetlenie gracza o jedną linię wyżej.

Teraz powinno być oczywiste jak uzyskać iluzję ruchu w pionie. Potrzebna jest procedura do przesuwania wszystkich danych kształtu o jeden bajt w górę lub w dół pamięci. Funkcje te wykonuje linia 250.

Początkowo, w linii 170, zmienna V przydzielona jest do przechowania aktualnej wartości pozycji pionowej gracza, w taki sam sposób jak zmienna H.

Linie 230 i 250 sprawdzają górną i dolną pozycję manipulatora i zwiększają lub zmniejszają odpowiednio wartość V. Jeśli manipulator został przesunięty w górę lub w dół, wtedy zmienna V1 ustawiana jest na 1. V1 działa jak flaga. Kiedy V1 jest jedynką, wtedy procedura przemieszczania pionowego w linii 250 jest aktywna. Kiedy V1 równe jest zero, żaden ruch pionowy nie jest potrzebny, procedura w linii 250 jest ignorowana.

Procedura ta podobna jest do procedury ładowania danych kształtu w linii 100 i wprowadza informacje o kształcie do obszaru danych gracza 0. Zamiast dodawania stałej wartości, podobnej do 140 w linii 100, do adresu bazowego obszaru danych gracza 0 (PMBASE+1024) dodawana jest wartość zmiennej V. Obiekt będzie przesuwany o odległość równą jednemu punktowi elementarnemu w górę lub w dół.

Instrukcja DATA zawierająca informacje o kształcie gracza 0, zawarta jest w linii 290. Zauważmy, że na początku i końcu tych danych występują zera, w celu wykasowania wcześniej wykreślonego punktu od góry lub dołu ostatniego kształtu gracza.

Aby zobaczyć co się zdarzy, kiedy pominiemy te zera, zmieńmy linie 290 poprzez wykasowanie obu zer i zmieńmy licznik pętli w liniach 100 i 250 z 12 na 10, aby czytana była poprawna ilość danych. Teraz, po uruchomieniu programu, zdarzą się straszne rzeczy! W miarę przesuwania manipulatora w górę lub w dół gracz pozostawia ślad.

Z pewnością nie będziemy zbyt zadowoleni z otrzymanego ruchu pionowego obiektu. Procedura ponownego wyświetlania w linii 250 zajmuje sporo czasu i zawsze będzie zauważalne opóźnienie, gdy program dokonuje aktualizacji informacji przechowywanej w obszarze danych gracza. Jeśli chcemy poprawić jakość przemieszczania, musimy wprowadzić procedurę w kodzie maszynowym.

Program nr 3 jest ulepszoną wersją poprzedniego. Zawiera dwie procedury w kodzie maszynowym, jedną dla ruchu w górę, a drugą dla ruchu w dół. Nie martwmy się, jeśli nie mamy pojęcia o kodzie maszynowym. Obie procedury mogą być skopiowane i włączone do używania we własnych programach bez dokładnego rozumienia jak one działają.

Po uruchomieniu tego programu zobaczymy, że działa on tak samo jak poprzedni. Dopiero podczas przesuwania w górę lub w dół zobaczymy, że animacja jest dużo płynniejsza.

W linii 170 czytane są dane zawarte w liniach 300 do 370 — procedury w kodzie maszynowym — i przechowywane w nieużywanej części obszaru danych gracz-pocisk. Ten nie wykorzystany obszar ma długość 768 bajtów dla rozdzielczości jednowierszowej graczy i 384 bajty dla rozdzielczości dwuwierszowej — wystarczająca ilość miejsca dla przechowania procedur w kodzie maszynowym.

Pierwsza procedura — o długości 21 bajtów — umieszczona jest w nie używanym obszarze od PMBASE w górę, druga procedura rozpoczyna się od PMBASE+21.

Procedury w takiej postaci będą sterować graczem lub pociskiem o wysokości do 20 punktów elementarnych. W celu użycia tej procedury dla wyższych graczy, musimy zmienić liczbę 22 w linii 340 na większą o 2 od wysokości gracza i liczbę 21, w linii 360, na większą o 1.

Przemieszczanie poziome obsługiwane jest w taki sam sposób jak w programie nr 2. W liniach 220 i 230 wstawiona została procedura śledzenia błędów, która zatrzymuje program, kiedy wartość zmiennej H stanie się mniejsza niż 48 lub większa niż 202. To znaczy, że gracz nie może opuścić obszaru pola gry i zapobiega pojawieniu się komunikatu błędu, który występuje, jeśli wartość H stanie się mniejsza od 0 lub większa od 255.

Linie 250 i 260 sterują przemieszczaniem pionowym poprzez wywoływanie procedur w kodzie maszynowym. V, zmienna współrzędnej pionowej, sprawdzana jest najpierw dla upewnienia się, że nie jest większa niż 32 lub mniejsza niż 212 — wtedy gracz jest na górnym lub dolnym brzegu pola gry — przed przekazaniem sterowania do odpowiedniej procedury maszynowej.

```
DM 5 REM ******************
MD 6 REM *                *
FP 7 REM *  Prosram nr 3  *
MF 8 REM *                *
DQ 9 REM ******************
AY 10 REM
SG 50 GRAPHICS 0:SETCOLOR 2,0,0:P
   OKE 755,0
NA 60 PMB=PEEK(106)-16
SN 70 POKE 54279,PMB
AB 80 PMBASE=PMB*256
LU 90 FOR J=0 TO 255:POKE PMBASE+
   1024+J,0:NEXT J
YE 100 FOR J=0 TO 12:READ A:POKE
   PMBASE+1024+140+J,A:NEXT J
RH 110 POKE 559,62
QK 120 POKE 704,204
MV 130 POKE 712,34
HU 140 POKE 53248,130
VM 150 POKE 53277,3
RF 160 POSITION 6,18:? "UZYJ MANI
   PULATORA DO PRZESUWANIA
          GRACZA"
BO 170 FOR I=0 TO 41:READ A:POKE
   PMBASE+I,A:NEXT I
XD 180 H=130:V=140
ZO 190 S=STICK(0):RESTORE
BA 200 IF S=11 THEN H=H-1
HN 210 IF S=7 THEN H=H+1
FP 220 IF H<48 THEN H=48
GN 230 IF H>202 THEN H=202
HK 240 POKE 53248,H
HW 250 IF S=14 AND V>32 THEN A=US
   R(PMBASE,PMBASE+1023+V):V=V-1
OJ 260 IF S=13 AND V<212 THEN A=U
   SR(PMBASE+21,PMBASE+1023+V):V=
   V+1
VV 270 POSITION 6,21:? "POZYCJA P
   OZIOMA =    ";CHR$(254);CHR$(25
   4);CHR$(254);H
JN 280 POSITION 6,22:? "POZYCJA P
   IONOWA =    ";CHR$(254);CHR$(25
   4);CHR$(254);V
PU 290 GOTO 190
JD 310 DATA 0,40,146,214,254,238,
   254,238,186,170,56,16,0
RM 330 DATA 104,104,133,204,104,1
   33,203,160,1,177
OR 340 DATA 203,136,145,203,200,2
   00,192,22,208,245,96
EE 360 DATA 104,104,133,204,104,1
   33,203,160,21,177
FR 370 DATA 203,200,145,203,136,1
   36,192,255,208,245,96
```

Wzór używania procedury jest następujący:
A=USR (adres kodu maszyn., adres gracza plus pozycja pionowa)
wykorzystując ten wzór, wywołanie procedury w linii 250 staje się następujące:

A=USR(PMBASE, PMBASE+1023+V)

Dla poprawnego działania procedury, V musi być zwiększane lub zmniejszane nie więcej niż o jeden naraz.

Po wywołaniu procedury i ponownym wyświetleniu gracza, V jest zmienne o 1, aby było gotowe do następnej operacji.

Inne własności

Opisany mechanizm umożliwia tworzenie bardzo szybkiej animacji. Istnieją czterej niezależni gracze. Mają oni własne zawartości rejestrów sterujących i obszary RAM; zatem ich działanie jest całkowicie niezależne. Oznaczeni są P0 do P3. Mogą być używani razem, jeden obok drugiego, dla otrzymania rozdzielczości 32-bitowej, lub mogą być używani niezależnie dla otrzymania czterech ruchomych obiektów.

Każdy gracz ma własny rejestr koloru, jest on całkowicie niezależny od rejestrów kolorów pola gry. Rejestry kolorów graczy nazywane są COLP(X). Umożliwia to otrzymanie większej ilości kolorów na ekranie. Jednak każdy gracz ma tylko jeden kolor; bez wykorzystania przerwań programu wyświetlania nie można uzyskać wielokolorowych graczy.

Każdy gracz ma zmienną szerokość, możemy ją ustawić normalną, podwójną lub poczwórną za pomocą rejestrów SIZEP(X). Jest to przydatne dla nadawania graczom różnych rozmiarów. Mamy także opcję wybierania rozdzielczości pionowej graczy. Możemy używać rozdzielczości jednowierszowej, w której każdy bajt w tablicy gracza zajmuje jedną linię ekranu, lub rozdzielczości dwuwierszowej, w której każdy bajt zajmuje dwie linie ekranu. Jest to jedyny przypadek, kiedy własności graczy nie są niezależne, wybór rozdzielczości pionowej dotyczy wszystkich graczy (steruje nią bit D4 rejestru DMACTL).

Pociski

Kolejną zaletą są pociski — obiekty graficzne o szerokości 2 bitów, skojarzone z graczami. Jeden pocisk przydzielony jest do każdego gracza. Informacje o kształcie pocisku pochodzą z tablicy umieszczonej tuż przed tablicami graczy. Cztery pociski upakowane są w tej samej tablicy. Mogą one poruszać się niezależnie od graczy; mają własne rejestry pozycji poziomej. Mają także własny rejestr wielkości, SIZEM, który określa ich szerokość, podobnie jak rejestry SIZEP(X) dla graczy. Jednak pociski nie mogą być ustawione na różne wielkości; są ustawiane razem. Na żądanie pociski mogą być połączone razem do postaci piątego gracza, w tym przypadku przyjmują one kolor rejestru 3 pola gry. Realizuje się to poprzez ustawienie bitu D4 rejestru sterowania priorytetem (PRIOR). W dalszym ciągu mogą one przemieszczać się niezależnie. Ustawienie tego bitu zmienia tylko kolor pocisków.

Pociski przesuwamy pionowo podobnie jak graczy: poprzez przemieszczenie danych obrazu w obszarze RAM pocisku. Jest to dosyć trudne w realizacji, ponieważ pociski umieszczone są w jednej tablicy.

Priorytety graczy i pola gry

Ważną własnością grafiki G—P jest to, że utworzone obiekty są całkowicie niezależne od pola gry. Możemy używać ich w dowolnym trybie graficznym. Powoduje to powstanie innego problemu: co się zdarzy, jeśli gracz zachodzi na obraz pola gry? Który obraz ma priorytet? Na szczęście mamy możliwość zdefiniowania priorytetów używanych do wyświetlania graczy. Jeśli zechcemy, wszyscy gracze mogą mieć priorytet nad rejestrami kolorów pola gry. Możemy ustalić, aby wszystkie rejestry pola gry (za wyjątkiem tła) miały priorytet nad wszystkimi graczami. Możemy również określić priorytety między poszczególnymi graczami i polem gry. Priorytety wybierane są z rejestru sterowania priorytetem — PRIOR. Ta własność umożliwia obiektom przechodzenie przed frontem jednego obrazu i poza innym, dają złudzenie efektu trójwymiarowości.

Układowe wykrywanie kolizji

Jedną z podstawowych zalet grafiki G—P, używaną w grach, jest możliwość układowego wykrywania kolizji. Możemy sprawdzić, czy jakiś obiekt graficzny (gracz lub pocisk) zderzył się z innym. Można wykryć następujące kolizje: pocisk-gracz, pocisk-pole gry, gracz-gracz, gracz-pole gry. Łącznie istnieje 54 możliwych kolizji, każda z nich ma przydzielony jeden bit, który może być sprawdzany. Jeśli bit jest ustawiony, wystąpiła kolizja. Bity te są odwzorowane w 15 rejestrach w GTIA (używane są tylko młodsze 4 bity). Rejestry te można tylko odczytywać, nie można ich oczyścić poprzez zapisanie zer. Aby je wyzerować (w celu dalszego wykrywania kolizji) musimy zapisać dowolną liczbę do rejestru HTCLR. Czyści to wszystkie rejestry kolizji.

Kolizja występuje wtedy, gdy obraz gracza nakłada się z innym obrazem; zatem bit kolizji nie będzie ustawiony dopóki kreślona jest część ekranu pokazująca kolizję. To znaczy, że wykrycie kolizji może nastąpić po około 20 ms od chwili przesunięcia gracza. Zalecanym rozwiązaniem jest wykonanie ruchu gracza i wykrywanie kolizji w czasie procedury przerwania pionowego. W tym przypadku najpierw należy sprawdzić, czy nie wystąpiła kolizja, następnie oczyścić rejestr kolizji i wykonać ruch graczem. Innym rozwiązaniem jest odczekanie 20 ms, po przesunięciu gracza, przed sprawdzeniem czy nie uległ on kolizji.

Zastosowania grafiki gracz—pocisk

Oprócz animacji, grafika G—P oferuje pewne dodatkowe możliwości. Gracze są wyjątkowym sposobem zwiększenia ilości kolorów obrazu. Cztery dodatkowe rejestry kolorów umożliwiają użycie czterech dodatkowych kolorów w każdej linii obrazu. Rozdzielczość 8-bitowa ogranicza oczywiście zakres ich zastosowań. Istnieje sposób obejścia tego ograniczenia. Weźmy gracza o poczwórnej szerokości i wstawmy go na ekran. Ustawmy tak priorytety, aby gracz miał niższy priorytet, niższy kolor pola gry. Następnie zamieńmy kolory pola gry i tła. Zatem widoczny kolor tła będzie w rzeczywistości kolorem pola gry. Gracz zniknie poza tym fałszywym tłem. Teraz wycinamy otwór poprzez wykreślenie prawdziwego koloru tła. Gracz pokaże się przed kolorem prawdziwego tła, lecz tylko w obszarze, gdzie było wykreślone prawdziwe tło. W ten sposób gracz może mieć więcej niż 8 bitów rozdzielczości poziomej.

Innym zastosowaniem grafiki G—P są znaki specjalne. Istnieje wiele znaków, które przekraczają pionowe granice w normalnych zestawach znaków. Jednym ze sposobów jest utworzenie zestawów znaków specjalnych, które rozwiążą ten problem. Innym sposobem jest użycie gracza. W ten sposób mogą być tworzone indeksy, znaki całek i inne symbole specjalne.

Szczególnie przydatnym zastosowaniem graczy jest wykorzystanie ich w charakterze kursora. Ze swoją zdolnością płynnego przemieszczania się po całym ekranie, bez naruszania jego zawartości, są oni idealnie dostosowani do takich zastosowań. Taki kursor może zmieniać kolor w miarę przemieszczania się po ekranie.

Grafika gracz—pocisk dostarcza wielu możliwości. Wykorzystanie jej w grach jako ruchomych obiektów jest oczywiste. Ma również zastosowanie poważne. Może zwiększyć ilość kolorów lub rozdzielczość obrazu. Może prezentować znaki specjalne. Może być używana w charakterze kursora. A zatem wykorzystujmy ją.

Ludwik PIELA,
Tomasz MROWIEC

Emulator BASIC 1.1 dla CPC 464 (1)

W niedługim czasie po ukazaniu się na rynku mikrokomputera CPC 464, pojawili się jego „młodsi bracia" — 664 i 6128. Do podstawowych różnic między CPC 464, a nowszymi modelami należy zaliczyć:

- zamianę pamięci zewnętrznej z magnetofonu na stację dyskietek
- powiększenie pamięci RAM z 64K do 128K (w przypadku 6128)
- rozszerzenie listy rozkazów interpretera BASIC o szereg instrukcji, głównie graficznych

O ile wyeliminowanie dwóch pierwszych różnic wymaga dosyć dużego wydatku, o tyle likwidację trzeciej możemy dokonać dzięki emulatorowi opublikowanemu w piśmie Schneider International 10/86. Celem zainstalowania rozszerzonej wersji interpretera należy wprowadzić krótki program LOADER. Program ten ładuje do pamięci RAM program maszynowy EMU.BIN, zawierający rozszerzenia. Nie są to RSX-y (tzn. komendy zewnętrzne względem interpretera — poprzedzane każdorazowo znakiem), lecz instrukcje stosowane na tych samych zasadach co komendy BASIC 1.0.

Program EMULATOR, składuje w pamięci zewnętrznej (kaseta lub dyskietka) program maszynowy EMU.BIN, ładowany w następnej fazie LOADER'em. Instrukcje BASIC 1.1 nie występujące w wersji 1.0 to:

- CURSOR
- ON BREAK ON BREAK STOP
- MASK
- GRAPHICS PEN
- GRAPHICS PAPER
- FRAME
- FILL
- COPYCHR$
- CLEAR INPUT

Dodatkowo komendy graficzne MOVE, MOVER, PLOT, PLOTR, DRAW, DRAWR zawierają dodatkowy parametr tryb tuszu. Powyższe komendy zostaną omówione i zilustrowane przykładami w kolejnym numerze „IKS'a".

J. R.

```
10 REM ****************************************
20 REM *   664/6128 - Emulator dla CPC 464    *
30 REM ****************************************
40 REM *       autor: M. Uphoff   1986        *
50 REM *                                      *
60 REM ****************************************
70 adr=&A1FF:MEMORY adr
80 FOR bl=1 TO 9
90 FOR i=1 TO 128
100 READ byte$:IF byte$="EOF" THEN 150
110 v=VAL("&"+byte$):s=s+v
120 adr=adr+1:POKE adr,v
130 NEXT i
140 NEXT bl
150 SAVE"emu.bin",b,&A200,&439
160 END
170 '
180 '*********************** Block 1
190 DATA 3E,C3,32,16,AC,21,83,A2
200 DATA 22,17,AC,32,19,AC,21,8E
210 DATA A2,22,1A,AC,32,07,AC,21
220 DATA 9A,A2,22,08,AC,32,04,AC
230 DATA 21,B4,A4,22,05,AC,32,0D
240 DATA AC,21,2A,A3,22,0E,AC,32
250 DATA 01,AC,21,AB,A2,22,02,AC
260 DATA 32,90,BB,21,D8,A2,22,91
270 DATA BB,32,DE,BB,21,11,A3,22
280 DATA DF,BB,32,C0,BB,21,F5,A2
290 DATA 22,C1,BB,32,C3,BB,21,FA
300 DATA A2,22,C4,BB,21,53,A3,22
310 DATA E3,BD,21,62,A3,22,E9,BD
320 DATA 3A,80,BC,FE,C3,C8,32,36
330 DATA A6,2A,81,BC,22,37,A6,3E
340 DATA C3,32,80,BC,21,B2,A2,22
350 '*********************** Block 2
360 DATA B1,BC,C9,11,E1,A5,CD,27
370 DATA E3,D0,F1,C3,61,DF,21,E1
380 DATA A5,CD,13,E3,D0,7E,23,C1
390 DATA C1,C9,FE,C5,D0,D6,BA,D1
400 DATA EB,4F,06,00,21,1C,A6,09
410 DATA C3,BB,DD,21,91,B2,75,C3
420 DATA 68,A2,E5,3A,36,A6,32,80
430 DATA BC,2A,37,A6,22,81,BC,CD
440 DATA B0,BC,21,80,BC,36,C3,21
450 DATA B2,A2,22,81,BC,E1,D8,C8
460 DATA FE,1A,37,3F,C0,B7,37,C9
470 DATA [illegible]
480 DATA [illegible]
490 DATA [illegible]
500 DATA 9F,DB,E1,E3,[illegible],CD,[illegible],A5
510 DATA 18,03,CD,BE,A5,F3,D9,CB
```

KRESKA (BASIC CPC 464) — zakończenie

Przedstawione w poprzednim odcinku komendy ¦SCREENCOPY i ¦SCREENSHOW posłużyły mi do zapamiętania w dowolnym momencie rysunku tworzonego przy wykorzystaniu programu KRESKA, a także umożliwiają one powrót do określonego etapu rysowania.

Ostatnią funkcją, którą proponuję dołączyć do programu, jest wprowadzanie tekstu do grafiki. Lokalizację tekstu na ekranie determinuje w tym wypadku bieżące położenie kursora graficznego. W programie umieściłem instrukcję TAG, która pozwala na mieszanie tekstu z grafiką. Rozkaz TAGOFF kasuje działanie rozkazu TAG (przywrócony jest druk tekstu na bieżącej pozycji kursora tekstowego).

Na tym zakończyłbym rozszerzanie programu KRESKA. Mam nadzieję, że kontynuowany przez wiele wydań „IKS-a" cykl artykułów stał się dla wielu czytelników okazją do własnych przemyśleń i inspiracją do poszukiwań.

Janek

```
190 LOCATE #2,2,2:PRINT #2,"(Z)AK. PRACY
 (O)DCZYT (W)PISANIE";
231 REM************************************

232 REM     ODCZYT RYSUNKU Z ROM
233 REM************************************

234 IF INKEY(34)=0 THEN |SCREENSHOW:GOTO
 180
235 REM************************************

236 REM     ZAPIS RYSUNKU DO ROM
237 REM************************************

238 IF INKEY(59)=0 THEN |SCREENCOPY:GOTO
 180
715 IF INKEY(2)=0 THEN GOTO 5000
4997 REM ******************************
4998 REM      WPROWADZANIE TEKSTU
4999 REM ******************************
5000 CLS #2:PRINT #2,"(T)EKST
      LUB ("+CHR$(240)+")";
5010 IF INKEY(0)=0 THEN GOTO 650
5020 IF INKEY(51)=0 THEN 5050
5040 GOTO 5010
5050 rb%=0:CLS #2:PRINT #2,"WPROWADZ. TE
KSTU WYLACZONE LUB P";
5060 LOCATE #2,1,2:PRINT #2,"UZYWAJ :"+C
HR$(240)+CHR$(242)+CHR$(243)+CHR$(241)+"
 COPY LUB JOYSTICK"
5070 GOSUB 1070:GOSUB 1210:GOSUB 1120:GO
SUB 1010:GOSUB 1070
5080 IF INKEY(27)=0 THEN GOTO 5000

5090 IF INKEY(F%)=0 AND rb%=0 THEN rb%=1
:FOR i=1 TO 200:NEXT i:LOCATE #2,18,1:PR
INT #2,"WLACZONE ":FOR i=1 TO 1000:NEXT
i:GOTO 5110 ELSE IF INKEY(F%)=0 AND rb%=
1 THEN rb%=0:LOCATE #2,18,1:PRINT #2,"WY
LACZONE":FOR i=1 TO 200:NEXT i:GOTO 5070

5100 GOTO 5070
5110 CLS #2:PRINT #2,"WCISNIJ klawisz -
ENTER '":PEN #3,kolor:INPUT #3,tekst$:CL
S #3:PEN #3,1:PRINT #3,CHR$(211)+CHR$(32
)+CHR$(211)+CHR$(32);:CLS #2:INPUT #2,"P
odaj tekst:",tekst$
5120 TAG #1
5125 PLOT XPOS,YPOS,kolor
5130 PRINT #1,tekst$;
5140 rb%=0:TAGOFF #1
5150 GOTO 5050
```

```
520 '********************** Block 3
530 DATA 59,D9,FB,C3,E1,E3,CD,55
540 DATA D3,0C,0E,A3,E3,C9,CD,4B
550 DATA C2,CD,D9,45,F3,D9,C8,59
560 DATA D9,FB,C9,CD,55,DD,D0,3E
570 DATA 04,CD,FB,C1,E5,C9,59,BC
580 DATA E1,C9,2B,7E,FE,20,28,FA
590 DATA FE,2C,C0,2B,E5,2A,34,AE
600 DATA CD,3F,DD,21,28,A6,CD,4A
610 DATA FF,E1,3B,09,FE,BB,C0,CD
620 DATA 93,BB,18,03,CD,E1,BB,D1
630 DATA C3,4F,D0,3A,2F,A6,2F,32
640 DATA 33,A6,CD,3C,18,AF,32,33
650 DATA A6,C9,3A,33,A6,B7,20,0F
660 DATA 3A,39,B3,B8,C2,CC,B1,3A
670 DATA 30,A6,B7,CA,CC,B1,C9,3A
680 DATA 46,B3,B7,3A,2F,A6,20,11
690 '********************** Block 4
700 DATA 37,32,2F,A6,3A,36,B3,47
710 DATA DA,CC,B1,3A,39,B3,47,18
720 DATA DE,E5,D5,C5,21,CF,B1,5F
730 DATA 3A,07,B2,ED,44,47,4F,CB
740 DATA 01,30,05,CB,03,30,01,B6
750 DATA 23,10,F4,4F,7B,32,2F,A6
760 DATA 3A,30,A6,B7,20,3A,2A,3B
770 DATA B3,7D,AC,A1,AC,C1,18,01
780 DATA F1,47,D1,E1,C3,CC,B1,CA
790 DATA A3,FE,CD,86,0C,32,3A,B3
800 DATA CD,1A,16,CD,FF,16,D2,68
810 DATA A4,E5,FD,E1,D5,DD,E1,CD
820 DATA A9,0D,3A,3A,B3,57,4E,A1
830 DATA 28,7E,3A,38,B3,5F,AE,A1
840 DATA 28,76,06,00,FD,23,CD,2D
850 DATA 0C,3A,34,B3,FD,95,38,52
860 '********************** Block 5
870 DATA 7A,AE,A1,28,4D,7B,AE,A1
880 DATA 20,EA,18,46,AE,77,CB,20
890 DATA E5,CB,91,DC,05,0C,7A,AE
900 DATA A1,28,12,7B,AE,A1,28,0B
910 DATA CB,C0,CB,48,28,07,DD,2B
920 DATA DF,8A,A4,DD,23,CB,A0,CB
930 DATA 07,E1,E5,CB,09,DC,F9,0B
940 DATA 7A,AE,A1,28,12,7B,AE,A1
950 DATA 28,0D,CB,E0,CB,68,20,07
960 DATA DD,23,DF,8A,A4,DD,23,CB
970 DATA 01,E1,3A,36,B3,FD,95,30
980 DATA 0F,FD,2B,CD,13,0C,7B,AE
990 DATA A1,28,05,7A,AE,A1,20,A4
1000 DATA 2A,34,A6,7D,B4,C8,2B,22
1010 DATA 34,A6,DF,79,A4,C3,D3,A3
1020 DATA 73,A4,FD,2A,89,AE,2B,56
1030 '********************** Block 6
1040 DATA 2B,5E,2B,22,89,AE,6E,26
1050 DATA 30,C9,E,A4,7D,C5,D5,2A
1060 DATA 89,AE,01,03,00,CD,18,F6
1070 DATA 38,17,EB,22,89,AE,DD,E5
1080 DATA D1,FD,E5,C1,2B,72,2B,73
1090 DATA 2B,71,2A,34,A6,23,22,34
1100 DATA A6,D1,C1,C9,FE,03,28,41
1110 DATA FE,AB,28,57,FE,9F,CA,4A
1120 DATA A5,FE2C,28,03,FE,28,C0
1130 DATA 3E,02,B9,20,64,F1,C1,F5
1140 DATA 3E,FB,B8,C0,79,FE,EE,28
1150 DATA 15,FE,F5,C0,7E,2B,FE,72
1160 DATA 20,FA,7E,3C,20,F6,23,CD
1170 DATA 3F,DD,CD,37,DD,2B,F1,C6
1180 DATA FB,CE,CD,37,DD,2C,C3,F5
1190 DATA F8,3E,0D,DB,C0,2A,34,AE
1200 '**********************. Block 7
1210 DATA CD,3F,DD,FE,7C,28,03,FE
1220 DATA 83,C0,E1,2A,C2,B0,CD,21
1230 DATA FB,37,C9,3E,05,BB,C0,3E
1240 DATA 1A,91,C0,B8,C0,F1,CD,F5
1250 DATA C1,CD,D3,C1,CD,37,DD,29
1260 DATA CD,60,BB,E5,CD,19,FA,E1
1270 DATA C9,3E,49,B9,C0,F1,C1,F5
1280 DATA 3E,D0,B8,C0,3E,93,B9,C0
1290 DATA F1,E5,00,00,AF,CD,0A,FF
1300 DATA E1,C9,F1,C1,F5,3E,C8,B8
1310 DATA C0,3E,D6,B9,C0,3E,BB,BE
1320 DATA C0,F1,11,61,A5,1B,C3,DA
1330 DATA CB,06,00,FF,FF,C9,00,CD
1340 DATA 55,DD,38,10,3E,02,CD,FB
1350 DATA C1,B7,CC,84,BB,C4,81,BB
1360 DATA CD,55,DD,D0,3E,02,CD,FB
1370 '********************** Block 8
1380 DATA C1,B7,CA,7E,BB,C3,7B,BB
1390 DATA 23,01,E4,BB,FE,BA,CA,18
1400 DATA CD,FE,CB,C2,C6,DE,CD,55
1410 DATA DD,38,0A,CD,4B,C2,CD,D8
1420 DATA A5,CD,55,DD,D8,3E,02,CD
1430 DATA FB,C1,32,30,A6,C9,CD,4B
1440 DATA C2,E5,DF,C7,A3,E1,C9,CD
1450 DATA 55,DD,38,0A,CD,67,CE,32
1460 DATA 2F,A6,CD,55,DD,D0,3E,02
1470 DATA CD,FB,C1,32,31,A6,C9,CD
1480 DATA 09,BB,38,FB,C9,CF,A9,92
1490 DATA CF,F6,97,CF,F4,95,CF,F1
1500 DATA 95,46,49,4C,CC,DD,47,52
1510 DATA 41,50,48,49,43,D3,DE,4D
1520 DATA 41,53,CB,DF,46,52,41,4D
1530 DATA C5,E0,43,55,52,53,4F,D2
1540 '********************** Block 9
1550 DATA E1,43,4C,45,41,52,20,49
1560 ATA 4E,50,55,D4,E2,43,4F,50
1570 DATA 59,43,48,52,A4,7E,44,45
1580 DATA 52,D2,49,00,AE,A5,88,A5
1590 DATA B7,A5,19,BD,67,A5,CF,A5
1600 DATA 94,95,AE,AF,BC,BD,00,FF
1610 DATA 00,FF,FF,00,00,00,00,00
1620 DATA 00,C0F
```

Mikrokomputery IBM PC (4)

Jacek WOJTALA

Każdy z mikrokomputerów osobistych wyposażony jest w system operacyjny czyli zespół programów umożliwiających wykorzystanie zasobów sprzętowych mikrokomputera, organizujących i kontrolujących jego pracę.

Mikrokomputery IBM PC/XT mogą pracować pod następującymi systemami operacyjnymi: DOS, CP/M-86, CONCURRENT DOS, CONCURRENT CP/M. Mikrokomputery IBM PC/AT ponadto mogą pracować pod systemem XENIX, wersją systemu UNIX.

System operacyjny DOS (Disk Operating System) jest najpopularniejszym z systemów wykorzystywanych w mikrokomputerach IBM PC, większość oprogramowania pracuje właśnie pod tym systemem.

Polecenia DOS dzielą się na dwa typy:

Polecenia wewnętrzne i **polecenia zewnętrzne. Polecenia wewnętrzne** wykonywane są natychmiast ponieważ wbudowane są w DOS. Jeżeli system został wczytany do pamięci operacyjnej, wówczas dyskietka ze zbiorami DOS nie jest potrzebna do wykonania tych poleceń.

Polecenia zewnętrzne umieszczone są na dyskietce jako zbiory programowe i przed ich wykonaniem muszą być przeczytane.

Dowolne zbiory z rozszerzeniem nazwy zbioru .BAT, .COM, .EXE są traktowane jako polecenia zewnętrzne. Pozwala to na tworzenie nowych, unikalnych poleceń i dołączenie ich do systemu operacyjnego

Struktura systemu DOS

BOOT STRAP jest to program ładowania, umieszczony na ścieżce Ø, w sektorze 1, na stronie Ø dyskietki, zajmuje 512 bajtów (1 sektor). Powoduje odszukanie w katalogach i wczytanie do pamięci operacyjnej programów IBMBIO.-COM, IBMDOS.COM.

IBMBIO.COM jest to program pośredniczący między procedurami umieszczonymi w BIOS (Basic Input Output System), a programem IBMDOS.COM. Program dokonuje również obróbki zbioru konfiguracyjnego CONFIG.SYS Zajmuje 9 sektorów.

IBMDOS.COM jest to program z rezydentem systemu operacyjnego, zarządza zbiorami na dyskach, wpisuje adresy procedur użytkowych, wczytuje do pamięci operacyjnej interpreter poleceń COMMAND.COM. Program zajmuje 34 sektory.

COMMAND.COM służy do interpretacji i wykonywania poleceń wewnętrznych oraz do wczytywania do pamięci operacyjnej poleceń zewnętrznych (lub programów użytkowych) i ich wykonywania. Zajmuje 36 sektorów.

PROGRAMY OBSŁUGI POLECEŃ ZEWNĘTRZNYCH

umieszczone są na dyskietce systemowej DOS lub w katalogu systemowym na dysku stałym.

Praca systemu DOS

Po włączeniu zasilania (start zimny) mikrokomputera w pierwszej kolejności wykonywane są testy sprawdzające procesor, pamięć, klawiaturę itp. Następnie jedna z proce-

dur BIOS-u powoduje ściągnięcie BOOT STRAP-u, ten z kolei wczytuje do pamięci IBMBIO.COM i IBMDOS.COM. Ostatni z programów wczytuje interpreter poleceń COMMAND.COM.

W dowolnej chwili praca systemu może być przerwana, można dokonać tzw. startu gorącego czyli spowodować ponowną inicjację systemu operacyjnego.

Zbiory wsadowe

Polecenia zawarte w zbiorze wsadowym, który musi mieć rozszerzenie nazwy zbioru .BAT, wykonywane są tak samo, jak gdyby zostały wprowadzone bezpośrednio. Istnieje siedem poleceń: ECHO, FOR, GOTO, IF, SHIFT, PAUSE, REM, które pozwalają kontrolować wykonywanie zbioru wsadowego. Ostatnim poleceniem zbioru może być nazwa innego zbioru wsadowego.

Przy każdej inicjacji systemu DOS poszukuje w katalogu systemowym dysku, z którego był inicjowany, specjalnego zbioru wsadowego o nazwie AUTOEXEC.BAT. Zbiór ten jest automatycznie wykonywany po uruchomieniu systemu.

Zbiór wsadowy może zawierać parametry, które będą zastępowane argumentami w trakcie jego wykonywania.

Konfiguracja systemu

Zbiór CONFIG.SYS jest specjalnym zbiorem, zawierającym listę poleceń konfiguracyjnych pozwalających m.in.: podać liczbę buforów systemowych; podać format daty i czasu; instalować programy obsługi urządzeń; podać liczbę zbiorów, które mogą być otwarte przez blok kontroli zbiorów.

Podczas każdego startu systemu, DOS szuka — w katalogu systemowym dysku, z którego był inicjowany — zbioru CONFIG. SYS. Polecenie zawarte w tym zbiorze: BREAK, BUFFERS, COUNTRY, DEVICE, FCBS, FILES, LASTDRIVE, SHELL powodują zmianę standardowych poleceń konfiguracyjnych systemu. Przykładowo posiadając program obsługi urządzenia, które symuluje stację dysków i po umieszczeniu odpowiedniego polecenia DEVICE w zbiorze CONFIG.SYS, można wykorzystując część pamięci operacyjnej komputera zainstalować **dysk wirtualny**

Katalogi strukturalne

Katalog systemowy dyskietki czy dysku stałego może zmieścić ograniczoną liczbę tzw. **katalogów wejściowych**. Katalogiem wejściowym może być zbiór, katalog niższego poziomu lub etykieta. Katalog systemowy dyskietki jednostronnej (160 lub 180 KB) może przechować 64 wejścia; katalog dyskietki dwustronnej (320 lub 360 KB) 112 wejść; katalog dyskietki dużej pojemności (1,2 MB) 222 wejścia; natomiast katalog dysku stałego 512 wejść. Aby uniknąć tych ograniczeń oraz aby skrócić czas poszukiwania konkretnego zbioru przez DOS, zbiory przechowywane są w grupach zwanych **katalogami niższych poziomów**. Katalogi te nie posiadają ograniczeń na ilość zbiorów w nich zapisanych, jedynym ograniczeniem jest wielkość dostępnego obszaru dyskowego.

Polecenia DOS

Wersja 3.10. DOS posiada, łącznie z poleceniami do kontroli wykonywania zbioru AUTOEXEC. BAT, a bez poleceń konfiguracyjnych zbioru CONFIG.SYS, 55 poleceń wewnętrznych i zewnętrznych. Polecenia mogą być wykorzystane, m.in. do następujących operacji:

— porównywania, kopiowania, wyświetlania, usuwania i zmiany nazw zbiorów,
— formatowania dysku stałego i dyskietek;
— uruchamiania programów użytkowych i systemowych, takich jak EDLIN, DEBUG i LINK;
— ustawiania różnych opcji pracy drukarki i monitora;
— żądania przerwania,
— skierowania wyjścia drukarki na urządzenie komunikacji asynchronicznej;
— odzyskiwania konkretnego zbioru z uszkodzonego dysku,
— przesłania zawartości ekranu graficznego na drukarkę;
— drukowania zbiorów na drukarce podczas wykonywania przez system innego zadania;
— wykonywania kopii zapasowych;
— definiowania konsoli pomocniczej,
— sortowania danych tekstowych;
— poszukiwania łańcuchów znaków w tekście;
— ustawiania znaku gotowości systemu;
— określania środowiska systemu;
— zabezpieczania zbiorów przed skasowaniem;
— zmiany etykiety dysku;
— tworzenia, usuwania i zmieniania katalogów niższego poziomu;
— wyświetlania struktury katalogu;
— ustawiania daty i czasu oraz ich formatu;
— sprawdzania błędów;
— przyłączania urządzeń do katalogu innego urządzenia.

Większość poleceń systemu DOS wersji 3.10 może być wykorzystana do pracy w sieci.

Do systemu operacyjnego dołączone są program EDLIN (edytor tekstów), program LINK (do łączenia programów), program DEBUG (do uruchamiania i testowania programów).

Jacek WOJTALA

Kompilator Hisoft — Pascal

Programy ilustrujące kurs języka Pascal prowadzonego w „IKS-ie" pisane są przy użyciu kompilatora TURBO PASCAL. Jest on bardzo wygodny w użyciu, lecz dostępny tylko dla komputerów pracujących pod CP/M. Nie jest więc dostępny na większości użytkowanych przez naszych Czytelników komputerach. Znacznie bardziej rozpowszechniony jest kompilator **HISOFT-PASCAL**. Trzeba przyznać, że kompilator ten jest mniej wygodny w użyciu, ale umożliwia uruchomienie tych programów na komputerach nie posiadających systemu CP/M. Omawiana poniżej wersja kompilatora dotyczy P.C. „Spectrum", lecz różnice występujące w innych wersjach są nieznaczne. Poprawnie wczytany kompilator rozpoczyna działanie od zadania pytań dotyczących lokalizacji poszczególnych bloków, dla większości programów wystarczy odpowiedzieć wciskając klawisz ENTER, co powoduje standardową organizację pamięci.

Po zakończeniu wczytywania możemy zacząć wpisywać nasz program. Linie programu należy ponumerować tak jak w języku Basic. **Po wpisaniu każdej linii wciskamy klawisz ENTER.**

W celu wprowadzenia poprawek w dowolnej linii przywołujemy ją instrukcją E*n* (*n* — numer edytowanej linii)

Na dole ekranu ukazuje się zapis linii w wersji pierwotnej, a poniżej numer tej linii, od którego należy wpisać zapis poprawny.

Istnieją trzy możliwości wprowadzania poprawek:
— wprowadzić pełny nowy zapis z klawiatury;
— przesunąć kursor za pomocą klawisza spacji do miejsca, w którym tekst ma być poprawiony, nacisnąć „I", w wyniku czego pojawi się znak kursora „x". Wpisywanie nowych znaków spowoduje rozsunięcie się pozostałych znaków;
— po ustawieniu kursora we właściwe miejsce nacisnąć klawisz „C", kursor staje się znakiem „+". Wprowadzany nowy tekst zapisywany jest na istniejącym.

Naciśnięcie klawisza „X" powoduje przejście kursora na koniec linii ze skasowaniem pozostałych znaków linii poprawionej. Kursor zmienia się na znak „x". Pierwsze naciśnięcie ENTER powoduje powrót do normalnego kursora, drugie naciśnięcie — wprowadzenie skorygowanej linii programu.

PROGRAM 22 ATARI

Maszyna do pisania

Zadaniem programu jest wykorzystanie drukarki ATARI 1029 do druku tekstu z polskimi literami. Tekst nie może zawierać więcej niż 80 znaków w wierszu.

Wprowadzony zestaw polskich znaków otrzymuje się wciskając klawisz CONTROL i "A" (uzyskuje się literę ą), „C" (ć), "E" (ę), "L" (ł), "N" (ń), "O" (ó), "S" (ś), "Z" (ż), "," (ź).

Przed przystąpieniem do pisania należy ustawić lewy i prawy margines. W czasie pisania istnieje możliwość (w dowolnym momencie) zmiany marginesów, w tym celu należy nacisnąć klawisz CONTROL i "Q". Jeżeli w trakcie pisania użytkownik pomyli się, powinien wtedy nacisnąć klawisz CONTROL i "P". Wpisany fragment wiersza jest wtedy kasowany, a piszący może wprowadzić go ponownie.

Przy dochodzeniu pisanego wiersza do prawego marginesu, trzy ostatnie znaki są sygnalizowane dźwiękiem. Po wprowadzeniu ostatniego znaku wyłączana jest automatycznie klawiatura i następuje wydruk. Jeśli piszący decyduje się wydrukować krótszy wiersz niż przewiduje to prawy margines, należy nacisnąć RETURN. Po wykonaniu przez drukarkę wydruku komputer oczekuje wprowadzenia następnego wiersza. Odstęp między wierszami wynosi cztery skoki wałka maszyny do pisania. Zwiększenie tego odstępu można uzyskać wciskając klawisz RETURN.

W celu stworzenia wizualnej kontroli nad wprowadzanym tekstem, w górnej części ekranu podane są kolejno: wartość lewego i prawego marginesu, zliczanie znaków w wierszu, zliczanie wierszy.

W programie nie zostały uwzględnione duże polskie znaki.

J. JANIEC

```
10 ? CHR$(125)
400 REM ************************
410 REM Wykorzystanie drukarki
420 REM ATARI 1029 Jako maszyny
430 REM do pisania z zestawem
440 REM polskich liter
450 REM
460 REM Janusz J.
470 REM Warszawa 1986,12


480 REM ************************
500 GRAPHICS 2:COLOR 1
510 POSITION 8,3:? #6;"MINI"
520 POSITION 7,5:? #6;"EDYTOR"
525 FOR A=0 TO 100:NEXT A
530 POKE 656,1:POKE 752,1:? "START
"
560 IF PEEK(53279)=6 THEN 600
570 POSITION 8,3:? #6;CHR$(205);CH
R$(201);CHR$(206);CHR$(201)
575 POSITION 7,5:? #6;CHR$(197);CH
R$(196);CHR$(217);CHR$(212);CHR$(2
07);CHR$(210)
580 FOR A=0 TO 100:NEXT A
590 GOTO 510
600 GRAPHICS 0:?
620 ? "          ** PROSZE CZEKAC **
"


1000 REM ************************
1010 N=57344:M=32768
1020 FOR I=0 TO 1023:POKE M+I,PEEK
(N+I):NEXT I
1040 A=M+776:B=M+520
1050 FOR I=0 TO 207:POKE B+I,PEEK(
A+I):NEXT I
1070 A=M+976:B=M+512
1080 FOR I=0 TO 7:POKE B+I,PEEK(A+
I):NEXT I
1100 FOR X=1 TO 21:READ MA,AA:POKE
 M+MA,AA:NEXT X
1140 POKE 756,128
1150 DATA 513,12,514,24,515,124
1160 DATA 527,3,537,12,538,24
1170 DATA 539,60,559,6,611,28
1180 DATA 612,56,625,12,626,24
1190 DATA 627,124,633,12,634,24
1200 DATA 635,60,664,6,665,12
1210 DATA 721,24,722,0,723,124
1230 DATA 0,9,11,45,73,1,0
1231 DATA 4,42,42,30,3,1,0
1233 DATA 0,6,9,41,73,9,0
1235 DATA 28,42,42,42,19,1,0
1242 DATA 0,0,4,127,17,0,0
1244 DATA 0,8,7,40,72,7,0
1245 DATA 0,6,9,41,73,6,0
1249 DATA 0,9,21,53,85,2,0
1256 DATA 0,17,19,85,25,17,0
1320 C=0:DIM A$(80):OPEN #3,4,0,"K
":OPEN #1,8,0,"P:":OPEN #2,8,0,"P
"
1380 ? CHR$(125):IF A=16 THEN 1440
1400 POSITION 2,2:? "LEWY MARGINES
:";:INPUT Y:IF Y<1 OR Y>80 THEN 14
00
1420 POSITION 2,4:? "PRAWY MARGINE
S:";:INPUT YY:IF YY<1 OR YY>80 THE
N 1420
1440 REM ************************
1442 W=5
1450 FOR X=1 TO Y:A$(X,X)=CHR$(32)
:NEXT X
1470 ? CHR$(125):POKE 752,1:B=Y:C=
C+1:POSITION 29,1:? "WIERSZ:";C
1500 GOTO 1520
1510 B=B+1:POKE 730,5:IF B>YY-3 TH
EN POKE 730,0
1515 IF B=YY OR A=155 THEN A$(B,B)
=CHR$(155):POKE 621,255:GOTO 1630
1520 POSITION 2,1:? "MARG.L:";Y;"
P:";YY;"   ZNAK:"
1525 K=B-Y:IF K<40 THEN 1530
1527 K=K-40:W=7
1530 POKE 621,0:POSITION K,W:? CHR
$(160):GET #3,A
1540 IF A=16 OR A=17 THEN A$="":C=
C-1:GOTO 1380
1555 A$(B,B)=CHR$(A):POSITION 24,1
:? B+1:POSITION K,W:? CHR$(A);
1580 IF B<YY-2 THEN 1510
1590 FOR X=1 TO 5:SOUND 0,50,10,15
:NEXT X:SOUND 0,0,0,0:GOTO 1510
1620 REM ************************
1630 FOR BB=1 TO B:A=ASC(A$(BB,BB)
)
1650 IF A=0 OR A=1 OR A=3 OR A=5 O
R A=12 OR A=14 OR A=15 OR A=19 OR
A=26 THEN 1700
1660 ? #2;CHR$(A);
1670 NEXT BB
1675 IF A$(B-1,B-1)=CHR$(155) THEN
 1690
1680 IF B=YY THEN LPRINT
1690 A=125:GOTO 1442
1700 RESTORE 1230+A
1710 ? #1;CHR$(27);CHR$(54);CHR$(2
7);CHR$(65);CHR$(0);CHR$(7);
1720 FOR X=0 TO 6:READ A
1740 ? #1;CHR$(A);:NEXT X
1760 GOTO 1670
```

Dodatkowo w tej fazie można używać dyrektyw:

D*n.m* — kasuje linie między *n* a *m*;

O — zakończenie edycji z pominięciem zmian;

L — wypisanie całej linii wraz z niewidoczną częścią za kursorem.

Po wyjściu z trybu edycji możemy uzyskać:

L*n.m* — listuje na ekranie linie od n do m,

M*n.m* — zastąpienie linii m tekstem z linii n (kopiowanie linii);

N*n.m* — przenumerowanie linii od linii n do końca programu, nr linii następnej jest o m większy od poprzedniej.

Gotowy program należy wyprowadzić na taśmę. Radzę robić to stosunkowo często, gdyż kompilator ma tendencje do „zawieszania się" (komputer przestaje reagować na naciskanie jakichkolwiek klawiszy). W tym przypadku jedyną radą jest wyłączenie go na moment i ponowne wczytanie kompilatora i programu.

Wyprowadzenia na taśmę lub dysk dokonujemy instrukcją **P***n.m.s*, instrukcją tą wyprowadzamy na taśmę linie o numerach od *n* do *m* w bloku o nazwie *s*. Ponowne wczytanie dokonujemy instrukcją **G.***s* (*s* — nazwa bloku).

Teraz mozemy przejść do kompilacji i uruchomienia programu. Instrukcja **C***n* powoduje kompilacje programu od linii *n* do końca. W razie wykrycia błędu kompilacja jest przerywana i drukowana informacja o błędzie

Po pomyślnym zakończeniu kompilacji drukowane jest pytanie **"RUN?"**. Naciśnięcie klawisza **"Y"** powoduje uruchomienie programu wynikowego. Ponowne uruchomienie uzyskujemy naciskając klawisz **"R"**.

W celu stworzenia kompletnego programu wynikowego składa się go z podprogramów standardowych, podprogramów wspomagających i przetłumaczonego programu używamy instrukcji TN (n-numer początkowej linii programu).

Do podstawowych błędów wykrywanych w trakcie kompilacji należą

1 — zbyt duża liczba;

2 — oczekiwano średnika;

3 — wykryto niezadeklarowany identyfikator;

4 — oczekiwano identyfikatora,

5—6 — oczekiwano "=";

8—9 — oczekiwano ":=", ")";

10 — niewłaściwy typ;

11 — oczekiwano ".",

12 — oczekiwano wskaźnika,

13—14 — oczekiwano stałej,

25—28 — oczekiwano "THEN", "DO", "TO", "(";

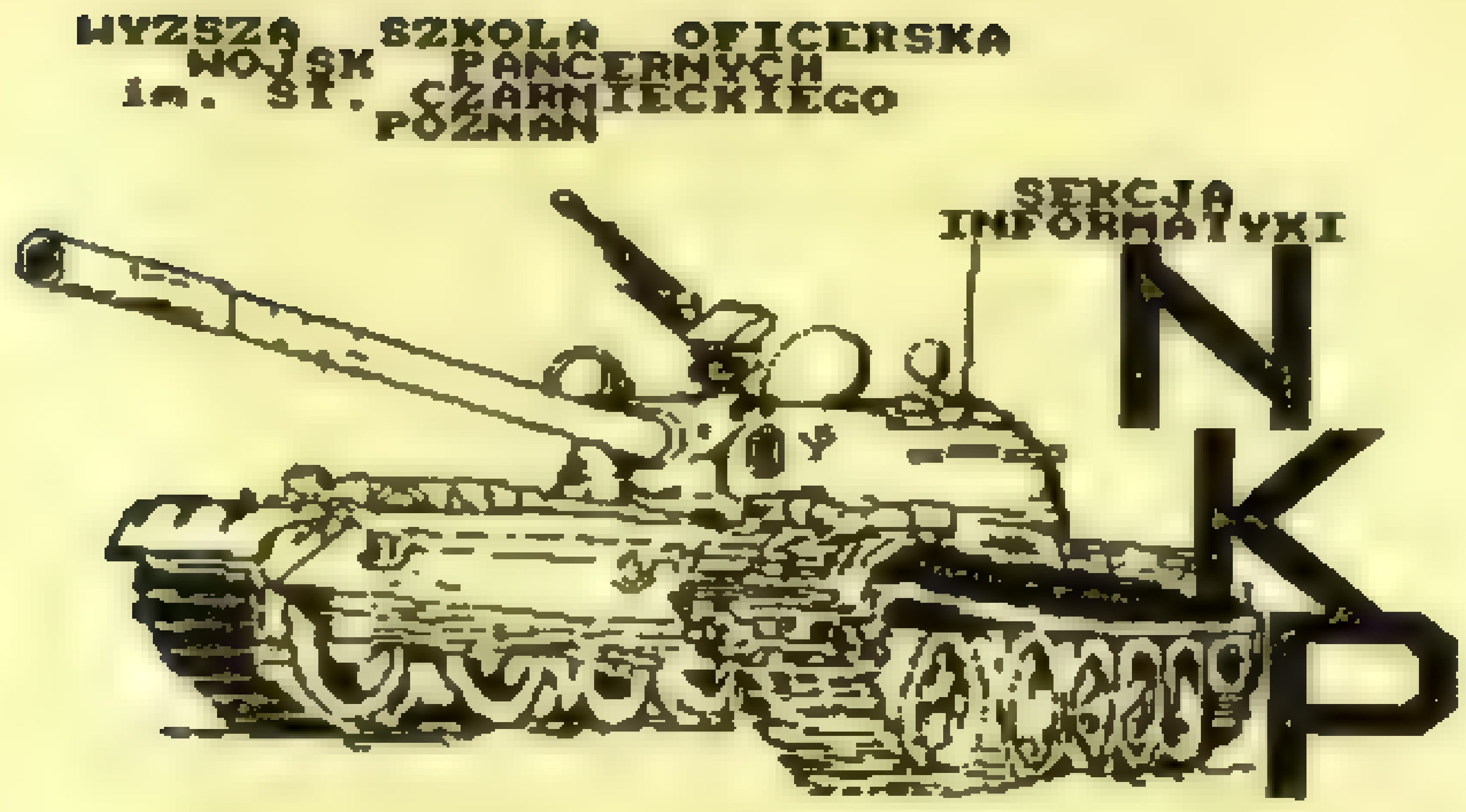
20—23 — oczekiwano "OF", ".", ";", "" "PROGRAM'

25 — oczekiwano "BEGIN'

opracował **CEZARY DOBROWOLSKI**

Czołgiści wśród mikrokomputerów

Foto: Jan Zelman



— *Zaczęło się to 5 lat temu* — wspomina por. **Mieczysław Skonieczny**, opiekun KNP — *szkoła dysponowała wówczas komputerem ze standardowym zestawem urządzeń zewnętrznych. Praca sekcji skupiała się na poznawaniu możliwości maszyny i podstaw informatyki. Dla nieprofesjonalistów były to sprawy chyba zbyt mało frapujące. Mieliśmy bowiem problemy z naborem chętnych.*

Punktem zwrotnym w działalności koła stało się wyposażenie szkoły w zestawy mikrokomputerowe. Zaczęli od trzech mikrokomputerów ZX Spectrum +. Rok temu otrzymali 8 zestawów Amstrad CPC 6128. Wówczas to zainteresowanie pracą sekcji informatyki, wśród podchorążych i kadetów znacznie wzrosło. Po ścisłej selekcji zaszczytu bycia jej członkiem dostąpiło 40 chętnych. Autorzy koncepcji uaktywnienia sekcji zadali sobie wiele trudu, opracowali zasady działalności, wzór statutu oraz prawa i obowiązki członków. Rozpisano również cele i zadania w szczegółach. Pierwsze tematy są łatwe: omówienie mikrokomputera Amstrad CPC 6128, ogólne uwagi o programowaniu, krótka charakterystyka języka BASIC czy omówienie zasad opracowania i edycji programu na mikrokomputerze. Później wzrasta stopień trudności, realizacja instrukcji w języku BASIC i instrukcje skoku, projekt rozwiązań graficznych ekranu, algorytm realizacji programu. Takie programowe zajęcia odbywają się 3—4 razy w miesiącu. Poza tym, członkowie sekcji uczestniczą w organizowaniu ważniejszych imprez na terenie uczelni. Ponad 4 miesiące trwał otwarty turniej szachowy, gdzie przeciwnikiem zawodników był mikrokomputer. Startowało w nim 60 szachistów. W wyniku pucharowego systemu wyłoniono pięciu zwycięzców. I tak to zabawa edukacyjna wpleciona została w proces nauczania.

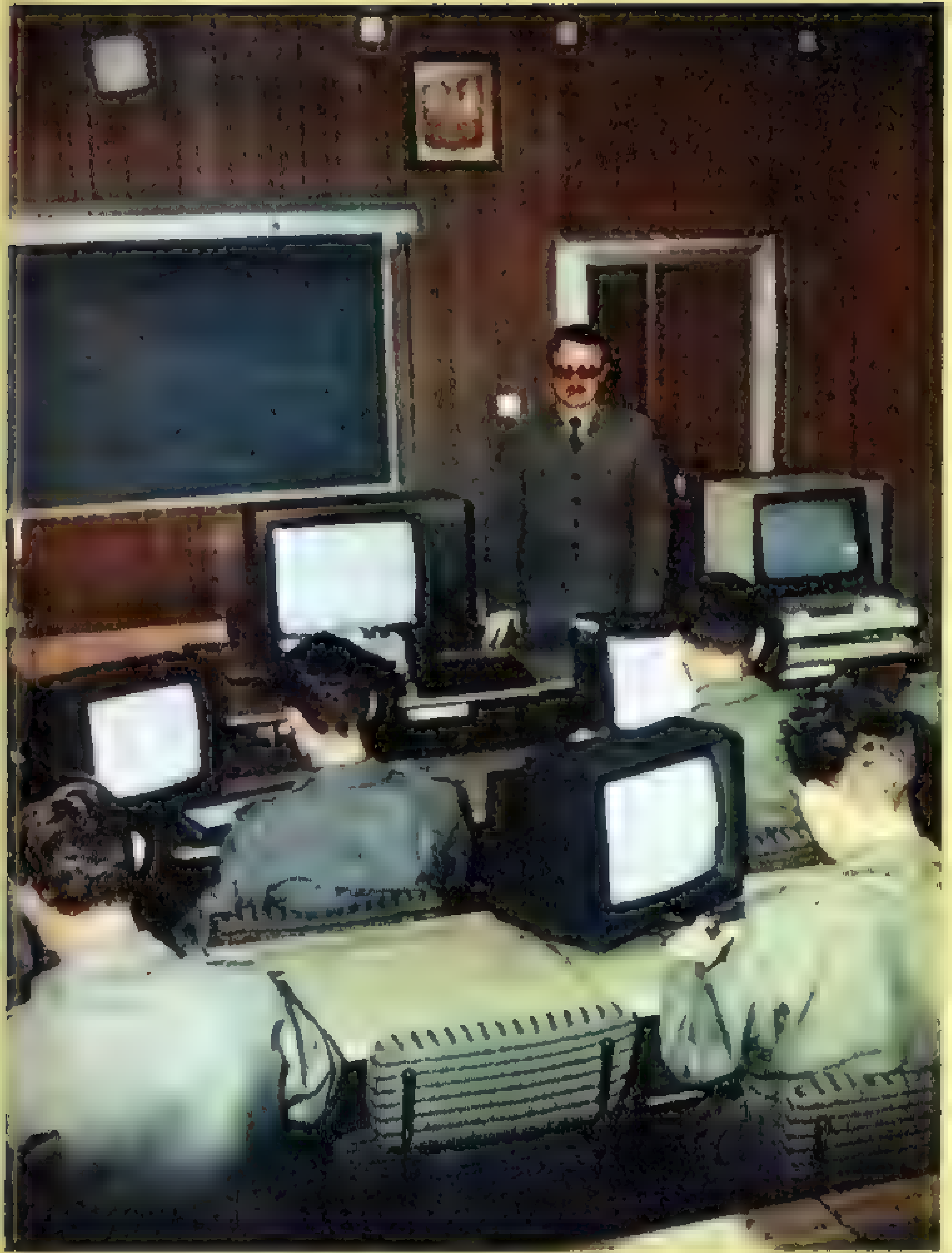
Sekcja informatyki KNP była również współorganizatorem ogólnokrajowego sympozjum naukowego nt. wykorzystania mikrokomputerów w systemie dydaktycznym. Członkowie różnych sekcji prezentowali wybrane programy dydaktyczne wspomagające proces nauczania. Pięć z nich opracowała sekcja informatyki. Trzeba przyznać, że podchorążym i kadetom stworzono bardzo dobre warunki do pracy naukowej. Posiadają odpowiednią salę komputerową, bibliotekę. Dobrym duchem sekcji jest jej kierownik st. kpr. kdt **Grzegorz Siankowski**. — *Moje zainteresowania* — mówi — *zrodziły się w czasie mikrokomputerowych gier. Najpierw była fascynacja samą grą, a później dochodzenie do tego, jak ona powstaje. Zaczęła się żmudna praca od podstaw: poznanie języka, komend, pisanie programu. Teraz najciekawsze jest oglądanie efektu programowania.*

Na przykład program „budowa i zasady działania mikrokomputera" tworzyli wspólnie z opiekunem sekcji prawie miesiąc. Bywało, że po ośmiu godzinach ślęczenia, oczy i głowa bolały, ale coś z tej pracy pozostało. I to cieszy, to wciąga bez reszty. Mieliśmy okazję obejrzenia kilku programów dydaktycznych i trzeba powiedzieć, że są to udane dzieła i o to przecież chodzi.

Jerzy RAUBE









Komputer u kierowców

Ten program opracowano w Wyższej Szkole Oficerskiej Wojsk Pancernych im. S. Czarnieckiego w Poznaniu. Przeznaczony jest do badania kierowców, a zastąpić może niekiedy urządzenia wykorzystywane w Wojskowej Poradni Psychotechnicznej.

W tym przypadku program symuluje pracę aparatu PIORKOWSKI służącego do oceny koordynacji wzrokowo-ruchowej i stopnia koncentracji uwagi kierowców.

Pakiet programów użytkowych obejmuje 4 segmenty:

— menu główne, które informuje użytkownika o możliwościach pakietu;
— program badający czas reakcji prostej i z wyborem kierowców;
— symulator aparatu PIORKOWSKI do badania koordynacji wzrokowo-ruchowej i stopnia koncentracji uwagi kierowców,
— program badający organizację przestrzenną i szybkość podejmowania decyzji przez kierowcę.

Kierowca poddawany jest testowi koordynacji wzrokowo-ruchowej i stopnia koncentracji uwagi w 2 etapach:

— etap próbny: emisja dowolnej ilości prób (w zależności od czasu reakcji badanego (w czasie 30 sekund);
— etap właściwy: emisja 150 prób w jednakowych odstępach czasu.

Etap próbny ma na celu przygotowanie badanego kierowcy do właściwej części testu i oswojenia użytkownika za sprzętem mikrokomputerowym. Na tym etapie nie jest badany czas reakcji. Kolejne losowe wyświetlenie lampki jest uzależnione od reakcji kierowcy.

Etap właściwy polega na losowym zapalaniu lampek w równych odstępach czasowych. Mikrokomputer zalicza ilość:

— poprawnych reakcji (badany kierowca zdążył nacisnąć właściwy klawisz),
— błędnych reakcji (badany kierowca nie zdążył nacisnąć właściwego klawisza lub nacisnął klawisz ze złym odpowiednikiem cyfrowym).

Po zakończeniu etapu właściwego są wyświetlane wyniki badania, które zawierają:

— sumaryczną ilość prób dobrych;
— sumaryczną ilość prób złych;
— ilość prób złych w 1 i 2 części etapu właściwego.

Etap właściwy podzielono na dwie części (każda po 75 prób), aby obiektywniej ocenić użytkownika uwzględniając proces zmęczenia badaniem. W dolnej części ekranu wyświetlana jest ocena słowna, określająca stopień przydatności do wykonywania zawodu kierowcy. Poniżej pokazuje się ocena w skali czterostopniowej.

Do segmentów menu głównego i programu właściwego dołączono podprogram, który umożliwia wypisywanie tekstów na ekranie monitora z uwzględnieniem liter charakterystycznych dla pisowni w języku polskim (np. ą, ę, ń, ł). Wpływa to na zwiększenie czytelności prezentowanych informacji przez użytkownika mikrokomputera. W programie wykorzystano duże możliwości kolorystyczne, dźwiękowe i graficzne mikrokomputera AMSTRAD CPC 6128.

Danuta KWASIŻUR, Mieczysław SKONIECZNY, Robert ZUGEHÓR

```
10 MODE 1:CLEAR INPUT:INK 0,0:INK 1,6:INK 2,24:BO
RDER 0:PAPER 0:CLS:GOSUB 900:WINDOW #1,1,40,10,16:
PAPER #1,2:CLS #1
20 PEN #1,0:LOCATE #1,2,4:PRINT#1,"APARATU 'PIORKO
WSKI' DO BADANIA KOORDY-":LOCATE #1,2,5:PRINT#1,"N
ACJI WZROKOWO-RUCHOWEJ I STOPNIA KON-"
30 LOCATE #1,2,6:PRINT#1,"CENTRACJI UWAGI KIEROWC6
```

```
W":SPEED INK 10,10
40 GRAPHICS PAPER 2:TAG:FOR x=550 TO -450 STEP -15
50 GRAPHICS PEN 1:PAPER 2:FRAME:MOVE x,240:PRINT " SYMULATOR MIKROKOMPUTEROWY ";
60 IF INKEY$=" " THEN 90
70 NEXT
80 GOTO 40
90 TAGOFF:MODE 1:GOSUB 800:PEN 2:PAPER 0:CLS:PRINT "Zadaniem programu jest symulacja badaniakierowc5w na aparacie typu 'PIORKOWSKI' wykorzystujlca mikrokomputer AMSTRAD    CPC 6128.
100 WINDOW #1,1,40,6,14:PAPER #1,2:CLS #1:PEN #1,0:LOCATE #1,2,1:PRINT #1,"Na ekranie monitora wy7wietli si[ obraz":LOCATE #1,2,2:PRINT#1,"10 ponumerowanych lampek. Zadaniem ba-"
110  LOCATE #1,2,3:PRINT #1,"danego jest nacis"ni[cie po zapaleniu":LOCATE #1,2,4:PRINT#1,"dowolnej lampki klawisza numerycznego":LOCATE #1,2,5:PRINT#1,"z odpowiadaj!c! lampce cyfr! w g5rnym":LOCATE #1,2,6:PRINT#1,"rz[dzie klawiatury mikrokomputera.Kie-"
120 LOCATE #1,2,7:PRINT #1,"rowca poddawany jest testowi koordyna-":LOCATE #1,2,8:PRINT #1,"cji wzrokowo-ruchowej i stopnia koncen-":LOCATE #1,2,9:PRINT#1,"tracji uwagi w dw5ch etapach:"
130 LOCATE 1,16:PRINT"1. Etap pr5bny - emisja dowolnej ilo7ci    pr5b (w zale9no7ci od czasu reakcji     badanego) w czasie 30 sek."
140 LOCATE 1,19:PRINT"2. Etap wla7ciwy - emisja 150 pr5b w je-   dnakowych odst[pach czasowych"
150 WINDOW #2,1,40,22,24:PAPER #2,3:PEN #2,0:CLS #2:LOCATE #2,2,1:PRINT #2,"Na zako3czenie badania kierowcy b[dzie":LOCATE #2,2,2:PRINT #2,"wy7wietlana jego ocena w skali cztero-":LOCATE #2,2,3:PRINT#2,"stopniowej"
160 IF INKEY$="" THEN 160
170 MODE 0:INK 1,26:WINDOW #1,4,17,12,14:PAPER #1,2:CLS #1:PEN #1,0:LOCATE #1,2,2:PRINT#1,"ETAP PR6BNY":INK 2,24,26
180 IF INKEY$="" THEN 180
190 INK 2,24:MODE 1:INK 2,14:INK 3,6:CLS:WINDOW #1,2,39,2,7:PAPER #1,2:CLS #1:PEN #1,0:WINDOW #2,1,36,10,13:PAPER #2,3:CLS #2:PEN #2,0:WINDOW #3,2,39,16,20:PAPER #3,2:CLS #3:PEN #3,0
200 LOCATE #1,2,2:PRINT #1,"Etap pr5bny ma na celu przygotowanie":LOCATE #1,2,3:PRINT#1,"badanego kierowcy do wla7ciwej cz[7ci":LOCATE #1,2,4:PRINT#1,"testu i oswojenie u9ytkownika ze":LOCATE #1,2,5:PRINT#1,"sprz[tem mikrokomputerowym."
210 LOCATE #2,2,2:PRINT#2,"NA TYM ETAPIE NIE JEST BADANY CZAS ":LOCATE #2,2,3:PRINT#2,"REAKCJI.":LOCATE #3,2,2:PRINT#3,"Kolejne, losowe wy7wietlenie lampki":LOCATE #3,2,3:PRINT#3,"jest uzale9nione od czasu reakcji ":LOCATE #3,2,4:PRINT#3,"kierowcy."
220 PEN 3:LOCATE 1,23:PRINT"Ten etap zako3czy si[ po uplywie 30 sek."
230 IF INKEY$=""THEN 230
240 GOSUB 1110
250 GOSUB 710
260 AFTER 1500,1 GOSUB 330
270 FOR g=1 TO 200:NEXT:GOSUB 790
280 a$=INKEY$:IF f=1 GOTO 340
290 IF a$="" THEN 280
300 IF nr$="10" THEN nr$="0"
310 IF NOT(nr$=a$) THEN GOTO 280
320 PAPER 0:INK nr,14:GOTO 270
330 f=1:s=REMAIN(1):RETURN
340 MODE 0:INK 0,0:PAPER 0:BORDER 0:CLS:GOSUB 800:INK 2,24:WINDOW #1,3,17,12,14:PAPER #1,2:CLS #1:PEN #1,0:LOCATE #1,2,2:PRINT #1,"ETAP W2A8CIWY":SPEED INK 20,20:INK 2,24,26
350 IF NOT(INKEY$=" ") THEN 350
360 INK 2,24:MODE 1:GOSUB 800:INK 0,0:INK 1,26:INK 2,14:INK 3,6:WINDOW #1,1,40,1,4:PAPER #1,2:CLS #1:PEN #1,0:LOCATE #1,1,2:PRINT#1,"Etap wla7ciwy polega na losowym zapala-":LOCATE #1,1,3:PRINT#1,"niu lampek w r5wnych odst[pach czasowych"
370 PAPER 0:PEN 1:LOCATE 1,6:PRINT"Mikrokomputer zlicza ilo7":":LOCATE 1,8:PRINT"- poprawnych reakcji (badany kierowca    zd!9yl nacisn!" wla7ciwy klawisz)     - bl[dnych reakcji (badany kierowca nie   zd!9yl nacisn!" wla7ciwego klawisza"
380 LOCATE 1,12:PRINT"  lub nacisn!l klawisz ze zlym odpowie-   dnikiem cyfrowym)"
390 LOCATE 4,15:PEN 3:PRINT "UWAGA!":WINDOW #2,1,40,17,25:PAPER #2,3:CLS #2:PEN #2,0:LOCATE #2,1,2:PRINT#2,"Mikrokomputer generuje numer wy7wietlanej lampki losowo tzn badany kierowca   nie mo9e przewidzie" kt5ra lampka zapa-"
400 LOCATE #2,1,5:PRINT#2,"li si[ w nast[pnej kolejno7ci. Jest mo- 9liwe 9e w nast[pnej pr5bie wy7wietli   si[ ta sama lampka. Efektem takiej sy-  tuacji na ekranie jest mignilcie tej    samej cyfry."
410 IF NOT (INKEY$=" ") THEN 410
420 GOSUB 1110
430 P=0:PEN 13:GOSUB 710
440 FOR r=1 TO 2:b=0
450 FOR n=1 TO 75:GOSUB 790
460 AFTER 70,1 GOSUB 700
470 CLEAR INPUT
480 a$=INKEY$:IF f=1 GOTO 540
490 IF a$="" THEN 480
500 IF nr$="10" THEN nr$="0"
510 IF nr$=a$ GOTO 530
520 b=b+1:GOSUB 690:GOTO 550
530 GOSUB 690:P=P+1
540 F=0
550 NEXT:bl(r)=b:NEXT
560 S=REMAIN(1):INK 1,26:INK 0,0:INK 2,24:INK 3,0:PAPER 2:BORDER 24:MODE 1:CLS:GOSUB 800:WINDOW #1,3,17,3,5:PAPER #1,1:CLS #1:PEN #1,0:LOCATE #1,2,2:PRINT#1,"WYNIKI TESTU":INK 0,0,24
570 PEN 3:LOCATE 3,7:PRINT "PRAWIDLOWO - ";:PRINT USING "###";P:LOCATE 3,9:PRINT "BLEDNIE    - ";:PRINT USING "###";Bl(1)+bl(2);
580 LOCATE 3,11:PRINT"1 CZ081 - ";USING "###":BL(1);:PRINT" BLEDOW":LOCATE 3,12:PRINT"2 CZ083 - ";USING "###":BL(2);:PRINT" BLEDOW"
590 INK 1,6:PEN 1:LOCATE 9,15:PRINT"##### OCENA #####":PEN 3
```

```
600 IF p>140 GOTO 660
610 IF p>124 GOTO 650
620 IF p>100  GOTO 640
630 LOCATE 1,17:PRINT"TWOJE WYNIKI S} BARDZO SZABE
.CZAS BADA- NIA KOORDYNACJI WZROKOWO-RUCHOWEJ I ST
O-PNIA KONCENTRACJI UWAGI DYSKWALIFIKUJE  CIO JAKO
 KIEROWCO WOJSKOWEGO. ZDECYDOWA-NIE MUSISZ ]WICZY]
.":PRINT:PEN 1:PRINT"UZYSKA2E8 OCENO  2":GOTO 670
640 LOCATE 1,17:PRINT"TWOJE WYNIKI S} S2ABE. CZAS
BADANIA     KOORDYNACJI WZROKOWO-RUCHOWEJ I STOPNI
A KONCENTRACJI UWAGI KWALIFIKUJE CIO JAKO KIEROWCO
 WOJSKOWEGO.ALE MUSISZ ZDECYDO- WANIE ]WICZY].":PR
INT:PEN 1:PRINT"UZYSKA2E8 OCENO  3":GOTO 670
650 LOCATE 1,17:PRINT"TWOJE WYNIKI S} DOBRE. CZAS
BADANIA     KOORDYNACJI WZROKOWO-RUCHOWEJ I STOPNI
A KONCENTRACJI UWAGI KWALIFIKUJE CIO JAKO KIEROWCO
 WOJSKOWEGO.NIE ZASZKODZI JEDNAKJAK BODZIESZ ]WICZ
Y2.":PRINT:PEN 1:PRINT"UZYSKA2E8 OCENO  4":GOTO 67
0
660 LOCATE 1,17:PRINT"TWOJE WYNIKI S} BARDZO DOBRE
. CZAS BADA-NIA KOORDYNACJI WZROKOWO-RUCHOWEJ I ST
O-PNIA KONCENTRACJI UWAGI KWALIFIKUJE CIO JAKO KIE
ROWCO WOJSKOWEGO. NIE MUSISZ    ]WICZY].":PRINT:PE
N 1:PRINT"UZYSKA2E8 OCENO  5":GOTO 670
670 IF NOT(INKEY$="s" OR INKEY$="S") THEN 670
680 GOTO 10
690 s=REMAIN(1):PAPER 0:INK nr,14:RETURN
700 SOUND 1,2000:P=P+1:GOSUB 690:F=1:RETURN
710 MODE 0:INK 11,0:INK 0,1:GRAPHICS PEN 11:INK 12
,14:PEN 13,0:BORDER 1:PAPER 0:CLS
720 FOR a=1 TO 10
730 INK a,14:x=64*(a-1):PLOT x,172:DRAW x+62,172:P
LOT x,174:DRAW x+62,174:DRAW x+62,238:DRAW x,238:D
RAW x,174:PLOT x,240:DRAW x+62,240:MOVE x+4,180:FI
LL a
740 NEXT
750 PEN 11:c=1:b=0:FOR a=1 TO 19 STEP 2
760 PAPER c:LOCATE a,12:PRINT CHR$(125+b);CHR$(126
+b)
770 LOCATE a,13:PRINT CHR$(127+b);CHR$(128+b):b=b+
4:c=c+1
780 NEXT:f=0:RETURN
790 nr=1+INT(10*RND):nr$=RIGHT$(STR$(nr),1):INK nr
,6:RETURN
800 SYMBOL AFTER 170
810 SYMBOL 171,0,0,120,12,124,204,118,3
820 SYMBOL 172,24,60,102,102,126,102,102,3
830 SYMBOL 173,24,0,60,102,96,102,60,0
840 SYMBOL 174,24,60,102,192,192,102,60,0
850 SYMBOL 175,0,0,60,102,126,96,60,6
860 SYMBOL 176,254,98,104,120,104,98,254,3
870 SYMBOL 177,56,24,28,24,56,24,60,0
880 SYMBOL 178,240,96,112,96,226,102,254,0
890 SYMBOL 179,24,0,220,102,102,102,102,0
900 SYMBOL 180,24,198,230,246,222,206,198,0
910 SYMBOL 181,24,0,60,102,102,102,60,0
920 SYMBOL 182,12,56,108,198,198,108,56,0
930 SYMBOL 183,24,0,60,96,60,6,124,0
940 SYMBOL 184,24,60,96,60,6,102,60,0
950 SYMBOL 185,24,0,126,76,24,48,126,0
960 SYMBOL 186,254,198,140,126,48,102,254,0
970 SYMBOL 187,12,24,126,76,24,48,126,0
980 SYMBOL 188,48,254,172,24,50,102,254,0
990 SYMBOL 189,0,0,0,0,12,12,24
1000 KEY DEF 10,1,55,171,172
1010 KEY DEF 11,1,56,173,174
1020 KEY DEF 3,1,57,175,176
1030 KEY DEF 20,1,52,177,178
1040 KEY DEF 12,1,53,179,180
1050 KEY DEF 4,1,54,181,182
1060 KEY DEF 13,1,49,183,184
1070 KEY DEF 14,1,50,185,186
1080 KEY DEF 5,1,51,187,188
1090 KEY DEF 15,1,189
1100 RETURN
1110 SYMBOL AFTER 125
1120 SYMBOL 125,129,129,131,131,129,129,129,129
1130 SYMBOL 126,129,129,129,129,129,129,129,129
1140 SYMBOL 127,129,129,129,129,131,131,128,128
1150 SYMBOL 128,129,129,129,129,193,193,1,1
1160 SYMBOL 129,131,131,134,134,128,128,131,131
1170 SYMBOL 130,193,193,97,97,97,97,193,193
1180 SYMBOL 131,134,134,134,134,135,135,128,128
1190 SYMBOL 132,1,1,97,97,225,225,1,1
1200 SYMBOL 133,131,131,132,132,128,128,131,131
1210 SYMBOL 134,193,193,97,97,97,97,193,193
1220 SYMBOL 135,128,128,134,134,131,131,128,128
1230 SYMBOL 136,97,97,97,97,193,193,1,1
1240 SYMBOL 137,129,129,131,131,133,133,137,137
1250 SYMBOL 138,129,129,129,129,129,129,129,129
1260 SYMBOL 139,143,143,129,129,131,131,128,128
1270 SYMBOL 140,225,225,129,129,193,193,1,1
1280 SYMBOL 141,135,135,134,134,134,134,135,135
1290 SYMBOL 142,225,225,33,33,1,1,193,193
1300 SYMBOL 143,128,128,134,134,131,131,128,128
1310 SYMBOL 144,97,97,97,97,193,193,1,1
1320 SYMBOL 145,131,131,134,134,134,134,135,135
1330 SYMBOL 146,193,193,97,97,1,1,193,193
1340 SYMBOL 147,134,134,134,134,131,131,128,128
1350 SYMBOL 148,97,97,97,97,193,193,1,1
1360 SYMBOL 149,135,135,132,132,128,128,128,128
1370 SYMBOL 150,225,225,97,97,97,97,193,193
1380 SYMBOL 151,129,129,129,129,129,129,128,128
1390 SYMBOL 152,129,129,129,129,129,129,1,1
1400 SYMBOL 153,131,131,134,134,134,134,131,131
1410 SYMBOL 154,193,193,97,97,97,97,193,193
1420 SYMBOL 155,134,134,134,134,131,131,128,128
1430 SYMBOL 156,97,97,97,97,193,193,1,1
1440 SYMBOL 157,131,131,134,134,134,134,131,131
1450 SYMBOL 158,193,193,97,97,97,97,225,225
1460 SYMBOL 159,128,128,134,134,131,131,128,128
1470 SYMBOL 160,97,97,97,97,193,193,1,1
1480 SYMBOL 161,131,131,134,134,134,134,134,134
1490 SYMBOL 162,193,193,97,97,97,97,97,97
1500 SYMBOL 163,134,134,134,134,131,131,128,128
1510 SYMBOL 164,97,97,97,97,193,193,1,1
1520 RETURN
```

Grafika na ekranie (1)

Grafika komputerowa fascynuje większość użytkowników mikrokomputerów. Konstrukcja obrazu wymaga znajomości zasad jego definiowania i przekształcania. W prezentowanym cyklu przedstawimy programy przydatne do przedstawiania i przekształcania położenia punktów i linii. Konstrukcja tych programów oparta będzie na zastosowaniu matematyki w dziedzinie grafiki komputerowej. Każdy program poprzedzony będzie krótkim opisem zastosowanej metody matematycznej. Wyboru metod matematycznych stosowanych w grafice komputerowej dokonano na podstawie pracy. D. F. Rogers'a i I. A. Adams'a: „MATHEMATICAL ELEMENTS FOR COMPUTER GRAPHICS" (New York, McGraw-Hill Book Company).

PRZEDSTAWIANIE PUNKTÓW

Na płaszczyźnie punkt reprezentowany jest przez dwie współrzędne. Można je traktować jako elementy macierzy wierszowej [x y]. W przestrzeni trójwymiarowej każdy punkt przedstawia macierz wierszowa [x y z]. Można je również przedstawić w postaci macierzy kolumnowej $\begin{bmatrix} x \\ y \end{bmatrix}$ lub $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$.

PRZEKSZTAŁCENIA I MACIERZE

Macierzowy zapis współrzędnych punktu jest wygodny do:

— reprezentacji maszynowej,

— zastosowania algebry macierzowej, co umożliwia wykonywanie transformacji obrazu

Wiele fizycznych zadań można sprowadzić do następującej formuły. Niech będą dane macierze A i B oraz zależność AT = B. Łatwo znaleźć macierz transformacji $T = A^{-1}B$,

gdzie A^{-1} jest macierzą odwrotną do macierzy kwadratowej A.

Interpretacja podanej formuły w grafice komputerowej

A — macierz zdefiniowanych współrzędnych punktów (współrzędne pierwotne),

B — macierz przekształconych współrzędnych punktów (współrzędne po transformacji)

T — macierz transformacji.

Interpretacja mnożenia macierzy jako operatora geometrycznego stanowi podstawę matematycznych przekształceń wykorzystywanych w grafice komputerowej.

PRZEKSZTAŁCENIA PUNKTÓW

Rozważmy rezultat macierzowego pomnożenia macierzy [xy] odpowiadającej punktowi *p* oraz macierzy transformacji [2 x 2]

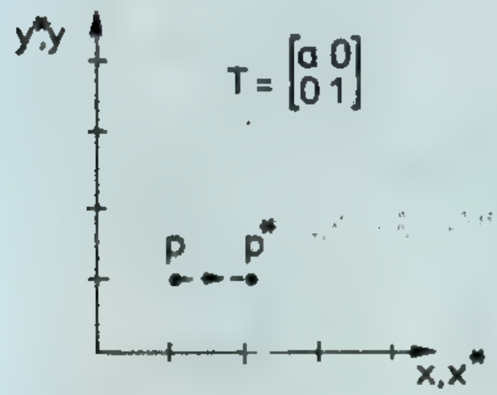
$$[xy]\begin{bmatrix} a & b \\ c & d \end{bmatrix} = [(ax + cy)\ (bx + dy)] = [x^*\ y^*]$$

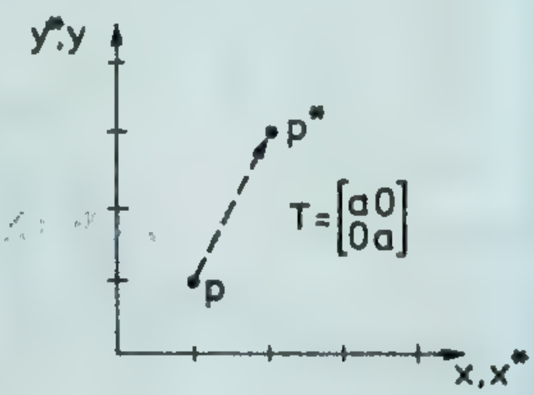
Ten matematyczny zapis oznacza, że współrzędne x i y przekształcone zostały w x^* i y^*, gdzie: $x^* = (ax + cy)$, $y^* = (bx + dy)$. Przeprowadźmy analizę otrzymanych wyników, rozpatrując x^* i y^* jako przekształcone (przetransportowane) współrzędne. Analiza ta pozwoli wyłonić podstawowe przekształcenia obrazów (zbiorów punktów).

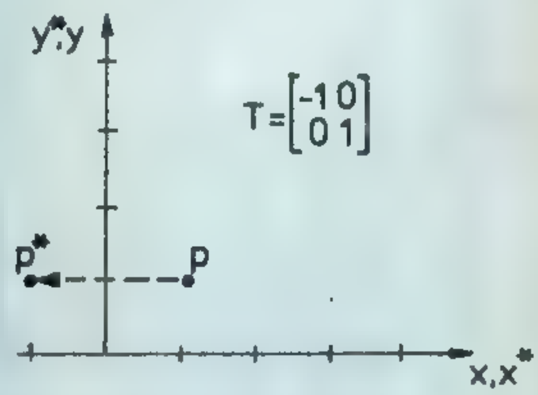
PRZYPADEK 1

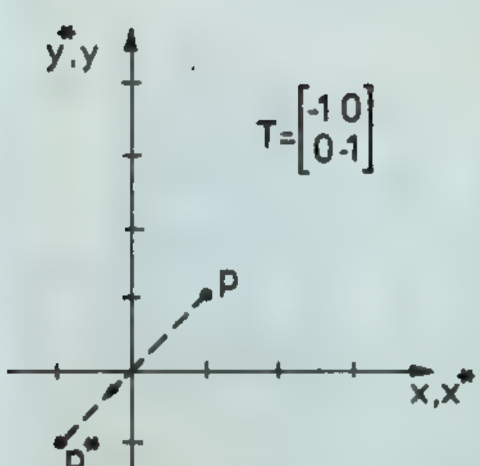
Niech d = 1, b = c = o.

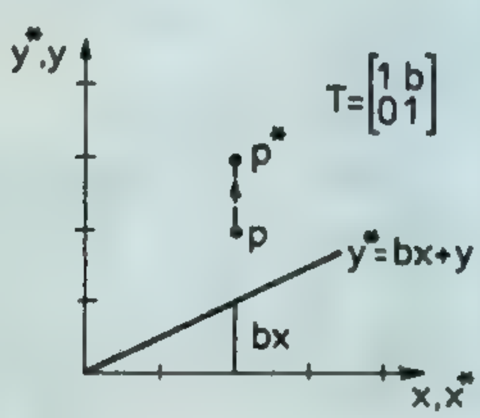
$$[xy]\begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} = [ax\ y] = [x^*y^*]$$

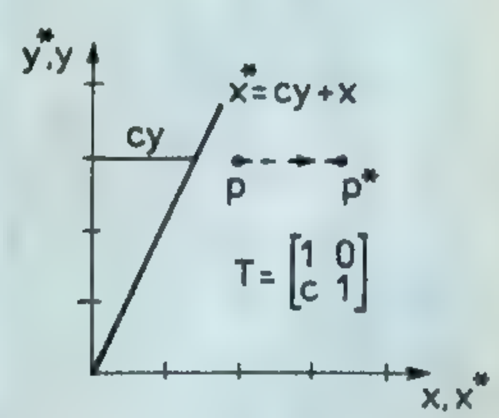












Przekształcenie takie — przedstawione na rys. 1 — nosi nazwę powinowactwa prostokątnego. W wyniku następuje przesunięcie punktu wzdłuż osi x.

PRZYPADEK 2

Niech b = c = 0

$[xy]\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = [ax\ dy] = [x^* y^*]$

Przekształcenie takie — przedstawione na rys. 2 — nazywamy skalowaniem w kierunku osi x i y.

Jeśli $a = d > 1$ to następuje zwiększenie współrzędnych punktu *p*

Jeśli $0 < a = d < 1$ to następuje zmniejszenie współrzędnych punktu *p*.

Gdy $a \neq b$ to skalowanie współrzędnych punktu *p* jest nierównomierne wzdłuż osi x i y.

PRZYPADEK 3

Niech $a = -1$, $b = c = 0$, $d = 1$

$[xy]\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = [-xy] = [x^* y^*]$

Przekształcenie takie — przedstawione na rys. 3 — nosi nazwę lustrzanego odbicia względem osi y.

PRZYPADEK 4

Niech $a = d = -1$, $b = c = 0$.

$[xy]\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = [-x\ -y] = [x^* y^*]$

Przekształcenie takie — przedstawione na rys. 4 — nazywamy symetrią środkową

PRZYPADEK 5

Niech $a = d = 1$, $c = 0$

$[xy]\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = [x\ (bx + y)] = [x^* y^*]$

Przekształcenie takie — przedstawione na rys. 5 — nosi nazwę skręcenia. Współrzędna y^* zależy liniowo od współrzędnych pierwotnych punktu *p*.

Analogiczne przekształcenie zachodzi, gdy $a = d = 1$ i $b = 0$. Ilustruje to rysunek 6.

Przyjmując, że położenie dowolnego odcinka może być przekształcone w dowolne nowe położenie prostą transformacją jego granicznych punktów, możemy wszystkie przedstawione powyżej transformacje wykorzystać do przekształcenia odcinków oraz figur z nich utworzonych.

Definiowanię figury złożonej z odcinków umożliwia nam zapis macierzowy, w kolejnych wierszach macierzy zapisujemy współrzędne punktów granicznych odcinków kreślących figurę

UWAGA

Wszystkie powyżej opisane przekształcenia z zastosowaniem ogólnej macierzy transformacji 2 x 2 dokonywane były względem początku układu współrzędnych określonego macierzą [Ø, Ø]. Jest to ograniczenie, które zostanie wyeliminowane w kolejnych odcinkach niniejszego cyklu

Przedstawiony na przykładzie punktu zbiór prostych przekształceń na płaszczyźnie obejmuje:
- powinowactwo prostokątne,
- skalowanie,
- lustrzane odbicie,
- symetrię środkową,
- skręcenie.

Zaprezentujemy teraz program umożliwiający wykonanie podstawowego przekształcenia na płaszczyźnie jakim jest skalowanie równomierne wzdłuż osi x i y. Dodatkowo program umożliwia realizację symetrii środkowej.

Program skalowania napisany został w języku Basic mikrokomputera Amstrad 6128. Użyto Basica ze względu na powszechną jego znajomość. Konstrukcja programu pozwoli łatwo przetłumaczyć go na inny język programowania. Pożądana jest jednak duża rozdzielczość ekranu z uwagi na zakres współrzędnych elementów obrazu. Program napisany został w sposób konwersacyjny. Jednakże ze względu na udostępnienie użytkującym jak największego zakresu współrzędnych obrazu, komunikacja programu z użytkownikiem sprowadzona została do jednego 40-znakowego wiersza. Stąd kilka słów komentarza. Rysowanie (definiowanie) obrazu odbywa się poprzez podawanie współrzędnych kolejnych punktów obrazu. Punkty te łączone są odcinkami. Po wprowadzeniu każdej współrzędnej punktu wypisywany jest komunikat *„koniec wprowadzania współ. —k, nie —n"*. Gdy chcemy zakończyć wprowadzanie współrzędnych naciskamy klawisz *K* oraz *ENTER*. W przeciwnym wypadku *N* oraz *ENTER*. Komunikat *„podaj skalę"* wypisywany jest po zakończeniu definiowania obrazu. Gdy wprowadzimy liczbę z zakresu (0, 1) to uzyskamy zmniejszenie skali. Liczba większa od 1 powoduje zwiększenie skali. Wprowadzając jako skalę liczbę ujemną uzyskujemy przekształcenie obrazu nazwane **SYMETRIĄ ŚRODKOWĄ**. Po wyskalowaniu obrazu, sygnalizowanym komunikatem *„wyskalowano: ponownie —r, koniec —k"*, uzyskujemy na ekranie monitora dwie figury. Jedną zdefiniowaną przez nas oraz drugą przekształconą operacją skalowania. Gdy chcemy kontynuować skalowanie naciskamy klawisz **r**. Wciśnięcie klawisza **K** kończy działanie programu. Wybierając kontynuację zauważymy komunikat *„kasowanie obrazu—k, rysowanie—r"*. Wybór rysowania powoduje definiowanie obrazu od początku. Kasowanie obrazu oznacza, że będziemy mogli pozostawić na ekranie jedną z obrazowanych figur. Wyboru dokonujemy po komunikacie *„nowego obrazu—n, starego—s"*. Nowy obraz oznacza figurę uzyskaną po przekształceniu uprzednio zdefiniowanej (starej). Po wyborze figury do skasowania wyłania się nam kolejna możliwość sygnalizowana komunikatem *„skalowanie—s, dorysowanie—d"*. Wybierając skalowanie dokonujemy ponownego skalowania obrazu. Dorysowanie umożliwia kontynuację definiowania kolejnych elementów obrazu.

Andrzej CETERA

```
30 REM SKALA
60 DIM wsp(101,2),mat(2,2),nwsp(101,2)
70 MODE 1:INK 0,13:INK 1,3:INK 2,12:INK 3,0
80 BORDER 0:PEN 1:PAPER 2
90 CLS
100 PRINT "Skalowanie obrazu w ukladzie"
110 PRINT "wspolrzednych:-319<x<319,-191<y<191"
120 PRINT
130 INPUT "Gdy jestes gotow to wcisnij t"; nk$
140 IF nk$="t" THEN 160
150 GOTO 130
160 GRAPHICS PAPER 1:REM Tlo grafiki
170 ORIGIN 320,208,0,640,8,400
180 CLEAR INPUT
190 WINDOW 1,40,25,25
200 GOSUB 1200:REM Rysowanie ukladu wspolrzed.
210 i=0 :REM Licznik punktow
220 INPUT "Podaj wsp. punktu x,y";x,y
230 i=i+1
240 IF i>100 THEN GOTO 370
250 wsp(i,1)=x
260 wsp(i,2)=y
270 GOSUB 790:REM wykreslanie punktu lub odcinka
280 INPUT "koniec wprowadzania wspol.-k,nie-n";nk$
290 IF nk$="n" THEN 220
```

```
300 IF nk$="k" THEN 320
310 GOTO 280
320 INPUT "Podaj skale";p
330 GOSUB 860 :REM Ustaw macierz przeksz.
340 GOSUB 1000 :REM Mnozenie macierzy
350 GOSUB 1110 :REM Zobrazowanie
360 GOTO 390
370 INPUT "Juz za duzo punktow:skala";p
380 GOTO 330
390 INPUT "wyskalowano:ponownie-r,koniec-k";nk$
400 IF nk$="r" THEN 430
410 IF nk$="k" THEN 1400
420 GOTO 390
430 INPUT "kasowanie obrazu-k,rysowanie-r";nk$
440 IF nk$="k" THEN 470
450 IF nk$="r" THEN 680
460 GOTO 430
470 INPUT"nowego obrazu-n,starego-s";nk$
480 IF nk$="n" THEN 630
490 IF nk$="s" THEN 510
500 GOTO 470
510 REM Kasowanie starego obrazu
520 GOSUB 1200:REM Rys. ukladu wspolrzednych
530 FOR k=1 TO i
540 wsp(k,1)=nwsp(k,1)
550 wsp(k,2)=nwsp(k,2)
560 NEXT k
570 GRAPHICS PEN 3
580 GOSUB 1300
590 INPUT "skalowanie-s,dorysowanie-d";nk$
600 IF nk$="s" THEN 320
610 IF nk$="d" THEN 220
620 GOTO 590
630 REM Kasowanie nowego obrazu
640 GOSUB 1200
650 GRAPHICS PEN 3
660 GOSUB 1300
670 GOTO 590
680 FOR n=1 TO i:REM Rysowanie od poczatku
690 FOR k=1 TO 2
700 wsp(n,k)=0
710 nwsp(n,k)=0
720 NEXT k
730 NEXT n
740 GOTO 200
750 REM ****************************************
760 REM     PODPROGRAMY
770 REM ****************************************
780 REM
790 REM Kreslenie punktu lub odcinka
800 REM
810 GRAPHICS PEN 3:REM Stare wspolrzedne
820 IF i=1 THEN PLOT wsp(i,1),wsp(i,2)
830 IF i>1 THEN DRAW wsp(i,1),wsp(i,2)
840 RETURN
850 REM
860 REM Ustawianie macierzy przeksztalcenia
870 REM
880 FOR k=1 TO 2
890 mat(k,k)=p
900 NEXT k
910 mat(1,2)=0
920 mat(2,1)=0
930 FOR k=1  TO i:REM Zerowanie nwsp
940 FOR l=1 TO 2
950 nwsp(k,l)=0
960 NEXT l
970 NEXT k
980 RETURN
990 REM
1000 REM Mnozenie macierzy
1010 REM
1020 FOR n=1 TO i
1030 FOR k=1 TO 2
1040 FOR m=1 TO 2
1050 nwsp(n,k)=nwsp(n,k)+wsp(n,m)*mat(m,k)
1060 NEXT m
1070 NEXT k
1080 NEXT n
1090 RETURN
1100 REM
1110 REM Rysowanie nowej figury
1120 REM
1130 GRAPHICS PEN 0:REM Nowe wspolrzedne
1140 PLOT nwsp(1,1),nwsp(1,2)
1150 FOR n=2 TO i
1160 DRAW nwsp(n,1),nwsp(n,2)
1170 NEXT
1180 RETURN
1190 REM
1200 REM Rysowanie ukladu wspolrzednych
1210 REM
1220 CLG 1
1230 GRAPHICS PEN 2:REM Pioro ukladu wspol.
1240 PLOT 0,-208
1250 DRAW 0,208
1260 PLOT -320,0
1270 DRAW 320,0
1280 RETURN
1290 REM
1300 REM Rysowanie starej figury
1310 REM
1320 PLOT wsp(1,1),wsp(1,2)
1330 FOR n=2 TO i
1340 DRAW wsp(n,1),wsp(n,2)
1350 NEXT n
1360 RETURN
1370 REM
1380 REM Koniec
1390 REM
1400 WINDOW 1,40,1,25
1410 CLS
1420 CLG 1
1430 END
```

Spectrum trwa nadal

Po Spectrum +2 na rynek wypuszczono kolejną wersję tego komputera Spectrum +3. Od poprzednika różni się przede wszystkim stacją dysku, która podobnie, jak w Amstradzie CPC 6128 jest integralną częścią klawiatury. Nowe Spectrum ma pamięć RAM 128 Kb ROM 64 Kb. Oczywiście tak jak wszystkie komputery posiada możliwość dołączenia kolejnej stacji dysków, drukarki i dwóch joysticków. Tyle danych katalogowych. Innych walorów użytkowych nie znamy.



Zmienne systemowe (ROM–SPECTRUM)

Krzysztof MAMCARZ

Kalkulator

MEM	23656/7
MEMBOT	23698..23727
BREG	23655

Kalkulator możemy uważać za odrębną część systemu operacyjnego wykonującą działania arytmetyczne i operacje na łańcuchach tekstowych. Jest swoistym „państwem w państwie". Posługuje się własnym „językiem" (tzw. literały— ang. literals) i korzysta z własnego stosu. Literały to 8-bitowe rozkazy wskazujące odpowiednią procedurę w ROM-ie. Umieszcza się je sekwencyjnie po instrukcji RST 40, która inicjuje pracę kalkulatora, sygnałem kończącym listę jest literał 56. Wszystkie operacje kalkulatora są przeprowadzone na 5-bajtowych reprezentacjach liczb zwanych liczbami zmiennoprzecinkowymi (ang. floating-point numbers). W dalszej części tak właśnie będzie rozumiane pojęcie liczby.

Operacje, wykonywane na liczbach nazywa się arytmetyką zmiennoprzecinkową. Aby w skuteczny sposób posługiwać się nią, nie trzeba wcale umieć przekształcać liczbę rzeczywistą na jej 5-bajtową postać. Poniższe przykłady wyjaśnią kilka podstawowych sztuczek z kalkulatorem.

Najważniejsza sprawa to kontrola stosu. Jego dno wskazuje zawsze zmienna **STKBOT** a szczyt **STKEND**. Poszczególne operacje przeprowadza się zgodnie z notacją Łukasiewicza, co w praktyce sprowadza się do tego, że wysłanie kodu operacji powinno być poprzedzone umieszczeniem na stosie odpowiedniej ilości parametrów. Przykładowo, aby dodać dwie liczby do siebie należy wysłać na stos pierwszą, potem drugą, a na końcu literał dodawania. Można wyróżnić następujące typy operacji:

1) pojedyncze, gdy liczba z wierzchołka stosu podlega obliczeniom i jest zastępowana przez wynik np. *sin, abs, not, truncate*,

2) podwójne, gdy w obliczeniach uczestniczy para liczb z wierzchołka stosu i jest zastępowana przez wynik np. *multiply, division* lub przez dwie liczby będące wynikiem np. *modulo*;

3) manipulacyjne, gdy liczba ze szczytu stosu jest przekazywana do obszaru pamięci kalkulatora lub dorzucana na stos np. *st-mem-1, get-mem-1, stk-1*;

4) wieloczynnościowe (ang. multi-purpose) — w przypadku generowania ciągów liczbowych np. współczynników wielomianów Czebyszewa

Pierwszym narzucającym się problemem jest przekazanie liczby z poziomu BASIC'A na stos kalkulatora. Można to zrobić przez zdefiniowanie funkcji użytkownika i odczyt **DEFADD** przez oznaczenie zmiennej (LET) i odczyt **DEST**. Ten drugi sposób sprawia nieco kłopotów, lecz zaczniemy właśnie od niego. Poniższy zestaw instrukcji asemblera GENS3M2 umieszcza na stosie kalkulatora liczbę rzeczywistą przekazaną ze zmiennej Basic'a. Literały są podawane dziesiętnie.

```
  5 ; ** DEST '87
 10               ORG   62000
 20 DEST          EQU   23629
 30 CHADD         EQU   23645
 40 MEMBOT        EQU   23698
 50 MEM           EQU   23656
 60 SEED          EQU   23670
 70 LOOKVS        EQU   #28B2
 80               LD    HL,(CHADD)
 90               LD    (SEED),HL
100               LD    HL,(DEST)
110               BIT   1,(IY+55)
120               JR    NZ,NEW
130               BIT   5,(HL)
140               JR    Z,NONS
150               BIT   6,(HL)
160               JR    NZ,OLD
170 NEW           LD    (CHADD),HL
180               CALL  LOOKVS
190               JR    C,VARNOT
200 OLD           INC   HL
210               LD    (MEM),HL
240               RST   40
250 getmem        DEFB  224
260 end           DEFB  56
270               LD    HL,MEMBOT
280               LD    (MEM),HL
290               POP   BC
300               LD    HL,(SEED)
310               LD    (CHADD),HL
320               RET
330 NONS          RST   8
340               DEFB  11
350 VARNOT        RST   8
360               DEFB  1
```

Po wczytaniu instrukcji np.:
LET liczba = 9.81: **PRINT USR** 62000
nastąpi uruchomienie procedury, która wykona następujące czynności:

linie 80..90	—	przechowanie adresu interpretowanego wyrażenia w **SEED**,
linie 100..110	—	odczyt **DEST** i flagi charakteryzującej wczytaną po **LET** zmienną;
linie 120..160	—	test typu zmiennej BASIC'a, jeśli już istnieje;
linie 170..190	—	odszukanie adresu zmiennej w pamięci, jeśli jest nowa lub wieloliterowa oznaczona wcześniej;
linia 210	—	przełączenie obszaru pamięci kalkulatora na początek liczby
linie 240..260	—	wejście do kalkulatora i przekazanie liczby z pamięci na stos,
linie 270..280	—	odtworzenie oryginalnej wartości zmiennej MEM,
linie 290..320	—	odtworzenie adresu przerwanej interpretacji i powrót do BASIC'a, jeśli to potrzebne;

Zmienna systemowa **MEM**, która odgrywa tu główną rolę wskazuje obszar pamięci kalkulatora. Zajmuje on 30 bajtów od **MEMBOT** w górę, co oznacza, że mieści 6 liczb. W pamięci mogą być składane liczby ze stosu, które będą potrzebne w innym momencie. Operacje tego typu wykonują literały *st-mem* i *get-mem* zaopatrzone w numer sektora. Sektorów jest 6 (gdyż tyle może zmieścić pamięć) ponumerowanych od 0 do 5.

▶

Korzystając z tego, że wśród literałów znajduje się procedura modulo, spróbujmy zaimplementować funkcję m-modulo-n. Określmy zatem:

10 **DEF FN** m (m,n) = **USR** 64000

Chcąc obliczyć wartość 5 mod 2 należy przykładowo umieścić linie:

20 **LET** mod=**FN** m (5,2)
30 **PRINT** mod

Tekst źródłowy asemblera wygląda następująco:

```
 10 ; ** MODULO '87
 20 ;GENS3M2 assembler
 30                 ORG   64000
 40 DEFADD          EQU   23563
 50 STK_NR          EQU   #33B4
 60                 LD    HL,(DEFADD)
 70                 INC   HL
 80                 INC   HL
 90                 CALL  STK_NR
100                 CALL  TEST
110                 LD    HL,(DEFADD)
120                 LD    BC,10
130                 ADD   HL,BC
140                 CALL  STK_NR
150                 CALL  TEST
160                 RST   40
170 mod             DEFB  50
180 delete          DEFB  2
190 end             DEFB  56
200                 POP   HL
210                 RET
220 TEST            RST   40
230 duplic          DEFB  49
240 less0           DEFB  54
250 jptrue          DEFB  0
260                 DEFB  8 ;do 340
270 duplic          DEFB  49
280 duplic          DEFB  49
290 trunc           DEFB  58
300 subtr           DEFB  3
310 not             DEFB  48
320 jptrue          DEFB  0
330                 DEFB  4 ;do 370
340 end             DEFB  56
350                 RST   8
360                 DEFB  9 ;inv. arg.
370 end             DEFB  56
380                 RET
```

Opis:

linie 60..90	— przekazanie parametru m na stos kalkulatora (adres 5-bajtowej liczby wskazuje para HL);
linia 100	— sprawdzenie bezpośrednio na stosie, czy liczba jest dodatnia i całkowita;
linie 110..150	— przekazanie parametru *n* na stos;
linie 160..190	— operacje na stosie; mod odkłada dwie liczby, z których pierwsza to m-mod-n, a druga wykazuje ile razy n mieści się w m (stąd operacja delete);
linie 200..210	— powrót do BASIC'a z pominięciem procedury umieszczającej na stosie zawartość rejestrów **BC**;

Skompilowany kod nie jest relokowalny w przeciwieństwie do poprzedniego. Etykiety składające się z małych liter są rozkazami kalkulatora i spełniają rolę poglądową. Nie trzeba ich wpisywać.

Kolejny przykład również wymaga określenia funkcji użytkownika

13 **DEF FN** s (a) = **USR** 60000

Rzecz dotyczy procedury obliczającej silnię liczby całkowitej nieujemnej. Po instrukcji np.

PRINT FN s (9)

nastąpi umieszczenie parametru na stosie kalkulatora (linie 60..90) oraz obliczenia (linie 100..270).

```
 10 ; ** SILNIA '86
 20 ;GENS3M2 assembler
 30                 ORG   60000
 40 DEFADD          EQU   23563
 50 STK_NR          EQU   #33B4
 60                 LD    HL,(DEFADD)
 70                 INC   HL
 80                 INC   HL
 90                 CALL  STK_NR
100                 RST   40
110 duplic          DEFB  49
120 not             DEFB  48
130 jptrue          DEFB  0
140                 DEFB  13 ;do 270
150 duplic          DEFB  49
160 stk1            DEFB  161
170 subtr           DEFB  3
180 stmem0          DEFB  192
190 not             DEFB  48
200 jptrue          DEFB  0
210                 DEFB  11 ;do 300
220 gtmem0          DEFB  224
230 multip          DEFB  4
240 gtmem0          DEFB  224
250 jump            DEFB  51
260                 DEFB  -10 ;do 150
270 end             DEFB  56
280                 LD    BC,1
290                 RET
300 end             DEFB  56
310                 POP   BC
320                 RET
```

Na początku parametr jest sprawdzany. Jeśli jest zerem, następuje powrót do BASIC'a z jedynką na stosie. Jeśli nie jest zerem wykonuje się pętla w liniach 150..260. Na dnie stosu zawsze znajduje się wynik kolejnych operacji mnożenia go przez parametr pomniejszony o jeden. W chwili, gdy doszłoby do mnożenia przez zero procedura kończy pracę (linie 300..320). Ponieważ możliwości Spectrum w zakresie przechowywania liczb są ograniczone, więc najwyższa liczba, której silnię można poznać wynosi 33. Poruszając się tylko w języku wewnętrznym również można dokonywać operacji na liczbach rzeczywistych i to w sposób całkiem łatwy. Oto przykład podniesienia stałej e do potęgi *pi*

```
 10 ; ** E^PI '87
 20                 ORG   61200
 30 CHADD           EQU   23645
 40 DEC_FP          EQU   #2CB8
 50                 LD    HL,E_CON
 60                 LD    (CHADD),HL
 70                 LD    A,(HL)
 80                 CALL  DEC_FP
 90                 LD    HL,PI_CON
100                 LD    (CHADD),HL
```

```
110                 LD   A,(HL)
120                 CALL DEC_FP
130                 RST  40
140                 DEFB 6
150                 DEFB 56
160                 POP  HL
170                 RET
180 PI_CON          DEFM "3.14159265"
190                 DEFB 13
200 E_CON           DEFM "2.71828183"
210                 DEFB 13
```

Przekazanie obu stałych na stos odbywa się na zasadzie interpretacji kolejnych cyfr umieszczonych w postaci znaków ASCII. Listę znaków kończy bajt 13 lub inny, różny od kodów cyfr, kropki lub litery *e*. Część interpretacyjną wykonuje procedura ROM umieszczając jednocześnie liczby na stosie. Po wejściu do kalkulatora wykonywana jest tylko operacja potęgowania (linia 140). Chcąc sprawdzić wynik należy wpisać.

PRINT USR 61200

Zmienna systemowa **BREG** pełni dość ciekawą rolę w działaniu kalkulatora. Analogia do rejestru **B** mikroprocesora nie jest przypadkowa, gdyż **BREG** wspólnie z literałem *dec-jr-nz* stanowi symulację pętli. Wejście do kalkulatora w czasie pętlowania powinno odbywać się przez **CALL** 3362 (hex.), **a nie RST 40**.

Na zakończenie wykaz użytych w programach rozkazów kalkulatora:

literał	nazwa	znaczenie
224	*get-mem-0*	przesyła na stos kalkulatora liczbę z obszaru pomięci (sektor 0);
50	*modulo*	oblicza m-modulo-n;
2	*delete*	obniża wierzchołek stosu o 5 bajtów powodując pozbycie się liczby;
56	*end-calc*	wskaźnik końca listy literałów; wyjście z kalkulatora;
49	*duplicate*	zdublowanie liczby na szczycie stosu;
54	*less 0*	test znaku liczby; wynikiem operacji jest wartość logiczna 1, jeśli liczba jest ujemna;
0	*jptrue*	skok, jeśli liczba na stosie nie jest zerem (fałszem); po literale umieszcza się adres względny, od którego mają być kontynuowane operacje kalkulatora;
58	*truncate*	powoduje pozbawienie liczby jej części ułamkowej (tzw. zaokrąglenie w kierunku zera);
3	*subtract*	odejmowanie dwu liczb na stosie;
48	*not*	wynikiem tej operacji jest 0, jeśli liczba na stosie nie jest zerem lub 1, jeśli jest;
161	*stk-1*	umieszczenie na stosie liczby 1;
192	*stk-mem-0*	odłożenie do pamięci (sektor 0) liczby ze szczytu stosu (stos nie ulega obniżeniu);
4	*multiply*	mnożenie dwu liczb na stosie;
51	*jump*	skok bezwarunkowy do adresu względnego podanego za literałem;
6	*to-power*	podniesienie liczby pod wierzchołkiem stosu do potęgi liczby znajdującej się na szczycie stosu.

Oczywiście podane przykłady, ani wyjaśnienia nie wyczerpują wszystkich możliwości kalkulatora, natomiast kończą listę opisywanych zmiennych systemowych ROM, które były pretekstem do zagłębienia się we wnętrze Spectrum...

Krzysztof MAMCARZ

PROGRAM 23

ATARI

Migający kursor w mikrokomputerze

Przedstawiony program uruchamia pulsowanie kursora w mikrokomputerze ATARI 800 XL. Dodatkowo pulsują również napisy wykonane w technice INVERSE VIDEO, co może być przydatne w innych programach w celu zwrócenia uwagi na pewne fragmenty tekstu.

Program po wpisaniu i uruchomieniu można wykasować instrukcją NEW. W pamięci mikrokomputera pozostanie wtedy procedura w języku maszynowym i program nadal będzie działał. W przypadku użycia klawisza RESET program można ponownie wywołać instrukcją:

I = USR (30 000).

Drugi z przedstawionych programów jest napisany w asemblerze i analizując go można łatwo zrozumieć, co powoduje miganie kursora.

Tomasz WOJTCZAK

```
DM 10 REM ... MIGAJACY KURSOR ...
BI 15 REM
RS 20 FOR A=30000 TO 30036
YH 25 READ D
UU 30 POKE A,D
GH 35 NEXT A
BP 40 I=USR(30000):REM UAKTYWNIENIE
HM 45 REM PROCEDURY W JEZYKU MASZYN
OWYM
RT 90 DATA 104,169,59,141,40,2,169,
117
UN 91 DATA 141,41,2,169,18,141,26,2
EC 92 DATA 173,243,2,201,2,240,7,23
8
BK 93 DATA 243,2,238,243,2,96,206,2
43
AP 94 DATA 2,206,243,2,96
BU 97 REM
AT 98 REM ... napisane w HACKER-STU
DIO,
XV 99 REM .   a dziala ?
```

Grafika C-64 (cz. IV)

Sławomir WASIELEWSKI

Opisane zostały już podstawowe tryby pracy mikrokomputera Commodore 64: znakowy w części II („IKS" nr 3, 1987) i graficzny w części III („IKS" nr 6, 1987). SPRITE, duszek, MOB, blokowy obiekt ruchomy są to nazwy używane dla określenia kolejnego trybu pracy C-64. W trybie tym mamy możliwość niezależnego definiowania i prostego sterowania ośmioma obrazkami graficznymi. Możemy łączyć go z dwoma pozostałymi trybami pracy. Cecha ta staje się bardzo ważna przy programowaniu specjalnych efektów graficznych (szczególnie w grach). Blokowy obiekt ruchomy należy traktować, jako pewnego rodzaju narzędzie graficzne wspomagające programistę.

SPRITE

SPRITE jest obiektem graficznym definiowalnym na powierzchni o wymiarach 24 x 21 punktów świetlnych ekranu. Nie wgłębiając się na razie w szczegóły wykonajmy następujące instrukcje (w trybie bezpośredniej pracy C-64) i zaobserwujmy ich efekt na ekranie:

POKE 53248, 150: POKE 53249, 150: POKE 53269, 1

Pokazał się nam obrazek o dziwacznych kształtach. Jest to jeden z ośmiu dostępnych ruchomych obiektów nie posiadający na razie sensownej postaci. Zmieniając wartości bajtów 53248 i 53249 możemy go przemieszczać po ekranie. Wykonanie instrukcji

POKE 53269, 0

powoduje zniknięcie obiektu.

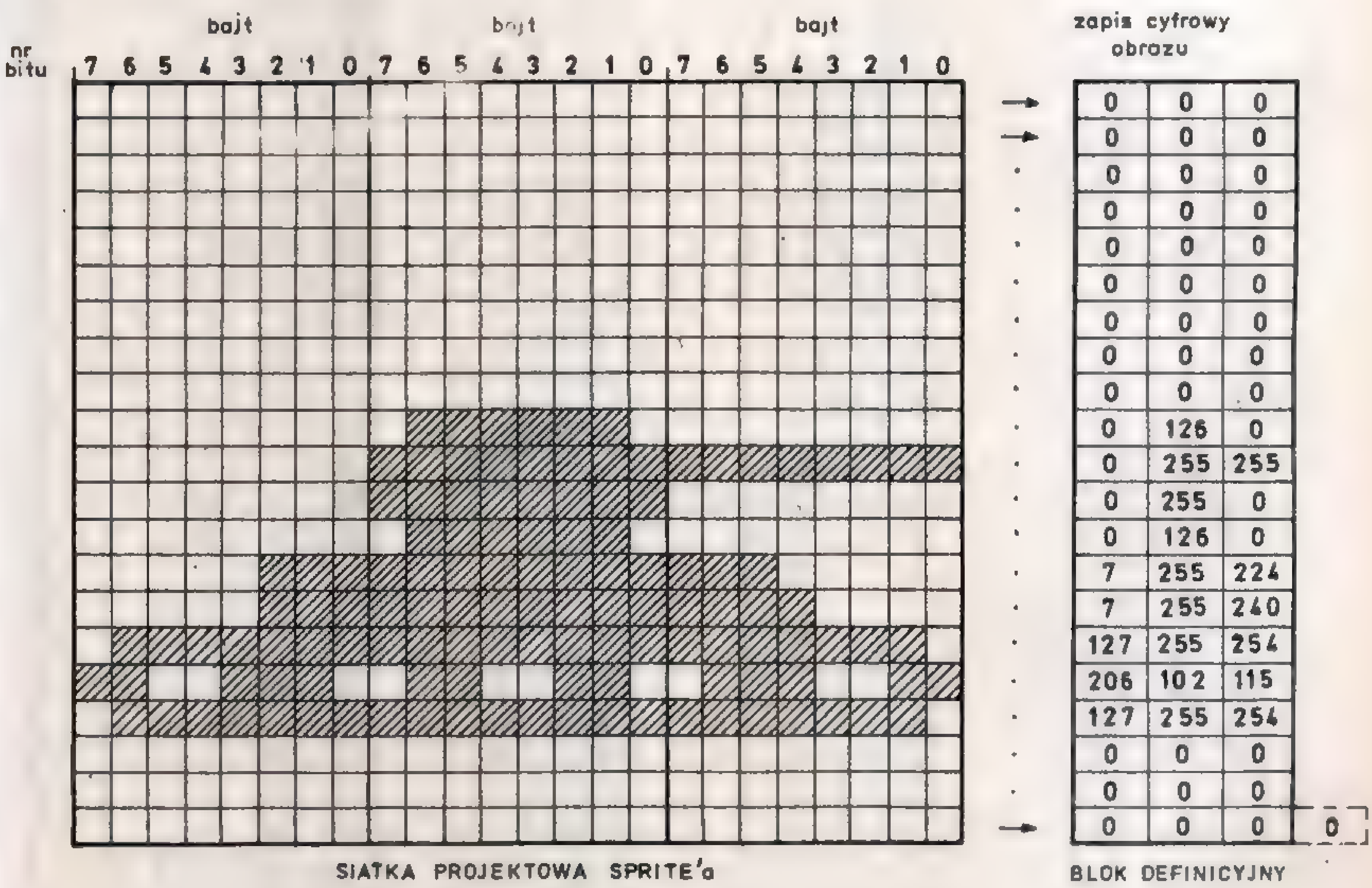
Do konstrukcji obrazu SPRITE'a na ekranie VIC wykorzystuje: blok definicyjny wskazany przez wskaźnik bloku oraz rejestry sterujące.

Blok definicyjny jest zapisanym w pamięci mikrokomputera obrazem SPRITE'a. Sposób tworzenia bloku definicyjnego ilustruje rys. 1.

Na siatce odpowiadającej jednemu obiektowi wrysowany został czołg. Wyodrębnianie kolejnych bajtów pamięci i wyznaczanie ich wartości odbywa się na ogólnych zasadach opisanych w części II. Blok definicyjny powinien składać się z 63 bajtów. Przyjmuje się jednak wielkość 64 bajtów, przy czym ostatni bajt nie jest brany pod uwagę przez VIC-a. Zapiszmy utworzony blok definicyjny w programie za pomocą instrukcji DATA.

```
1 DATA 0, 0, 0, 0, 0, 0
2 DATA 0, 0, 0, 0, 0, 0
3 DATA 0, 0, 0, 0, 0, 0
4 DATA 0, 0, 0, 0, 0, 0
5 DATA 0, 0, 0, 0, 126, 0
6 DATA 0,255, 255, 0, 255, 0
7 DATA 0, 126, 0, 7, 255, 224
8 DATA 7, 255, 240, 127, 255, 254
9 DATA 206, 102, 115, 127, 255, 254
10 DATA 0, 0, 0, 0, 0, 0
11 DATA 0, 0, 0, 0
```

Tworzenie bloku definicyjnego jest najbardziej zmudnym zajęciem podczas projektowania SPRITE'a. **Tym, których zniechęca ręczne przeliczanie i wprowadzanie do pamięci mikrokomputera wymaganych danych polecam wykorzystanie do tego celu programu użytkowego „PROJEKTANT-SPRITES" zamieszczonego w „IKS-ie" nr 3, 1986.**



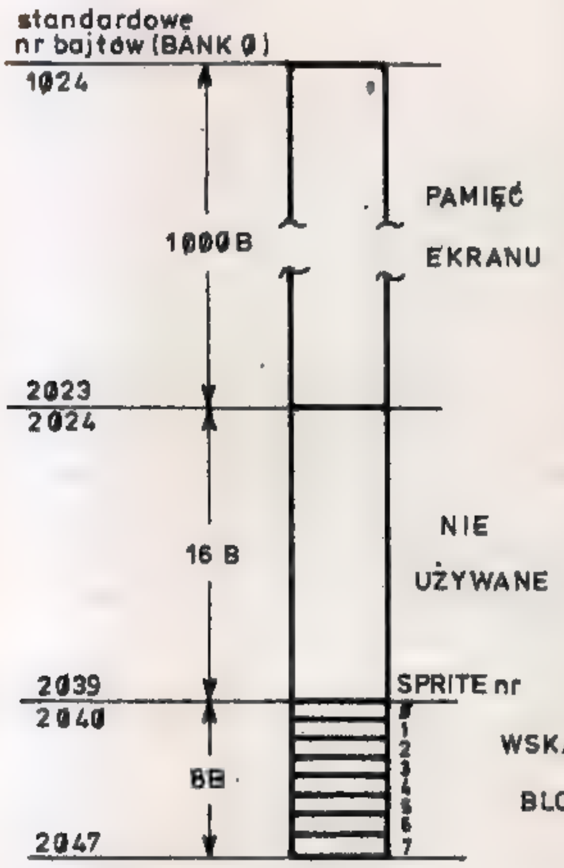
Rys. 1 Tworzenie bloku definicyjnego SPRITE'a.

VIC w każdym BANKU pozwala na 256 (256 x 64 =16 384) różnych lokalizacji bloków definicyjnych (od Ø do 255). Umieśćmy nasz blok definicyjny jako ostatni w BANKU Ø:

20 **FOR** I=Ø TO 63: **READ** A: **POKE** 255*64+I, A: **NEXT** I

W przypadku pracy VIC-a w BANKU innym niż zerowy musimy do adresu początkowego wybranego bloku (255*64) dodać adres początkowy użytego BANKU.

Jak poinformować VIC-a, w którym bloku umieściliśmy postać SPRITE'a? Do tego celu służą wskaźniki bloków definicyjnych. W każdym BANKU jest ich osiem, gdyż można wyświetlać jednocześnie do ośmiu ruchomych obiektów. Lokalizacja wskaźników zależy od umiejscowienia PAMIĘCI EKRANU wewnątrz BANKU. Zajmują one bowiem ostatnie osiem bajtów w 1K obszarze wyznaczonym adresem początkowym PAMIĘCI EKRANU (rys. 2).



Rys. 2 Lokalizacja wskaźników bloków definicyjnych w pamięci mikrokomputera

Ponieważ nie zmienialiśmy standardowego położenia PAMIĘCI EKRANU, w naszym przypadku wskaźniki te mieszczą się między bajtami 2Ø4Ø — 2Ø47. Wprowadzenie do odpowiedniego wskaźnika liczby z zakresu Ø — 255 wskazuje VIC-owi, który blok definicyjny ma być interpretowany. Teraz już wiemy dlaczego uzyskany początkowo na ekranie SPRITE miał nieokreślony kształt. Wykonajmy ponownie te same trzy instrukcje POKE oraz dodatkowo

POKE 2Ø4Ø, 255

SPRITE nadal ma tę samą postać. Wprowadźmy do 255 bloku definicyjnego zaprojektowaną postać obiektu ruchomego komendą RUN. Na ekranie uzyskaliśmy czołg.

W celu przemieszczania obiektów ruchomych po ekranie i uzyskiwania pewnych ich cech służą niektóre z rejestrów sterujących VIC-a. Wykaz interesujących nas rejestrów wraz z krótkim opisem realizowanych funkcji zamieszczony jest w tabeli.

bajt pamięci	realizowana funkcja
53248	SPRITE Ø współrzędna X
53249	SPRITE Ø współrzędna Y
5325Ø	SPRITE 1 współrzędna X
53251	SPRITE 1 — „ — Y
53252	SPRITE 2 — „ — X
53253	— „ — — „ — Y
53254	SPRITE 3 — „ — X
53255	— „ — — „ — Y
53256	SPRITE 4 — „ — X
53257	— „ — — „ — Y
53258	SPRITE 5 — „ — X
53259	— „ — — „ — Y
5326Ø	SPRITE 6 — „ — X
53261	— „ — — „ — Y
53262	SPRITE 7 — „ — X
53263	— „ — — „ — Y
53264	znacznik pozycji X
53269	aktywność SPRITE'a (1 — aktywny)
53271	powiększenie w pionie (na wysokość)
53275	priorytet SPRITE — podłoże (Ø — SPRITE ma wyższy)
53276	wprowadzenie trybu wielobarwnego (1 — wprowadzony)
53277	powiększenie w poziomie (na szerokość)
53278	sygnalizacja kolizji SPRITE — SPRITE
53279	— „ — — „ — SPRITE — podłoże
53281	kolor punktu Ø (ekran, SPRITE) w t. dwukolorowym; kolor punktu ØØ w t. wielobarwnym
53285	kolor punktu Ø1 SPRITE w t. wielobarwnym
53286	kolor punktu 11 SPRITE — „ —
53287	SPRITE Ø: kolor punktu 1 w t. dwukolorowym lub kolor punktu 1Ø w t. wielobarwnym
53288	SPRITE 1: — „ —
53289	SPRITE 2: — „ —
5329Ø	SPRITE 3: — „ —
53291	SPRITE 4: — „ —
53292	SPRITE 5: — „ —
53293	SPRITE 6: — „ —
53294	SPRITE 7: — „ —

TABELA:
Wykaz rejestrów sterujących obiektami ruchomymi.

Niektóre z wymienionych rejestrów określają jakąś cechę SPRITE'a zawartością pojedynczego bitu odpowiadającego jego numerowi. Bit nr Ø jest odpowiedzialny za zaistnienie tej cechy u SPRITE'a nr Ø, bit nr 1 u SPRITE'a nr 1, itd. aż do ostatniego bitu (nr 7). Ogólne postacie instrukcji zmieniających zawartość pojedynczych bitów w rejestrze są następujące:

— ustawienie bitu na 1

POKE rejestr, **PEEK** (rejestr) **OR** 2↑bit

— ustawienie bitu na Ø

POKE rejestr, **PEEK** (rejestr) **AND** (255-2↑bit)

gdzie: rejestr — adres rejestru (bajtu pamięci),
bit — numer bitu odpowiadający wybranemu obiektowi.

Zapoznamy się teraz bliżej z wymienionymi w tabeli funkcjami. Pozwolą one nam w sposób efektywny wykorzystać możliwości VIC-a do programowania obiektów ruchomych.

AKTYWNOŚĆ

Każdy z ośmiu obiektów ruchomych może być niezależnie wprowadzany w stan aktywności. Jest to stan, w którym VIC interpretuje wszystkie niezbędne informacje doty-

cące danego SPRITE'a i na ich podstawie wyświetla jego obraz na ekranie.

Do określenia stanu aktywności poszczególnych obiektów służy rejestr sterujący VIC-a umieszczony pod adresem 53269. Ustawienie jakiegoś bitu na „1" w tym rejestrze oznacza stan aktywny odpowiadającego mu obiektu. Jeśli chcemy, na przykład uaktywnić tylko obiekty numer Ø i 1 musimy wykonać instrukcję:

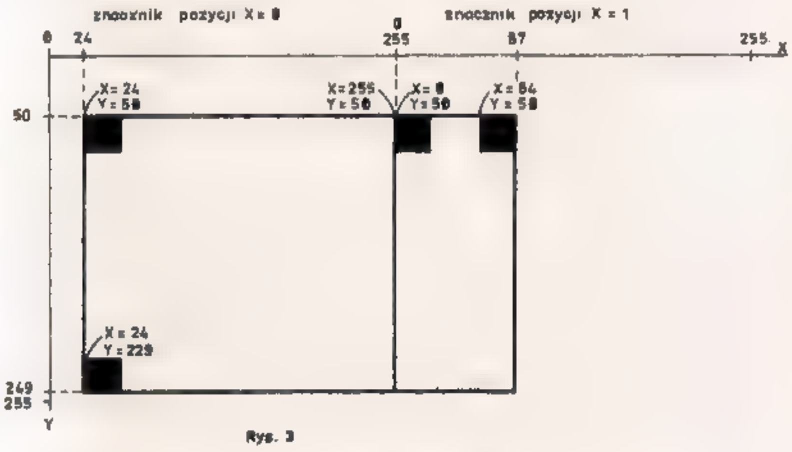
POKE 53269, **PEEK** (53269) **OR** 3

SPRITE może być widoczny na ekranie lub nie, zależnie od stanu pozostałych odpowiadających mu rejestrów VIC-a. Powrót do stanu nieaktywnego uzyskujemy zerując odpowiednie bity. W naszym przypadku.

POKE 53269, **PEEK** (53269) **AND** 252

POZYCJONOWANIE

Do ustalenia pozycji ruchomych obiektów na ekranie służy grupa rejestrów o adresach od 53248 do 53264. Pozycja obiektu wyznaczana jest na podstawie wartości współrzędnej X na osi poziomej i współrzędnej Y na osi pionowej (rys. 3).



Rys. 3 Pozycjonowanie SPRITE'a na ekranie.

Współrzędne X i Y ustalają pozycję lewego górnego punktu SPRITE'a. Na rys. 1 jest to lewy górny kwadrat siatki projektowej. Każdy SPRITE ma przydzieloną parę rejestrów do wprowadzania wartości obu współrzędnych. Ponieważ największa liczba jaką można zapisać w jednym bajcie wynosi 255, do pozycjonowania na osi poziomej (X) wykorzystuje się dla każdego obiektu dodatkowy jeden bit w rejestrze o adresie 53264. Bit ten nazywany jest znacznikiem pozycji X. Ustawienie znacznika na Ø oznacza, że wartość współrzędnej X dotyczy lewego odcinka osi poziomej. Znacznik równy 1 określa prawy odcinek osi poziomej X.

Wykonajmy następujący przykład w trybie bezpośrednim:

— utwórzmy blok definicyjny obiektu w kształcie prostokąta

FOR I=Ø **TO** 63: **POKE** 254*64+I, 255: **NEXT** I

— postać tę nadamy obiektowi nr Ø

POKE 2Ø4Ø, 254: **POKE** 53269, 1

— wyświetlimy prostokąt w lewym górnym rogu ekranu

POKE 53248, 24: **POKE** 53249, 5Ø: **POKE** 53264, Ø

Zmieniając edytorem wartość ostatnich trzech rejestrów (współrzędne X, Y oraz znacznik pozycji X) możemy zbadać położenie SPRITE'a w różnych punktach ekranu.

Wiedząc już jak zmieniać położenie obiektu spróbujmy uruchomić nasz czołg w realizowanym programie i przejechać nim po ekranie. Wprowadźmy niezbędne instrukcje i wykonajmy komendę RUN.

3Ø **POKE** 2Ø4Ø, 255: **POKE** 53269, 1: **POKE** 53249, 1ØØ
4Ø **POKE** 53264, Ø: **FOR** X=Ø **TO** 255: **POKE** 53248, X: **NEXT** X
5Ø **POKE** 53264, 1: **FOR** X=1 **TO** 88: **POKE** 53248, X: **NEXT** X

POWIĘKSZANIE

Normalnej wielkości SPRITE ma 24 punkty szerokości i 21 punktów wysokości. Istnieje możliwość dwukrotnego powiększenia SPRITE'a. Powiększenie na szerokość uzyskujemy ustawiając odpowiedni bit na 1 w rejestrze o adresie 53277, a na wysokość w rejestrze 53271. Powrót do normalnych wymiarów uzyskamy po wyzerowaniu odpowiednich bitów.

Ostatnio używaliśmy czołgu jako obiektu ruchomego o nr. Ø. Wyświetlmy więc go ponownie w centralnej części ekranu.

POKE 53248, 15Ø: **POKE** 53249, 15Ø: **POKE** 53264,Ø

a teraz powiększymy go na szerokość

POKE 53277,1

i na wysokość

POKE 53271,1

Powracamy do normalnych wymiarów:

POKE 53277,Ø: **POKE** 53271,Ø

PRIORYTET

Przy użyciu większej liczby obiektów ruchomych mogą zaistnieć sytuacje, gdy obiekty będą nachodzić na siebie. Należy zatem wcześniej zadbać o to, który SPRITE ma być widziany „przed" innym lub „za" innym. Do tego celu wykorzystuje się priorytety przyporządkowane na stałe wszystkim obiektom ruchomym. SPRITE Ø ma największy priorytet i jest widziany zawsze „przed" wszystkimi pozostałymi. SPRITE 7 ma najmniejszy priorytet i jest widziany „za" wszystkimi pozostałymi. Inaczej mówiąc: SPRITE o numerze niższym ma zawsze wyższy priorytet.

W trakcie łączenia różnych trybów graficznych należy ustalić priorytet każdego używanego SPRITE'a względem pozostałych elementów graficznych (podłoża). Do ustalenia tego priorytetu służy rejestr o adresie 53275. Wartość zerowa bitu odpowiadającego wykorzystywanemu obiektowi oznacza, że ma on wyższy priorytet względem podłoża. Na ekranie widoczny będzie SPRITE, a za nim schowane będą występujące w tym samym miejscu elementy graficzne zrealizowane w innym trybie pracy VIC-a.

Należy zaznaczyć, że przesłaniane są tylko szczegóły obrazu graficznego, które w reprezentacji bitowej są jedynkami (np. zaczernione kwadraty w siatce projektowej na rys. 1)

Zmieniając priorytet czołgu (SPRITE Ø) względem podłoża i przesuwając kursor w miejsce, gdzie znajduje się czołg na ekranie możemy zaobserwować opisane efekty. Zmianę priorytetu uzyskamy instrukcjami:

— czołg ma niższy priorytet

POKE 53275,1

— czołg ma wyższy priorytet

POKE 53275,Ø

KOLIZJA

Sytuacja kolizji występuje, gdy nastąpi zetknięcie się kilku niezerowych (w reprezentacji bitowej) elementów graficznych. Do kolizji może dojść między obiektami ruchomymi oraz między podłożem a obiektem ruchomym. VIC sam wykrywa sytuacje kolizji i informuje nas o tym za pomocą rejestrów o adresach 53278 i 53279. Rejestr 53278 sygnalizuje wystąpienie kolizji SPRITE — SPRITE ustawiając na 1 bity odpowiadające obiektom ruchomym, między którymi wystąpiła kolizja. Rejestr 53279 sygnalizuje wystąpienie kolizji typu SPRITE — podłoże ustawiając na 1 bity odpowiadające obiektom, które zetknęły się z jakimkolwiek elementem graficznym podłoża.

Sygnały o sytuacjach kolizyjnych są dostarczane ciągle przez VIC-a. Zerowanie rejestrów kolizji następuje automatycznie po odczytaniu ich zawartości instrukcją PEEK. Należy pamiętać o tym, że SPRITE nieaktywny może również powodować kolizje.

KOLORY

Podobnie, jak i w pozostałych trybach pracy VIC-a, można w obiektach ruchomych wykorzystywać grafikę dwukolorową i wielobarwną. Za pomocą rejestru o adresie 53276 możemy wybrać rodzaj grafiki dla każdego obiektu oddzielnie. Wartość zerowa bitu oznacza grafikę dwukolorową. Ustawienie na 1 jakiegoś bitu wprowadza odpowiadający mu SPRITE w grafikę wielobarwną.

W grafice dwukolorowej VIC pobiera kod kolorów (Ø — 15) każdego z obiektów ruchomych z dwóch rejestrów. Rejestr odpowiedzialny za kolor podłoża SPRITE'a (zera w reprezentacji bitowej, puste kwadraty w siatce projektowej) jest ten sam dla wszystkich obiektów i mieści się pod adresem 53281. Rejestr ten równocześnie określa kolor punktów o zerowej reprezentacji bitowej na ekranie. Kolor grafiki SPRITE'a (jedynki w reprezentacji bitowej) jest określany niezależnie dla każdego z nich za pomocą odpowiedniego rejestru od 53287 do 53294.

W grafice wielobarwnej, podobnie jak w trybie tekstowym czy trybie wysokiej rozdzielczości jeden punkt w obrazie SPRITE'a na ekranie odpowiada dwóm bitom pamięci. Kolory są kodowane w następujący sposób:

punkt	rejestr zawierający kod koloru	
ØØ	53281	wspólne dla wszystkich obiektów ruchomych
Ø1	53285	
11	53286	
1Ø	rejestr koloru odpowiedniego SPRITE'a (53287 — 53294)	

I to już wszystko. Przekazany zakres wiedzy na temat grafiki C-64 powinien wystarczyć aby samodzielnie pisać programy graficzne. Należy jednak zdawać sobie sprawę z tego, że uzyskanie wyrafinowanych efektów graficznych wiąże się z wykorzystaniem dodatkowych technik programowania mikrokomputerów (np. technika przerwań), a w szczególności programowania w języku wewnętrznym mikrokomputera Commodore 64.

Sławomir WASIELEWSKI

ATAROWISKO





(Foto: Jan ZELMAN)





PROGRAM 24

ZX SPECTRUM

B O M B A

Hasło *„zagrajmy w bombę"* kierujemy do adeptów języka LOGO, którzy w wolnej chwili chcieliby zostać bohaterami. Otrzymujecie informację (oczywiście pewną), że w jednym z 357 911 pokoi wieżowca podłożono bombę. Należy ją rozbroić. Zabieracie specjalny czujnik i wchodzicie do budynku w miejscu o umownych współrzędnych 0,0,0. Budynek ma kształt sześcianu. Pokój na drugim końcu przekątnej ma współrzędne 70,70,70.

Czujnik natychmiast podaje pewien sygnał, który należy wykorzystać. Jedyną wskazówką, jaką można podać jest zakres sygnału wynoszący 0...1000. Im większy wskaźnik, tym bliżej bomby. Macie 15 szans na wybranie takich trzech współrzędnych, które wskażą poszukiwany pokój. Współrzędne należy wczytywać oddzielając je spacją. Na końcu wprowadzania naciskamy ENTER.

Gra nie wymaga intensywnego angażowania wielu szarych komórek pod warunkiem, że szybko znajdziecie trochę reguł.

Krzysztof MAMCARZ

```
?
TO REKORD
IF :numer < ITEM 2 :best [type :
imie type char 32 pr :numer ( ma
ke "best list :imie :numer )] [t
ype ( item 1 :best ) type char 3
2 pr ( item 2 :best )]
END

TO WPROW
SETCUR SE 22 :line
MAKE "dane RL
MAKE "ilpar COUNT :dane
IF :ilpar = 3 [] [wprow]
MAKE "wys FIRST :dane
IF :wys < 0 [pr " pr [wpadles do
 piwnicy!!!] TOPLEVEL]
IF :wys > 70 [pr " pr [spadles z
 dachu!!!] TOPLEVEL]
MAKE "sz ITEM 2 :dane
IF OR :sz < 0 :sz > 70 [pr " pr
[wypadles przez okno!!!] TOPLEVE
L]
MAKE "dl LAST :dane
IF OR :dl < 0 :dl > 70 [pr " pr
[wypadles przez okno!!!] TOPLEVE
L]
MAKE "czujnik ( SUM :wys / 10 :s
z :dl * 10 ) - ( SUM :a / 10 :b
:c * 10 )
IF :czujnik < 0 [make "czujnik -
 :czujnik]
MAKE "syg 1000 - :czujnik
IF :syg = 1000 [pr " pr [BRAVO!
BOMBA ROZBROJONA!] REKORD TOPLEV
EL]
IF :numer = 15 [pr " pr [NIESTET
Y! BOMBA WYBUCHA!] TOPLEVEL]
MAKE "line :line + 1
END

TO WYDRUK
SETCUR SE 0 :line
TYPE :numer
SETCUR SE 4 :line
TYPE :wys
SETCUR SE 8 :line
TYPE :sz
SETCUR SE 12 :line
TYPE :dl
SETCUR SE 16 :line
TYPE :syg
MAKE "numer :numer + 1
END

TO INIT
HT TS CS SETBR 4
SETCUR [0 0]
PR [wpisz imie...]
MAKE "imie RL
MAKE "ilpar COUNT :imie
IF :ilpar = 1 [] [toplevel]
CS SETCUR [0 0]
REPEAT 32 [type char 42]
PR [nr..wys.sz..dl..sygnal]
REPEAT 32 [type char 42]
MAKE "a RANDOM 71
MAKE "b RANDOM 71
MAKE "c RANDOM 71
MAKE "wys 0
MAKE "sz 0
MAKE "dl 0
MAKE "syg 1000 - ( SUM :a / 10 :
b :c * 10 )
MAKE "numer 0
MAKE "line 3
WYDRUK
END

TO BOMBA
IF ( .EXAMINE 65535 ) = 0 [make
"best [EMIL 5] .DEPOSIT 65535 1]
INIT
REPEAT 15 [wprow wydruk]
END
```

```
********************************
nr..wys.sz..dl..sygnal
********************************
0   0   0   0   408.2 70 70 70
1   70  70  70  727.8 70 50 60
2   70  50  60  847.8 70 50 30
3   70  50  30  852.2 70 40 50
4   70  40  50  957.8 62 40 54
5   62  40  54  918.5 64 40 46
6   64  40  46  996.4 66 42 46
7   66  42  46  996   66 38 46

BRAVO! BOMBA ROZBROJONA!
krzysztof 8
```

GIEŁDA POMYSŁÓW

Meritum umie generować ludzki głos

Można się o tym przekonać wpisując poniższy program, który koduje w pamięci dźwięk wprowadzany z magnetofonu.

Głos do pamięci mikrokomputera wprowadza się tak jak przy wczytywaniu programów. Najlepszy efekt osiąga się stosując firmowy magnetofon dołączany do zestawu.

Najpierw należy przygotować około piętnastosekundowy fragment nagrania (ciągłej mowy, z dość wysokim poziomem zapisu), następnie połączyć magnetofon z mikrokomputerem, włączyć odtwarzanie i w chwili usłyszenia dźwięku nacisnąć klawisz „Z". Dźwięk będzie wpisywany do pamięci, a po jej wypełnieniu zostanie odtworzony. Prostota wprowadzenia dźwięku okupiona jest zniekształceniami, które powoduje interface magnetofonu. Odtwarzany dźwięk przypomina dźwięki z radiostacji krótkofalarskich. Na wydruku kodu maszynowego wyróżnione są bajty regulujące częstotliwość próbkowania sygnału wejściowego (pierwszy) i prędkość odtwarzania (dwa następne). Można zmieniać ich wartości (heksadecymalnie).

Piotr BIEDRZYCKI

```
10     FOR A=32528 TO 32622
20    READ A$
30    N=ASC( LEFT$(A$,1))-48
40    N=N-INT(N/12)*7
50    M=ASC(RIGHT$(A$,1))-48
60    M=M-INT(M/12)*7
70    B=N*16+M: POKE A,B :W=W+B
75      NEXT A
80   ' KOD MASZYNOWY PROGRAMU
90   DATA CD,17,7F,CD,48,7F,C9,C5
100  DATA D5,E5,11,FF,7E,21,00,7F
110  DATA 36,00,01,01,38,ED,B8,21
120  DATA 00,47,0E,FF,11,80,00,06
130  DATA 08,ED,78,A3,B6,07,77,16
140  DATA 0B,15,20,FD,ED,51,10,F1
150  DATA 23,7C,FE,7F,20,E9,18,23
160  DATA C5,D5,E5,21,00,47,0E,FE
170  DATA 16,00,7E,1E,08,07,ED,79
180  DATA 06,07,10,FE,ED,51,06,08
190  DATA 10,FE,1D,20,F0,23,7C,FE
200  DATA 7F,20,E7,E1,D1,C1,C9
210  IF W<>10207 THEN PRINT
     "BLAD W ZBIORZE DANYCH ":END
230  CLS : POKE 16527,127
240     PRINT@ 448,
"/Z/ - ZAPIS GLOSU Z MAGNETOFONU
/O/ - ODTWARZANIE DZWIEKU":PRINT:
PRINT"      WYBIERZ OPCJE  /Z,O/"
250  A$=INKEY$:IF A$="O" OR A$="Z"
     THEN 260 ELSE GOTO 250
260  IF A$="O" THEN POKE 16526,19
270  IF A$="Z" THEN POKE 16526,16
280  PRINT@ 782,"-";A$;"-";:A$=""
290  U=USR(0):PRINT@ 782,"- -";
400      GOTO 250
```

LIGA MYŚLĄCYCH

Zadanie 1

Zapytano rybaka, ile waży złowiona przez niego ryba. Rybak odpowiedział: $\frac{2}{5}$ kg i jeszcze 2 razy po $\frac{2}{5}$ swego ciężaru. Ile ważyła złowiona przez rybaka ryba?

Zadanie 2

Pamiętny w historii rok pewnego odkrycia wyraża się liczbą czterocyfrową. Należy znaleźć tę liczbę wiedząc, iż suma cyfr tej liczby wynosi 16, cyfra tysięcy jest cztery razy mniejsza od cyfry setek, cyfra setek jest dwa razy większa od cyfry jedności, a liczba trzycyfrowa utworzona z szukanej liczby przez skreślenie w niej cyfry jedności jest o 100 większa od liczby dwucyfrowej utworzonej z szukanej liczby przez skreślenie w niej cyfry jedności i cyfry tysięcy.

Zadanie 3

Wieśniaczka sprzedała pierwszej osobie $\frac{1}{2}$ całej ilości jajek i jeszcze 2 jajka. Drugiej osobie sprzedała połowę reszty jajek i jeszcze jedno jajko. Po drugiej sprzedaży zostało wieśniaczce jeszcze 8 jajek. Ile wszystkich jajek przyniosła wieśniaczka na targ? Ile jajek kupiła pierwsza osoba, a ile druga?

Zadanie 4

Codziennie z miejscowości A, w której mieści się urząd pocztowy, wyjeżdża listonosz do miejscowości B, C, D, by dostarczyć listy.

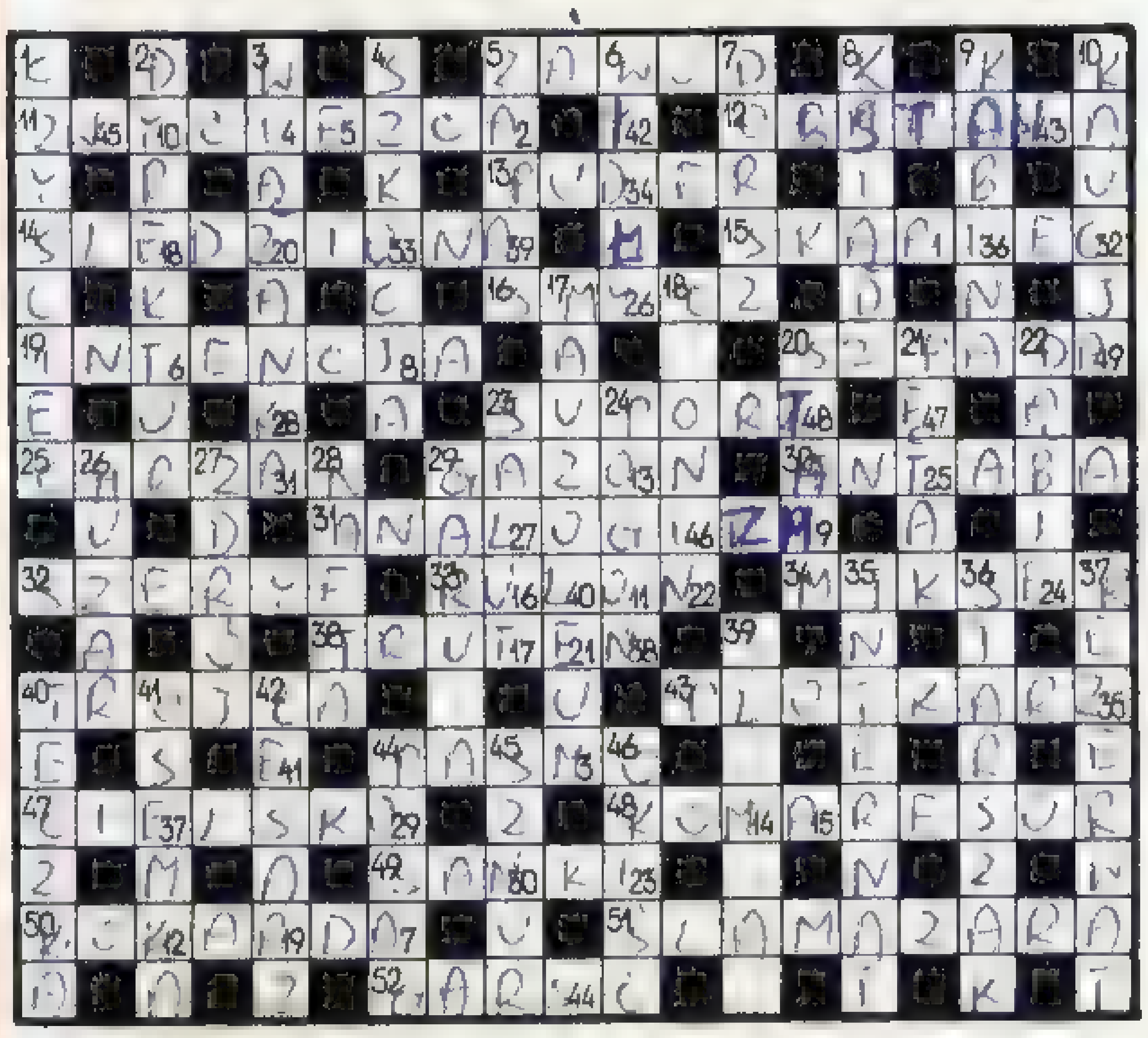
W jakiej kolejności powinien listonosz objeżdżać te miejscowości aby trasa objazdu z A przez pozostałe miejscowości i z powrotem do A była możliwie najkrótsza, jeśli długość drogi z A do B wynosi 5 km, od A do C — 7 km, od A do D — 7 km, od B do D — 8 km, od B do C — 10 km, od C do D — 6 km?

Zadanie 5

Gdy Jan zapytał Andrzeja ile ma lat, usłyszał odpowiedź: gdy ja byłem w twoim wieku byłeś ode mnie cztery razy młodszy, a gdy ty będziesz w moim wieku ja będę miał 40 lat. Ile lat ma Jan, a ile Andrzej?

Rozwiązania zadań prosimy przesyłać do redakcji do końca października br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody-niespodzianki.

Krzyżówka nr 8



POZIOMO: 5) profesja, 11) triumfator, 12) ludzie stanowiący ubezpieczenie, zapewniający ochronę kogoś lub czegoś, 13) środek kosmetyczny lub leczniczy, 14) wydłużony narząd kręgowców, stanowiący swego rodzaju magazyn krwi, 15) sknera, kutwa, 16) do prowadzenia psa, 19) motyw działania; zamiar, zamysł, 20) broń sportowa używana w szermierce, 23) ruchomy zespół obrabiarki (np. tokarki, strugarki), 25) składnik ćwikły, 29) ozdobny trawnik lub kwietnik, 30) metalowa sztaba umacniająca zamknięcie drzwi, bramy, 31) w logice; wnioskowanie przez analogię, 32) w USA urzędnik lokalny o uprawnieniach adm.-sądowych, 33) zwój o cylindrycznym kształcie, 34) zmechanizowane urządzenie używane przy sporządzaniu różnych potraw, deserów i napojów cocktailowych, 38) samiec pszczoły miodnej, 40) rel. grupa trzech bóstw występująca w wielu wierzeniach, 43) siewca kłamliwych pogłosek, 44) ... nieszczęść lub sukcesów, 47) opasłe, olbrzymie ciało, 48) sprężarka, 49) popularny zimowy sprzęt sportowy, 50) ozdoba warkocza, 51) ktoś powolny, pozbawiony energii, 52) dłoń złożona w sposób umożliwiający zaczerpnięcie, uchwycenie czegoś.

PIONOWO: 1) wg religii katolickiej: miejsce pozagrobowej okresowej pokuty dusz ludzi zmarłych, 2) stoi na czele instytucji, przedsiębiorstwa, 3) ślubna lub nagrobna, 4) kraj z Edynburgiem, 5) zasób, rezerwa, 6) gable, 7) pospolita ryba morska, 8) duchowny katolicki, 9) kajuta okrętowa, 10) suma pieniężna złożona jako zastaw, 17) monumentalny grobowiec w formie wolno stojącej budowli, 18) pisarz szkocko-angielski (1896 — 1981), autor „Cytadeli", 20) najwybitniejszy trener polskich pięściarzy, 21) brzdąc, smarkacz, 22) dzielnica Szczecina lub jezioro, nad którym leży to miasto, 23) uroczyste oddanie salw artyleryjskich, 24) klub sportowy ze Szczecina, 26) dawny żołnierz lekkiej jazdy, ubrany i uzbrojony na sposób węgierski, 27) źródło, 28) łatwo zapalna ciecz, produkt destylacji ropy naftowej, 29) żegl. wiatr wiejący z boku, 35) bursa, 36) dziecko z grupy starszych w przedszkolu, żłobku, 37) teren wyłączony spod eksploatacji, znajdujący się pod specjalną ochroną, 39) dawny zakład, w którym uczniowie otrzymywali darmo naukę i utrzymanie, 40) aktówka, 41) liczna osada wioślarska, 42) np. Napoleon Bonaparte, 44) wiano, 45) cienki powróz, 46) śnieg na gałęziach drzew, zwieszający się w kształcie kiści.

Po prawidłowym rozwiązaniu krzyżówki litery z kratek ponumerowanych dodatkowo w prawym dolnym rogu, czytane w kolejności odpowiadających im liczb od 1 do 49, dadzą hasło, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji do końca października 1987 r. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe

MIKROCIEKAWOSTKI:



Rysunek z kolekcji GEM wykonany na zestawie firmy ICL
komputer — DRS 300
drukarka — Ink Yet Coluor Printer
(drukarka wykorzystuje miniaturowe działka z różnymi kolorami tuszu)

W naszym komputerlandzie

Spektruś postanowił zwiększyć w swoim zespole inicjatywę w dziedzinie racjonalizacji. Zainspirował więc do działania swojego młodego i zdolnego dziełoroba Jaśka Atari. Jaśko przeszedł się po pracowni i rozejrzał po swojemu. Ujrzał, że jeden z półautomatów wyrabia 20 części na godzinę. — To stanowczo za mało — pomyślał. Zabrał się do roboty.

Po tygodniu specjalna komisja stwierdziła, że pomysł jest dobry — dokumentacja w porządku, a korzyści finansowe udowodnione. W maszynie wymieniono mechanizm napędowy. Półautomat wyrabiał teraz 40 części na godzinę, Jaśko zaś otrzymał grubą kopertę.

Do Spektrusia przyszedł jednak brakarz, który stwierdził, że półautomat wyrabia rzeczywiście dużo, ale z jakością jest gorzej. Tylko jedna trzecia część nadaje się do użytku.

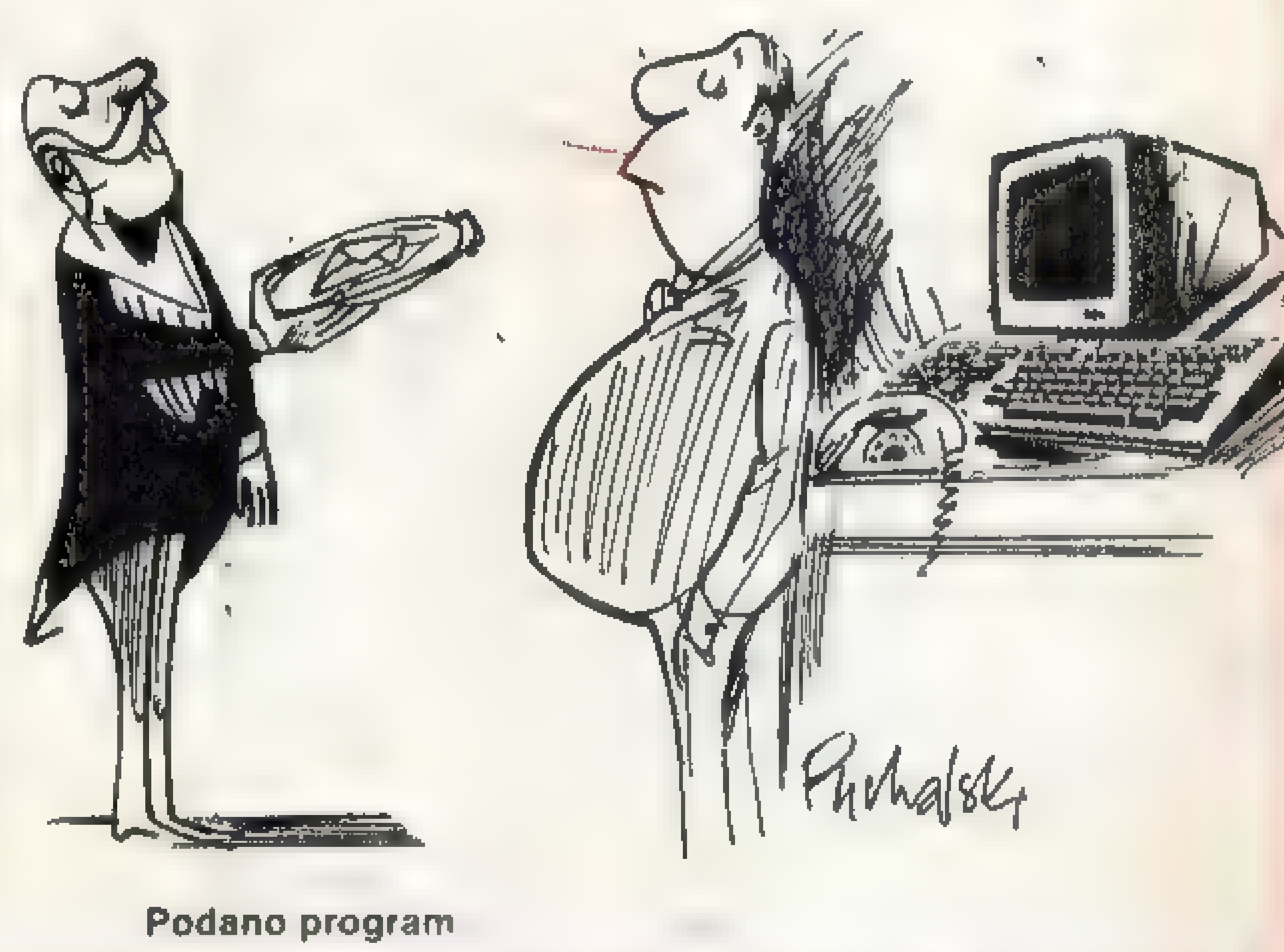
Jeszcze tego samego dnia Jaśko stanął przed półautomatem. Pomyślał i zabrał się do roboty. Po dwóch tygodniach komisja przyjęła projekt, wszyscy klaskali, a Jaśko otrzymał grubą kopertę.

Po wymianie mechanizmu napędowego półautomat wyrabiał 20 części na godzinę, brakarz nie mógł się nachwalić tych części, lecz roboty nie chciały pracować, bo urządzenie hałasować poczęło potwornie.

Racjonalizator Jaśko znowu był na posterunku. Teraz przygotowywał projekt rozwiązania z zakresu bhp. Zaproponowaną przez niego wymianę układu napędowego komisja przyjęła bez uwag. Jaśko otrzymał grubą kopertę. Półautomat wyrabiał 20 dobrych części, a co najważniejsze można było wyjąć watę z uszu, bo pracował cicho.

Zadowolony był też Spektruś, gdy od dyrektora generalnego otrzymał grubą kopertę za wdrożenie w ostatnim czasie trzech wybitnych projektów.

Podglądał: Eugeniusz MLECZAK



Podano program

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szanlawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 9189. Nr. ind. 361682. K-65

INFORMATYKA KOMPUTERY SYSTEMY



CENA — 80 ZŁ

DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 10/1987 ISSN 0860—2794



Każdy nadesłany do redakcji list jest uważnie przeczytany. Staramy się w redagowaniu pisma wykorzystywać pomysły naszych Czytelników. Są listy, na które odpowiadamy również listami, są także takie, na które odpowiedzią jest artykuł czy program zamieszczony na naszych łamach.

Czytelnicy zawsze towarzyszyli powstawaniu każdego numeru „IKS-a". Z waszych listów, a ostatnio także z nadesłanych ankiet wiemy, czym powinniśmy się zajmować. Dlatego też na naszych łamach przeważać będą informacje na temat możliwości posiadanego przez was sprzętu. Oczywiście pisać nadal będziemy o Atari, Commodore, Spectrum, ale także o Amstradzie, którego liczba użytkowników stale wzrasta i IBM czyli olbrzymiej już rodzinie komputerów od lat wyznaczających światowy standard. Zatem posiadacze ośmiobitowych mikrokomputerów nadal będą mogli znaleźć u nas informacje o tym, co jeszcze potrafią ich komputery, a o czym fabryczne instrukcje często milczą. Czasami krótki, zawierający kilkanaście instrukcji program odkryć może nieznane dotąd walory np. starego Spectrum.

To oczywiście nie wszystko. Już od dzisiejszego numeru drukować będziemy co miesiąc angielsko-polski słownik informatyczny. Mamy nadzieję, że ułatwi on studiowanie trudnej obcej literatury fachowej. Na łamy „IKS-a" trafiła również fantastyka, której autorami są nasi Czytelnicy — publikujemy bowiem opowiadania nagrodzone w naszym konkursie.

Jeśli chcecie, aby „IKS" był naprawdę waszą gazetą piszcie do nas. Zawartość „IKS-a" zależy od Czytelników.

W NUMERZE:

Czy będziemy mieli polski mikrokomputer? — str. 4
W szponach Atari (7) — str. 5—8
Przystawka magnetofonowa do Atari — str. 11—13
Procedury Rom SPECTRUM (1) — str. 14—15
Opowiadanie SF „Finał" — str. 20
Basic (9) — str. 21—25
Badanie kierowców — str. 25—27
Emulator Basic 1.1 dla CPC 464 (2) — str. 27
Informatyczny słownik angielsko-polski — str. 29—30

TMMT

Ogólnopolska Federacja Klubów Komputerowych Młodych Mistrzów Techniki ZSMP powstała w styczniu 1986 roku. Obecnie w jej skład wchodzą 264 kluby, wśród których najwięcej jest placówek działających przy zakładach pracy i szkołach. Podstawowym celem Federacji, celem wspólnym dla wszystkich jej ogniw jest szeroko rozumiana edukacja informatyczna młodzieży.

Oczywiście klub to nie szkoła. Wprawdzie uczestnicy zajęć — w większości zresztą uczniowie, zdobywają wiedzę, jednak proces ten ma niewiele wspólnego z lekcjami. Porównać go można raczej do nauki własnej, do samokształcenia, którego kierunki i intensywność zależą od indywidualnych zainteresowań konkretnego człowieka. Zadaniem klubu jest więc tworzenie optymalnych warunków do rozwoju pasji jego członków. Kto ma styczność z jakąkolwiek placówką KK MMT wie jak trudne jest to zadanie.

Oczekiwania fanów informatyki, potrzeby które zgłaszają w dziedzinie sprzętu (jego ilości czy jakości), literatury, wydawnictw, dostępu do oprogramowania, niejednokrotnie przekraczają możliwości organizacyjne i finansowe klubu i jego sponsorów. Jednocześnie wiele problemów łatwiej pokonać, wiele spraw szybciej i taniej można załatwić występując grupowo. Stara to zasada i wielokrotnie już potwierdzona w praktyce. Ona też przesądza o sensowności istnienia Federacji KK MMT, powołanej przecież przez organizatorów i działaczy klubów egzystujących wcześniej w pojedynkę, zdanych tylko na własne siły przy pokonywaniu wspólnych przecież dla wszystkich problemów.

Korzystając z gościnnych łamów „IKS-a" będziemy, od tego numeru na bieżąco, informowali o naszych propozycjach i przedsięwzięciach. Sądzę, że będą to sprawy w równej mierze interesujące dla klubów pracujących w formule KK MMT, nie związanych z Federacją, jak też dla indywidualnych posiadaczy mk. Chcemy jednocześnie zaproponować wszystkim chętnym współdziałanie. Więcej — liczymy bardzo na głosy oceniające poszczególne elementy naszej oferty, na rady, opinie i podpowiedzi, na podzielenie się z nami waszymi przemyśleniami i doświadczeniami. Im więcej będzie zabierających głos, im szersze kręgi obejmie rozmowa o naszych sprawach, tym bardziej prawdopodobne jest wypracowanie najlepszych, najbardziej efektywnych metod działania.

Czekając na wasze listy już teraz proponujemy:

1. Pomoc prawną i organizacyjną przy powoływaniu klubów komputerowych.

2. Nieodpłatne pośrednictwo przy zakupach sprzętu komputerowego dla klubów bądź ich sponsorów. Przykładowe ceny: Amstrad CPC 6128 z monitorem zielonym — 520 tys., IBM PC/XT (RAM 640, monitor firmowy, karta Hercules, CGA, 1x360 kB — dysk firmy NEC) — 1280 tys. zł, hard-dysk 20 MB — 1200 tys. — gwarancje — od 12 do 18 miesięcy. Upusty, które uzyskaliśmy wynoszą od 10 do 50 % od cen rynkowych.

3. Legitymacje, znaczki, plakietki, plakaty itp. elementy propagandowe (dla placówek KK MMT nieodpłatnie bądź w cenie równej kosztom produkcji).

4. Wydawnictwa serii „Mikrokomputerowe ABC" (nieodpłatnie dla placówek KK MMT).

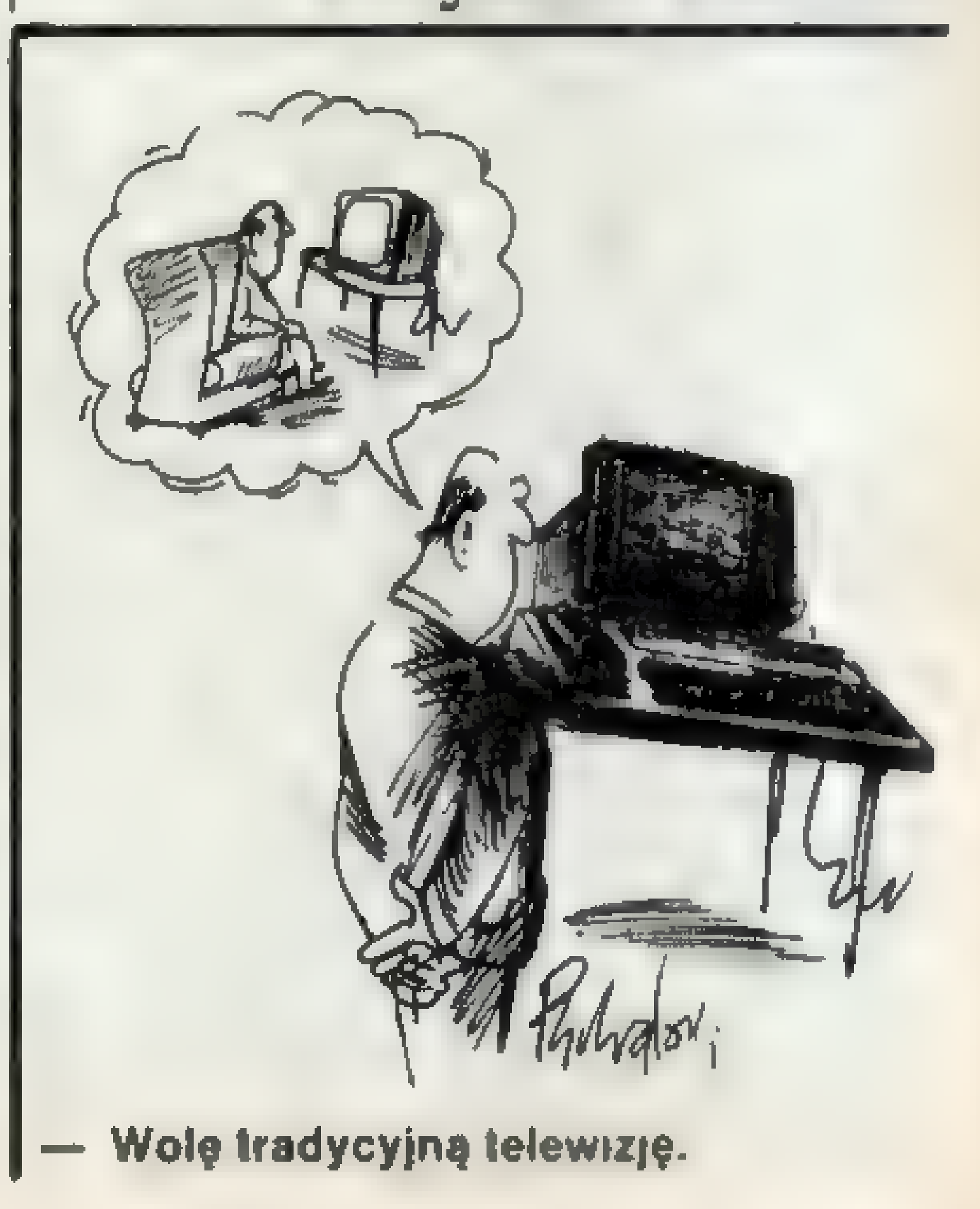
Jednocześnie jeszcze w tym roku będziemy kompletowali i rozsyłali (również nieodpłatnie), klubom zrzeszonym w Federacji, pakiety książek, wydawnictw oraz programów uznanych za najlepsze z dostępnych w kraju. Listę zawierającą konkretne tytuły opublikujemy.

A może, ...może zechcecie nam podpowiedzieć, jakie waszym zdaniem książki i które programy (narzędziowe, użytkowe, dydaktyczne i zabawowe) powinny znaleźć się w takim pakiecie? Jeżeli tak, kontaktujcie się z Sekretariatem Rady Ogólnopolskiej Federacji Klubów Komputerowych Młodych Mistrzów Techniki, Warszawa, ul. Smolna 40 pok. 414 lub dzwońcie tel. 26-54-01 wew. 362 lub 232.

Jeżeli chcecie dowiedzieć się szczegółów czytajcie „IKS-a", słuchajcie też kolejnych emisji „Radiokomputera", który będziemy dla was współredagować. Dla posiadaczy mk. kompatybilnych z ZX Spectrum wiadomość dodatkowa — w każdy czwarty piątek miesiąca „Radiokomputer" emitował będzie „Komputerowy Biuletyn Informacyjny" w postaci programów przyswajalnych przez popularnego „Spektrusia".

I na razie to byłoby już wszystko. Pozdrawiam, zapraszam do współpracy.

Krzysztof KRUPA



— Wolę tradycyjną telewizję.

ATARI STAR TEXTER

W IKS-ie nr 6 z bieżącego roku zaprezentowany został jeden z wielu opracowanych dla mikrokomputerów ATARI edytorów tekstu - ATARI WRITER (W szponach ATARI, str. 8-9). Autorzy artykułu stwierdzili, że poważną wadą tego programu jest brak możliwości uzyskiwania znaków specyficznych dla polskiego alfabetu zarówno na ekranie monitora jak i na drukarce. Rzeczywiście, dla polskiego użytkownika jest to bardzo istotna wada, eliminująca praktycznie ten program jako narzędzie do redagowania i drukowania artykułów, pism,itp.

Chciałbym przedstawić dzisiaj czytelnikom inny edytor tekstu - STAR TEXTER, wprowadzony na rynek oprogramowania w 1985 roku przez firmę SYBEX Verlag GmbH. Autorem programu jest Toni Schwaiger, a wersję programu na ATARI opracował Carsten Reinhardt. Najpoważniejszą zaletą tego edytora, w stosunku do takich jak ATARI WRITER, jest to, że umożliwia on w łatwy sposób definiować i wykorzystywać, praktycznie dowolny kształt liter,które są wyświetlane na monitorze i drukowane na drukarce, przy czym wszystko to dzieje się bez jakiejkolwiek ingerencji w konstrukcję programu jak i współpracującej z nim drukarki.

Na początek kilka informacji ogólnych o tym edytorze. STAR TEXTER do prawidłowego funkcjonowania wymaga stacji dysków. Bufor pamięci, w którym przechowywuje on zredagowany tekst,posiada pojemność ok. 20 kB. Jest on zorganizowany w 250 linii po 80 znaków każda, co umożliwia obróbkę tekstu o objętości 7-8 stron maszynopisu formatu A-4. Teksty o większej objętości muszą być dzielone na części, zapisywane na dyskietce, które są redagowane oraz drukowane oddzielnie. Edytor STAR TEXTER składa się z trzech zasadniczych programów użytkowych: programu instalacyjnego o nazwie INSTALL.BAS, właściwego edytora oraz programu umożliwiającego definiowanie potrzebnych użytkownikowi znaków. Program ten napisany jest w języku BASIC i zapisany jest w zbiorze o nazwie STARFONT.BAS. Ponieważ dzięki temu programowi STAR TEXTER posiada cechę tak istotnie, z punktu widzenia polskiego użytkownika, wyróżniającą go spośród innych edytorów, warto chyba prezentację całego edytora rozpocząć od niego.

STARFONT.BAS jest programem, dzięki któremu użytkownik może definiować własne zestawy znaków, bądź modyfikować już istniejące. Mogą to być właśnie specyficzne znaki dla polskiego alfabetu lub inne. Na przykład: matematyk bez problemu może pisać takie wzory:

```
100          b
 Σa(n)  lub ∫f(x)
n=1          a
```

Ktoś, kto bardzo lubi skróty, może w redagowanym piśmie podawać swój numer telefonu poprzedzony znakiem ☎ zamiast skrótem „tel.",natomiast ten, kto ma przyjaciela w Moskwie, może za pomocą tego edytora napisać do niego list zaczynający się tak: Дорогой Борис .... Prakycznie kształt definiowanych znaków zależy jedynie od wyobraźni użytkownika i oczywiście od możliwości komputera, tworzony znak musi się mieścić w matrycy 8x8 pikseli - co może niezadowolić zwolenników kaligrafii, lecz proszę pamiętać, że ATARI jest tylko mikrokomputerem domowym. Na dyskietce zawierającej STAR TEXTER-a można zdefiniować 10 różnych zestawów znaków. Jeżeli liczba ta nie zaspokoi potrzeb użytkownika, można na kolejnej dyskietce zdefiniować i zapisać kolejnych dziesięć zestawów.

Drugą cenną zaletą edytora STAR TEXTER jest możliwość zaadoptowania go do współpracy z dowolnym typem drukarki, którą można podłączyć do mikrokomputera ATARI. Standardowo, edytor zawarty na dyskietce umożliwia wybranie jednej z czterech drukarek:

1. ATARI 1029
2. ATARI 1027
3. EPSON (lub inna) z interfejsem
4. EPSON (lub inna) z łączem wykorzystującym gniazda dżojstików.

Zadanie przystosowania edytora do współpracy z drukarkami innymi niż wyżej wymienione realizuje program INSTALL.BAS. Po jego uruchomieniu wyświetlane są cztery typy drukarek jak wyżej. Jeżeli użytkownik nie posiada jednej z czterech wymienionych drukarek, to w zależności od typu łącza, którym dukarka połączona jest z komputerem (interfejs lub Game Port), musi on wybrać liczbę 3 lub 4. Następnie należy podać programowi INSTALL.BAS sekwencję kodów (liczb dziesiętnych) wywołujących określoną funkcję drukarki. Na przykład: jeżeli drukarka użytkownika włącza podkreślenia, otrzymując sekwencję znaków ESC,CONTROL+Y,to w programie INSTALL.BAS odpowiednia funkcja włączająca podkreślenia musi być wywołana sekwencją kodów 27,25. Program INSTALL.BAS umożliwia zdefiniowanie, poprzez podanie sekwencji kodów, maksymalnie 10 funkcji ponumerowanych od 0 do 9. W redagowanym tekście można używać funkcji - przez podanie odpowiadających im liczb w celu wyróżnienia jego fragmentów. Właściwy edytor stwarza użytkownikowi możliwości redagowania tekstu zbliżone do tych, które oferuje ATARI WRITER. Można w redagowanym tekście wstawiać lub kasować znaki, wyrazy, zdania i całe paragrafy. Można kopiować całe fragmenty tekstu z jednego miejsca w inne, wyrównywać marginesy lub nie, można zlecić edytorowi wyszukanie w napisanym tekście określonych wyrazów i zastąpić je innymi. W zależności od możliwości drukarki współpracującej z komputerem można fragmenty tekstu wyróżniać przez drukowanie ich innym krojem pisma. Na przykład: pracując z drukarką ATARI 1029 można stosować p i s m o rozszerzone. Przy wykorzystaniu innych drukarek można pisać pismem wysokim, niskim, podniesionym lub opuszczonym. Po napisaniu tekstu można zlecić edytorowi jego sformatowanie, wówczas zostanie on przedstawiony na ekranie dokładnie w takiej postaci, w jakiej trochę później zostanie wydrukowany. Do obejrzenia tekstu sformatowanego służy zawarta w edytorze opcja przedstawiająca tekst w trybie 80 kolumnowym - oznacza to, że w jednej linii ekranu wyświetlanych jest 80 znaków. Tryb ten jest bardzo wygodny dla użytkownika, ponieważ wiernie odzwierciedla kształt tekstu, jaki będzie on miał po wydrukowaniu, eliminując jednocześnie żmudne przesuwanie kursora od lewego marginesu do prawego i odwrotnie. Tekst przedstawiony w trybie 80 kolumnowym jest trudniejszy do czytania, z tego powodu, że normalnie znak (wyświetlany w trybie 40 kolumnowym) składa się z 64 pikseli, natomiast w tym trybie jest on „przepołowiany" - składa się z 32 pikseli (4x8).

Inną cenną zaletą edytora jest to, że oferuje on użytkownikowi w trakcie pisania tekstu wykorzystanie, czasami bardzo potrzebnego narzędzia - kalkulatora. Jeżeli w trakcie pisania użytkownik potrzebuje obliczyć wartość skomplikowanego wyrażenia matematycznego,nie musi sięgać po kartkę, ołówek czy elektroniczny kalkulator. Wystarczy, że w jakimś wolnym wierszu wpisze to wyrażenie tak, jak się pisze wyrażenia matematyczne w języku BASIC, wywoła kalkulator z edytora i za chwilę otrzyma wyświetlony na ekranie wynik. Kalkulator w STAR TEXTERZE dopuszcza używanie pięciu operacji, to jest: odejmowania, dodawania, mnożenia, dzielenia i potęgowania. Oprócz tego dozwolone jest stosowanie nawiasów, funkcji matematycznych dostępnych w języku BASIC oraz operatorów logicznych.

Przedstawiając czytelnikom IKS-a jeszcze jeden edytor tekstów opracowany dla mikrokomputerów ATARI,nie chcę stwierdzić, że jest to narzędzie lepsze od innych. Napewno STAR TEXTER jest interesujący dla polskiego użytkownika ze względu na dostępność „polskich" znaków zarówno na ekranie jak i drukarce. Próbkę możliwości STAR TEXTERA obrazuje niniejszy artykuł, napisany, zredagowany i wydrukowany na drukarce ATARI 1029 przy jego pomocy.

Redakcja dziękuje Towarzystwu Wynalazców Ratio za udostępnienie polskiej wersji edytora Star-Texter, którego możliwości zaprezentowaliśmy powyżej.

Czy będziemy mieli polski mikrokomputer?

O historii polskich mikrokomputerów można by napisać poważną rozprawę. Być może wkrótce rozszaleje się dyskusja w poszukiwaniu odpowiedzi dlaczego nam się nie udało. Argumentów z pewnością nie zabraknie.

Po elwrowskim Juniorze, po reklamowanym **Kraku 86** i okrzyczanej **Mazovi** pojawił się na wybrzeżu **BOSMAN-8**. Mikrokomputer skonstruowano w ubiegłym roku; pierwsze prototypy pokazano na Targach Poznańskich zapowiadając możliwość rychłego uruchomienia seryjnej produkcji w gdańskim UNIMORze.

Produkcji, ale czego? **BOSMAN-8** jest mikrokomputerem ośmiobitowym zbudowanym w oparciu o procesor Z80 A. Posiada pamięć RAM o pojemności 512 KB i 16-kilowy EPROM. Standardowo ma dwa napędy dysków elastycznych 3, 3.5 lub 5 1/4 calowych. Pracuje pod nadzorem systemu operacyjnego CPM/R (zgodnym z CPM 2.2) rezydującym w ROM-ie, który umożliwia dołączenie dowolnych urządzeń zewnętrznych z interfejsem szeregowym V24 lub równoległym (Centronics, DZM-180).

Istotną zaletą tego komputera jest dysk krzemowy o pojemności 416 KB, który znacznie usprawnia pracę programów często odwołujących się do pamięci dyskowej. Zwiększenie szybkości i niezawodności mikrokomputera uzyskano również przez umieszczenie systemu operacyjnego w pamięci EPROM.



To oczywiście nie wszystkie niespodzianki W pamięci stałej rezyduje również program poprawiający sprawność komunikacji użytkownika z komputerem. Przede wszystkim pozwala on na przetwarzanie pliku komend zwanego plikiem wsadowym, umożliwia wprowadzenie informacji na drukarkę równolegle z wykonywaniem innego programu czyli tzw. drukowanie w tle.

Program D łączy w sobie możliwości systemowych poleceń CP/M takich jak: DIR, ERA, REN, PIP, STAT, TYPE, DUMP, USER.

BOSMAN-8 jest łatwy w obsłudze, dobre oprogramowanie sprawia, że nadaje się on szczególnie do przetwarzania danych w biurach i małych przedsiębiorstwach, przetwarzania tekstów, obliczeń inżynierskich, prac projektowych i programowych (kompilatory języków: Fortran, Basic, Cobol, C. Forth, Pascal), organizacji sieci lokalnych.

Jeszcze jeden ważny szczegół, BOSMAN-8 z odpowiednim oprogramowaniem może pracować jako końcówka dużych komputerów z wielodostępem np. R-32, IBM-360, VAX, ODRA.

Inaczej mówiąc konstruktorzy włożyli wiele wysiłku, aby z poczciwego już przecież Z80 A wycisnąć wszystko co można. I tu kolejny już raz w historii polskiej myśli technicznej należy schylić czoła przed pomysłowością i odwagą naszego konstruktora. BOSMAN-8, pracownikom Instytutu Okrętowego Politechniki Gdańskiej (bo oni właśnie są konstruktorami tego mikrokomputera), udał się.

Tu jednak wyraźnie podkreślić trzeba, że BOSMAN IBM-em nie jest! Będzie natomiast sukcesem, jeśli uda się uruchomić jego masową produkcję. Konstruktorzy mają swoje racje. Sprzęt klasy IBM nie jest przecież wszystkim potrzebny, a jego obecność w wielu miejscach byłaby po prostu marnotrawstwem. Nasza rodzima informatyka opierać się będzie w najbliższej i chyba, niestety, dalszej przyszłości o lokalne systemy informatyczne. Na budowę rozbudowanych sieci przy obecnym stanie łączności nas po prostu nie stać. Na przykład w wielu przedsiębiorstwach można by sieci konstruować w oparciu właśnie o BOSMAN-a, pozostawiając kompúter lepszej klasy jako jednostkę centralną. Okazuje się, że BOSMAN całkiem dobrze pracuje jako terminal szesnastobitowego IBM-a. Ale to są problemy techniczne. W wielu przypadkach drugorzędną sprawą jest czas uzyskania informacji. Zysk rzędu kilku nawet minut jest w przypadku naszej organizacji niezauważalny. Na przykład zestawienia statystyczne przygotowane na nowoczesnym DRS300 w czasie paru minut (firma ICL) mogą zupełnie spokojnie przeleżeć na biurku referenta nawet kilka dni, zupełnie go nie drażniąc.

Czy dzięki BOSMANOWI zmieni się informatyczny krajobraz naszego kraju?

inż. Andrzej NOWAK

z-ca dyr. Instytutu Okrętowego Politechniki Gdańskiej

— BOSMAN powstał z konieczności. Okazało się, że zrobienie własnego komputera jest tańsze niż zakupienie takiego urządzenia np. w Zabrzu. Sądzę, że udało się zrobić mikrokomputer nowoczesny i niezawodny. Nasze prototypy nie wprawiły jeszcze twórców w zakłopotanie.

Oczywiście nie jest to światowa nowość. Nasza propozycja jest optymalnym rozwiązaniem na miarę możliwości materiałowych i technologicznych, a także rzeczywistych potrzeb.



inż. Mieczysław BARGIEŁ

kierownik Zakładowego Ośrodka Opracowań i Produkcji Doświadczalnego Sprzętu Informatycznego (UNIMOR)

— Od prototypu do seryjnej produkcji droga jeszcze daleka. UNIMOR jest w stanie BOSMAN-a robić. Oczywiście istnieje jeszcze problem kosztów. W tej chwili podstawowa konfiguracja tego mikrokomputera kosztowałaby w granicach 900 — 1300 tys. Przy uruchomieniu produkcji seryjnej być może dałoby się ją obniżyć. Z pewnością taniej kosztowałyby nas dolarowe podzespoły (chociażby napędy). Dotychczas kupujemy je pojedynczo, a taki zakup jest z reguły droższy od hurtu. Poza tym nie zawsze możemy liczyć na rodzimych kooperantów, nie wszyscy potrafią uruchomić nowoczesną produkcję.

Edytor gracz-pocisk

Tomasz MROWIEC
Ludwik PIELA

Możemy wyrzucić papier milimetrowy i tablice konwersji dwójkowo-dziesiętnej. Program nr 4 umożliwi projektowanie graczy i pocisków na ekranie za pomocą manipulatora. Dzięki jego odpowiedniej konstrukcji, łatwo możemy przedstawić graczy dowolnej wielkości, włączając obiekty o podwójnej szerokości lub łączenie dwóch graczy w jednego. Po wykonaniu tego możemy obejrzeć gracza „w akcji", a nawet automatycznie wygenerować procedurę w języku BASIC do używania graczy i pocisków we własnych programach. Wszystkie możliwości programu opisane są w dalszej części.

Używanie edytora

Obejrzyjmy prosty przykład użycia programu. Wybierzmy gracza o rozdzielczości dwuwierszowej, pojedynczej szerokości i długości ośmiu linii. Umieśćmy kursor w prawym, górnym rogu sztalugi, naciśnijmy **„K"** dla wybrania kreślenia, przesuńmy kursor do lewego, dolnego rogu i naciśnijmy klawisz. Mamy gracza składającego się z przekątnej. Naciśnijmy **„S"** w celu zatrzymania, po wyświetleniu menu naciśnijmy **„L"**. Komputer wyświetli teraz gracza w postaci danych numerycznych. Zauważmy, że każda liczba reprezentuje wartość bitową punktów elementarnych G-P w miarę przesuwania się w dół gracza, linia po linii. Aby zobaczyć naszego gracza w trybie grafiki G-P, należy nacisnąć **„0"**, po kilku sekundach pojawi się on na ekranie.

A co będzie, gdy zechcemy zobaczyć nasz obiekt w rozdzielczości jednowierszowej i o wielkości cztery? W oknie tekstowym pojawiają się odpowiednie instrukcje i bez trudu mogą być przeprowadzone ' odpowiednie zmiany. Załóżmy, że chcemy dokonać zmiany obrazu. Po kilku naciśnięciach klawiszy znowu jesteśmy na sztaludze. Różni się ona teraz od oryginalnej, zarówno wielkością, jak i kształtem, odzwierciedlając zmiany. Program zapewnia całkowitą swobodę przy przechodzeniu z jednego obszaru do drugiego. Umożliwia to wykonanie tylu zmian, ile potrzeba, przed zapisaniem ostatecznych danych obrazu na taśmie. Możemy dodać lub wykasować linie u góry lub u dołu, a procedura wypełniania umożliwi wypełnienie obszaru.

Poeksperymentujmy z łączeniem dwóch lub czterech graczy, zmianą parametrów wielkości i rozdzielczości, a uzyskamy lepsze zrozumienie znaczenia i wzajemnych relacji tych parametrów.

Jednym z unikalnych narzędzi używanych w programie jest wykorzystanie funkcji SGN i operatorów logicznych do przekształcania odczytanych wartości STICK(0) na współrzędne X, Y. Aktualne przykłady w programie mogą być trochę niejasne. Procedura jest następująca:

```
10 S=STICK(0)
20 IF S<12 THEN X=X+SGN (8-S):
:X = X + (X<XMIN) — (X>XMAX) : S =
S + (X<8)*4 + 4
30 IF S<15 THEN Y = Y + SGN(13 5—S)
: Y = Y + (Y<YMIN)—(Y<YMAX)
```

Możecie nie wierzyć, lecz te trzy linie odczytają dziewięć pozycji manipulatora, tłumacząc odczyty na nowe współrzędne X, Y i dopasują te współrzędne do granic wyrażonych przez wartości X i Y, maksymalne i minimalne. Dla nieobytych z instrukcjami logicznymi, wartość w nawiasach porównania lub równości staje się jedynką, jeśli warunek jest spełniony i zerem, jeśli niespełniony. Takie instrukcje mogą często zredukować dwie linie kodu, używające instrukcji IF, do jednej linii.

Zwróćmy uwagę na ostrzezenie, jeśli planujemy wykorzystanie programu. Przeznaczony on jest do użycia z rozdzielczością jednowierszową. Adresy dla graczy, w pamięci G-P, indeksowane są przez numer strony. Jednak przy rozdzielczości dwuwierszowej pamięć dla graczy dwa i cztery rozpoczyna się w odstępie połowy strony i nie może być osiągnięta. Program kreślenia G-P obchodzi to ograniczenie poprzez rozszerzenie długości graczy jeden i trzy w celu włączenia graczy dwa i cztery odpowiednio. Działa to jednak tylko wtedy, gdy sąsiedni gracze przesuwani są razem.

Instrukcja programu

A. Pytania początkowe

1. Wprowadzenie rozdzielczości. Rozdzielczość wszystkich graczy i pocisków musi być taka sama w jednym programie.

2. Wprowadzenie gracza lub pocisku. Gracz ma szerokość ośmiu punktów elementarnych G-P; pocisk ma szerokość dwóch punktów.

3. Wprowadzenie liczby graczy. Mamy do dyspozycji czterech graczy, każdy o szerokości ośmiu punktów G-P. Można podać dowolną ilość graczy umieszczonych obok siebie i przesuwanych razem.

4. Wprowadzenie wielkości. Parametr ten może być zmieniony później w programie.

5. Wprowadzenie liczby linii pionowych. Linie mogą być dodane lub wykasowane później w programie.

B. Sztaluga kreślenia

Sztaluga kreślenia ma kolor zielony. Kreślony gracz pojawi się w kolorze pomarańczowym. Kursor wskazywany jest przez jaśniej zabarwiony obszar zielony lub pomarańczowy. W celu zmiany koloru należy nacisnąć przycisk manipulatora. Do przesuwania kursora używamy manipulatora.

Opcje klawiatury:

(0-9) Steruje szybkością ruchu kursora. 0-szybko, 9-wolno, 2-szybkość początkowa

(K)reślenie: kreśli linie od aktualnej pozycji kursora. Przesuńmy kursor do nowej pozycji i naciśnijmy przycisk w celu wykreślenia linii między dwoma punktami.

(W)ypełnianie: Używane do wypełnienia obszaru. Musi on być ograniczony przez kolor pomarańczowy na wszystkich bokach. Należy umieścić kursor poniżej najwyższej zielonej spacji i jeśli to możliwe, powyżej najniższej zielonej spacji oraz nacisnąć W. Dla obszarów o urozmaiconym kształcie procedura ta może wymagać powtórzenia.

(L)inie: Dodaje lub kasuje linie od góry lub dołu sztalugi.

(S)top: Zatrzymuje kreślenie. Po wykreśleniu skomplikowanego rysunku może być zauważalne oczekiwanie podczas tłumaczenia diagramu na dane numeryczne.

C. Menu opcji

(O)bejrzeć: Obserwacja gracza w grafice G-P. Przesuwamy gracza przy użyciu manipulatora. W czasie realizacji tej opcji można nacisnąć:

(W)ielkość — do zmiany wielkości gracza

```
ZL 4 REM ********************
FG 5 REM *                  *
AR 6 REM *   Program nr 4   *
FI 7 REM *                  *
ZP 8 REM ********************
UE 10 DIM A$(1),P(3),PY(3),PX(3):
   PA=2:LA=1590:CO=24:PLX=53248:P
   LY=1780:PLL=1784
DX 15 IF PEEK(1536)<>162 THEN GOS
   UB 9000
YV 20 PM=PEEK(106)-16:PMB=PM*256:
   POKE 54279,PM:FOR A=0 TO 3:P(A
   )=PMB+A*24:NEXT A
GW 30 GRAPHICS 0:? :? "PODAJ ROZD
   ZIELCZOSC PIONOWA:":? :? "  (1
   )=DWUWIERSZOWA":? :? "  (2)=JE
   DNOWIERSZOWA":?
TX 40 TRAP 30:INPUT RE:IF RE>2 OR
    RE<1 THEN 30
DI 50 ? :? "CZY CHCESZ DEFINIOWAC
   GRACZA CZY":? "POCISK";:TRAP
   50:INPUT A$:IF A$<>"G" AND A$<
   >"P" THEN 50
OE 60 IF A$="P" THEN MI=1:NP=1:GO
   TO 80
BK 70 ? :? "ILU GRACZY CHCESZ DEF
   INIOWAC (1-4)";:TRAP 70:INPUT
   NP:IF NP<1 OR NP>4 THEN 70
IG 80 NP=NP-1:? :? "PODAJ ROZMIAR
    POZIOMY (1,2 LUB 4)";:TRAP 80
   :INPUT SI:IF SI<1 OR SI>4 OR S
   I=3 THEN 80
RK 90 GOSUB 170
DR 100 ? :? "ILE LINII BEDZIE MIA
   L GRACZ (1-24)";:TRAP 100:INPU
   T VL:IF VL<1 OR VL>24 THEN 100
NA 110 GOSUB 200:SETCOLOR 2,12,6:
   COLOR 3:X=20*G:Y=10*G-1:PLOT X
   ,Y:DRAWTO X+XD,Y
KB 120 IF STRIG(0)=0 THEN POKE 71
   0,42+(PEEK(710)=42)*156:PLOT X
   ,Y:DRAWTO X+XD,Y
JL 130 S=STICK(0):IF S<15 THEN GO
   SUB 250
VZ 140 IF PEEK(53775)<255 THEN 30
   0
SW 150 FOR A=1 TO 25*PA:NEXT A
MO 160 GOTO 120
UG 170 IF RE*(NP+1)*SI<17 THEN RE
   TURN
BP 180 GRAPHICS 0:? :? "NIE MOGE
   ZDEFINIOWAC 4 GRACZY O ROZDZIE
   LCZOSCI JEDNOWIERSZOWEJ I ROZM
   IARZE >2."
TP 190 ? " ROZMIAR BEDZIE ZMIENIO
   NY NA 2.":SI=2:? :? "NACISNIJ
   RETURN":INPUT A$:RETURN
BP 200 XD1=RE*SI:XD=XD1-1:H=(NP+1
   )*XD1:IF MI THEN H=H/4
AI 210 G=1+((H>5) OR (VL>20))+2*(
   (H>10) OR (VL>40)):GRAPHICS 19
   +(G>1)*G
UI 220 YMIN=INT(10*G-VL/2):YMAX=I
   NT(10*G+VL/2-1):XMIN=20*G-4*H:
   XMAX=20*G+4*H-1-XD
VA 230 SETCOLOR 0,2,8:SETCOLOR 1,
   12,4:COLOR 2:FOR Y=YMIN TO YMA
   X:PLOT XMIN,Y:DRAWTO XMAX+XD,Y
   :NEXT Y:RETURN
VM 250 GOSUB 290
RO 260 IF S<12 THEN X=X+SGN(8-S)*
   XD1:X=X+(X<XMIN)*XD1-(X>XMAX)*
   XD1:S=S+(S<8)*4+4
ZE 270 IF S<15 THEN Y=Y+SGN(13.5-
   S):Y=Y+(Y<YMIN)-(Y>YMAX)
TQ 280 LOCATE X,Y,A:POKE 710,PEEK
   (707+A)+2:COLOR 3:PLOT X,Y:DRA
   WTO X+XD,Y:RETURN
JB 290 COLOR 1+(PEEK(710)=198):PL
   OT X,Y:DRAWTO X+XD,Y:RETURN
VW 300 GOSUB 400:IF A=ASC("K") TH
   EN 450
CJ 310 IF A=ASC("W") THEN 600
QU 320 IF A=ASC("S") THEN GOSUB 7
   00:GOTO 900
RN 330 IF A=ASC("L") THEN GOSUB 7
   00:GOTO 750
UX 340 IF A<58 AND A>47 THEN PA=A
   -47
MO 350 GOTO 120
IY 400 OPEN #1,4,0,"K:":GET #1,A:
   CLOSE #1
WX 410 IF PEEK(53755)<255 THEN 41
   0
ZE 420 RETURN
BZ 450 X1=X:Y1=Y
ZO 460 S=STICK(0):IF S=15 THEN 46
   0
UI 470 GOSUB 260
WV 480 IF STRIG(0)=0 THEN 510
KA 490 S=STICK(0):IF S<15 THEN GO
   SUB 250
PY 500 GOTO 480
WS 510 COLOR 1:X2=X:Y2=Y:Y3=Y1:A=
   XD1*SGN(X2-X1):B=SGN(Y2-Y1):C=
   X2-X1+A:D=Y2-Y1+B
SW 520 IF C=0 OR D=0 THEN FOR A=0
    TO XD:PLOT X1+A,Y1:DRAWTO X2+
   A,Y2:NEXT A:GOTO 570
UU 530 FOR X=X1 TO X2 STEP A:FOR
   Y=Y3 TO Y2 STEP B:PLOT X,Y:DRA
   WTO X+XD,Y
HB 540 IF (X-X1+A)/C=(Y-Y1+B)/D T
   HEN Y=Y+B:GOTO 560
RS 550 IF (X-X1+A)/C>(Y-Y1+B)/D T
   HEN NEXT Y
BJ 560 Y3=Y:NEXT X:X=X-A:Y=Y-B
DU 570 GOSUB 280:GOTO 120
SH 600 X1=X:COLOR 2:PLOT X,Y:DRAW
   TO X+XD,Y:B=670
RS 610 GOSUB B:LOCATE X,Y-1,A:IF
   A=2 THEN Y=Y-1:GOTO 610
TQ 620 GOSUB B:LOCATE X-1,Y,A:IF
   A=2 THEN X=X-1:GOTO 620
IB 630 GOSUB B:COLOR 1:PLOT X,Y
US 640 GOSUB B:LOCATE X+1,Y,A:IF
   A=2 THEN X=X+1:PLOT X,Y:GOTO 6
   40
PB 650 X=X1:Y=Y+1:GOSUB B:LOCATE
   X,Y,A:IF A=2 THEN 620
EO 660 GOSUB 280:GOTO 130
UV 670 IF X=XMAX OR X=XMIN OR Y=Y
   MAX OR Y=YMIN THEN POP :GOSUB
   280:GOTO 120
ZS 680 RETURN
FA 700 GOSUB 290:FOR A=0 TO NP:B=
   0:X1=XMIN+8*A*XD1:FOR Y=YMIN T
   O YMIN+VL-1
DO 710 C=128:D=0:FOR X=X1 TO X1+7
   *XD1 STEP XD1:IF MI THEN FOR X
   =XMIN TO XMIN+XD1 STEP XD1
WN 720 LOCATE X,Y,E:IF E=1 THEN D
   =D+C
PB 730 C=C/2:NEXT X:B=B+1:POKE P(
   A)+B,D:NEXT Y:NEXT A:RETURN
LV 750 D=0:T=0:? :? "DODAJ LUB WY
   KASUJ LINIE":INPUT A$:IF A$="W
   " THEN D=1:GOTO 770
LR 760 IF A$<>"D" THEN 750
TP 770 ? :? "U GORY CZY U DOLU":I
   NPUT A$:IF A$="G" THEN T=1:GOT
   O 790
NN 780 IF A$<>"D" THEN 770
KZ 790 ? :? " ILE LINII":TRAP 790
   :INPUT C:IF D THEN C=C-(C-VL+1
   )*(C>=VL):GOTO 840
BB 800 IF VL+C>24 THEN C=24-VL:?
   :? "ZA DUZO":? :? "MOZESZ DODA
   C TYLKO ";C;" LINII":? :? " NA
   CISNIJ RETURN":INPUT A$
MC 810 IF T THEN FOR A=0 TO NP:FO
   R B=VL TO 0 STEP -1:POKE P(A)+
   B+C,PEEK(P(A)+B):NEXT B:NEXT A
   :GOSUB 860
LJ 820 VL=VL+C:IF  NOT D AND  NOT
    T THEN GOSUB 870
LL 830 GOTO 1400
IR 840 IF T THEN FOR A=0 TO NP:FO
   R B=C TO VL:POKE P(A)+B-C,PEEK
   (P(A)+B):NEXT B:NEXT A
DR 850 FOR A=0 TO NP:FOR B=VL-C+1
    TO VL:POKE P(A)+B,0:NEXT B:NE
   XT A:VL=VL-C:GOTO 1400
GZ 860 FOR A=0 TO NP:FOR B=1 TO C
   :POKE P(A)+B,0:NEXT B:NEXT A:R
   ETURN
VT 870 FOR A=0 TO NP:FOR B=VL-C+1
    TO VL:POKE P(A)+B,0:NEXT B:NE
   XT A:RETURN
OV 900 GRAPHICS 0:? :? "CO CHCESZ
    ZROBIC:":? :? " OBEJRZEC GRAC
   ZA?":? CHR$(29);" ZAPISAC DANE
   ?"
CZ 905 ? CHR$(29);" WYSWIETLIC DA
   NE?"
NL 910 ? CHR$(29);" POPRAWIC GRAC
   ZA?":? CHR$(29);" ROZPOCZAC Z
   NOWYM GRACZEM?":? CHR$(29);" S
   TOP?"
FT 920 GOSUB 400:IF A=ASC("O") TH
   EN 1000
CN 930 IF A=ASC("Z") THEN 1500
EX 940 IF A=ASC("W") THEN 1450
KQ 950 IF A=ASC("P") THEN GOSUB 1
   70:GOTO 1400
NN 960 IF A=ASC("R") THEN RUN
UF 970 IF A=ASC("S") THEN STOP
PG 980 GOTO 900
YO 1000 TRAP 40000:GOSUB 1240
RS 1010 GRAPHICS 7:GRAPHICS 3:POK
   E 559,46+16*(RE=2):POKE 53277,
   3:GOSUB 1220:Y=64*RE-VL/2
IB 1020 FOR A=1774 TO 1787:POKE A
   ,0:NEXT A
PB 1030 IF RE=1 AND NP>0 THEN FOR
    A=0 TO 1:POKE PLL+A,VL+128:NE
   XT A:GOTO 1050
CD 1040 FOR A=0 TO NP:POKE PLL+A,
   VL:NEXT A
DI 1050 PXM=255-NP*8*SI
LQ 1060 FOR A=0 TO 3:PY(A)=PMB+51
   2*RE+128*RE*A:NEXT A
AK 1070 FOR A=0 TO NP:FOR B=1 TO
   VL:POKE PY(A)+B,PEEK(P(A)+B):N
   EXT B:NEXT A
JL 1080 POKE 1788,PM+2*RE:Z=USR(1
   696)
TT 1090 ? :? "'RETURN' - PROGRAM
   GLOWNY ":? "'K' ZMIANA KOLORU"
   :? "'W' ZMIANA WIELKOSCI"
QA 1100 ? "'R' ZMIANA ROZDZIELCZO
   SCI Z ";RE;" NA ";3-RE;
ED 1110 S=STICK(0):X=0
PF 1120 IF S<12 THEN X=SGN(8-S):S
   =S+(S<8)*4+4:X=X+(PX(0)+X<0)-(
   PX(0)+X>PXM)
RN 1130 FOR A=0 TO NP:PX(A)=PX(A)
   +X:NEXT A
WI 1140 IF S<15 THEN Y=Y+SGN(13.5
   -S):Y=Y+(Y<0)-(Y>255-VL)
BJ 1150 FOR A=0 TO NP:POKE PLY+A,
   Y:POKE PLX+A,PX(A):NEXT A
WI 1160 IF PEEK(53775)=255 THEN 1
   110
HY 1170 GOSUB 400:IF A=155 THEN G
   OSUB 1230:GOSUB 170:GOTO 900
KS 1180 IF A=ASC("W") THEN 1250
ZG 1190 IF A=ASC("K") THEN 1300
FN 1200 IF A=ASC("R") THEN RE=3-R
   E:GOSUB 1230:GOTO 1010
NU 1210 GOTO 1110
SB 1220 PX(0)=128-4*(NP+1)*SI:FOR
    A=0 TO NP:PX(A)=PX(0)+8*SI*A:
   POKE PLX+A,PX(A):POKE 53256+A,
   SI-1:NEXT A:RETURN
KC 1230 FOR A=0 TO 3:POKE PLX+A,0
   :POKE PLY+A,0:NEXT A:RETURN
GI 1240 FOR A=704 TO 707:POKE A,C
   O:NEXT A:RETURN
YU 1250 ? "WIELKOSC = ";SI:? "NAC
   ISNIJ 1, 2 LUB 4  DLA  ZMIANY
         WIELKOSCI"
EP 1255 ? "NACISNIJ RETURN ABY WY
   JSC Z PROCEDURY ZMIANY WIELKOS
   CI";
ZL 1260 GOSUB 400:IF A=155 THEN 1
   [illegible]
GZ 1270 IF A=49 OR A=50 OR A=52 T
   HEN SI=A-48:GOSUB 1220
RB 1280 GOTO 1250
IC 1300 A=INT(CO/16):B=CO-A*16:?
   :? "KOLOR=";A,"JASKRAWOSC = ";
   B
EA 1310 ? "POZIOME RUCHY MANIPULA
   TORA - ZMIANA  KOLORU; PIONOW
   E - ZMIANA JASKRAWOSCI"
HF 1320 ? "<RETURN> WYJSCIE Z PRO
   CEDURY";
IZ 1330 S=STICK(0):IF PEEK(53775)
   <255 THEN 1380
XN 1340 IF S=15 THEN 1330
YI 1350 IF S=7 OR S=11 THEN A=A+S
   GN(8-S):A=A-(A>15)*16+(A<0)*16
XD 1360 IF S=13 OR S=14 THEN B=B-
```

```
    2*SGN(13.5-S):B=B-(B>14)*16+(B
    <0)*16
GV 1370 CO=16*A+B GOSUB 1240:GOTO
     1300
ZT 1380 GOSUB 400 IF A=155 THEN 1
    030
PF 1390 GOTO 1300
YC 1400 GOSUB 200:COLOR 1:FOR A=0
     TO NP:X=XMIN+A*XD1*8:FOR Y=1
    TO VL
JY 1410 B=PEEK(P(A)+Y):C=128:FOR
    X1=0 TO 7:IF MI THEN FOR X1=0
    TO 1
AQ 1420 IF B>=C THEN B=B-C:GOSUB
    1440
ZL 1430 C=C/2:NEXT X1:NEXT Y:NEXT
     A:X=20*G:Y=10*G-1 GOSUB 280 G
    OTO 120
YZ 1440 PLOT X+X1*XD1,YMIN+Y-1:DR
    AWTO X+X1*XD1+XD,YMIN+Y-1 RETU
    RN
SH 1450 GRAPHICS 0:FOR A=0 TO NP:
    ? :IF MI THEN ? "POCISK":GOTO
    1470
WM 1460 ? "GRACZ ";A
QM 1470 FOR B=1 TO VL:? PEEK(P(A)
    +B),",", NEXT B:? CHR$(126) NE
    XT A
IS 1480 ? :? " NACISNIJ <RETURN>
    ABY WROCIC DO PROGRAMU":GOSUB
    400:GOTO 900


AY 1500 ? :? "JAKI JEST NR TEGO G
    RACZA, POCISKU LUB 1-SZEJ GRUP
    Y GRACZY":TRAP 900:INPUT PN
BM 1510 ? :? "UMIESC CZYSTA TASME
     W MAGNETOFONIE,  NACISNIJ PL
    AY, RECORD I RETURN"
AG 1520 OPEN #1,8,0,"C:":FOR A=1
    TO 64:? #1;"R.");:NEXT A
ZE 1530 FOR A=0 TO NP:LI=31400+(P
    N+A)*100:? #1:IF MI THEN 1600
JX 1540 ? #1;LI;"REM** GRACZ ";PN
    +A;":ROZDZIELCZOSC,DLUGOSC,WIE
    LKOSC,KOLOR.":GOSUB LA
WT 1550 ? #1;LI;"DATA ";RE;",";VL
    ;",";SI;",";CO:GOSUB LA
EI 1560 FOR B=1 TO VL:IF B=1 OR B
    =26 THEN ? #1:? #1;LI;"DATA ";
    :GOSUB LA:GOTO 1580
AX 1570 ? #1;",";
TT 1580 ? #1;PEEK(P(A)+B);:NEXT B
    :NEXT A:CLOSE #1:TRAP 900:INPU
    T A:GOTO 900
JY 1590 LI=LI+10:RETURN
TM 1600 LI=31000+(PN+A)*100:? #1;
    LI;"REM** POCISK ";PN;"ROZDZIE
    LCZOSC,DLUGOSC,WIELKOSC.":GOSU
    B LA
JL 1620 ? #1;LI;"DATA ";RE;",";VL
    ;",";SI:GOSUB LA:GOTO 1560
YQ 9000 FOR I=1536 TO 1706:READ A
    :POKE I,A:NEXT I:RETURN
OS 9010 DATA 162,3,189,244,6,240,
    89,56,221,240,6,240,83,141,254
    ,6,106,141
DR 9020 DATA 255,6,142,253,6,24,1
    69,0,109,253,6,24,109,252,6,13
    3,204,133
FO 9030 DATA 206,189,240,6,133,20
    3,173,254,6,133,205,189,248,6,
    170,232,46,255
GN 9040 DATA 6,144,16,168,177,203
    ,145,205,169,0,145,203,136,202
    ,208,244,76,87
XO 9050 DATA 6,160,0,177,203,145,
    205,169,0,145,203,200,202,208,
    244,174,253,6
GW 9060 DATA 173,254,6,157,240,6,
    189,236,6,240,48,133,203,24,13
    8,141,253,6
JY 9070 DATA 109,235,6,133,204,24
    ,173,253,6,109,252,6,133,206,1
    89,240,6,133
MC 9080 DATA 205,189,248,6,170,16
    0,0,177,203,145,205,200,202,20
    8,248,174,253,6
FQ 9090 DATA 169,0,157,236,6,202,
    48,3,76,2,6,76,98,228,0,0,104,
    169
OC 9100 DATA 7,162,6,160,0,32,92,
    228,96
```

(K)olor — do zmiany koloru gracza. Poziomy ruch manipulatora zmienia kolor. Pionowy ruch zmienia jaskrawość. Kody liczbowe koloru i jaskrawości są wyświetlane po dokonaniu zmiany.

(R)ozdzielczość — do zmiany rozdzielczości.

(W)yświetlić: Wyprowadza dane obrazu gracza.

(P)oprawić: Powrót do sztalugi kreślenia. Jeśli wielkość lub rozdzielczość były zmieniane w czasie obserwacji, wtedy odpowiednio modyfikowana jest sztaluga. Jeśli ograniczenia wielkości zostały przekroczone, wtedy wielkość zredukowana jest do dwóch.

(Z)apamiętać: Przechowuje dane numeryczne obrazu na taśmie. Włóż czystą taśmę do magnetofonu. Naciśnij RECORD i PLAY. Odpowiedz na pytanie o numer gracza. Jeśli to pierwszy gracz, odpowiedz zero. Jeśli poprzedni gracze byli przechowani, wprowadź kolejny numer. Na przykład: jeśli pierwsze kreślenie miało szerokość dwóch graczy, będą to gracze zero i jeden. Dlatego wprowadź teraz numer dwa. Dane będą przesyłane na taśmę przy użyciu instrukcji PRINT.

(R)ozpocząć z nowym graczem. Używane do kasowania obrazu aktualnego gracza i rozpoczęcia kreślenia nowego.

D. Wykorzystanie zapisanych danych

Zapisane na taśmie dane mogą być wprowadzone do komputera przez instrukcję ENTER. Mogą być one połączone z rezydującym w pamięci programem. Instrukcje DATA będą miały numery zaczynające się od 31 000.

A. Jeśli program nie ma być łączony z rezydującym programem, wprowadź NEW i naciśnij <RETURN>.

B. Umieść taśmę w magnetofonie.

C. Naciśnij PLAY.

D. Wprowadź ENTER„C" i naciśnij dwa razy <RETURN>

E. Przed zakończeniem wprowadzania na ekranie wystąpi dwa razy READY.

F. Wyprowadź (za pomocą LIST) to, co wprowadziłeś.

Efekty specjalne

Każda próba przemieszczania punktów w trybie graficznym o dużej rozdzielczości, pionowo lub ukośnie, z dużą szybkością, jest zwykle trudna, jeśli nie niemożliwa do wykonania w BASIC'u. Jest to dosyć trudne do uzyskania nawet w języku maszynowym.

Chcemy zademonstrować prostą technikę otrzymania skomplikowanych ruchów, dla uzyskania specjalnych efektów, wykonaną całkowicie w BASIC'u. **Aby to zobaczyć, należy wprowadzić program nr 5 i uruchomić go.** Składa się on z dwóch części. Druga działa w nieskończonej pętli i w celu jej przerwania należy nacisnąć klawisz RESET.

Jak widać, program napisany jest całkowicie w BASIC'u i nie ma tu skomplikowanych obliczeń. Aby się dowiedzieć, jak to działa, proszę uruchomić program przez napisanie **GOTO 1200**.

Idea tej techniki była krótko opisana w poprzednim odcinku. Opis dotyczył używania maski pola gry do zwiększenia rozdzielczości gracza. W tym programie po prostu rozwinięto tę ideę przez użycie maski jako matrycy, która tworzy iluzję ruchu w pionie i po przekątnej w czasie, gdy pionowy pasek gracza przesuwany jest poziomo po ekranie.

Poszczególne etapy uzyskiwania takich efektów są następujące:

1) Kreślenie maski poprzez wypełnienie ekranu lub wybranego obszaru kolorem pola gry: ustawienie koloru i jego jaskrawości do takich jak tło.

2) Kreślenie matrycy w kolorze tła (aby była niewidoczna).

3) Tworzenie pionowego paska o odpowiedniej wysokości i szerokości za pomocą grafiki gracz—pocisk.

4) Ustawienie priorytetu pola gry nad graczem i przesuwanie paska poziomo poza maską.

Aby lepiej zobaczyć, jak to jest zrobione, prześledźmy pierwszą część programu:

linia 160 — wybranie trybu graficznego;

linia 790 — czyszczenie ekranu w procesie kreślenia poprzez ustawienie jego koloru na czarny;

linia 490 — kreślenie maski w kolorze 3 pola gry;

linie 500—520 — kreślenie matrycy dla wystrzału w kolorze 0 (tła);

linie 530—570 — kreślenie jabłka w kolorze 1 pola gry;

linie 680—730 — kreślenie matrycy dla rozbryzgu jabłka w kolorze 0;

linie 740—770 — kreślenie karabinu w kolorze 2 pola gry;

linie 820—890 — przygotowanie grafiki gracz—pocisk;

linia 850 tworzy gracza nr 2 w postaci paska o szerokości 5 bitów rozciągającego się od góry do dołu ekranu;

linie 860—870 — tworzą gracza nr 1 jako pocisk karabinowy;

linia 830 ustala priorytet (za pomocą POKE 623,2), gracz nr 1 jest przed frontem maski, a gracz nr 2 jest poza maską;

linie 190 i 250 — ustawianie kolorów;

linie 260 do 320 — gracz nr 2 przesuwa się poziomo przez ekran i jego kolor zmienia się od białego do czerwonego (gdy przechodzi poza jabłkiem).

Ludwik PIELA

Tomasz MROWIEC

```
FK 1 REM ****************
FJ 2 REM *              *
LU 3 REM * PROGRAM nr 5 *
FL 4 REM *              *
FO 5 REM ****************
JI 10 N1=784 N2=862 N3=0 GOTO 150
   REM SKOK DO PROGRAMU GLOWNEGO
KT 20 REM OPOZNIENIE
JO 30 FOR D=1 TO DELAY2 NEXT D RE
   TURN
NL 40 REM DZWIEK
MM 50 SOUND 0,17,10,10
LR 60 FOR D=1 TO 4 NEXT D
MW 70 SOUND 0,0,0,0 RETURN
ZT 80 REM OSCYLOSKOP
UY 90 FOR H=43 TO 200 STEP 2
DG 100 POKE 53250,H
GR 110 IF H=55 OR H=93 OR H=131 O
   R H=169 THEN GOSUB 50
KP 120 FOR D=1 TO 1 NEXT D
XI 130 NEXT H:GOTO 90
CQ 140 REM USTAWIENIE PARAMETROW
IC 150 DELAY2=500 FLAG=0 G=3
DR 160 GRAPHICS 7 GOSUB 790 POKE
   709,10 POKE 752,1 POKE 657,10
   ? "STRZAL W JABLKO"
ZW 170 POKE 656,1:POKE 657,12:? "
   (za 14 sek.)"
SL 180 GOSUB 820:GOSUB 490
HH 190 POKE 708,68 POKE 709,6 POK
   E 712,0
AU 200 SOUND 0,30,10,14:FOR D=1 T
   O 25:NEXT D:SOUND 0,0,0,0
YM 210 REM STRZAL W JABLKO
BF 220 POKE 657,17:? CHR$(199);CH
   R$(207),CHR$(212),CHR$(207),CH
   R$(215),CHR$(217) GOSUB 30
TA 230 ? CHR$(125):POKE 657,18:?
   "LEL" GOSUB 30
LS 240 ? CHR$(125):POKE 657,18:?
   CHR$(208);CHR$(193);CHR$(204):
   GOSUB 30
LB 250 POKE 706,12 POKE 705,79
TY 260 FOR L=1 TO 35 SOUND 0,L,4,
   15 NEXT L
PN 270 FOR V=15 TO 0 STEP -G
JF 280 POKE 705,64+V SOUND 0,L,4,
   V
CU 290 NEXT V:POKE 705,0
GI 300 FOR H=60 TO 122 STEP G:POK
   E 53250,H:NEXT H
GD 310 POKE 708,0 POKE 706,68
EE 320 FOR H=123 TO 200 STEP G:PO
   KE 53250,H NEXT H
VF 330 POKE 706,0 POKE 53250,0:FO
   R D=1 TO DELAY2 NEXT D:IF FLAG
    THEN 430
AI 340 POKE 752,1 ? CHR$(125) POK
   E 657,6:? "TO SAMO W WOLNIEJSZ
   YM TEMPIE."
YB 350 FOR D=1 TO DELAY2:NEXT D:G
   =1
HP 360 POKE 706,68:FOR D=1 TO DEL
   AY2:NEXT D:? CHR$(125)
DL 370 FOR H=200 TO 123 STEP -3:P
   OKE 53250,H:NEXT H
GB 380 POKE 708,68:POKE 706,12
FR 390 FOR H=122 TO 60 STEP -3:PO
   KE 53250,H:NEXT H
XI 400 FOR D=1 TO DELAY:NEXT D
DK 410 POKE 53250,250:FLAG=1:DELA
   Y2=800:GOTO 220
XW 420 REM USTAWIENIE PARAMETROW
   OSCYLOGRAMU
GO 430 POKE 53249,0:POKE 53250,0:
   GOSUB 790
FT 440 POKE 709,6:? CHR$(125):POK
   E 657,15:? "OSCYLOSKOP"
UV 450 GOSUB 910
RG 460 FOR I=PMB+N1 TO PMB+N2:POK
   E I,3:NEXT I
DX 470 POKE 706,14:? CHR$(125):GO
   TO 90
QZ 480 REM RYSOWANIE JABLKA
TY 490 COLOR 3:FOR Y=0 TO 79:PLOT
    0,Y:DRAWTO 159,Y:NEXT Y
VR 500 COLOR 0:FOR Y=38 TO 42 STE
   P 2
LO 510 FOR X=14 TO 72 STEP 2:PLOT
    X,Y:NEXT X:NEXT Y
VC 520 PLOT 74,38:PLOT 74,42
ML 530 COLOR 1:Y1=35 X2=82
LS 540 FOR X1=77 TO 74 STEP -1
MJ 550 Y1=Y1+1:X2=X2+1
QT 560 PLOT X1,Y1:DRAWTO X2,Y1
GW 570 NEXT X1
UY 580 FOR Y1=40 TO 41
FO 590 PLOT 74,Y1:DRAWTO X2,Y1
VQ 600 NEXT Y1:Y1=41
FE 610 FOR X1=75 TO 78
NO 620 Y1=Y1+1:X2=X2-1
QO 630 PLOT X1,Y1:DRAWTO X2,Y1
GR 640 NEXT X1
UX 650 PLOT 80,36:DRAWTO 80,34:PL
   OT 81,34
KY 660 PLOT 81,33:PLOT 82,33:PLOT
    82,32
IZ 670 COLOR 3:PLOT 79,36:PLOT 81
   ,36
XT 680 COLOR 0
YJ 690 PLOT 83,35:DRAWTO 120,0:PL
   OT 84,36:DRAWTO 140,0
TP 700 PLOT 85,37:DRAWTO 159,4:PL
   OT 86,38:DRAWTO 159,16
TI 710 PLOT 87,39:DRAWTO 159,28:P
   LOT 87,40:DRAWTO 159,40
PH 720 PLOT 87,41:DRAWTO 159,52:P
   LOT 86,42:DRAWTO 159,64:PLOT 8
   5,43:DRAWTO 159,76
QB 730 PLOT 84,44:DRAWTO 140,79:P
   LOT 83,45:DRAWTO 120,79
GQ 740 POKE 709,0:COLOR 2:FOR Y=3
   7 TO 43
MB 750 PLOT 0,Y:DRAWTO 13,Y
MN 760 NEXT Y
ZU 770 PLOT 14,37:PLOT 14,43:? CH
   R$(125) RETURN
OS 780 REM CZYSZCZENIE EKRANU
PR 790 FOR N=0 TO 4 POKE 708+N,0
   NEXT N:POKE 710,N3
ZE 800 RETURN
EO 810 REM USTAWIENIE PARAMETROW
   P/M
HU 820 POKE 705,0:POKE 706,0:A=PE
   EK(106)-24:POKE 53249,0:POKE 5
   3250,0
BI 830 POKE 54279,A:PMB=256*A:POK
   E 53277,2:POKE 623,2:POKE 559,
   46
WV 840 FOR I=PMB+640 TO PMB+895:P
   OKE I,0:NEXT I
ZS 850 FOR I=PMB+N1 TO PMB+N2:POK
   E I,31:NEXT I
HY 860 RESTORE 870:FOR I=PMB+694
   TO PMB+698:READ A:POKE I,A:NEX
   T I
HT 870 DATA 192,240,254,240,192
IW 880 POKE 53257,0:POKE 53258,0:
   POKE 53249,62
ZW 890 RETURN
HL 900 REM WYKRESLANIE OSCYLOGRAM
   U
IA 910 TRAP 1000
TP 920 COLOR 3:FOR Y=0 TO 79:PLOT
    0,Y DRAWTO 159,Y NEXT Y
YU 930 COLOR 0:X=0
LZ 940 PLOT X,32:DRAWTO X,2
QW 950 X=X+12
ID 960 DRAWTO X,2:DRAWTO X,32
RA 970 X=X+12
PB 980 DRAWTO X,32:DRAWTO X,2
RL 990 GOTO 950
NU 1000 TRAP 40000:Y=59:HT=-15:SL
   =-3
QI 1010 FOR X=0 TO 114 STEP 38
XO 1020 PLOT X,Y
CJ 1030 FOR XX=0 TO 19 STEP 19
WL 1040 DRAWTO X+XX+10,Y
WS 1050 DRAWTO X+XX+13,Y+SL
XO 1060 DRAWTO X+XX+14,Y+SL
QG 1070 DRAWTO X+XX+14,Y+SL+HT
XU 1080 DRAWTO X+XX+14,Y+SL
YQ 1090 DRAWTO X+XX+15,Y+SL
BX 1100 DRAWTO X+XX+18,Y
XA 1110 SL=-SL HT=-HT
VL 1120 NEXT XX:NEXT X:DRAWTO 159
   ,Y:RETURN
CB 1200 N1=780 N2=882 N3=4:GOTO 1
   50
```

PROGRAM AMSTRAD 25

3 punkty

Program wymaga wczytania współrzędnych trzech punktów na płaszczyźnie, po czym podaje komunikat o ich wzajemnym położeniu. Jeśli okaże się, że wszystkie leżą na jednej prostej, dodatkowo pojawi się jej równanie w postaci y=f(x). Metoda zastosowana w programie polega na porównaniu długości odcinków łączących punkty.

Program pozbawiony jest instrukcji skoków, lecz jego struktura nie dorównuje Pascalowi. Problem tu przedstawiony nie daje się jednak uchwycić w Pascalu. Dlaczego? Spróbujcie odpowiedzi poszukać sami...

Krzysztof MAMCARZ

```
10 ' ** 3 punkty '86
20 DEFINT i
30 DEFSTR w
40 DEFREAL a-d,m,n,x,y
50 '
60 DEF FN dlugodc(x1,x2,y1,y2)=ROUND(SQR
((x1-x2)^2+(y1-y2)^2),5)
70 DEF FN m(x1,x2,y1,y2)=(y2-y1)/(x2-x1)
80 DEF FN n(x1,y1,m)=y1-m*x1
90 DIM x(3)
100 DIM y(3)
110 '
120 MODE 1
130 wsp="xayaxbybxcyc"
140 PRINT "PODAJ WSPOLRZEDNE PUNKTOW A,B
,C"
150 FOR i=1 TO 3
160 PRINT MID$(wsp,4*i-3,2);"=";
170 INPUT x(i)
180 PRINT MID$(wsp,4*i-1,2);"=";
190 INPUT y(i)
200 NEXT
210 PRINT
220 '
230 ab=FN dlugodc(x(1),x(2),y(1),y(2))
240 bc=FN dlugodc(x(2),x(3),y(2),y(3))
250 ac=FN dlugodc(x(1),x(3),y(1),y(3))
260 IF ab=0 AND bc=0 AND ac=0 THEN PRINT
 "A=B=C":PRINT"przez 1 punkt przechodzi
pek prostych":END
270 IF ab=0 OR bc=0 OR ac=0 THEN PRINT "
przez dwa punkty przechodzi":PRINT"ZAWSZ
E jedna prosta":END
280 c=ROUND(ABS(ac-bc),4)
290 ab=ROUND(ab,4)
300 IF (ab=c) THEN ELSE PRINT "punkty A,
B,C nie sa wspolliniowe...":END
310 PRINT "punkty A,B,C sa wspolliniowe"
320 IF x(1)=x(2) AND x(2)=x(3) THEN PRIN
T"prosta x=";x(1):END
330 IF y(1)=y(2) AND y(2)=y(3) THEN PRIN
T"prosta y=";y(1):END
340 m=FN m(x(1),x(2),y(1),y(2)):n=FN n(x
(1),y(1),m)
350 PRINT "prosta y=";
360 IF m<>0 AND m<>1 THEN PRINT m;CHR$(8
);"x"; ELSE IF m=1 THEN PRINT"x";
370 IF n>0 THEN PRINT"+";n ELSE IF n<>0
THEN PRINT n
380 END
```

PROGRAM 26

ATARI

Interpolacja Lagrange'a

Zdarzają się sytuacje, gdy mając funkcję w postaci dyskretnej, tzn. zbioru punktów Xi i Yi (i = 1, 2,..., N); trzeba uzyskać jej przybliżony wzór. Jedną z metod otrzymania tego wzoru jest utworzenie, a następnie przekształcenie układu równań, z którego wyliczymy współczynniki wielomianu interpolującego funkcję daną tablicą. W poniższym przykładzie zawężono otrzymywane wielomiany do stopnia pierwszego i drugiego, a więc do ich wyznaczenia potrzebne będą dwie lub trzy współrzędne punktów. Kolejność wprowadzania współrzędnych punktów jest dowolna przy czym jako pierwszy deklarujemy X a po nim Y. Jeżeli jeden z X-ów tablicy jest równy zero, wprowadzamy go w ostatniej kolejności, gdyż ze względu na sposób wyprowadzenia wzorów zadanie może okazać się niewykonalne.

Janusz W. JANIEC

```
50 REM ************************
55 REM *                      *
60 REM *     INTERPOLACJA     *
65 REM *                      *
70 REM *     .LAGPANGE'A      *
75 REM *                      *
80 REM ************************
100 OPEN #1,4,0,"K:"
110 ? CHR$(125)
120 ? :? "Ile masz danych punkto
w w tablicy ?"
150 ? :GET #1,ST:PO=ST-49
155 IF PO<1 OR PO>2 THEN 110
160 ? CHR$(28);CHR$(28);"Wprowad
zaj kolejno dane z tablicy  "
170 ? :? "[ Xn,Yn,Xn+1,Yn+1,...]
":?
175 TRAP 410
180 ON PO GOTO 200,300
200 INPUT X1,Y1,X2,Y2
210 B=(Y2*X1-Y1*X2)/(X1-X2)
220 A=(Y1-B)/X1
230 ? CHR$(125)
240 POSITION 2,10:? "Y=";A;"*X+"
;
250 IF B<0 THEN ? CHR$(30);B:GOT
O 420
260 ? B:GOTO 420
300 INPUT X1,Y1,X2,Y2,X3,Y3
310 XA=X1-X2:XB=X2/X1:XC=X1^2:XD
=XA*XB
320 XE=XD*XC:XF=X3^2:XG=XB^2
330 C=(Y3*XE+XF*(Y2*X1-Y1*(XD+XG
*X1))+XC*X3*(XG*Y1-Y2))/(XE-XF*X
D+(1-XG)*(XF*X1-X3*XC))
340 B=(Y2-C+XG*(C-Y1))/XD
350 A=(Y1-B*X1-C)/XC
360 ? CHR$(125)
380 POSITION 2,10:? "Y=";A;"*X2+
";
385 IF B<0 THEN ? CHR$(30);B;:GO
TO 395
390 ? B;
395 ? "*X+";:IF C<0 THEN ? CHR$(
30);C:GOTO 420
400 ? C:GOTO 420
410 IF PEEK(195)=11 THEN ? :? "M
IANOWNIK JEST=0, TRZEBA ZMIENIC'
X'"
420 POSITION 2,15:? "[ Wcisnij d
owolny klawisz ]"
430 ? :GET #1,P:GOTO 110
440 REM ************************
```

* SUMA KONTROLNA / ETYKIETA *

LK	50	QW	55	CC	60	QX	65
VI	70	QY	75	LN	80	PA	100
AU	110	RL	120	FT	150	HP	155
VT	160	PI	170	MI	175	SK	180
JM	200	UB	210	BG	220	AZ	230
QE	240	FE	250	PA	260	RY	300
GR	310	FH	320	KP	330	AY	340
HP	350	BG	360	DJ	380	EZ	385
UG	390	CB	395	OX	400	ST	410
MN	420	RN	430	PN	440		

PROGRAM 27

ATARI

Disasembler

Analizowanie programów napisanych w kodzie maszynowym jest czynnością bardzo żmudną i pracochłonną, lecz niezbędną szczególnie wtedy, gdy chcemy dobrze poznać algorytm działania programu. Najwięcej pracy w takiej sytuacji pochłania „przejście" z kodu maszynowego programu, zwanego również kodem wynikowym, na jego kod źródłowy, który jest czytelny dla programisty. Kod źródłowy składa się z mnemonicznych nazw rozkazów mikroprocesora i przyporządkowanych im wartości argumentów (operandów), a jego uzyskiwanie z kodu wynikowego nazywa się disasemblacją. Komputer może w tej operacji z powodzeniem zastąpić programistę — wystarczy mieć odpowiedni program zwany disa-

Disasemblerem jest prezentowany program o nazwie DISASM. Wykonuje on disasemblację obszaru pamięci, którego początek i koniec określa użytkownik programu przez podanie odpowiednich adresów. Program do disasemblacji musi być najpierw wprowadzony do pamięci komputera. Program DISASM wczytuje do pamięci komputera programy, które mają być disasemblowane, zarówno z dysku, jak i z taśmy. Wyniki disasemblacji wyprowadzane są na monitor i jednocześnie na drukarkę, jeśli jest podłączona do komputera. Mają one następującą postać: adres pamięci, w którym znajduje się rozkaz, nazwa mnemoniczna rozkazu i jego argument w postaci liczby dziesiętnej, kod dziesiętny rozkazu i argumentu oraz kod heksadecymalny rozkazu i argumentu. Adres pamięci na monitor wyprowadzany jest w postaci dziesiętnej, a na drukarkę dodatkowo w postaci liczby heksadecymalnej.

Wszystkie rozkazy skoków w programie disasemblowanym są wyróżnione przez wydrukowanie dodatkowego znaku przed ich nazwą mnemoniczną. Znakiem tym jest

— gwiazdka, gdy rozkaz skoku jest skokiem bezwzględnym rozkazy JMP i JSR)

— plus, gdy rozkaz skoku jest skokiem względnym do „przodu" to znaczy, gdy skok wykonywany jest do adresu większego niż adres, w którym znajduje się rozkaz skoku;

— minus, gdy rozkaz skoku jest skokiem względnym do „tyłu", to znaczy, gdy skok wykonywany jest do adresu mniejszego niż adres, w którym znajduje się rozkaz skoku

Rozkazami skoku względnego są rozkazy BPL, BMI, BVC, BCC, BCS, BNE, BEQ

Miejsca, do których przekazywane jest sterowanie w disasemblowanym programie po wykonaniu rozkazu skoku względnego, podawane są w postaci adresów bezwzględnych obliczonych przez dodanie (gdy skok do „przodu") lub odjęcie (gdy skok do „tyłu") wartości adresu, w którym znajduje się rozkaz skoku i wartośc jego argumentu. Takie rozwiązanie bardzo ułatwia analizowanie uzyskanego w wyniku disasemblacji kodu źródłowego

Dla tych czytelników, którzy wolą posługiwać się adresami pamięci w postaci liczb heksadecymalnych, lub będą chcieli wprowadzać wyniki inaczej niż to zrobiono w programie DISASM, podaję dokładną zawartość 70-bajtowej zmiennej tekstowej o nazwie A$. Zawiera ona wynik disasemblacji jednego rozkazu który następnie jest wyprowadzany na monitor i drukarkę Poszczególne bajty tej zmiennej zawierają

A$(1,5) — adres pamięci, w którym znajduje się disasemblowany rozkaz w postaci liczby dziesiętnej

A$(7,11) — ten sam adres w postaci liczby heksadecymalnej

A$(13,24) — nazwa mnemoniczna rozkazu oraz jego argument w postaci liczby dziesiętnej,

A$(29 39) — kod dziesiętny rozkazu i argumentu

A$(46,53) — kod heksadecymalny rozkazu i argumentu

Pozostałe bajty tej zmiennej zawierają spacje

H. KRASUSKI

```
1 REM ****************************
2 REM
3 REM DISASSEMBLER
4 REM
5 REM ****************************
10 ? CHR$(125):? "Prosze chwilke
 poczekac... "
11 DIM SPEC$(16),CIO$(6):CIO$(1,
1)=CHR$(104):CIO$(2,2)=CHR$(162)
:CIO$(3,3)=CHR$(32)
12 CIO$(4,5)="LV":CIO$(6,6)=CHR$
(228):CBUF=1000
14 J0=0:J1=1:J2=2:J3=3:J4=4
15 J18=9*J2
16 J9=900:DRNAG=1100
25 DIM ROZ$(4),A$(70),C$(969),HE
X$(16),MK$(76)
27 RESTORE 30
28 FOR I=1 TO 76:READ D:MK$(I)=C
HR$(D):NEXT I
30 DATA 169,0,133,18,133,19,133,
20,104,104,133,204,104,133,203,1
60,0,177,203,153,206,0,200,192,4
,208,246
32 DATA 165,207,201,63,208,1,96,
24,165,203,105,4,133,203,144,2,2
30,204,160,0,177,203,197,205,240
,19,200
34 DATA 192,13,208,245,24,165,20
3,105,13,133,203,144,2,230,204,2
4,144,199,152,133,18,96
40 RESTORE 45
42 FOR I=1 TO 969:READ D:C$(I)=C
HR$(D):NEXT I
45 DATA 32,65,68,67,2,105,101,11
7,2,109,125,121,2,2,97,113,2,32,
65,78,68,2,41,37,53,2,45,61,57,2
,2,33,49
47 DATA 2,32,65,83,76,10,2,6,22,
2,14,30,2,2,2,2,2,2,43,66,67,67,
2,2,2,2,2,2,2,2,144,2,2,2,43,6
6,67,83
49 DATA 2,2,2,2,2,2,2,2,176,2,
2,2,43,66,69,81,2,2,2,2,2,2,2,
2,240,2,2,2,32,66,73,84,2,2,36,2
,2,44,2
51 DATA 2,2,2,2,2,2,43,66,77,73,
2,2,2,2,2,2,2,2,48,2,2,2,43,66
,78,69,2,2,2,2,2,2,2,2,208,2,2
,2,43
53 DATA 66,80,76,2,2,2,2,2,2,2,2
,2,16,2,2,2,32,66,82,75,2,2,2,2,
2,2,2,2,0,2,2,2,2,43,66,86,67,2,
2,2,2,2
55 DATA 2,2,2,2,80,2,2,2,43,66,8
6,83,2,2,2,2,2,2,2,2,112,2,2,2
,32,67,76,67,2,2,2,2,2,2,2,2,24,
2,2,2,2
57 DATA 32,67,76,68,2,2,2,2,2,2,
2,2,216,2,2,2,2,32,67,76,73,2,2,
2,2,2,2,2,2,88,2,2,2,2,32,67,76,
86,2,2,2
59 DATA 2,2,2,2,2,184,2,2,2,2,32
,67,77,80,2,201,197,213,2,205,22
1,17,2,2,193,209,2,32,67,80,88,2
,224,228
61 DATA 2,2,236,2,2,2,2,2,2,2,32
,68,69,67,2,2,198,214,2,206,222,
2,2,2,2,2,2,32,68,69,88,2,2,2,2,
2,2,2,2
63 DATA 202,2,2,2,2,32,68,69,89,
2,2,2,2,2,2,2,2,136,2,2,2,2,32,6
9,79,82,2,73,69,85,2,77,93,89,2,
2,65,81,2
65 DATA 32,73,78,67,2,2,230,246,
2,238,254,2,2,2,2,2,2,32,73,78,8
8,2,2,2,2,2,2,2,2,232,2,2,2,2,32
,73,78,89
67 DATA 2,2,2,2,2,2,2,2,200,2,2,
2,2,42,74,77,80,2,2,2,2,2,76,2,2
,2,2,2,2,108,42,74,83,82,2,2,2,2
,2,32,2,2
69 DATA 2,2,2,2,2,32,76,68,65,2,
169,165,181,2,173,189,185,2,2,16
1,177,2,32,76,68,88,2,162,166,2,
182,174,2
71 DATA 190,2,2,2,2,2,32,76,68,8
9,2,160,164,180,2,172,188,2,2,2,
2,2,2,32,76,83,82,74,2,70,86,2,7
8,94,2,2,2
73 DATA 2,2,2,32,78,79,80,2,2,2,
2,2,2,2,2,234,2,2,2,2,32,79,82,6
5,2,9,5,21,2,13,29,25,2,2,1,17,2
,32,80,72
75 DATA 65,2,2,2,2,2,2,2,2,72,2,
2,2,2,32,80,72,80,2,2,2,2,2,2,2,
2,8,2,2,2,2,32,80,76,65,2,2,2,2,
2,2,2,2
77 DATA 104,2,2,2,2,32,80,76,80,
2,2,2,2,2,2,2,2,40,2,2,2,2,32,82
,79,76,42,2,38,54,2,46,62,2,2,2,
2,2,2,32
79 DATA 82,79,82,106,2,102,118,2
,110,126,2,2,2,2,2,2,32,82,84,73
,2,2,2,2,2,2,2,2,64,2,2,2,2,32,8
2,84,83
81 DATA 2,2,2,2,2,2,2,2,96,2,2,2
,2,32,83,66,67,2,233,229,245,2,2
37,253,249,2,2,225,241,2,32,83,6
9,67,2,2
83 DATA 2,2,2,2,2,2,8,2,2,2,2,32
,83,69,68,2,2,2,2,2,2,2,2,248,2,
2,2,2,32,83,69,73,2,2,2,2,2,2,2,
2,120,2,2
85 DATA 2,2,32,83,84,65,2,2,133,
149,2,141,157,153,2,2,129,145,2,
32,83,84,88,2,2,134,2,150,142,2,
2,2,2,2,2
87 DATA 2,32,83,84,89,2,2,132,14
8,2,140,2,2,2,2,2,2,2,32,84,65,8
8,2,2,2,2,2,2,2,2,170,2,2,2,2,32
,84,65,89
89 DATA 2,2,2,2,2,2,2,2,168,2,2,
2,2,32,84,83,88,2,2,2,2,2,2,2,2,
186,2,2,2,2,32,84,88,65,2,2,2,2,
2,2,2,2
90 DATA 138,2,2,2,2,32,84,88,83,
2,2,2,2,2,2,2,2,154,2,2,2,2,32,8
4,89,65,2,2,2,2,2,2,2,2,152,2,2,
2,2,32
91 DATA 67,80,89,2,192,196,2,2,2
04,2,2,2,2,2,2,2,32,63,63,63,2,2
,2,2,2,2,2,2,2,2,2,2,2
92 GOSUB CBUF:? "Adres poczatkow
y obszaru ";:INPUT AP:AP1=AP:LN=
J1
93 ? :? "Adres koncowy obszaru "
;:INPUT AK
94 POKE 82,J0:HEX$="0123456789AB
CDEF":DRU=J0:? :? "----- -------
--- ---------- ---------"
95 ? "adres r-z w jez. ":? "dzi
es.assemblera  kod dzies.  kod h
eks."
96 ? "----- ----------  -------
-- ---------"
97 TRAP 99:LPRINT :DRU=J1:GOSUB
DRNAG
99 TRAP 40000:BRKF=J0
100 BYTE=PEEK(AP):POKE 205,BYTE:
IF BYTE=2 THEN ROZ$=" ???":Z=9:G
OTO 400
110 X=USR(ADR(MK$),ADR(C$)):Z=PE
EK(18)+J1:FOR A=J1 TO J2+J2:ROZ$
(A,A)=CHR$(PEEK(A+205)):NEXT A:G
OTO 400
390 A=INT(A/16)
395 B=J1+A-INT(A/16)*16
397 ROZ$(C,C)=HEX$(B,B):RETURN
400 IF ROZ$=" BRK" AND AP<AK THE
N AP=AP+J1:BRKF=J1:GOTO 100
401 FOR A=J1 TO 70 STEP 10:A$(A)
="          ":NEXT A
402 IF ROZ$=" ???" THEN Z=9
405 A$(J3+J2-INT(CLOG(AP)))=STR$
(AP):A$(13,16)=ROZ$
410 A=AP:C=J2+J2:GOSUB 395
420 FOR C=J3 TO J1 STEP -J1:GOSU
B 390:NEXT C
430 A$(7,11)=ROZ$
435 IF BYTE=J3 THEN BYTE=44
438 IF BYTE=J0 THEN A$(31,31)="0
":GOTO 450
440 A$(31-INT(CLOG(BYTE)))=STR$(
BYTE)
450 A=BYTE:C=J2:GOSUB 395
460 C=1:GOSUB 390:A$(46,47)=ROZ$
(J1,J2)
470 GOTO 460+Z*30
490 A$(J18,J18)="A":AP=AP+J1:FL=
J3:GOTO J9
520 A$(19)=STR$(PEEK(AP+J1)):A$(
J18,J18)="#":AP=AP+J2:FL=J1:GOTO
 J9
550 A$(J18)=STR$(PEEK(AP+J1)):AP
=AP+J2:FL=J1:GOTO J9
580 A$(J18)=STR$(PEEK(AP+J1)):A$
(21,22)=",X":AP=AP+J2:FL=J1:GOTO
 J9
610 A$(J18)=STR$(PEEK(AP+J1)):A$
(21,22)=",Y":AP=AP+J2:FL=J1:GOTO
 J9
640 A$(J18)=STR$(PEEK(AP+J1)+PEE
K(AP+J2)*256):AP=AP+J3:FL=J2:GOT
O J9
670 A$(J18)=STR$(PEEK(AP+J1)+PEE
K(AP+J2)*256):A$(23,24)=",X":AP=
AP+J3:FL=J2:GOTO J9
700 A$(J18)=STR$(PEEK(AP+J1)+PEE
K(AP+J2)*256):A$(23,24)=",Y":AP=
AP+J3:FL=J2:GOTO J9
730 AP=AP+J1:FL=J3:GOTO J9
760 A=PEEK(AP+J1):IF A>127 THEN
A=A-256:A$(13,13)=CHR$(45)
770 A$(J18)=STR$(A+AP+J2):AP=AP+
J2:FL=J1:GOTO J9
790 A$(J18,J18)="(":A$(J18+J1)=S
TR$(PEEK(AP+J1)):A$(22)=",X)":AP
```

```
=AP+J2:FL=J1:GOTO J9
820 A$(J18,J18)="(":A$(J18+J1)=S
TR$(PEEK(AP+J1)):A$(22)="),Y":AP
=AP+J2:FL=J1:GOTO J9
850 A$(J18)="(":A$(J18+J1)=STR$(
PEEK(AP+J1)+PEEK(AP+J2)*256):A$(
24)=")":AP=AP+J3:FL=J2:GOTO J9
880 FL=J3:AP=AP+J1
900 RESTORE
910 IF AP>AK+J1 THEN GOTO 92
915 A=PEEK(AP-FL)
917 IF A=J0 AND FL<J3 THEN A$(35
,35)="0":A$(49,50)="00":GOTO 925
920 IF FL<J3 THEN A$(35-INT(CLOG
(A)),35)=STR$(A):C=J2:GOSUB 395:
C=J1:GOSUB 390:A$(49,50)=ROZ$(J1
,J2)
925 A=PEEK(AP-J1)
927 IF A=J0 AND FL=J2 THEN A$(39
,39)="0":A$(52,53)="00":GOTO 980
930 IF FL=J2 THEN A$(39-INT(CLOG
(A)),39)=STR$(A):C=J2:GOSUB 395:
C=J1:GOSUB 390:A$(52,53)=ROZ$(J1
,J2)
950 A$(56)=" "
955 IF BRKF=J1 THEN ? :? CHR$(12
7);CHR$(30);"BRK":? :BRKF=J0:IF
DRU THEN LPRINT :LPRINT " ","
    FFV"
1080 RETURN
1090 REM ------------------------
1091 REM
1092 REM MODUL DRUKU NAGLOWKA
1093 REM
1094 REM ------------------------
1100 LPRINT "------------    -----
-----------          ------------
        ----------"
1102 LPRINT "   adres         rozk
az w jezyku           kod dzies.
     kod hex."
1104 LPRINT "dzies.hex.          a
ssemblera"
1106 LPRINT "------------    -----
-----------          ------------
        ----------"
960 ? A$(J1,5);A$(13,24);A$(28,3
9);A$(45,53)
980 IF DRU THEN LPRINT A$(J1,11)
;"   ";A$(12,28);"          ";A$(2
9)
990 GOTO 100
995 REM -------------------------
996 REM
997 REM MODUL WCZYTYWANIA PRO-
998 REM GRAMU DO PAMIECI
999 REM -------------------------
1000 ? "Chcesz wprowadzic progra
m do pamieci":? "ktory bedzie di
sassemblowany ? (T/N)"
1002 INPUT SPEC$:IF SPEC$(1,1)="
N" THEN 1080:IF SPEC$(1,1)<>"T"
THEN GOTO 1002
1008 ? :? "Wolny obszar pamieci"
:? "od ";:WP=PEEK(144)+256*PEEK
(145):? WP;:? "  do  ";
1010 WP1=PEEK(741)+256*PEEK(742)
:? WP1
1020 ? "Podaj adres bufora ";:IN
PUT ABUF
1030 IF ABUF>WP1 OR ABUF<WP THEN
 ? "Zly adres ":GOTO 1020
1040 ? "Podaj nazwe zbioru ";:IN
PUT SPEC$
1050 GS=0:IF SPEC$(1,1)="C" THEN
 GS=128
1060 TRAP 1040:CLOSE #2:OPEN #2,
4,GS,SPEC$:BH=INT(ABUF/256):BL=A
BUF-(BH*256):POKE 866,7:POKE 872
,0
1065 POKE 873,PEEK(742)-PEEK(145
):POKE 868,BL:POKE 869,BH:X=USR(
ADR(CIO$))
1070 TRAP 40000:LZB=PEEK(873)*25
6+PEEK(872):? "Zaladowano ";LZB
;" bajtow"
```

*** SUMA KONTROLNA / ETYKIETA ***

HQ	1	UR	2	JI	3	UT	4
HU	5	YH	10	GK	11	VE	12
OJ	14	LN	15	EH	16	LR	25
JV	27	WQ	28	VH	30	YO	32
OS	34	MI	40	XW	42	LB	45
CB	47	GR	49	XV	51	ZN	53
NL	55	GO	57	IH	59	MU	61
PY	63	UZ	65	VU	67	KV	69
OG	71	EG	73	AC	75	CU	77
EB	79	IP	81	BI	83	KG	85
RU	87	SL	89	GJ	90	CG	91
IU	92	YN	93	ST	94	ST	95
PX	96	JE	97	ZV	99	ZV	100
GW	110	XY	390	FX	395	HI	397
WL	400	IX	401	LR	402	OS	405
WS	410	GT	420	ZU	430	CX	435
IL	438	BG	440	KM	450	BX	460
QO	470	AO	490	YS	520	MV	550
MK	580	NO	610	OY	640	NP	670
PJ	700	PH	730	NP	760	OH	770
BV	790	FO	820	RL	850	VY	880
RC	900	FT	910	AF	915	UK	917
EM	920	LZ	925	BY	927	EQ	930
SX	950	WI	955	XZ	960	OR	980
MG	990	VL	995	CA	996	VS	997
UA	998	VX	999	XU	1000	TZ	1002
QF	1008	BS	1010	WD	1020	UJ	1030
FR	1040	EJ	1050	RD	1060	WD	1065
TS	1070	PR	1080	XY	1090	UD	1091
JR	1092	UL	1093	YO	1094	SM	1100
WI	1102	LQ	1104	TK	1106	OY	1110

Przystawka magnetofonowa dla Atari

Jedną z najpopularniejszych pamięci masowych, stosowanych w mikrokomputerach domowych, jest magnetofon kasetowy. Komputery Atari 130 XE, 800 XL wykorzystują w tym celu specjalizowany magnetofon (np. XC 12, XC 11). Zakup takiej pamięci związany jest ze stosunkowo dużym wydatkiem.

W tym artykule znajdziecie opis przystawki umożliwiającej wykorzystanie dowolnego magnetofonu kasetowego (mono lub stereofonicznego) w roli pamięci masowej dla Atari. Koszt przystawki wynosi ok. 2÷3 tys. złotych.

Zasada działania

Zapis informacji (programów) na taśmie magnetofonowej realizowany przez Atari oparty jest na zasadzie modulacji częstotliwości. Oznacza to, że jedynce logicznej odpowiada sygnał o częstotliwości 5,1 kHz, a zeru logicznemu przebieg o częstotliwości 3,9 kHz. Ciąg bitów, który ma być zapisany w kasecie, zamieniony jest w komputerze na sygnały o tych częstotliwościach. Zapis dokonywany jest z szybkością 600 bitów/sekundę.

Zapisanych w ten sposób informacji komputer nie potrafi samodzielnie odczytać. Potrzebny jest układ, który sygnały o częstotliwościach 5,1 kHz i 3,9 kHz (zapisane na taśmie magnetofonowej) zamieni ponownie na sygnały zrozumiałe dla komputera. Przystawka dokonuje tej zamiany i jednocześnie zapewnia, że poziomy napięć tych sygnałów odpowiadają poziomem napięć TTL; tzn. wysokiemu poziomowi napięcia (+ 5 V) odpowiada jedynka logiczna, a zeru logicznemu odpowiada niski poziom napięcia (ØV).

Schemat blokowy przystawki przedstawia rys. 1. Przystawka połączona jest z komputerem za pośrednictwem gniazda łącza szeregowego. Z magnetofonem przystawka połączona jest typową 5-stykową wtyczką magnetofonową. Zasilana jest napięciem + 5 V z komputera.

W czasie wczytywania programu (magnetofon włączony na odtwarzanie) sygnał m.cz. z magnetofonu doprowadzony jest do wzmacniania m.cz. (wzmacniacz operacyjny A). Wzmocniony sygnał podawany jest na aktywne filtry pasmowoprzepustowe **B**, **C**. Filtr **B** nastrojony jest na częstotliwość 3,9 kHz, a filtr **C** na 5,1 kHz. Gdy na wejściu filtra **B** pojawi się przebieg o częstotliwości 3,9 KHz, to będzie on wzmocniony. Jeżeli na wejściu tego filtra pojawi się przebieg o innej częstotliwości (np. 5,1 kHz) to filtr nie będzie tego sygnału wzmacniał.

Podobnie pracuje filtr **C**. Ustawiony jest jednak na inną częstotliwość. Jest to częstotliwość 5,1 kHz. Wyjścia filtrów dołączone są do komputera napięcia zbudowanego na wzmacniaczu **D**. Ta część przystawki jest odpowiedzialna za właściwe dekodowanie informacji. Jeżeli na taśmie pojawi się przebieg o częstotliwości 5,1 kHz („1" logiczna), poprzez magnetofon i wzmacniacz **A**

▶

podawany jest on na filtry **B** i **C**. W tym wypadku zadziała jedynie filtr **C**. Na jego wyjściu pojawi się wzmocniony sygnał, który steruje wejściem odwracającym komparatora. Napięcie na wejście nieodwracające w tym momencie jest niższe. W związku z tym na wyjściu komputera będzie niski poziom napięcia (prawie 0 V). Napięciem tym sterowany jest końcowy stopień odwracający fazę napięcia. Na wyjściu tego stopnia (czyli jednocześnie na wejście komputera) będzie wysoki poziom napięcia (prawie + 5 V), czyli poziom odpowiadający jedynce logicznej.

Kiedy na taśmie pojawi się przebieg o częstotliwości 3,9 kHz, to zadziała filtr **B** i na wyjście komputera będzie wysoki poziom napięcia (napięcie na wyjście „+" wyższe od napięcia na wejście „−"). Po odwróceniu fazy w układzie końcowym na wejściu komputera będzie niski poziom napięcia, odpowiadający zeru logicznemu.

W czasie nagrywania programów na kasecie (magnetofon w trybie zapis) sygnał z komputera podawany jest na dzielnik napięcia. Tak zmniejszony podawany jest na wejście magnetofonu i zapisany na taśmie.

Opis konstrukcji

Schemat ideowy układu zamieszczonego na rysunku 2. Do budowy wykorzystano jeden układ scalony typu **LM 324**. Jest to poczwórny wzmacniacz operacyjny. Cena tego układu na warszawskiej giełdzie wynosi od 400 do 600 złotych. Należy w tym miejscu podkreślić, że ten układ nie może być zastąpiony polskimi wzmacniaczami operacyjnymi (pojedynczymi) typu ULY 7741, ULY 7701 ze względu na to, że nie pracują one poprawnie przy zasilaniu napięciem + 5 V. W opisywanej konstrukcji sygnały z filtrów **B** i **C** podawane są na komparator poprzez mostek prostowniczy **D3** ÷ **D6**. Rolę komparatora odgrywa jeden ze wzmacniaczy operacyjnych. Sygnał z komparatora steruje, poprzez filtr **RC**, tranzystorem **T1**. Przebiegi z kolektora tranzystora podawane są do wejścia „**IN**" komputera.

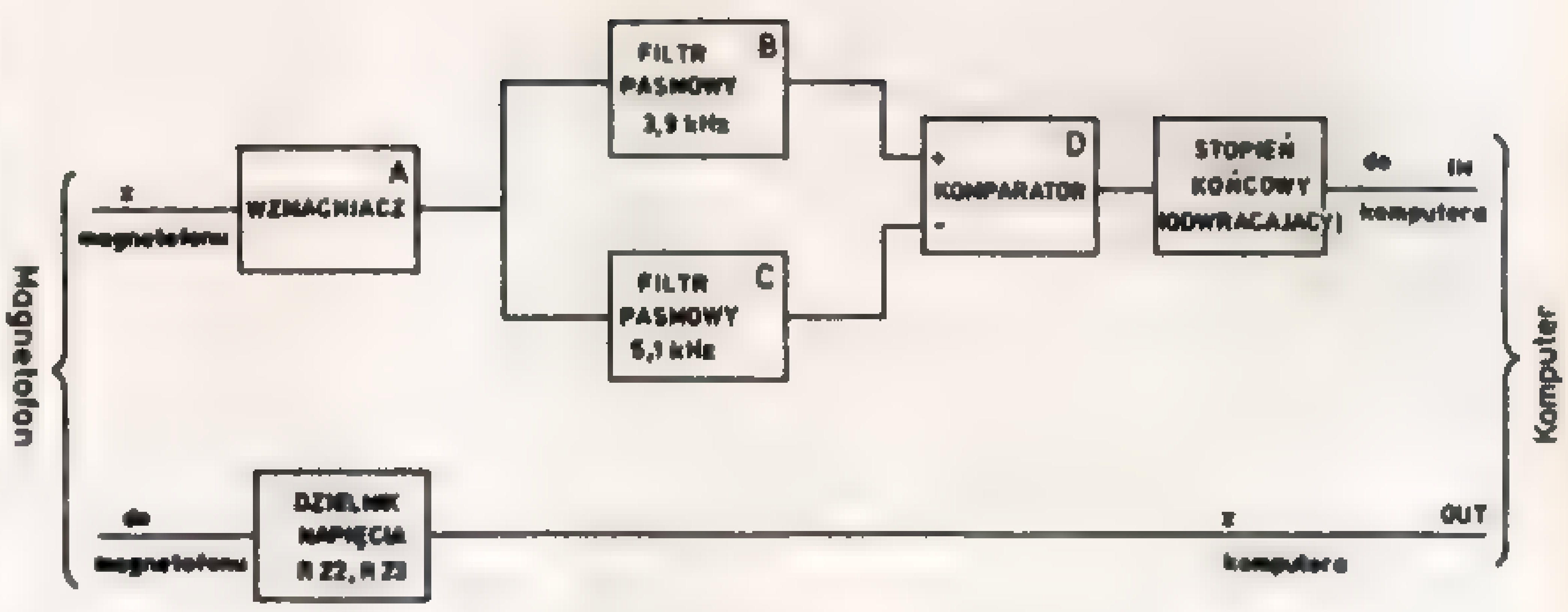
Przystawkę zmontowano na jednostronnie drukowanej płytce o wymiarach 10 x 6 cm (rys. 3). Przystawka łączona jest z komputerem wtyczką dołączoną do gniazda interfejsu szeregowego.

Jeśli nie mamy możliwości zakupienia firmowej wtyczki, to wykonać ją można we własnym zakresie według rysunku 5. Jako elementów stykowych użyło styków z gniazda szufladowego Eltra typu 881025. Po wykonaniu obu detali łącza i uzbrojeniu części A w styki sklejamy je klejem (np. distalem). Połączenia. Wtyczkę z przystawką łączymy tzw. skrętkami (przewód sygnałowy okręcony przewodem masy).

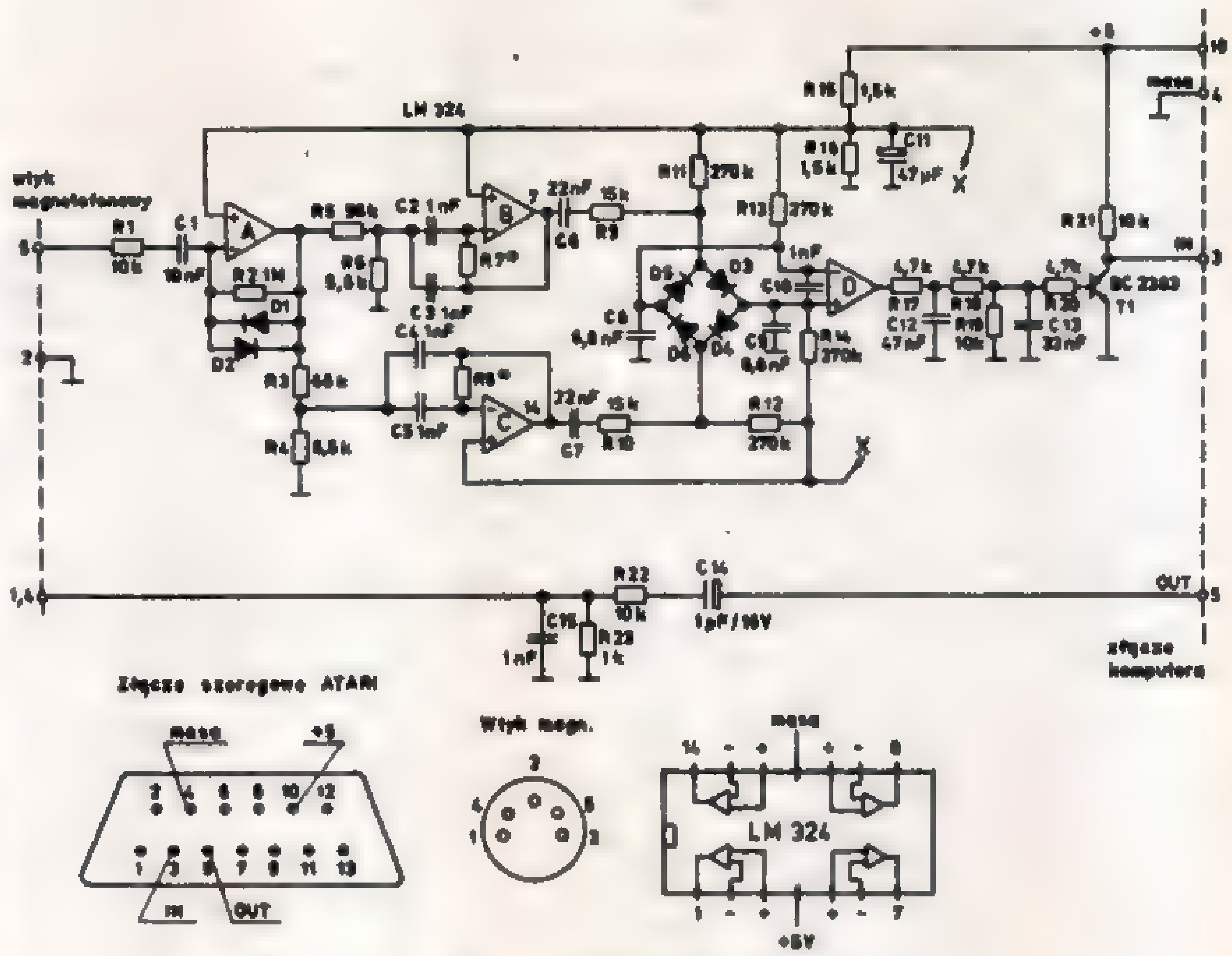
Wtyk magnetofonowy łączymy z układem dwużyłowym przewodem ekranowym.

Uruchomienie

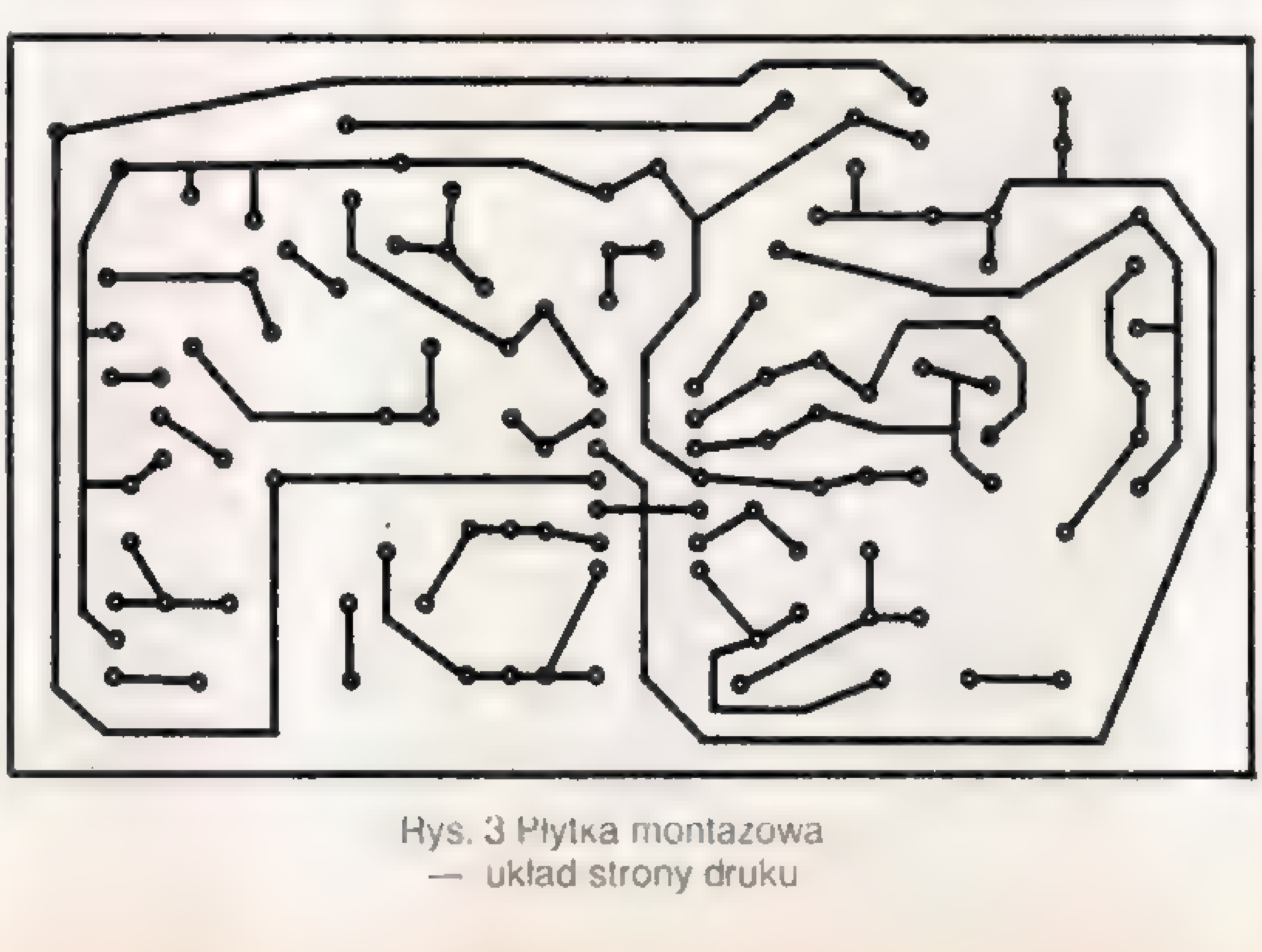
Po zlutowaniu wszystkich elementów i sprawdzeniu poprawności montażu możemy przystąpić do strojenia. Przed tym należy zamiast rezystorów R7 i R8 wlutować potencjometry montażowe 470 kΩ. Przystawkę zasilamy napięciem + 5 V (nie z komputera), łączymy z magnetofonem, do którego wkładamy kasetę z nagranym dowolnym programem dla Atari. Do zestrojenia przystawki musimy użyć oscyloskopu (jest w pracowni fizycznej każdej szkoły średniej), w którym podstawę czasu ustawiamy na 5 ÷ 10 ms. Włączamy magnetofon na odtwa-



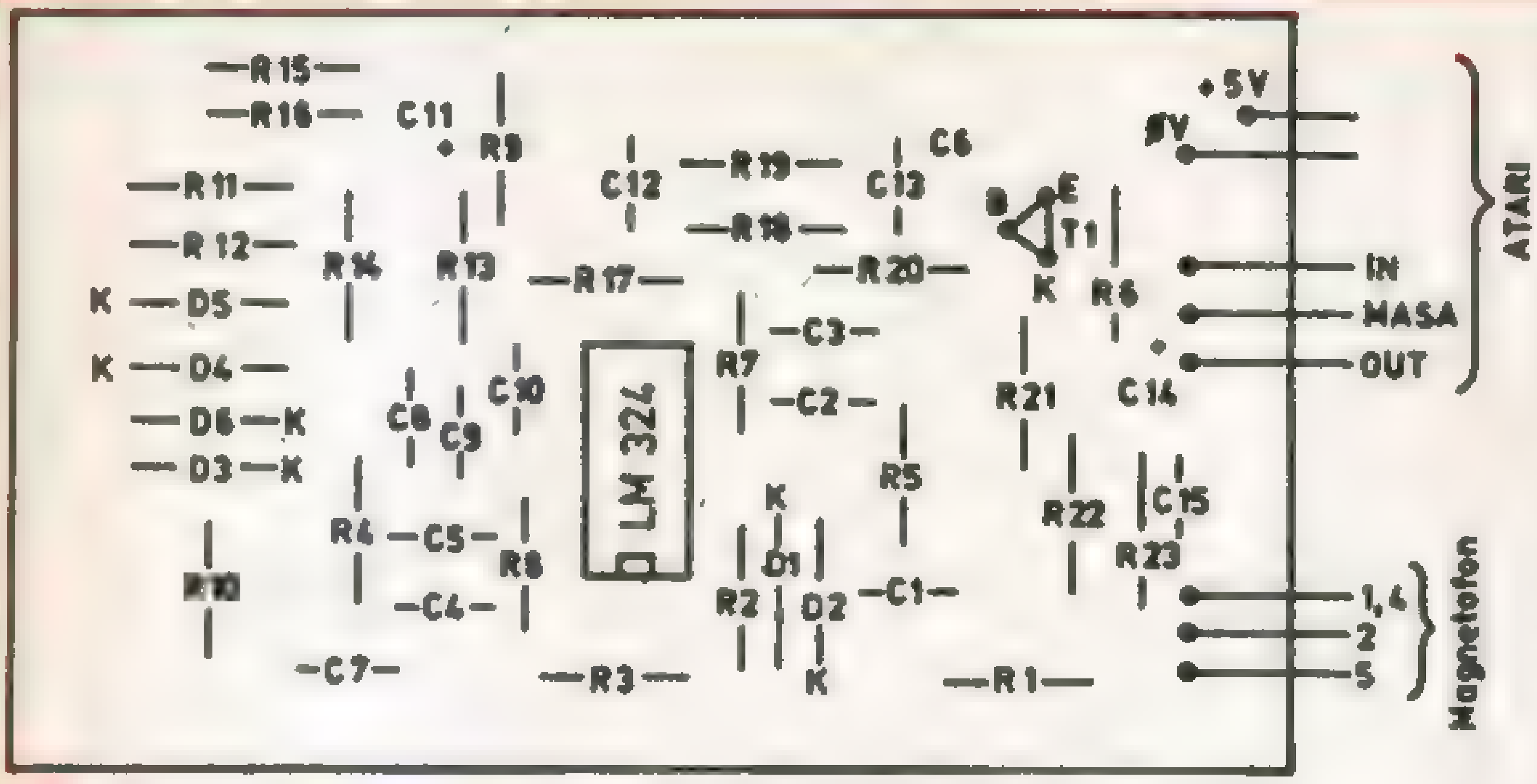
Rys. 1 Schemat blokowy przystawki



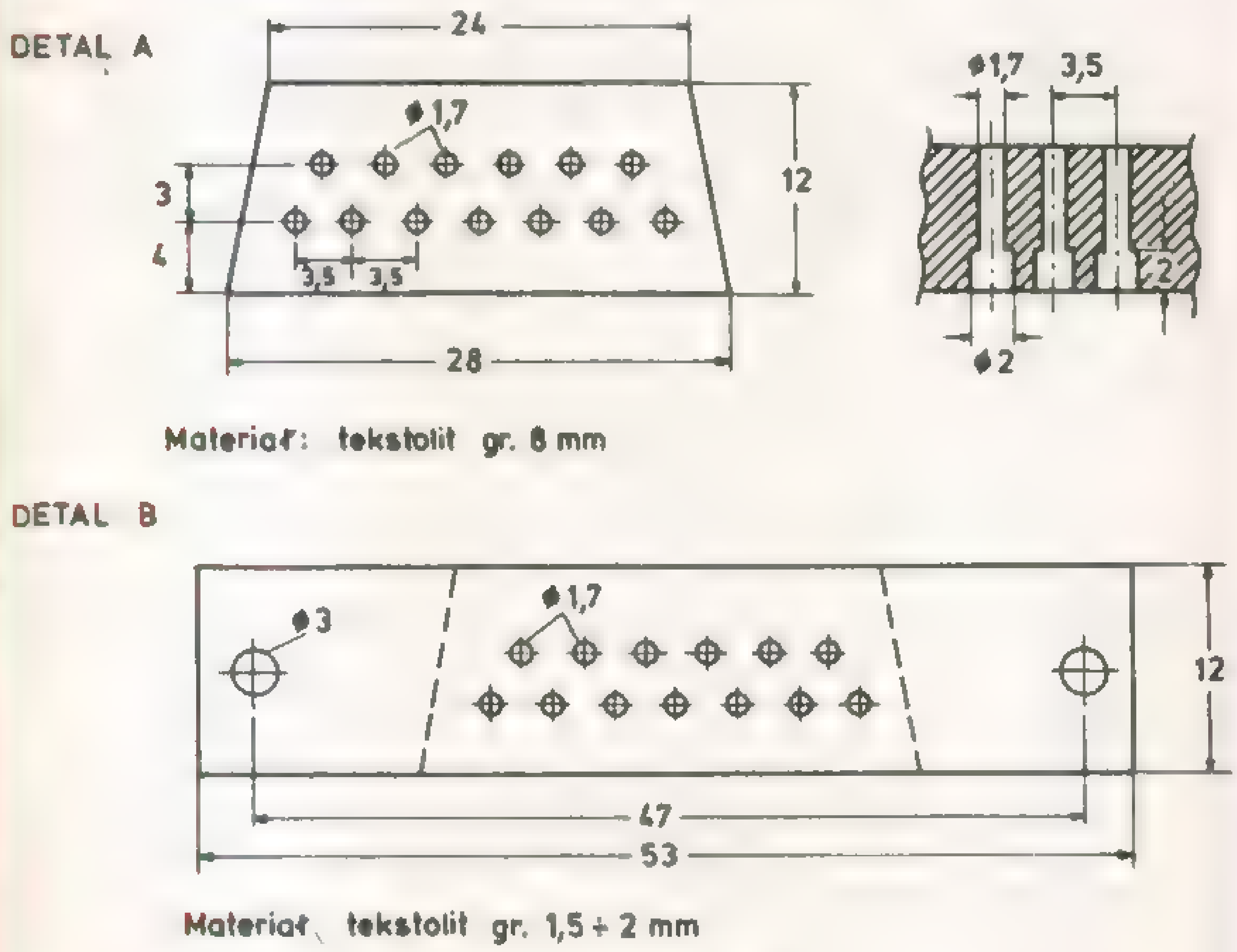
Rys. 2 Schemat ideowy przystawki



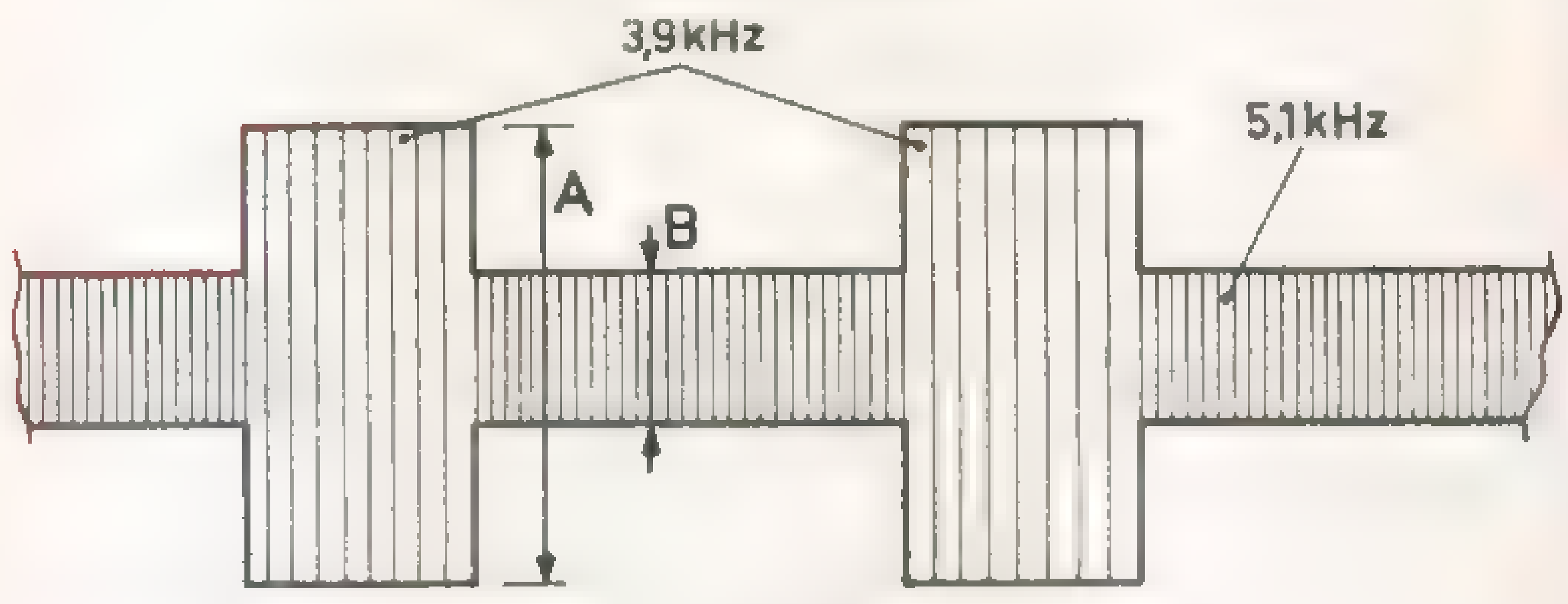
Rys. 3 Płytka montażowa — układ strony druku



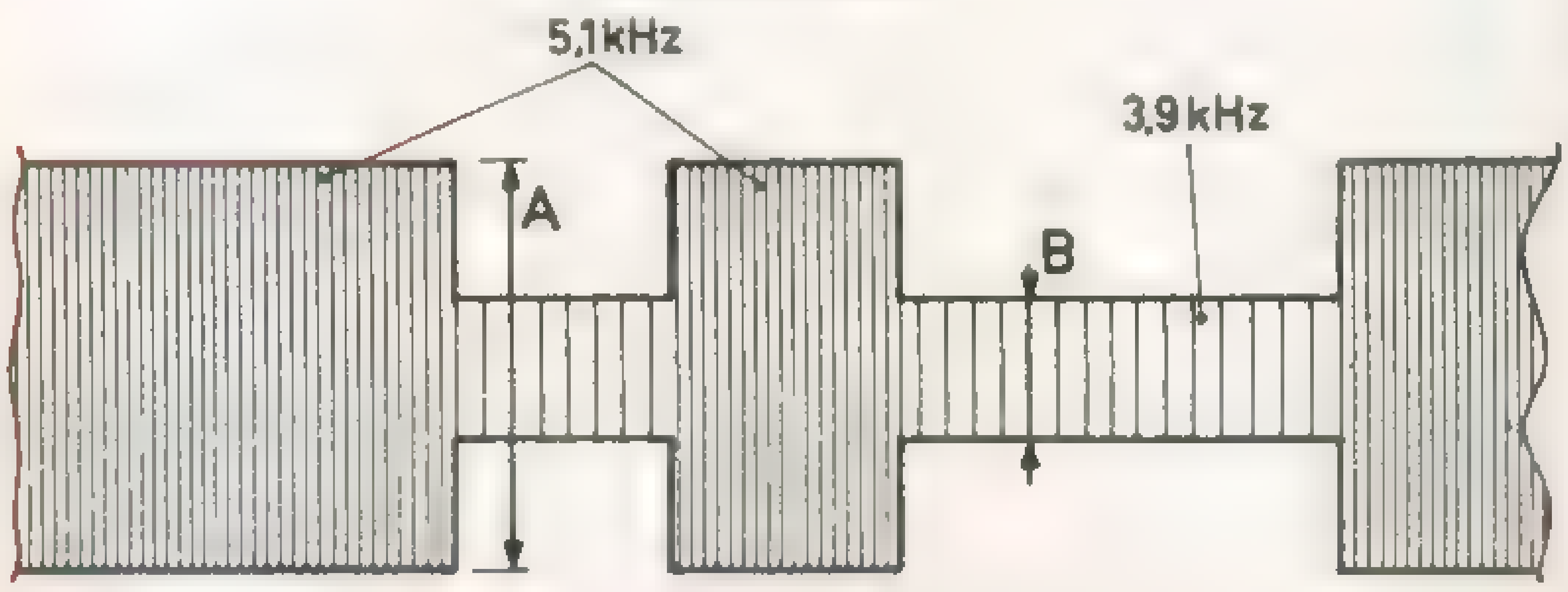
Rys. 4 Rozmieszczenie elementów



Rys. 5 Wtyk do komputera



Rys. 6 Oscylogram sygnału na końcówce nr 7 układu scalonego



Rys. 7 Oscylogram sygnału na końcówce nr 14 układu scalonego

rzanie i kolejno dołączamy oscyloskop do końcówek 7 i 14 układu scalonego. Kręcąc potencjometrem staramy się uzyskać na oscyloskopie przebiegi podobne do oscylogramów z rys. 6 i 7. W danym momencie stroimy jeden filtr. Podczas strojenia należy uzyskać jak największą amplitudę A przy jak najmniejszej jednocześnie amplitudzie B. W najgorszym wypadku **amplituda A powinna być przynajmniej dwa razy większa od amplitudy B.**

Od tej regulacji zależy poprawna praca przystawki. Po zestrojeniu filtrów mierzymy rezystancję potencjometrów i zamiast nich wlutowujemy rezystory. Możemy sprawdzić jeszcze przebieg na kolektorze tranzystora. W czasie trwania sygnału „pilota" rozpoczynającego program (sygnał o częstotliwości ? kHz) na kolektorze powinno być napięcie wyższe od + 3,5 V. Po pojawieniu się treści programu na kolektorze powinniśmy obserwować przebieg prostokątny, zmieniający się w takt zapisanej na taśmie informacji.

W czasie użytkowania musimy przestrzegać następujących zasad:

— w pierwszej kolejności łączymy przystawkę z komputerem i dopiero potem włączamy zasilanie komputera,

— w czasie pracy komputera nie wolno odłączać wtyczki przystawki z komputera,

— przystawkę możemy łączyć z magnetofonem w dowolnej chwili,

— obsługą magnetofonu musi zajmować się użytkownik tak samo jak mikrokomputera ZX Spectrum).

Opisana przystawka współpracuje bezawaryjnie od ponad roku z komputerem Atari 130 XE.

Spis części

[illegible]

UWAGA:

Wartości rezystorów R 7 i R 8 ustalamy w [illegible]

[illegible]

Procedury Rom SPECTRUM (1)

Krzysztof MAMCARZ

Pojawia się coraz więcej konkursów na oprogramowanie. Wzrastają też wymagania dotyczące opracowania graficznego programu, dźwięku, szybkości działania, walorów edukacyjnych. Użytkownicy Spectrum są w gorszej sytuacji, ponieważ BASIC korzysta z niewielu możliwości systemowych i stąd tak częsta potrzeba sięgania do języka wewnętrznego. Proponujemy przedstawienie najciekawszych procedur znajdujących się w pamięci stałej ROM myśląc, że przydadzą się w programach pisanych przez Was.

Procedury te mają dwa podstawowe atuty: po pierwsze już są w pamięci, a po drugie nie zaburzają relokowalności programów w języku wewnętrznym. Każda z nich zostanie przedstawiona w postaci tabelki zawierającej numer porządkowy, nazwę (wg Complete ROM Dissassembly), adres heksagonalny i dziesiętny, cel sprecyzowany w jednym zdaniu, parametry wejściowe oraz stan wyjściowy rejestrów i flag. Dokładniejsze opisy, jeśli będzie to konieczne, znajdą się pod tabelką. Pojawiające się nazwy zmiennych systemowych nie będą wyjaśniane. Zainteresowanych ich znaczeniem odsyłam do wcześniejszego cyklu „Sztuki i sztuczki". Brak uwag o stanie rejestrów oznacza, że procedura zmienia ich zawartość. Rejestr IY będzie pominięty w rozważaniach, gdyż jego zawartość powinna być stała i wynosić 23610. Jeśli procedura zmienia stan rejestrów pomocniczych, to będzie to specjalnie podkreślone. Przy powrocie do BASIC-a para HL, powinna zawsze zawierać adres 10072.

Zaczynamy od klawiatury

1	KEYBOARD	028Eh	0703
CEL	przeszukanie klawiatury, odczyt jej stanu oraz zdekodowanie przyciśniętego klawisza według tabel translacji.		
WE WY	nieistotne uaktualnienie zmiennych systemowych KSTATE, LAST i FLAGS (5 bit); IX przechowany		

Procedura 1 jest osią systemu. To jedyny interfejs programowy umożliwiający komunikowanie się człowieka z komputerem. Uruchamiana jest po każdym sygnale przerwania w procedurze 38h (RST 56).

2	KEY—SCAN	028Eh	0654
CEL	przeszukanie klawiatury		
WE WY	nieistotne D = 255 jeśli nie naciśnięto klawisza, 39 jeśli naciśnięto CAPS SHIFT, 24 jeśli naciśnięto SYMBOL SHIFT, E = 255 jeśli nie naciśnięto klawisza; w przeciwnym wypadku zawiera numer klawisza (0 — 39), Z — naciśnięto właściwą kombinację klawiszy, NZ — naciśnięto niewłaściwą kombinację klawiszy; IX przechowywany.		

Przez właściwą kombinację klawiszy procedura rozumie każdy klawisz pojedynczy z pominięciem CAPS i SYMBOL SHIFT oraz każde dwa klawisze, z których jeden to CAPS lub SYMBOL SHIFT. Szczególna sytuacja zachodzi, gdy naciśnięto oba klawisze specjalne (SHIFT). Wtedy stan CAPS znajdzie się w rejestrze D, a stan SYMBOL w rejestrze E.

Numery klawiszy są na tym poziomie rozumiane następująco:

36	28	20	12	4	3	11	19	27	35
37	29	21	13	5	2	10	18	26	34
38	30	22	14	6	1	9	17	25	33
39	31	23	15	7	0	8	16	24	32

Przykład przedstawia wykorzystanie procedury do testowania wybranego klawisza.

```
10 ; ** test '87
20 ; GENS3M21
30            ORG      60000
40  KEYSCN    EQU      028Eh
50            DI
60  PRACUJ    CALL     KEYSCN
70            LD       A,255
80            CP       D
90            JR       NZ,PRACUJ
100           LD       A,33
110           CP       E
120           JR       NZ,PRACUJ
130           EI
140           RET
```

Pierwsza część pętli (linie 70..90) sprawdza rejestr D, a druga rejestr E (linie 100..120). Wymagane wartości wprowadzamy do akumulatora. W tym wypadku w programie testowany jest klawisz ENTER.

2.	BREAK—KEY	1F54h	8020
CEL:	testowanie kombinacji CAPS SHIFT + SPACE (BREAK		
WE: WY:	Nieistotne NC — naciśnięto BREAK; CF — nie naciśnięto BREAK		

Testowanie BREAK może się odbywać przy wyłączonych przerwaniach. Program będzie wyglądał następująco:

```
10 ; ** break '85
20 ; GENA3M21
30          ORG      60000
40  BREAK   EQU      1F54h
50          DI
60  TEST    CALL     BREAK
70          JR       C,TEST
```

Dekodowanie przyciśniętego klawisza jest dostosowane do warunków pracy SPECTRUM. Algorytm musi uwzględniać klawisze specjalne, odgrywające dość istotną rolę. Nic nie stoi na przeszkodzie, aby algorytm ten zastąpić i dostosować do własnych potrzeb. Co prawda, nie można zastąpić procedury 38h, ale można przełączyć tryb przerwań (IM 2).

Oto przykład innego podejścia do zagadnienia przeszukiwania i dekodowania:

```
10 ; ** SCAN '87
20 ; GENS3M21
30          ORG       60000
40 U_UDG    EQU       65528
50          LD        HL,U_UDG
60          LD        BC,FEFEh
70 NEXT     IN        A,(C)
80          CPL
90          AND       1Fh
100         LD        (HL),A
110         INC       HL
120         RLC       B
130         JR        C,NEXT
140         RET
```

Odczyt stanu klawiatury następuje przez port o adresie FEh (254), który podaje się w rejestrze C. Drugi bajt, który reprezentuje jeden z ośmiu rzędów klawiatury, powinien być podany w rejestrze B. Wtedy odczytujemy port dzięki instrukcji IN A, (C). Każdy z rzędów odpowiada więc za 5 klawiszy. Akumulator przechwytuje 5 bitów, których stan jest reakcją na nasze zachowanie przy klawiaturze.

Ta metoda odczytu nie przewiduje traktowania wybranych klawiszy w jakiś specjalny sposób. Stany 40 klawiszy zostają po prostu odwzorowane w pamięci RAM w postaci mapy bitowej. Dalej można z nimi zrobić, co się komu podoba. I nie ma znaczenia, ile klawiszy przyciśnięto.

Dla uwidocznienia tego mapa bitowa została umieszczona w obszarze grafiki użytkownika, aby można było ją oglądać. Umożliwi to następujący program.

```
10 REM ** mapa '87
20 RANDOMIZE USR 60000
30 PRINT AT 1,1;CHR$ (164)
40 GOTO 20
```

W następnych odsłonach przedstawione będą procedury współpracujące z ekranem. A może znacie ciekawe rozwiązania w Spectrum, które uzupełniałyby przekazywane wiadomości? Jeśli tak, to napiszcie.

Krzysztof MAMCARZ

Tajemnice matematyki

Dzielniki liczby całkowitej

Problem znajdowania dzielników dowolnej liczby całkowitej intrygował wielu sławnych matematyków, a ich poszukiwania wzoru opisującego liczby pierwsze długo były daremne. Kiedy wielki matematyk **Fermat**, w roku 1640 podał swoją formułę

$$2^{2^n} + 1$$

łatwo pokazał, że początkowe liczby tego rodzaju, czyli 3, 5, 17, 257, 65537 są liczbami pierwszymi. Nie potrafił jednak podać ogólnego dowodu swojej hipotezy. Trzeba było stu lat, aby inny matematyk — **Euler** udowodnił, że $2^{32}+1$ nie jest liczbą pierwszą. Program nasz nie sprosta liczbie $2^{32} + 1 = 100895598169$. Podaje jednak, dla każdej liczby mającej maksymalnie 9 cyfr, dzielniki będące liczbami pierwszymi oraz sprawdza czy dzielniki te dzielą wybraną liczbę w kolejnych swoich potęgach.

Maksymalną potęgę, przy której dzielnik podniesiony do niej dzieli podaną liczbę, nazywamy krotnością tego dzielnika. Na podstawie krotności dzielników pierwszych program wylicza liczbę wszystkich dzielników wybranej liczby, w tym liczba 1 i podana liczba traktowane są jako dzielniki, zgodnie ze wzorem

$$(a_1 + 1) * (a_2 + 1) * ... * (a_n + 1)$$

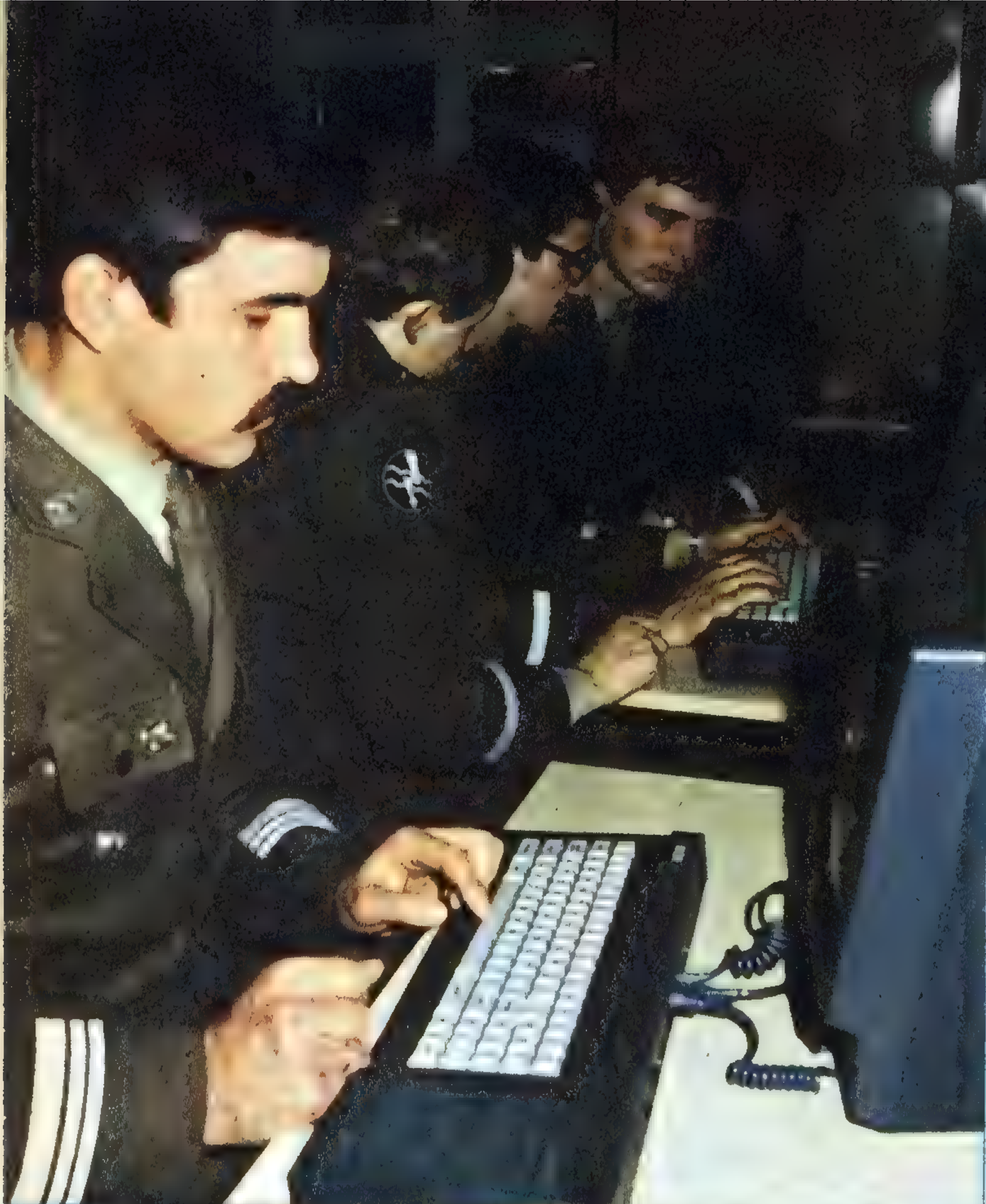
gdzie $a_1, a_2, ..., a_n$ są krotnościami dzielników pierwszych. Przykładowo $12 = 2^2*3^1$, ma zatem $(2+1)*(1+1) = 6$ dzielników, są to liczby: 1, 2, 3, 4, 6, 12.

A. T.

```
  10 REM Ile dzielnikow ma liczb
a ?
  15 CLS : BORDER 3
  19 PRINT AT 10,8;"Liczymy (t/n
)  ?": BEEP .1,3
  20 IF INKEY$="n" THEN STOP
  25 IF INKEY$<>"t" THEN BEEP .2
,4: GO TO 20
  28 LET a$="  "
  30 PRINT AT 10,6;"Podaj liczbe
 calkowita"
  36 INPUT "a =  ";a
  37 CLS
  40 LET a=ABS (a)
  45 PRINT "    Dzielnik 1"
  47 LET c=a: LET lo=1
  50 LET b=1+INT (a/2)
  55 FOR i=2 TO b
  60 LET licz=0
  65 LET r=a-i*INT (a/i)
  70 IF r<>0 THEN GO TO 85
  75 LET licz=licz+1
  76 LET a$="lz"
  80 LET a=a/i: GO TO 65
  85 IF licz=0 THEN GO TO 90
  86 PRINT "    Dzielnik ";i;"   k
rotnosc ";licz: BEEP .1,3
  87 LET lo=lo*(licz+1)
  88 IF a=1 THEN LET i=b
  89 IF i>a THEN LET i=b
  90 NEXT i
  96 PRINT "    Dzielnik ";c
 100 IF a$<>"lz" THEN GO TO 110
 105 PRINT AT 18,6;"liczba zlozo
na - ";c
 106 PRINT AT 18,6;"ilosc dzieln
ikow - ";lo: BEEP 2,3: PRINT AT
20,3; FLASH 1;"(nacisnij dowolny
 klawisz)": PAUSE 0: GO TO 15
 110 PRINT AT 18,6;"liczba pierw
sza - ";c
 115 PRINT AT 18,6;"liczba dziel
nikow - 2": BEEP 2,3: PRINT AT 2
0,3; FLASH 1;"(nacisnij dowolny
klawisz)": PAUSE 0: GO TO 15
```



— A gdzie nasza, programistka?
— Komputer dał jej zwolnienie.



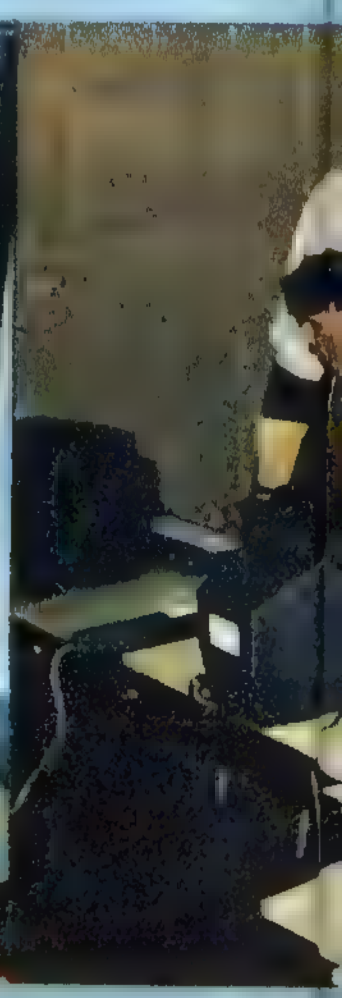


TELEB

Rozpoczęła już prac
Łączności w Zegrzu p
muszą nikogo namawia
cowni zasiedli pierwsi
mienił się w plac budow
doczne.

Nowocześnie wyposa
klub mikrokomputerow
teresuje informatyka m
Jest w tym zasługa k
rownika Ośrodka Oblic
Wszystkim informaty
„komputerowej" działalr









IT — ZEGRZE

pracownia komputerowa w Wyższej Szkole Oficerskiej Wojsk
d Warszawą. Pracownicy ośrodka obliczeniowego uczelni nie
ć do solidnej pracy. Zresztą sami dali przykład; zanim w pra-
łuchacze przeprowadzono tu gruntowny remont. Ośrodek za-
vlany, a informatycy bywało, że w budowlańców. Efekty są wi-

żona sala służyć będzie podchorążym; działać będzie również
„Telebit". Drzwi otwarte są dla każdego — wszyscy, których in-
ogą znaleźć tu swoje miejsce.
omendanta WSOWŁ płk. dypl. Zbigniewa Chruścińskiego i kie-
zeniowego mjr. mgr. inż. Tadeusza Szuszkiewicza.
kom z Zegrza i sympatykom tej dyscypliny, życzymy sukcesów w
ności.







PROGRAM 28

C 64

Tkanina

Krótki program graficzny, pozwalający na otrzymanie na ekranie różnorodnych wzorów w różnych kolorach. Może to być nie tylko interesująca zabawa, ale także pomoc dla tych, którzy zajmują się projektowaniem odzieży.

Tadeusz CISEK

```
 1 REM TKANINA
 5 PRINTCHR$(147)
 6 PRINT'TKANINA'
 60 PRINT'WYBOR WZORU:'
 70 PRINT'1) PASEK SZEROKI'
 75 PRINT'2) PASEK POJEDYNCZY'
 80 INPUT'3) PASEK WIELOKOLOROWY';C
 90 N=INT(RND(0)*9+1)
 100 FOR X=1 TO N
 110 A$(X)=CHR$(INT(RND(0)*62+161))
 120 NEXT X
 130 PRINTCHR$(147)
 140 FOR M=1TO6:IFC=1THEN GOSUB 250
 150 FOR X=1TON:IFC=2 THEN GOSUB 250
 160 FOR Y=1TO40:IF C=3 THEN GOSUB 250
 170 PRINTA$(X);
 180 NEXT Y,X,M
 210 GOTO 60
 250 A=INT(RND(0)*16+1)
 260 ON A GOSUB 300,310,320,330,340,350,3
60,370,380,390,400,410,420,430,440,450
■
 270 RETURN
 300 PRINTCHR$(5);:RETURN
 310 PRINTCHR$(28);:RETURN
 320 PRINTCHR$(30);:RETURN
 330 PRINTCHR$(31);:RETURN
 340 PRINTCHR$(129);:RETURN
 350 PRINTCHR$(144);:RETURN
 360 PRINTCHR$(149);:RETURN
 370 PRINTCHR$(150);:RETURN
 380 PRINTCHR$(151);:RETURN
 390 PRINTCHR$(152);:RETURN
 400 PRINTCHR$(153);:RETURN
 410 PRINTCHR$(154);:RETURN
 420 PRINTCHR$(155);:RETURN
 430 PRINTCHR$(156);:RETURN
 440 PRINTCHR$(158);:RETURN
 450 PRINTCHR$(158);:RETURN
READY.
```

PROGRAM 29

ZX 81

READ, DATA i RESTORE

W kilku listach powtarzały się prośby o programy na mikrokomputer ZX-81, który choć zapomniany, ma widać jeszcze zwolenników. Ponieważ możliwości ZX-81 są bardzo ograniczone, może przyda się sposób symulacji komend READ, DATA i RESTORE, które implementacja BASIC'a pomija.

```
 10 REM 5,-1001,0,-3,
 20 LET D=0
 30 LET C=0
 40 LET E=16427
 50 GOSUB 200
 60 IF D=1 THEN LET C=-C
 70 LET D=0
 80 LET E=A+E+1
 90 PRINT C
100 LET C=0
110 INPUT A$
120 GO TO 50
200 FOR A=0 TO 50
210 LET B=PEEK (E+A)-28
220 IF B=188 THEN RETURN
230 IF B=90 THEN STOP
240 IF NOT B=192 THEN GO TO 270
250 LET D=1
260 GO TO 280
270 LET C=C*10+B
280 NEXT A
290 STOP
```

Dane umieszczone są tu w postaci listy w linii 10 po instrukcji REM. Po każdej liczbie powinien nastąpić przecinek (również po ostatniej). Podprogram 200 odczytuje kolejne cyfry, a wartość liczby wkłada do zmiennej C. W ten sposób liczbę można przechwycić do własnych celów.

Symulacja komendy RESTORE polega na przypisaniu zmiennej E adresu początkowego danych, czyli adresu pierwszego znaku po REM. Dane można umieścić w dowolnej linii pod warunkiem, że znany będzie ów adres. Poniższy program wyszukuje linie z komendą REM i podaje potrzebne adresy

```
1000 LET A=PEEK (16392)+256*PEEK (16393)
1010 FOR B=16424 TO A
1020 IF PEEK (B)=254 THEN PRINT B+1
1030 NEXT B
```

Jeśli REM znajduje się tylko w pierwszej linii, to wystarczy przypisać zmiennej E wartość 16427 (jak w linii 40) aby osiągnąć efekt czytania danych od początku

K. M.

PROGRAM 30

ZX SPECTRUM

Kostka

To prosty program dla 1 lub dwóch graczy. Jeśli będzie tylko jeden, jako drugi zawodnik wystąpi komputer. Obsługa programu ogranicza się do wyboru liczby graczy, podania imion lub imienia, podania maksymalnego wyniku, do jakiego ma toczyć się gra. O tym, kto pierwszy zaczyna, decyduje przypadek i sygnalizowane to jest przez migający napis. Uzyskana ilość oczek dodawana jest do punktacji danego zawodnika. Jeśli ktoś wyrzucił „6" ma prawo powtórzyć rzut. Start „losowania" — dowolny klawisz. Komu uda się uzyskać maksymalny wynik, otrzymuje 1 duży punkt i cała zabawa zaczyna się od nowa. Przy każdej następnej partii maksymalny wynik może (lecz nie musi) być zmieniony. Na koniec jedna uwaga, przy wpisywaniu programu należy literę „A" w liniach 1520 — 1570 wpisać w trybie „graficznym".

Z. WIŚNIEWSKI

```
   1 REM
   4 REM [illegible]
   5 FOR x=0 TO 7
   6 READ n: POKE USR "a"+x,n
   7 NEXT x
   8 DATA 0,126,126,126,126,126,
126,0
  10 REM [illegible]
  12 DIM p$(32): DIM m(1): DIM n
(1)
  13 PRINT AT 0,0;"Zawodnik 1";A
T 0,21;"Zawodnik 2"
  15 INPUT "ilu graczy (1 lub 2)
";i
  20 IF i<1 OR i>2 THEN GO TO 15
  25 IF i=1 THEN INPUT "podaj im
ie ";a$: LET b$="Komputer"
  30 IF i=2 THEN INPUT "podaj im
ie "1" ";a$: INPUT "podaj imie "
2" ";b$
 131 LET l=0: LET p=0
 132 INPUT "max. punktacja = ";m
 133 PRINT BRIGHT 1;AT 2,1;a$;AT
 2,22;b$: PRINT PAPER 6;AT 4,0;"
punkty";AT 4,21;"punkty": PRINT
AT 4,7;l;AT 4,28;p
 140 FOR x=108 TO 139: PLOT x,69
: PLOT x,98: NEXT x
 150 FOR x=69 TO 98: PLOT 108,x:
 PLOT 139,x: NEXT x
 200 REM [illegible]
 205 LET g=(INT (RND*2)+1)*500
 210 GO TO g
 500 REM [illegible]
 505 PRINT BRIGHT 1; FLASH 1;AT
0,0;"Zawodnik 1": PRINT AT 0,21;
"Zawodnik 2"
 510 PRINT #0; PAPER 6;AT 1,0;"S
TART - DOWOLNY KLAWISZ": PAUSE 0
: BEEP 0.1,30: PRINT #0;AT 1,0;"
                                "
 540 GO SUB 1500
 550 LET l=l+a
 560 PRINT AT 4,7;l
 570 IF a=6 THEN GO TO 500
 575 IF l>=m THEN GO TO 2500
1000 REM [illegible]
1005 PRINT AT 0,0;"Zawodnik 1";
BRIGHT 1; FLASH 1;AT 0,21;"Zawod
nik 2"
1009 IF i=1 THEN LET d$="TERAZ J
A - CZEKAJ      ": PRINT #0; PAP
ER 6;AT 1,0;d$: BEEP 1,-25: PAUS
E 100: PRINT #0;AT 1,0;"
                                "
1010 IF i=2 THEN LET d$="START -
 DOWOLNY KLAWISZ": PRINT #0; PAP
ER 6;AT 1,0;d$: PAUSE 0: BEEP 0.
1,25: PRINT #0;AT 1,0;"
                                "
1015 GO SUB 1500
1020 LET p=p+a
1030 PRINT AT 4,28;p
1040 IF a=6 THEN GO TO 1000
1050 IF p>=m THEN GO TO 2500
1060 PRINT AT 0,21;"Zawodnik 2"
1070 GO TO 500
1500 REM [illegible]
1505 FOR x=1 TO 30
1510 LET a=INT (RND*6)+1
1520 IF a=1 THEN PRINT AT 10,14;
"   ";AT 11,14;" ■ ";AT 12,14;"
   "
1530 IF a=2 THEN PRINT AT 10,14;
"■  ";AT 11,14;"   ";AT 12,14;"
  ■"
1540 IF a=3 THEN PRINT AT 10,14;
"■  ";AT 11,14;" ■ ";AT 12,14;"
  ■"
1550 IF a=4 THEN PRINT AT 10,14;
"■ ■";AT 11,14;"   ";AT 12,14;"■
 ■"
1560 IF a=5 THEN PRINT AT 10,14;
"■ ■";AT 11,14;" ■ ";AT 12,14;"■
 ■"
1570 IF a=6 THEN PRINT AT 10,14;
"■ ■";AT 11,14;"■ ■";AT 12,14;"■
 ■"
1580 BEEP 0.008,x
1590 NEXT x
1600 RETURN
2500 REM [illegible]
2505 PAUSE 100: CLS
2510 PRINT PAPER 6;AT 0,8;"KLASY
FIKACJA"
2520 IF l>p THEN LET m(1)=m(1)+1
: PRINT AT 6,3;"1";"  ";a$;AT 8,
3;"2";"  ";b$; PAPER 6;AT 6,16;m
(1);AT 8,16;n(1)
2530 IF l<p THEN LET n(1)=n(1)+1
: PRINT AT 6,3;"1";"  ";a$;AT 8,
3;"2";"  ";b$; PAPER 6;AT 6,16;m
(1);AT 8,16;n(1)
2540 REM [illegible]
2545 LET t$=p$+"JESZCZE RAZ ? (T
-TAK , N- NIE) "+p$
2550 GO SUB 2700
2560 GO TO 2540
2700 FOR k=1 TO LEN t$-31
2710 PRINT #0; PAPER 6; INK 1;AT
 1,0;t$(k TO k+31): BEEP 0.005,I
NT (RND*30): PAUSE 3
2720 IF INKEY$="t" THEN CLS : GO
 TO 131
2730 IF INKEY$="n" THEN STOP
2740 NEXT k: RETURN
```

Opowiadaniem tym rozpoczynamy prezentację utworów, nagrodzonych i wyróżnionych w naszym konkursie „Przed nami XXI wiek". „Finał", jak Czytelnicy zapewne pamiętają, uzyskał równorzędną, trzecią nagrodę.

Korzystając ze sposobności, uprzejmie prosimy Autorów nagrodzonych i wyróżnionych prac o wyrozumiałość — i zgodę na redakcyjne opracowanie oraz skróty, dokonywane w Ich tekstach. Przyczyna — prosta: brak miejsca. W „IKS-ie" możemy publikować teksty o maksymalnej objętości do 6 stron znormalizowanego maszynopisu. Ze swej strony — dołożymy wszelkich starań, aby nic nie uronić z artystycznej wartości Waszych prac, Szanowni Autorzy.

FINAŁ

Sorg stał zadumany na płaskim brzegu metanowego morza, planety Yarr, jak ją sobie ochrzcił. Stąd mógł najlepiej obserwować zachód ogromnego czerwonego olbrzyma, gwiazdy tej planety. Był to widok pełen majestatu i jakiejś grozy. Być może nastroje te podtrzymywała świadomość, że w tym kosmosie jest to ostatnia istniejąca gwiazda, ostatnia planeta. A sam Sorg?

Ostatnia iskierka świadomej materii. Wielkie wyzwanie rzucone śmierci przez dawno wymarłą cywilizację.

Dawno wymarłą... jak dawno? Dla Sorga nie było to specjalnie istotne. Już teraz nie. W każdym razie były to miliardy lat. Lat ziemskich, liczonych czasem obiegu Ziemi, kolebki cywilizacji, wokół Słońca, niepozornej gwiazdy tej planety. Trzy, cztery... sześć miliardów lat? Rzecz bez znaczenia.

Od chwili śmierci ostatnich ludzi czas Sorga liczył się miarą rozszerzającego się stale kosmosu, rozrzedzającej się materii. Zwiększały się odległości między galaktykami, między gwiazdami, pękła iluzoryczna homeostaza systemów planetarnych. Energia rozpraszała się w narastającym tempie. Dokonały żywota galaktyki i gwiazdy, a ich materia rozproszyła się w przestrzeni, stale rosnącej przestrzeni...

Teraz był już tylko Sorg i to ostatnie wielkie czerwone słońce, które także niebawem dokona żywota, wybuchając supernową, rozproszy większość swej materii, jeżeli niecałą i w ogniu tego ostatniego spektaklu przyrody spali się też Yarr...

Sorg przeklinałby swych twórców, gdyby mógł czuć cokolwiek. Jego świadomość była czysta, bez biochemicznych wyznaczników osobowości. Nie miał temperamentu, nie odczuwał gniewu, nienawiści czy miłości... Lecz przecież ciągle towarzyszyła mu jakaś nieuchwytna forma cierpienia. Wiedział, że to niemożliwe, a jednak... chwytał się też często na tym, że z pewną, hmm, tęsknotą (o ile to możliwe) myślał o śmierci, zgaśnięciu... absolutnym wyłączeniu się. To jednak było niemożliwe. Na razie. Nie spełnił jeszcze swego posłannictwa, którego sam się przecież zupełnie dobrowolnie podjął... Czy je spełni? I na co się to zda?

Słońce zaszło. Niebo na zachodzie grało jeszcze blaskami czerwieni. Sorg ruszył w zamyśleniu wzdłuż brzegu. Jego osiem pajęczych nóg pracowało cierpliwie bez udziału woli. Czuł wielkie znużenie, choć i to było przecież niemożliwe. Był najdoskonalszym chyba wytworem swej cywilizacji; wszystkie ogromne możliwości w okresie jej największego rozkwitu sprzęgły się, by mógł powstać. Od tego czasu istnieje: zwyczajny człowiek, jakim był Sorg Altenberg — stał się Tym, Który Przetrwa. Był praktycznie wieczny, niezależny od czasu, przestrzeni, energii.

Pamiętał erę wielkości cywilizacji, jedynej (jak się okazało) rozumnej formy życia w kosmosie. Wspaniałe osiągnięcia myśli, cuda techniki; on sam zresztą był takim największym cudem. Był to czas rozkwitu, ekspansji, adaptacji coraz to nowych światów, coraz to nowych galaktyk do potrzeb Człowieka.

Potem nieuchronnie nadszedł czas regresji starości i śmierci gatunku homo. Prawa biologii okazały się nieubłagane. „Cokolwiek rozkwita, niezawodnie sczeźnie..." — pamiętał słowa jakiegoś barbarzyńskiego poety. Potem mógł już tylko oglądać, jak rozsypuje się w pył to, co cywilizacja stworzyła.

Był jeszcze on sam. Rozpaczliwy protest Rozumu przeciwko przemijaniu. Tylko po co, jaki może być sens w tym przedstawieniu, które zainscenizowała Przyroda? Krzywizna przestrzeni wszechświata okazała się ujemna. Oznaczało to ciągłe, nieskończone rozszerzanie się, rozpraszanie się energii. Nie było sensu czekać tu dłużej. Sorg zdecydował się na opuszczenie Yarr. Czas był już najwyższy, niebawem nawet dla niego stałoby się tu nieprzyjemnie. Skoczył na bezpieczną odległość kilkunastu lat świetlnych w pustą przestrzeń i stąd mógł obserwować ostatni akt tej sztuki.

Czerwony olbrzym gwałtownie skurczył się, zmieniając w oślepiająco jasną kulę, która następnie gwałtownie zaczęła się rozszerzać. Sorg widział nieskończenie wiele takich spektakli, zawsze jednak chętnie je oglądał. A ten, ostatni, miał swój specyficzny nastrój. Rozpraszały się resztki skoncentrowanej energii, resztki światła...

Teraz i zawsze już tylko pustka. Pustka i on, Sorg, już teraz w ostateczny sposób absolutnie sam. Na zawsze... Wtedy pojął, zrozumiał. Jak westchnienie ulgi spłynęło na niego zrozumienie. W tej chwili wypełnił swoje posłannictwo. Zrozumiał, że to, co usiłowali pojąć najwięksi mędrcy, z czym biedziły się niezliczone pokolenia uczonych, filozofów, mistyków wreszcie, poszukując sensu, najogólniejszych prawideł, budując modele, systemy praw, teorii mniej lub bardziej spójnych... Zrozumiał, że tu sensu po prostu nie ma i nie było. Wszystko było bezsensowne!

Sorg bez wahania uruchomił mechanizm samozniszczenia. Dookólną ciemność rozdarł na krótką chwilę jeden jasny błysk.

Andrzej DYRSZKA

BASIC (9)

W tym wykładzie podamy podstawowe wiadomości o tablicach zmiennych indeksowanych (numerycznych) oraz o stałych w programach zapisywanych w instrukcji biernej DATA (wraz z instr. READ i RESTORE).

Wykład ósmy: tablice, zmienne indeksowane (numeryczne) instrukcje: DIM, DATA, READ, RESTORE

W poprzednich wykładach wszystkie programy napisane były przy zastosowaniu zmiennych prostych (numerycznych lub alfanumerycznych). Zmienne proste nie pozwalają na zapamiętanie w programie dużej liczby wartości, a następnie wykonanie tych samych sekwencji instrukcji dla różnych zmiennych. Wprowadza się więc grupowanie ciągu zmiennych w tablice, które są identyfikowalne przez swoje nazwy. Dostęp do poszczególnych elementów tablicy następuje przez podanie nazwy tablicy i indeksów określających żądany element

Tak więc poindeksowany zbiór danych tego samego typu nazywa się **tablicą.** Wartościami indeksów mogą być pojedyncze liczby całkowite (tablica nazywa się wtedy jednowymiarową), pary liczb całkowitych (tablica dwuwymiarowa) itd. Tablica jako całość jest identyfikowana przez swą nazwę

Element tablicy, odpowiadający konkretnemu układowi wartości identyfikujących go indeksów, nazywa się **zmienną indeksowaną.** Zmienną taką oznacza się symbolem złożonym z napisanych po sobie nazwy tablicy i ujętych w nawiasy (bądź rozdzielonych przecinkami) indeksami

W języku BASIC należy określić nazwę i rozmiar tablic, jakie będą używane w danym programie Do tego celu służy deklaracja tablicy

Deklaracją tablicy w BASIC-u ZX Spectrum jest instrukcja bierna postaci

n **DIM** nazwa tablicy (rozmiar)

gdzie

n — numer wiersza

nazwa tablicy — składa się z
— jednej litery dla tablic numerycznych
— jednej litery i znaku $ dla tablic alfanumerycznych

rozmiar — zależy od liczby wymiarów (i elementów) tablicy

Tablice mogą być jednowymiarowe (nazywane wektorami) dwuwymiarowe (nazywane macierzami) lub trzywymiarowe itd

Instrukcja DIM może występować w dowolnym miejscu programu (ale przed odwołaniem się do tej tablicy). Jeżeli w programie zadeklarowano kilka tablic o tej samej nazwie, to w chwili uruchomienia aktualna jest deklaracja w linii o najwyższym numerze W tym samym programie może wystąpić zmienna prosta numeryczna o tej samej nazwie co tablica. W trakcie rezerwowania miejsca dla danej tablicy numerycznej następuje nadawanie jej elementom wartości początkowych zero

W wykładzie tym omawiać będziemy tylko tablice numeryczne, a tym samym zmienne indeksowe numeryczne. A teraz zilustruje my przykładami to, co zostało już powiedziane

Przykład 7.1

Dany jest wektor x o 10-elementach. Wyznaczyć sumę elementów o wartościach większych od zadanej stałej c

a) program

```
  10 REM Przyklad 7.1
  15 REM wektor definiujemy jako
         tablice jednowymiarowa,
         10 elementowa
  20 DIM x(10)
  25 INPUT "stala c= ";c
  30 REM wczytanie wartosci elem
en       tow tablicy x
  40 PRINT TAB (10);"TABLICA X"
  45 PRINT
  50 FOR i=1 TO 10
  60 INPUT "podaj wartosc x(i)=
";x(i)
  70 PRINT x(i);"   ";
  80 NEXT i
  85 LET s=0
  90 FOR i=1 TO 10
 100 IF x(i)>c THEN LET s=s+x(i)
 110 NEXT i
 115 PRINT
 120 PRINT '"Suma elementow o wa
rtosciach     wiekszych od zadane
j stalej      c = ";c;"  wynosi :
";s
 130 STOP
```

[illegible] przykładowych danych

```
          TABLICA X

1  2  3  4  5  6  7  8  9  10

Suma elementow o wartosciach
wiekszych od zadanej stalej
c = 5  wynosi :40
```

Instrukcja

20 DIM x (10)

deklaruje tablicą jednowymiarową o nazwie x i o 10 — elementach (liczba 10 deklaruje rozmiar, w tym wypadku liczbą elementów tablicy)

Tablicę jednowymiarową (wektor) można przedstawić w postaci wiersza (lub kolumny), tak więc dla naszej tablicy x w postaci



liczby 1 ÷ 10 kolejne liczby naturalne będące indeksami elementów (i — indeks; i = 1, 2,...10)

x (i) — zmienna indeksowana numeryczna — i-ty element tablicy

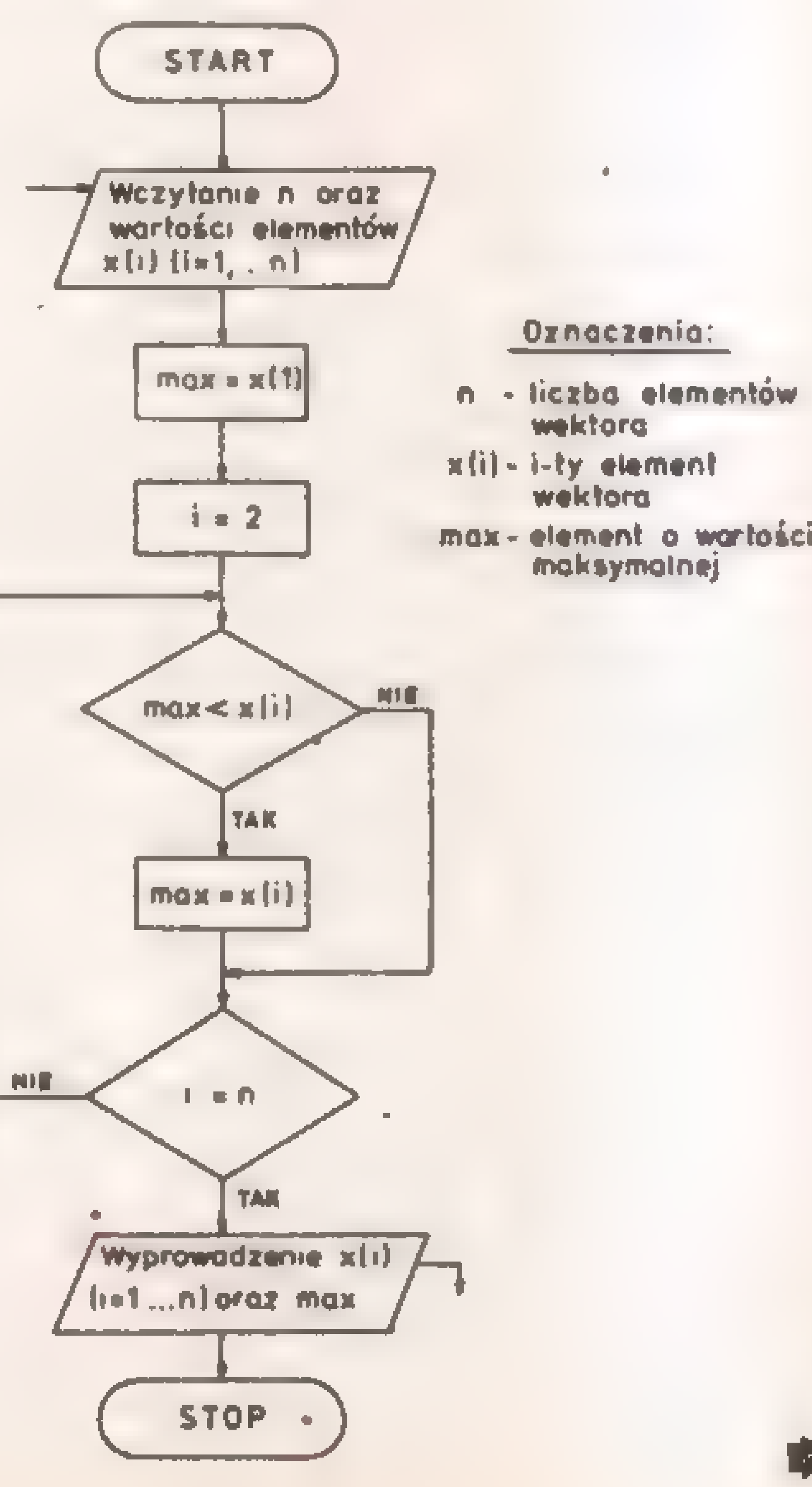
Instrukcja

60 INPUT „podaj wartość x (i) =-„; x(i) nadaje wartość i-temu elementowi tablicy

Przykład 7.2

Napisać program znajdujący wartość maksymalną w wektorze n-elementowym

a) schemat blokowy



Oznaczenia:
n — liczba elementów wektora
x (i) — i-ty element wektora
max — element o wartości maksymalnej

b) program

```
 10 REM Przyklad 7.2
 15 REM n - liczba elementow we
            ktora
       i - indeks
    x(i) - i-ty element wektora
            (zmienna indeksowana)
     max - wybrany element o war
            tosci maksymalnej
 20 INPUT "wprowadz n :";n
 25 REM liczba elementow tej ta
          blicy zostaje ustalona
          po wczytaniu wartosci n
 30 DIM x(n)
 35 REM wczytanie wartosci ele-
          mentow tablicy x
 40 FOR i=1 TO n
 45 INPUT "wprowadz x(i):";x(i)
 50 NEXT i
 60 LET max=x(1)
 70 FOR i=2 TO n
 80 IF max<x(i) THEN LET max=x(
i)
 90 NEXT i
100 PRINT TAB (5);"wektor dany
:"
105 PRINT
110 FOR i=1 TO n
120 PRINT x(i);"   ";
130 NEXT i
135 PRINT
140 PRINT '"wartosc maksymalna
: ";max
150 STOP
```

... przykładowych danych

```
     wektor dany :

8  5  4  6  9  11  3  22

wartość maksymalna : 22
```

c) uwaga

Jezeli chcielibyśmy znaleźć wartość minimalną w wektorze n--elementowym, wystarczy instrukcję o numerze 80 zapisać w postaci

80 IF max > x (i) THEN LET max = x (i)

oraz zamiast słowa maksymalnej (wartości) wpisać słowa minimalnej wartości

Instrukcja

30 DIM x (n)

deklaruje tablicę jednowymiarową o nazwie x i o n-elementach. Zarezerwowanie odpowiedniego miejsca dla tej tablicy w pamięci komputera następuje po nadaniu wartości zmiennej n. Nadawanie tej wartości może nastąpić za pomocą instrukcji INPUT lub LET i musi nastąpić przed deklaracją tablicy.

Tablice jedno-, dwu- lub więcej wymiarowe zadeklarowane w powyższy sposób nazywa się często tablicami dynamicznymi. Odpowiednie instrukcje

INPUT „wprowadź x (i)" "; x (i)

powodują nadanie aktualnej wartości elementom tych tablic. Wartości te „pozostają" tak długo aż

a) nie nadamy im nowych wartości instrukcją INPUT lub LET
b) nie usuniemy programu z pamięci komputera, wykonując komendę NEW lub LOAD, lub wyłączymy komputer

Korzystając z „pozostawania" tych wartości w tablicy, możemy odwoływać się do nich w różnych miejscach programu i tak np. odwołujemy się dodatkowo

W instrukcji o nr 100 w przykładzie 7.1 oraz w instrukcjach o nr 80 i 120 w przykładzie 7.2

Przykład 7.3

W ciągu pewnego czasu dla grupy 15 uczniów przeprowadzono 5 sprawdzianów z wybranego przedmiotu. Obliczyć średnią ocenę otrzymaną przez każdego ucznia

a) analiza zadania

Oceny każdego ucznia możemy zapisać w sektorze jednowymiarowym pięcioelementowym, ale tych wektorów mielibyśmy 15. Zapiszemy oceny w postaci tablicy

```
 10 REM Przyklad 7.3
 20 DIM o(15,5)
 25 REM wartosci elementow ta-
          blicy wprowadzamy wier-
          szami
 30 FOR i=1 TO 15
 35 FOR j=1 TO 5
 40 INPUT "wprowadz o(i,j): ";o
(i,j)
 45 NEXT j
 50 NEXT i
 60 FOR i=1 TO 15
 65 REM dla kazdego i oblicamy
          nowa sume - sr=0
 70 LET sr=0
 75 PRINT "uczen ";i'"oceny :"
 80 FOR j=1 TO 5
 90 LET sr=sr+o(i,j)
 95 PRINT o(i,j);"   ";
100 NEXT j
105 LET sr=sr/5
110 PRINT '"ocena srednia : ";s
r
115 PRINT
120 NEXT i
130 STOP
```

gdzie i-ty wiersz — i-ty uczeń
j-ta kolumna — j-ta ocena
o_{ij} — oznacza j-tą ocenę i-tego ucznia (i = 1,2,...,15, j = 1,2,3,4,5)

Jest to tablica dwuwymiarowa (macierz), którą deklarujemy w następujący sposób

20 DIM o (15.5)

Każdy element tej tablicy jest opisany dwójką liczb naturalnych będących jego indeksami i tak.

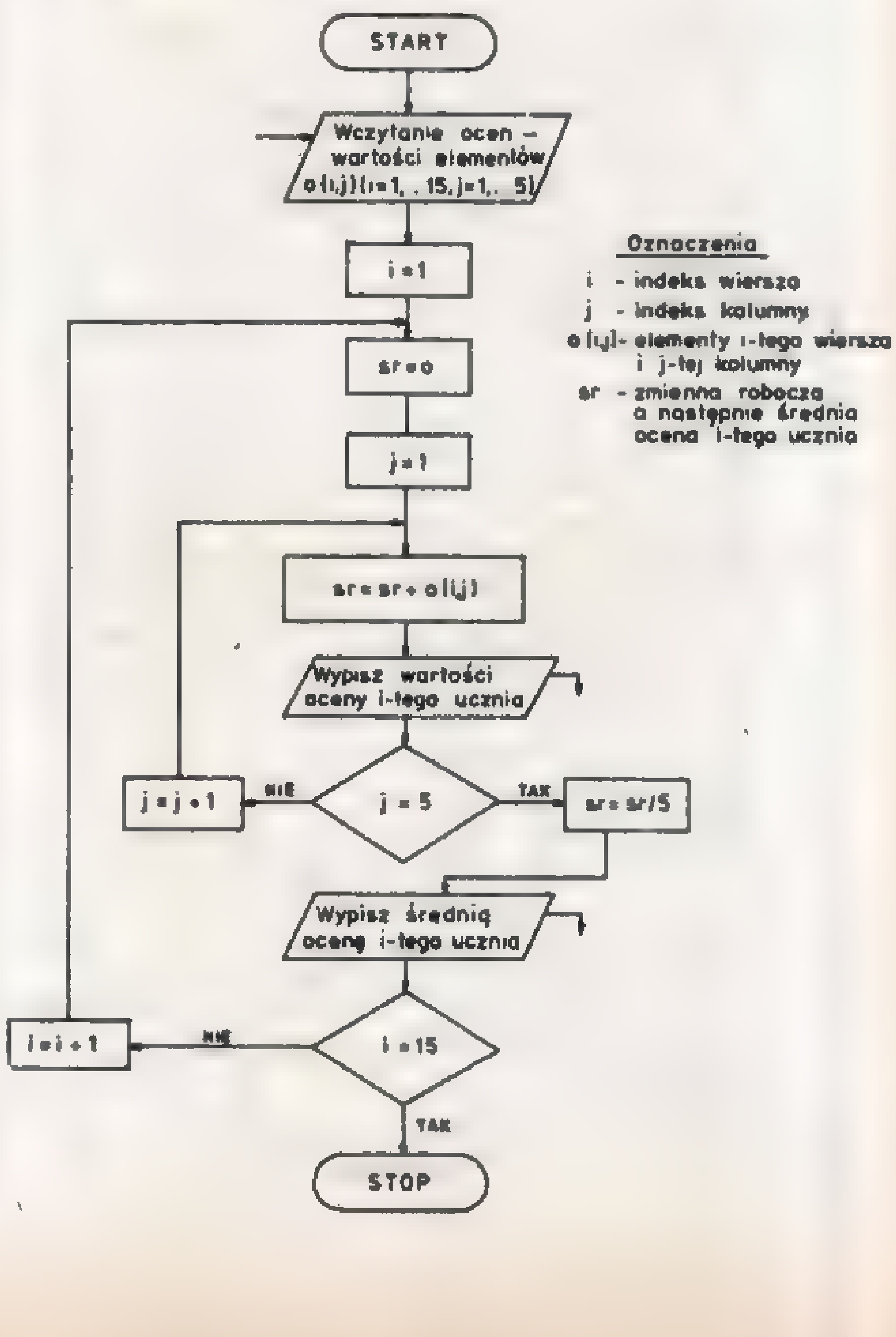
o(1,2) — jest to element pierwszego wiersza i drugiej kolumny
o(5,3) — jest to element piątego wiersza i trzeciej kolumny
o(i,j) — jest to element i-tego wiersza i-j-tej kolumny

Tak więc pierwszy indeks oznacza numer wiersza, a indeks drugi — numer kolumny

Oceny średnie obliczamy wg wzoru

$$sr = \frac{\sum_{j=1}^{5} o_{ij}}{5} \quad \text{dla każdego i (dla każdego ucznia) i = 1 2 ...15}$$

b) schemat blokowy



Oznaczenia:

i — indeks wiersza
j — indeks kolumny
o(i,j) — element i-tego wiersza i j-tej kolumny
sr — zmienna robocza, a następnie średnia ocena i-tego ucznia

Przykład 7.4

Znaleźć sumę oraz liczbę elementów dodatnich leżących powyżej głównej przekątnej macierzy Z (z_{ij} i,j = 1,2, ... n)

```
 10 REM Przyklad 7.4
 15 REM n - liczba wierszy(ko-
            lumn macierzy z
         i - indeks wiersza
         j - indeks kolumny
    z(i,j) - element macierzy z
         s - suma elementow do-
            datnich dla ktorych
            i>j
         l - liczba elementow
            dodatnich,dla kto-
            rych i>j
 20 INPUT "wprowadz n : ";n
 25 REM liczba elementow tej
        tablicy zostaje usta-
        lona po wczytaniu n
 30 DIM z(n,n)
 35 REM wartosci elementow ta-
        blicy wprowadzamy wier-
        szami
 40 PRINT "macierz dana : ": PR
INT
 45 FOR i=1 TO n
 50 FOR j=1 TO n
 55 INPUT "wprowadz z(i,j) :";z
(i,j)
 60 PRINT z(i,j);"  ";
 65 NEXT j: PRINT
 70 NEXT i
 75 LET s=0: LET l=0
 80 FOR i=1 TO n
 85 FOR j=i+1 TO n
 90 IF z(i,j)>0 THEN LET s=s+z(
i,j): LET l=l+1
 95 NEXT j
 100 NEXT i
 105 PRINT '"liczba elementow do
datnich      powyzej glownej prz
ekatnej :"'"  l = ";l;", ich sum
a = ";s
 110 STOP
```

Zapiszemy macierz Z w postaci

$$Z = \begin{bmatrix} z_{11} & z_{12} & \dots & z_{1n} \\ z_{21} & z_{22} & \dots & z_{2n} \\ \vdots & & & \vdots \\ z_{n1} & z_{n2} & \dots & z_{nn} \end{bmatrix}$$

Jeżeli liczba wierszy danej macierzy równa się liczbie jej kolumn, to macierz nazywa się kwadratową, jeżeli nie — prostokątną. Położenie elementu z(i,j) w macierzy Z można określić w następujący sposób

element z (i,j) (i,j = 1, 2,. . .n) dla
i > j znajduje się powyżej głównej przekątnej
i = j znajduje się na głównej przekątnej
i < j znajduje się poniżej głównej przekątnej

Badanie znaku i położenia elementów z(i,j) będziemy przeprowadzać w kolejnych wierszach macierzy. Macierz Z zostanie zadeklarowana jako dynamiczna tablica dwuwymiarowa (nxn)-elementowa.

Narysowanie schematu blokowego pozostawiamy czytelnikowi.

Instrukcja

20 DIM o(15,5)

deklaruje tablicę dwuwymiarową o nazwie o i liczbie elementów równej 75 (15x5),

Wiersze \ Kolumny	1	2	3	4	5
1	$o_{1,1}$	$o_{1,2}$	$o_{1,3}$	$o_{1,4}$	$o_{1,5}$
2	$o_{2,1}$	$o_{2,2}$	$o_{2,3}$	$o_{2,4}$	$o_{2,5}$
⋮	⋮				
15	$o_{15,1}$	$o_{15,2}$	$o_{15,3}$	$o_{15,4}$	$o_{15,5}$

natomiast instrukcja

30 DIM z(n,n)

deklaruje dynamiczną tablicę dwuwymiarową o nazwie z i liczbie elementów równej nxn, liczba tych elementów jest znana w momencie nadania wartości zmiennej n.

Każda tablica jest przedstawiona w pamięci komputera jako wektor. Tak więc kolejna tablica dwuwymiarowa jest przedstawiona jako ciąg kolejnych kolumn tworzących wektor o liczbie elementów równej liczbie elementów danej tablicy. A dostęp do tego wektora musi się odbywać zawsze tak jak do tablicy dwuwymiarowej.

Tablicą dwuwymiarową jest również tablica złożona z jednego wiersza (x) o 15 elementach lub kolumny (z) o 12 elementach, zadeklarowana w następujący sposób

10 DIM x(1,15)

lub

20 DIM z(12,1)

BASIC na ZX Spectrum nie dopuszcza napisania deklaracji tablic w następujący sposób

10 DIM x(1,15), z(12,1)

Przed każdą nazwą tablicy należy napisać słowo DIM. Tak więc deklarację powyższych tablic możemy napisać też w postaci

10 DIM x(1,15) : DIM z(12,1)

Przykład 7.5.

Dany jest n-elementowy wektor w (w_i, i=1, 2, ... n)
Odwrócić kolejność elementów wektora wg wzoru

$$\left.\begin{aligned} W_i &= W_{n-i+1} \\ W_{n-i+1} &= W_i \end{aligned}\right\} \quad i = 1, 2, \dots [\tfrac{n}{2}]$$

([X] — część całkowita x)

Program

```
 10 REM Przyklad 7.5
 15 REM n - liczba elementow
            wektora
        i - indeks
     w(i) - i-ty element wektora
 20 INPUT "wprowadz n : ";n
 25 REM  wektor deklarujemy ja-
         ko tablice jdnowymia-
         rowa, n-elementowa

 30 DIM w(n)
 35 PRINT "wektor dany : ": PRIN
T
 40 FOR i=1 TO n
 45 INPUT "wprowadz w(i) : ";w(
i)
 50 PRINT w(i);"  ";
 55 NEXT i: PRINT
 60 LET k=INT (n/2)
 65 REM r - zmienna robocza,wy-
          korzystywana do
          "przechowywania"
          wartosci w(i) przy
          przestawianiu

 70 FOR i=1 TO k
 75 LET r=w(i)
```

```
 80 LET w(i)=w(n-i+1)
 85 LET w(n-i+1)=r
 90 NEXT i
 95 PRINT '"nowy wektor : ": PR
INT
 100 FOR i=1 TO n
 105 PRINT w(i);"  ";
 110 NEXT i
 115 STOP
```

b) wyniki dla przykładowych danych

```
wektor dany :

2 4  6  8  10  12  14  16  18

nowy wektor :

18  16  14  12  10  8  6  4  2
```

Przykład 7.6.

Zaprogramować w BASICU:

1) wypełnienie tablicy A (a_i, $i = 1,2,..n$) liczbami naturalnymi tzn $a_1 = 1$, $a_2 = 2$,... $a_n = n$
2) obliczenie sumy

$$s = \sum_{n=1}^{n} \varepsilon \cdot a_i$$

gdzie: $\varepsilon = -1$ dla a_i podzielnych przez 3
$\varepsilon = +1$ dla pozostałych

3) wpisanie liczb a_i podzielnych przez 3 do tablicy o nazwie B

a) analiza zadania

Elementom tablicy A będziemy nadawać wartości w trakcie obliczeń za pomocą instrukcji podstawienia. Do sprawdzenia czy dana wartość dzieli się przez 3, uzywamy funkcji standardowej INT (część całkowita liczby, moze być ujemna; podaje najbliższą liczbę całkowitą mniejszą od argumentu). Jezeli liczba dzieli się przez 3, to sumujemy ze znakiem — oraz wpisujemy tę wartość do tablicy B, jezeli nie, to sumujemy ze znakiem +

Narysowanie schematu blokowego pozostawiamy czytelnikowi

```
 10 REM Przyklad 7.6
 15 REM n - liczba naturalna
            (liczba elementow
            tablicy A)
        i - kolejna liczba na-
            turalna(indeks ta-
            blicy A)
     A(i) - element tablicy A
     B(j) - element tablicy B
        j - indeks  tablicy B
            (liczba elementow
            w tablicy B)
        s - suma elementow ta-
            cy A(z odpowiednim
            znakiem)
 20 INPUT "wprowadz n : ";n
 30 DIM A(n): DIM B(n)
 35 REM nie znamy liczby elemen
        tow tablicy wynikowej,
        wiec deklarujemy ja ja-
        ko tablice o liczbie
        elementow = liczbie ele
        mentow tablicy wejscio-
        wej
 40 LET s=0: LET j=0
 50 FOR i=1 TO n
 60 LET A(i)=i
 65 LET E=1
 70 LET r=i/3
 75 IF r<>INT (r) THEN GO TO 90
 80 LET j=j+1: LET E=-1
 85 LET B(j)=i
 90 LET s=s+E*A(i)
 100 NEXT i
 105 PRINT "tablica liczb natura
lnych                od 1 do ";n
 110 PRINT
 115 FOR i=1 TO n
 120 PRINT "  ";A(i);
 125 NEXT i: PRINT
 130 PRINT '"tablica liczb natur
alnych           podzielnych prze
 135 PRINT
 140 FOR k=1 TO j
 145 PRINT "  ";B(k);
 150 NEXT k: PRINT
 155 PRINT 'TAB (8);"Suma = ";s
 160 STOP
```

W programach z przykładów 7.5 i 7.6 zastosowano instrukcje INPUT i LET do nadawania wartości elementom tablic (zmiennym indeksowanym)

Ponadto w programach tych pokazano, iz na zmiennych indeksowanych numerycznych mozemy wykonywać te same działania jakie wykonujemy na zmiennych prostych numerycznych

Z dotychczasowych wykładów wiemy, ze aby wpisać w formie konwersacyjnej pewną liczbę wartości zmiennych, używamy instrukcji INPUT. Czasami zdarza się, ze znamy te wartości przed wykonaniem programu, wtedy mozemy je umieścić w programie dzięki instrukcji READ (czytaj) i DATA (dane). Wykonując program komputer sam je umieści w odpowiednim momencie w pamięci. Instrukcje te mozna też zastosować do nadawania wartości elementom tablic (zmiennych indeksowanych)

Przykład 7.7

Dany jest program

```
 10 REM Przyklad 7.7
 15 REM wypelnianie tablic x
        i y tymi samymi wartos-
        ciami
 20 DIM x(3,5): DIM y(3,5)
 25 REM nadawanie wartosci ele-
        mentom tablicy x - wier
        szami
 30 FOR i=1 TO 3
 40 FOR j=1 TO 5
 50 READ x(i,j)
 60 NEXT j
 70 NEXT i
 80 RESTORE
 90 REM nadawanie wartosci ele-
        mentom tablicy y - ko-
        lumnami
 100 FOR j=1 TO 5
 110 FOR i=1 TO 3
 120 READ y(i,j)
 130 NEXT i
 140 NEXT j
 160 PRINT TAB (5);"tablica x"
 165 PRINT
 170 FOR i=1 TO 3
 180 FOR j=1 TO 5
 190 PRINT "  ";x(i,j);
 200 NEXT j
 205 PRINT
 210 NEXT i
 215 PRINT
 220 PRINT TAB (5);"tablica y"
 225 PRINT
 230 FOR i=1 TO 3
 240 FOR j=1 TO 5
 250 PRINT "  ";y(i,j);
 260 NEXT j
 265 PRINT
 270 NEXT i
 280 STOP
 290 DATA 1,2,3,4,5,6,7,8,9,10
 300 DATA 11,12,13,14,15
```

W programie tym zastosowano instrukcje READ, DATA i RESTORE. W trakcie wprowadzania linii programu jest mozliwe tworzenie zestawu danych, którymi będzie operował program. Do tego celu słuzy instrukcja bierna DATA postaci

n DATA lista

gdzie

n — numer wiersza

lista — dowolna liczba stałych numerycznych lub alfanumerycznych oddzielonych od siebie przecinkami

Występujące w programie instrukcje DATA tworzą według rosnących numerów wierszy, jeden zestaw danych. I tak w naszym programie instrukcje DATA o numerach wierszy 230 i 240 tworzą jeden zestaw danych postaci

1,2,3,4,5 6,7,8 9,10,11,12,13,14,15

Instrukcje DATA mozna napisać w dowolnym miejscu programu. Zaleca się, aby instrukcje te pisać na końcu programu, grupując wszystkie z danego programu obok siebie

Do przenoszenia wartości z zestawu DATA pod zmienne programu służy instrukcja czynna READ postaci

n READ lista

gdzie: n — numer wiersza

lista — nazwy zmiennych prostych numerycznych lub alfanumerycznych, zmiennych indeksowanych numerycznych lub alfanumerycznych oddzielone od siebie przecinkami

Wykonanie instrukcji READ powoduje pobieranie kolejnych wartości stałych z zestawu danych (poczynając od najniższego numeru wiersza instrukcji DATA) i podstawianie ich pod kolejne nazwy zmiennych występujące na liście READ. Jeżeli zestaw danych z pierwszej instrukcji DATA zostanie wyczerpany, to pobieranie następuje z zestawu danych następnej instrukcji DATA o wyższym numerze wiersza (pierwszym wyższym)

W tym programie wykonanie instrukcji READ w ciągu instrukcji

```
30 FOR i=1 TO 3
40 FOR j=1 TO 5
50 READ x(i,j)
60 NEXT j
70 NEXT i
```

spowoduje wyczerpanie wszystkich danych z zestawu danych instrukcji DATA, „zabrakło" więc danych do wykonania ciągu instrukcji o numerach 100—140 (brak DATA o wyższych numerach wierszy). Wykonanie instrukcji RESTORE powoduje przydzielenie do aktualnego czytania zestawu danych numerycznych w instrukcji DATA (230 i 240) poczynając od najniższego numeru wiersza instrukcji DATA (tego samego zestawu danych)

Instrukcja RESTORE ma postać:

n RESTORE

gdzie:

n — numer wiersza

Instrukcji RESTORE używa się wtedy, gdy trzeba wykonać obliczenia na tym samym zestawie danych (umieszczonych w DATA) lub przyporządkowania tych samych wartości kilku zmiennym w kilku miejscach tego samego programu. Wykonanie tej instrukcji powoduje potraktowanie kolejnej instrukcji READ tak, jakby była pierwszą instrukcją READ w programie, tzn. będzie ona pobierać wartości z pierwszej instrukcji DATA w danym programie (instrukcja o najniższym numerze wiersza)

Przykład 7.8.

Dane jest 8 wektorów 5-elementowych o znanych wartościach elementów. Znaleźć liczbę wektorów, których długość nie przekracza zadnej wielkości d

a) analiza zadania

Długość wektora x obliczamy według wzoru

$$l = \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2}$$

```
10 REM Przyklad 7.8
15 REM s - suma kwadratow x(i)
       l - dlugosc wektora
       d - zadana wielkosc
       m - liczba wektorow
           o dlugosci <=d
20 INPUT "wprowadz d :";d
30 DIM x(5)
35 LET m=0
40 FOR i=1 TO 8
45 PRINT '"wektor danu :"
50 LET s=0
55 FOR j=1 TO 5
60 READ x(j)
65 PRINT "  ";x(j);
70 LET s=s+x(j)*x(j)
75 NEXT j
80 LET l=SQR (s)
90 IF l<=d THEN LET m=m+1
95 PRINT
100 PRINT "dlugosc wektora : ";
l
105 NEXT i
110 PRINT '"liczba wektorow,kto
rych dlugosc nie przekracza zada
nej wielkos- ci d = ";d;"    wyn
osi = ";m
115 STOP
120 DATA 1,3,5,7,9,2,4,6,8,10
125 DATA 1,2,5,8,9,3,4,6,7,10
130 DATA 1,2,3,4,5,6,7,8,9,10
135 DATA 10,1,9,2,8,3,7,4,6,5
```

Wartości elementów wektorów znamy przed wykonaniem programu, więc możemy je umieścić w programie stosując instrukcję DATA. Nadawać będziemy wartości elementów pierwszego wektora, obliczać jego długość i porównywać z wielkością d, potem drugiemu, trzeciemu wektorowi itd

Narysowanie schematu blokowego pozostawiamy czytelnikowi

b) program

Jeżeli chcielibyśmy programy z przykładów 7.7. i 7.8 wykonać dla innego zestawu danych, należałoby zmienić listę stałych (numerycznych) w instrukcji DATA. Zmiany tej dokonujemy przed wykonaniem programu

Tak więc, jeżeli program ma wykonywać obliczenia dla różnych partii danych, należy zastosować instrukcje INPUT

W instrukcji DATA umieszcza się występujące zazwyczaj w programie stałe, z których korzystamy kilka razy w danym programie

Zadania:

Napisać programy w BASIC-u realizujące następujące zadanie

1. Dane są macierze prostokątne A,B,C (i = 1,2,...,m, j = 1,2,...,n)
   Oblicz macierz D według wzoru
   D = p · (A + B) + C
   gdzie: p jest zadaną liczbą (skalarem)
2. Dana jest macierz kwadratowa A(a_{ij}, i,j = 1,2,...,n)
   Obliczyć sumę tych elementów, które należą do zadanego przedziału [c,d]
3. Dane jest n-wektorów k-elementowych. Znaleźć liczbę wektorów, których długość nie przekracza zadnej wielkości d
   Uwaga: jest to uogólnione zadanie z przykładu 7.8. Wektory należy umieścić w tablicy dwuwymiarowej, w której wiersz reprezentuje wektor k-elementowy (porównaj przykład 7.3).

ZZW

BADANIE KIEROWCÓW

Mieczysław SKONIECZNY

Ten program służy do badania orientacji przestrzennej i szybkości podejmowania decyzji przez kierowcę. Pierwowzorem prezentowanego symulatora mikrokomputerowego jest aparat krzyżowy DUFOUR

Prezentowany program mikrokomputerowy, eksploatowany na mikrokomputerze typu AMSTRAD CPC 6128, składa się z dwóch etapów

— próbnego — złożonego z 4 impulsów. Jego celem jest przygotowanie kierowcy do badania właściwego oraz oswojenie go ze sprzętem mikrokomputerowym,

— właściwego — złożonego z 10 impulsów, gdzie przy każdej emisji liczony jest przez komputer czas wykonania zadania

Jeden impuls polega na wyświetleniu na ekranie monitora dwóch obiektów

— statycznego (kwadratowa ramka)

— dynamicznego (sterowanego przez badanego) — wypełniony kwadrat

Zadaniem badanego jest umieszczenie obiektu dynamicznego (kwadratu) w obiekcie statycznym (ramce). Sterowanie kwadratem odbywa się za pomocą czterech klawiszy funkcyjnych, rozmieszczonych z prawej strony klawiatury

↑ — ruch do góry

↓ — ruch w dół

→ — ruch w prawo

← — ruch w lewo

Rozmieszczenie obiektów na ekranie jest generowane losowo przez mikrokomputer.

Aby zapewnić porównywalność impulsów, odległość obu obiektów w momencie startowym jest stała. Po pomyślnym zareagowaniu na impuls badany kierowca słyszy efekt dźwiękowy i na dole ekranu wyświetlany jest czas jego reakcji. Na zakończenie programu podawany jest średni czas reakcji i ocena końcowa w skali czterostopniowej. Dodatkowo pojawia się ocena słowna, określająca stopień przydatności do wykonywania zawodu kierowcy

Kryteria graniczne poszczególnych ocen w skali czterostopniowej wyznaczono metodami statystycznymi. Zbadano próbą losową 150 kierowców wojskowych i uzyskane przez nich wyniki poddano analizie statystycznej, uwzględniającej charakterystyki rozkładu normalnego

UWAGI PROGRAMOWE

1. Jeżeli użytkownik chciałby zmienić graficzną reprezentację obiektu dynamicznego i statycznego, może uczynić to w prosty sposób. W tym celu wystarczy usunąć podprogram z linii 640 który rysuje na ekranie kwadrat i ramkę. Zamiast niego proponujemy wprowadzić ciąg instrukcji:

```
640 LOCATE y — 1, x — 1: PRINT CHR$ (144), CHR$ (145), CHR$ (146)
642 LOCATE y — 1, x : PRINT CHR$ (147)," "; CHR$ (148)
644 LOCATE y — 1, x + 1: PRINT CHR$ (149); CHR$ (150), CHR$ (151)
646 RETURN
```

Efektem działania tego podprogramu jest wrysowanie w określone (współrzędnymi x,y) miejsce ekranu obiektu statycznego. Dodatkowo należy zwiększyć zestaw symboli graficznych, definiowanych przez użytkownika, zastępując instrukcję 700 ciągiem instrukcji:

```
700 SYMBOL AFTER 143
701 SYMBOL 143, 60, 126, 255, 255, 255, 255, 126, 60
702 SYMBOL 144, 0, 0, 0, 1, 7, 15, 30
703 SYMBOL 145, 0, 0, 0, 255, 255, 255, 129
704 SYMBOL 146, 0, 0, 0, 128, 224, 240, 240, 120
705 SYMBOL 147, 30, 28, 28, 28, 28, 28, 28, 30
706 SYMBOL 148, 120, 56, 56, 56, 56, 56, 56, 120
707 SYMBOL 149, 30, 15, 15, 7, 1
708 SYMBOL 150, 0, 129, 255, 255, 255
709 SYMBOL 151, 120, 240, 240, 224, 128
```

Aby wprowadzone zmiany nie kolidowały z dotychczasową konstrukcją programu, należy usunąć instrukcję 620. Gdyby użytkownik chciał stosować inne obiekty dynamiczne, musi dokonać analogicznych modyfikacji programu

2. W etapie właściwym po każdej poprawnej reakcji na impuls w dowolnej części ekranu wyświetlana jest informacja o czasie reakcji badanego kierowcy. Informację tę można uzupełnić podając dodatkowo dotychczasowy średni czas reakcji oraz komentarz oceniający dotychczasową działalność badanego. W tym celu należy instrukcję 660 zastąpić następującymi instrukcjami:

```
656 MODE 1 PEN 2:G1 = TIME CZ = CZ + (G1 — G) /300 LOCATE 8,1
657 PRINT "CZAS REAKCJI NA IMPULS NR." R
658 PEN 3 LOCATE 18,2 PRINT (G1 — G) /300," SEK
659 PEN 2 LOCATE 5,4 PRINT" DOTYCHCZASOWY ŚREDNI CZAS REAKCJI
660 PEN 3 LOCATE 18,5 PRINT CZ/R " SEK
661 IF R = 1 THEN GOTO 670
662 IF (G1 — G) /300<CZ/R THEN GOTO 665
663 LOCATE 1,7 PRINT"TEMPO SŁABNIE. POPRAW SZYBKOŚĆ REAKCJI
664 GOTO 670
665 LOCATE 4,7 PRINT"JESTEŚ DOBRY. UTRZYMAJ TO TEMPO
670 a$ = INKEY$
680 IF INKEY$ = "" THEN 680
```

Dodatkowo należy usunąć instrukcję o etykiecie 220

3. Stosunkowo prostymi modyfikacjami programowymi można uzyskać sterowanie obiektem dynamicznym za pomocą innych klawiszy. W tym celu należy dołączyć do programu źródłowego następujący podprogram

```
1002 MODE 1:LOCATE 1,24 INPUT „CZY CHCESZ ZDEFINIOWAĆ WŁASNE KLAWISZE STERUJĄCE (T/N) ",K$
1005 IF K$ = "T" OR K$ = "t" THEN GOTO 1020
1010 IF K$ = "n" OR K$ = "N" THEN GOTO
1015 LOCATE 17,25 PRINT" ":GOTO 1000
1020 LOCATE 1,1:INPUT "RUCH DO GÓRY — ", B$ (1)
1025 LOCATE 1,3 INPUT "RUCH NA DÓŁ — ", B$ (2)
1030 LOCATE 1,5 INPUT "RUCH W PRAWO — ", B$ (3)
1035 LOCATE 1,7 INPUT "RUCH W LEWO — ", B$ (4)
1040 RETURN
1050 B$ (1) = CHR$ (240)
1052 B$ (2) = CHR$ (241)
1054 B$ (3) = CHR$ (242)
1056 B$ (4) = CHR$ (243)
1058 GOTO 1040
```

Dodatkowo należy wprowadzić instrukcję 39 **GOSUB** 1002 oraz zmodyfikować instrukcje 410—440 w następujący sposób

```
410 IF a$ =B$ (1) GOTO 450
420 IF a$ =B$ (2) GOTO 490
430 IF a$ =B$ (3) GOTO 530
440 IF a$ =B$ (4) GOTO 570
```

Mieczysław SKONIECZNY

```
10 CLEAR INPUT:BORDER 0:PAPER 0:CLS:MODE 1:INK 0,0
:INK 2,24:GOSUB 960
20 WINDOW #1,6,32,5,12:PAPER #1,2:CLS #1:PEN #1,0:
LOCATE #1,11,2:PRINT#1,"BADANIE":LOCATE #1,2,4:PRI
NT#1,"ORGANIZACJI PRZESTRZENNEJ":LOCATE #1,2,5:PRI
NT#1,"I SZYBKOSCI PODEJMOWANIA":LOCATE #1,2,6:PRIN
T#1,"DECYZJI PRZEZ KIEROWCO":INK 2,24,26

30 IF INKEY$="" THEN GOTO 30
40 MODE 1:INK 1,14:INK 2,24:INK 3,6:WINDOW #1,1,40
,1,7:PAPER #1,2:CLS #1:PEN #1,0:LOCATE #1,2,2:PRIN
T#1,"Program ma za zadanie zbada* organiza-":LOCAT
E #1,2,3:PRINT#1,"cjE przestrzenni i szybko7* pode
jmow-"
50 LOCATE #1,2,4:PRINT#1,"mia decyzji u badanego u
9ytkownika-kie-":LOCATE #1,2,5:PRINT#1,"rowcy wojs
kowego. Sklada on siE z eta-":LOCATE #1,2,6:PRINT#
1,"pu pr5bnego i wla7ciwego."
60 WINDOW #2,2,39,9,15:PAPER #2,1:CLS #2:PEN #2,0:
LOCATE #2,2,2:PRINT#2,"Etap pr5bny jest zlo9ony z
4 pr5b ma-":LOCATE #2,2,3:PRINT#2,"j!cych na celu
przygotowanie badanego":LOCATE #2,2,4:PRINT#2,"do
etapu wla7ciwego i oswojenie ze"
70 LOCATE #2,2,5:PRINT#2,"sprzEtem mikrokomputerow
ym. Nie jest":LOCATE #2,2,6:PRINT#2,"badany czas r
eakcji."
80 WINDOW #3,2,39,17,22:PAPER #3,3:PEN #3,0:CLS #3
:LOCATE #3,2,2:PRINT#3,"Etap wla7ciwy sklada siE z
 10 pr5b,":LOCATE #3,2,3:PRINT#3,"gdzie w ka9dej z
 nich jest naliczany":LOCATE #3,2,4:PRINT#3,"przez
 mikrokomputer czas reakcji ba-"
90 LOCATE #3,2,5:PRINT#3,"danego.":PEN 2:LOCATE 1,
25:PRINT"NACISNIJ 'SPACJO'"
100 IF INKEY$="" THEN 100
110 MODE 1:WINDOW #1,1,40,1,7:PAPER #1,2:CLS #1:PE
N #1,0:LOCATE #1,2,2:PRINT#1,"Jedna pr5ba polega n
a wy7wietleniu na":LOCATE #1,2,3:PRINT#1,"ekranie
monitora 2 obiekt5w:":LOCATE #1,3,4:PRINT#1,"- sta
tycznego (kwadratowa ramka)"
120 LOCATE #1,3,5:PRINT#1,"- dynamicznego (sterowa
nego przez ba-":LOCATE #1,5,6:PRINT#1,"danego) - w
ypelniony kwadrat."
130 WINDOW #2,1,40,9,16:PAPER #2,1:PEN #2,0:CLS #2
:LOCATE #2,2,2:PRINT#2,"Zadaniem kierowcy jest umi
eszczanie":LOCATE #2,2,3:PRINT#2,"obiektu dynamicz
nego (kwadratu) w obie-":LOCATE #2,2,4:PRINT#2,"kc
ie statycznym (ramce)."
140 LOCATE #2,2,5:PRINT#2,"Sterowanie kwadratem od
bywa siE za po-":LOCATE #2,2,6:PRINT#2,"moc! 4 kla
wiszy funkcyjnych rozmie-":LOCATE #2,2,7:PRINT#2,"
szczonych z prawej strony klawiatury:"
150 PEN 3:LOCATE 4,18:PRINT CHR$(240);" - ruch do
g5ry":LOCATE 4,20:PRINT CHR$(241);" - ruch na d5l"
:LOCATE 4,22:PRINT CHR$(243);" - ruch w prawo":LOC
ATE 4,24:PRINT CHR$(242);" - ruch w lewo":PEN 1:PR
INT "  ":PRINT "NACISNIJ 'SPACJO'"
160 IF INKEY$="" THEN 160
170 MODE 1:WINDOW#1,1,40,2,12:PAPER#1,1:CLS#1:PEN#
1,0:LOCATE#1,1,2:PRINT#1,"Rozmieszczanie obiekt5w
na ekranie jest generowane losowo przez mikrokompu
ter.  Jednak,9eby zapewni* por5wnywalno7* pr5bodle
glo7* obu obiekt5w w momencie star-"
180 LOCATE #1,1,6:PRINT#1,"towym jest stala.":LOCA
TE #1,1,8:PRINT#1,"Po pomy7lnym zako3czeniu pr5by
badany   slyszy efekt dzwiEkowy i u dolu ekranu  j
est wy7wietlany czas reakcji."
190 WINDOW#2,2,40,15,20:PAPER#2,2:CLS#2:PEN#2,0:LO
CATE #2,2,2:PRINT#2,"Na zako3czenie programu jest
podawany":LOCATE #2,2,3:PRINT#2,"czas reakcji i oc
ena badanego w skali":LOCATE #2,2,4:PRINT#2,"czter
ostopniowej.":PEN 2:LOCATE 1,24:PRINT"NACISNIJ 'SP
ACJO'
200 IF INKEY$="" THEN 200
210 MODE 0:BORDER 14:PAPER 1:CLS:WINDOW #1,3,18,12
,16:PAPER #1,3:CLS#1:PEN #1,0:LOCATE #1,3,3:PRINT#
1,"ETAP PR6BNY":INK 0,0,24
220 IF INKEY$="" THEN 220
230 MODE 1:INK 0,14:INK 1,0:INK 2,1:INK 3,26:BORDE
R 14:PAPER 0:P1=0:CLS
240 FOR r=1 TO 5
250 GOSUB 520
260 FOR z=1 TO 1000:NEXT
270 NEXT
280 P1=1:INK 1,6
290 MODE 0:INK 1,14:INK 2,24:INK 3,6:INK 0,0:BORDE
R 14:PAPER 1:CLS:WINDOW #1,3,18,12,16:PAPER #1,3:C
LS#1:PEN #1,0:LOCATE #1,2,3:PRINT#1,"ETAP W2ASCIWY
":INK 0,0,24
300 IF NOT(INKEY$=" ") THEN 300
310 FOR H=1 TO 500:NEXT
320 cz=0
330 FOR R=1 TO 10
340 GOSUB 520
350 CZ=CZ+((G1-G)/300)
360 NEXT
370 FOR H=1 TO 2000:NEXT
380 GOSUB 960:MODE 1:INK 0,0:INK 1,0:INK 2,24:INK
3,6:BORDER 24:PAPER 2:CLS:WINDOW #1,3,17,2,4:PAPER
 #1,0:CLS #1:PEN #1,2:LOCATE #1,2,2:PRINT#1,"WYNIK
I TESTU"
390 PEN 1:LOCATE 1,7:PRINT"SREDNI CZAS REAKCJI - "
;:PRINT USING "###.##";CZ/10;:PRINT" SEK"
400 PEN 3:LOCATE 9,12:PRINT"##### OCENA #####":PEN
 1
410 cz=cz/10
420 IF cz<4.74 GOTO 490
430 IF cz<5.57 GOTO 480
440 IF cz<8  GOTO 470
450 GOTO 460
460 LOCATE 1,14:PRINT"TWOJE WYNIKI S) BARDZO S2ABE
. BADANIE   ORGANIZACJI PRZESTRZENNEJ I SZYBKOSCI
  PODEJMOWANIA DECYZJI DYSKWALIFIKUJE CIO JAKO KIE
ROWCO WOJSKOWEGO.MUSISZ ZDECYDO-WANIE JWICZY1.":PR
INT:PEN 3:PRINT"UZYSKAZES OCENO  2":GOTO 500
470 LOCATE 1,14:PRINT"TWOJE WYNIKI S) S2ABE. BADAN
IE ORGANIZA-CJI PRZESTRZENNEJ I SZYBKOSCI PODEJMOW
```

```
A-NIA DECYZJI KWALIFIKUJE CIO JAKO KIE-   ROWCO WO
J9KOWEGO.ALE MUSISZ ZDECYDOWANIEJWICZY].":PRINT:PE
N 3:PRINT"UZYSKA2EB OCENO  3":GOTO 500
480 LOCATE 1,14:PRINT"TWOJE WYNIKI S) DOBRE. BADAN
IE ORGANIZA-CJI PRZESTRZENNEJ I SZYBKOSCI PODEJMOW
A-NIA DECYZJI KWALIFIKUJE CIO JAKO KIE-   ROWCO WO
JSKOWEGO.NIE ZASZKODZI JEDNAK   JAK BODZIESZ JWICZ
Y2.":PRINT:PEN 3:PRINT"UZYSKA2EB OCENO  4":GOTO 50
0
490 LOCATE 1,14:PRINT"TWOJE WYNIKI S) BARDZO DOBRE
. BADANIE   ORGANIZACJI PRZESTRZENNEJ I SZYBKOSCI
  PODEJMOWANIA DECYZJI KWALIFIKUJE CIO    JAKO KIE
ROWCO WOJSKOWEGO.NIE MUSISZ     JWICZY].":PRINT:PE
N 3:PRINT"UZYSKA2EB OCENO  5":GOTO 500
500 IF NOT(INKEY$="s" OR INKEY$="S") THEN 500
510 GOTO 10
520 MODE 1:INK 0,14:INK 1,0:INK 2,1:INK 3,26:BORDE
R 14:PAPER 0:CLS
530 PEN 1
540 x=2+INT(22*RND)
550 y=2+INT(37*RND)
560 IF x>12 GOTO 580
570 x1=x+12:GOTO 590
580 x1=x-12
590 IF y>20 GOTO 610
600 y1=y+15:GOTO 620
610 y1=y-15
620 CLEAR INPUT:PEN 1:GRAPHICS PEN 3:GOSUB 890:LOC
ATE y1,x1:PRINT CHR$(143);
630 q=TIME
640 a$=INKEY$
650 IF a$="" THEN GOTO 640
660 IF a$=CHR$(240) GOTO 700
670 IF a$=CHR$(241) GOTO 740
680 IF a$=CHR$(242) GOTO 780
690 IF a$=CHR$(243) GOTO 820
700 IF x1=1 GOTO 640
710 FRAME:LOCATE y1,x1:PRINT CHR$(128);:LOCATE y1,
x1-1:PRINT CHR$(143);:x1=x1-1
720 FRAME
730 GOTO 860
740 IF x1=25 GOTO 640
750 FRAME:LOCATE y1,x1:PRINT CHR$(128);:LOCATE y1,
x1+1:PRINT CHR$(143);:x1=x1+1
760 FRAME
770 GOTO 860
780 IF y1=1 GOTO 640
790 FRAME:LOCATE y1-1,x1:PRINT CHR$(143);CHR$(128)
;:y1=y1-1
800 FRAME
810 GOTO 860
820 IF y1=40 GOTO 640
830 FRAME:LOCATE y1,x1:PRINT CHR$(128);:LOCATE y1+
1,x1:PRINT CHR$(143);:y1=y1+1
840 FRAME
850 GOTO 860
860 IF X=X1 AND Y=Y1 GOTO 900
870 GOSUB 890
880 GOTO 640
890 Z=((Y-1)*16)-2:V=((25-X)*16)-2:LOCATE y,x:PRIN
T CHR$(143);:PLOT Z,V:DRAW Z+18,V:DRAW Z+18,V+18:D
RAW Z,V+18:DRAW Z,V:z=z-2:v=v-2:PLOT z,v:DRAW z+22
,v:DRAW z+22,v+22:DRAW z,v+22:DRAW z,v:RETURN
900 SOUND 1,478:SOUND 1,358:GOSUB 890:SPEED INK 10
,10:INK 1,6,26
910 IF P1=0 GOTO 930
920 PEN 2:LOCATE 1,24:q1=TIME:PRINT "CZAS - ";:PRI
NT USING "##.##";(q1-q)/300;:PRINT" sek"
930 a$=INKEY$
940 FOR I=1 TO 1000:NEXT
950 RETURN
960 SYMBOL AFTER 170
970 SYMBOL 171,0,0,120,12,124,204,118,3
980 SYMBOL 172,24,60,102,102,126,102,102,3
990 SYMBOL 173,24,0,60,102,96,102,60,0
1000 SYMBOL 174,24,60,102,192,192,102,60,0
1010 SYMBOL 175,0,0,60,102,126,96,60,6
1020 SYMBOL 176,254,98,104,120,104,98,254,3
1030 SYMBOL 177,56,24,28,24.56,24,60,0
1040 SYMBOL 178,240,96,112,96,226,102,254,0
1050 SYMBOL 179,24,0,220,102,102,102,102,0
1060 SYMBOL 180,24,198,230,246,222,206,198,0
1070 SYMBOL 181,24,0,60,102,102,102,60,0
1080 SYMBOL 182,12,56,108,198,198,108,56,0
1090 SYMBOL 183,24,0,60,96,60,6,124,0
1100 SYMBOL 184,24,60,96,60,6,102,60,0
1110 SYMBOL 185,24,0,126,76,24,48,126,0
1120 SYMBOL 186,254,198,140,126,48,102,254,0
1130 SYMBOL 187,12,24,126,76,24,48,126,0
1140 SYMBOL 188,48,254,172,24,50,102,254,0
1150 SYMBOL 189,0,0,0,0,0,12,12,24
1160 KEY DEF 10,1,55,171,172
1170 KEY DEF 11,1,56,173,174
1180 KEY DEF 3,1,57,175,176
1190 KEY DEF 20,1,52,177,178
1200 KEY DEF 12,1,53,179,180
1210 KEY DEF 4,1,54,181,182
1220 KEY DEF 13,1,49,183,184
1230 KEY DEF 14,1,50,185,186
1240 KEY DEF 5,1,51,187,188
1250 KEY DEF 15,1,189
1260 RETURN
```

M. Skoneczny

Emulator Basic 1.1 dla CPC 464 (2)

Zgodnie z zapowiedzią z poprzedniego numeru przystąpię teraz do przedstawienia komend, o które wzbogacony został BASIC 1.0.

CURSOR (przełącznik systemowy) (przełącznik użytkownika).

Funkcja ustawia przełączniki następująco włączony (1) i wyłączony (0). Kursor jest wyświetlany, jeśli oba przełączniki są włączone. Musi wystąpić przynajmniej jeden parametr. Nie można pominąć obu.

ON BREAK CONT i ON BREAK STOP

Pierwsza z tych funkcji likwiduje działanie klawisza ESC, którego wciśnięcie powoduje przerwanie programu. Działanie tego rozkazu jest znoszone przez drugą funkcję.

Mask (wyrażenia całkowite), (początkowe ustawienie punktu).

Rozkaz ustawia maskę i jest stosowany podczas kreślenia linii. Binarna wartość „wyrażenia całkowitego" z przedziału 0 — 255 ustawia bity w każdej przyległej grupie 8 punktów w stan włączony (1) lub wyłączony (0). Parametr (początkowe ustawienie punktu) określa, czy pierwszy punkt linii będzie wyświetlany (1), czy też nie (0). Musi wystąpić przynajmniej jeden z parametrów.

GRAPHICS PEN (nr tuszu), (tryb tła)

Rozkaz ustala kolor atramentu dla kreślonej grafiki (0—15). Tryby: O—tło nieprzezroczyste, 1—tło przezroczyste. Tryb tła wpływa na tło znaków wyświetlanych przy aktywnym TAG oraz na kolor przerw w liniach przy określonych parametrach instrukcji MASK.

GRAPHICS PAPER (nr tuszu)

Ustala kolor tła w trybie graficznym. Jest również kolorem tła przy czyszczeniu okna graficznego (CLG).

FRAME

Umożliwia poprawę jakości obrazu przy animacji. Dzięki tej funkcji uzyskujemy płynny ruch na ekranie.

FILL (nr INK)

Wypełnia dowolny obszar okna graficznego kolorem wyspecyfikowanym w INK.

COPYCHR$ (#nr strumienia)

Kopiuje znaki z bieżącej pozycji kursora znakowego z wyspecyfikowanego strumienia na pozycję instrukcją LOCATE.

CLEAR INPUT

Zeruje bufor klawiatury, usuwając z niego uprzednio wpisane znaki.

Dodatkowo, funkcje graficzne (DRAW, DRAWR, MOVE, MOVER, PLOT, PLOTR) występujące w BASIC'u 1.0 zostały rozszerzone o dodatkowy parametr — tryb tuszu i mają następującą postać: funkcja graficzna (wsp. X), (wsp. Y), (kolor tuszu), (tryb tuszu). Tryb tuszu określa sposób wzajemnego oddziaływania na siebie koloru tła (PAPER) i koloru tuszu (PEN). Parametr ten może przyjmować następujące wartości:

a. 0 — tryb normalny
b. 1 — tryb XOR
c. 2 — tryb AND
d. 3 — tryb OR.

Program demonstracyjny oraz przykład działania funkcji CLEAR INPUT ilustrują działania dodatkowych funkcji. Praca z emulatorem znacznie zwiększa możliwości tworzenia obrazów graficznych przy wykorzystaniu interpretera języka BASIC.

J. R.

```
10 REM**************************************
20 REM     PROGRAM DEMONSTRACYJNY     *
30 REM**************************************
40 MODE 1:INK 0,0:INK 1,9:INK 2,11:INK 3,26
50 WINDOW #1,25,40,1,25:PAPER #1,2:PEN #1,3:CLS #1:WINDOW #2,1,40,25,25:PAPER #2
,1:PEN #2,0:CLS #2
60 WINDOW #3,1,24,1,24
70 GOSUB 1690
80 LOCATE #2,10,1:PRINT #2,"DOKONAJ WYBORU OPCJI";
90 a$=INKEY$:IF a$="" THEN 90
100 IF INKEY(69)=0 THEN 1470
110 IF INKEY(54)=0 THEN 1590
120 IF INKEY(62)=0 THEN 1250
130 IF INKEY(61)=0 THEN 990
140 IF INKEY(58)=0 THEN 370
150 IF INKEY(53)=0 THEN 690
160 IF INKEY(52)=0 THEN 180
170 GOTO 90
180 REM ************ CURSOR ************
190 CLS #1:CLS #2:CLS #3:GOSUB 1830
200 LOCATE #1,2,4:PRINT #1,"*DEMONSTRACJA*";
210 LOCATE #1,2,6:PRINT #1,"    CURSOR";
220 CURSOR 1:LOCATE #0,22,5
230 LOCATE #3,5,5:PRINT #3,"napisz litere :"
240 LOCATE #2,17,1:PRINT #2,"CURSOR 1";
```

```
250 LOCATE #3,22,5
260 a$=INKEY$:IF a$="" THEN 260
270 LOCATE #0,22,5:PRINT #0," ":PRINT #3,a$;
280 CURSOR 0:FOR i=1 TO 2000:NEXT i
290 LOCATE #2,24,1:PRINT #2,"0";
300 LOCATE #3,5,5:PRINT #3,"napisz litere :   "
310 LOCATE #3,22,5
320 a$=INKEY$:IF a$="" THEN 320
330 PRINT #3,a$;
340 LOCATE #2,5,1:PRINT #2,"DOWOLNY KLAWISZ - POWROT DO MENU";
350 a$=INKEY$:IF a$="" THEN 350
360 GOTO 70
370 REM ########### ON BREAK ###########
380 CLS #1:CLS #2:CLS #3:GOSUB 1830
390 LOCATE #1,2,4:PRINT #1,"$DEMONSTRACJA$";
400 LOCATE #1,2,6:PRINT #1,"   ON BREAK";
410 LOCATE #1,2,7:PRINT #1," CONT    STOP";
420 FOR i=1 TO 1500:NEXT i
430 LOCATE #2,10,1:PRINT #2,"BREAK wylaczony"
440 GRAPHICS PEN 1,0:GRAPHICS PAPER 0
450 LOCATE #3,2,12:PRINT #3,CHR$(143):LOCATE #3,23,12:PRINT #3,CHR$(143)
460 TAG:MOVE 33,224
470 ON BREAK CONT
480 FOR z=1 TO 3
490 FOR x=33 TO 336
500 FRAME:MOVE x-1,224:PRINT " ";
510 MOVE x,224
520 PRINT CHR$(243);
530 NEXT x
540 MOVE 336,224
550 FOR x=336 TO 32 STEP -1
560 FRAME:MOVE x+1,224:PRINT " ";
570 MOVE x,224
580 PRINT CHR$(242);
590 NEXT x
600 NEXT z
610 GRAPHICS PEN 3:MOVE 384,200:DRAW 640,200:GRAPHICS PEN 1
620 LOCATE #2,16,1:PRINT #2,"wlaczony ";
630 LOCATE #1,2,15:PRINT #1,"po wystapieniu";
640 LOCATE #1,3,16:PRINT #1,"    BREAK";
650 LOCATE #1,2,17:PRINT #1,"wypalcowac RUN";
660 LOCATE #1,2,18:PRINT #1," wcisnac ENTER";
670 ON BREAK STOP
680 GOTO 480
690 REM ############ MASK  ############

700 CLS #1:CLS #2:CLS #3:GOSUB 1830
710 LOCATE #1,2,4:PRINT #1,"$DEMONSTRACJA$";
720 LOCATE #1,2,6:PRINT #1,"     MASK";
730 GRAPHICS PAPER 0:GRAPHICS PEN 3
740 y=397
750 lp%=0
760 FOR i=1 TO 10
770 MASK &X1
780 MOVE 3,y:DRAW 384,y
790 y=y-1
800 lp%=lp%+1
810 IF lp%>350 THEN 950
820 NEXT i
830 y=y-1:MOVE 3,y:MASK &X11111111:DRAW 384,y:y=y-1
840 lp%=lp%+1:IF lp%>350 THEN 950
850 FOR i=1 TO 10
860 MASK &X10000
870 MOVE 3,y:DRAW 384,y
880 y=y-1
890 lp%=lp%+1
900 IF lp%>350 THEN 950
910 NEXT i
920 y=y-1:MOVE 3,y:MASK &X11111111:DRAW 384,y:y=y-1
930 lp%=lp%+1:IF lp%>350 THEN 950
940 GOTO 760
950 CLS #2:LOCATE #2,5,1:PRINT #2,"DOWOLNY KLAWISZ - POWROT DO MENU";
960 a$=INKEY$:IF a$="" THEN 960
970 MASK &X11111111
980 GOTO 70
990 PEM ##### GRAPHICS PAPER &  PEN ####
1000 CLS #1:CLS #2:CLS #3:GOSUB 1830
1010 LOCATE #1,2,4:PRINT #1,"$DEMONSTRACJA$";
1020 LOCATE #1,2,6:PRINT #1,"   GRAPHICS";
1030 LOCATE #1.2,7:PRINT #1," PAPER    PEN";
1040 TAG:GRAPHICS PAPER 2
1050 GRAPHICS PEN 0,0
1060 FOR I=1 TO 2500:NEXT I
1070 MOVE 50,250:PRINT "Ala";
1080 GRAPHICS PEN 3,1
1090 FOR I=1 TO 2500:NEXT I
1100 MOVE 100,100:PRINT "As";
1110 GRAPHICS PEN 1,0
1120 GRAPHICS PAPER 3
1130 FOR I=1 TO 2500:NEXT I
1140 MOVE 300,200:PRINT "OLA";
1150 FOR I=1 TO 2500:NEXT I
1160 GRAPHICS PAPER 1
1170 GRAPHICS PEN 0,0
1180 MOVE 250,360:PRINT "DOM";
1190 FOR I=1 TO 2500:NEXT I:GRAPHICS PEN 2
1200 TAGOFF:MOVE 150,150:DRAW 150,300:DRAW 250,300:DRAW 250,150:DRAW 150,150:MOV
E 220,220:FILL 1
1210 LOCATE #2,5,1:PRINT #2,"DOWOLNY KLAWISZ - POWROT DO MENU";
1220 a$=INKEY$:IF a$="" THEN 1220
1230 GOTO 70
1240 REM ############# FRAME ###########
1250 CLS #1:CLS #2:CLS #3:GOSUB 1830
1260 LOCATE #1,2,4:PRINT #1,"$DEMONSTRACJA$";
1270 LOCATE #1,2,6:PRINT #1,"     FRAME";
1280 LOCATE #2,10,1:PRINT #2,"funkcja FRAME aktywna";
1290 TAG:MOVE 200,50
1300 FOR y=50 TO 300
1310 FRAME
1320 MOVE 200,y-1
1330 PRINT " ";
1340 MOVE 200,y
1350 PRINT CHR$(239);
1360 NEXT y
1370 TAGOFF:CLS #3:GOSUB 1840
1380 LOCATE #2,24,1:PRINT #2,"wylaczona";
1390 TAG:MOVE 320,0
1400 FOR y=50 TO 300
1410 MOVE 200,y-1
1420 PRINT " ";
1430 MOVE 200,y
1440 PRINT CHR$(239);
1450 NEXT y
1460 GOTO 70
1470 REM ######### COPYCHR$ ###########
1480 CLS #1:CLS #2:GOSUB 1840
1490 LOCATE #1,2,4:PRINT #1,"$DEMONSTRACJA$";
1500 LOCATE #1,2,6:PRINT #1,"   COPYCHR$";
1510 LOCATE #3,5,5:PRINT #3,"XYZ";
1520 FOR I=1 TO 2500:NEXT I
1530 LOCATE #3,5,5:b$=COPYCHR$(#3)
1540 LOCATE #3,10,10:PRINT #3,b$;
1550 LOCATE #2,5,1:PRINT #2,"DOWOLNY KLAWISZ - POWROT DO MENU";
1560 a$=INKEY$:IF a$="" THEN 1560
1570 GOTO 70
1580 REM ########## FILL  #############
1590 CLS #1:CLS #2:GOSUB 1840
1600 LOCATE #1,2,4:PRINT #1,"$DEMONSTRACJA$";
1610 LOCATE #1,2,6:PRINT #1,"     FILL";
1620 MOVE 50,50:DRAW 150,300:DRAW 230,150:DRAW 50,50:MOVE 150,200
1630 FOR I=1 TO 2500:NEXT i
1640 FILL 2: MOVE 260,150:FILL 1
1650 LOCATE #2,5,1:PRINT #2,"DOWOLNY KLAWISZ - POWROT DO MENU";
1660 a$=INKEY$:IF a$="" THEN 1660
1670 GOTO 70
1680 REM ########## MENU ###############
1690 CLS #1:CLS #2:CLS #3:GOSUB 1840
1700 LOCATE #1,2,2:PRINT #1,"#### MENU ####";
1710 LOCATE #1,2,5:PRINT #1,"(A) COPYCHR$";
1720 LOCATE #1,2,6:PRINT #1,"(B) FILL";
1730 LOCATE #1.2,7:PRINT #1,"(C) FRAME";
1740 LOCATE #1,2,8:PRINT #1,"(D) GRAPHICS "
1750 LOCATE #1,2,9:PRINT #1,"    PAPER/PEN"
1760 LOCATE #1,2,10:PRINT #1,"(E) ON BREAK "
1770 LOCATE #1,2,11:PRINT #1,"    CONT"
1780 LOCATE #1,2,12:PRINT #1,"    ON BREAK"
1790 LOCATE #1,2.13:PRINT #1,"    STOP"
1800 LOCATE #1.2,14:PRINT #1,"(F) MASK"
1810 LOCATE #1.2.15:PRINT #1,"(G) CURSOR";
1820 RETURN
1830 REM ####### RYSOWANIE RAMEK #######
1840 MOVE 0,16:DRAW 640,16,3:MOVE 384,398:DRAW 638,398,3:DRAW 638,16:DRAW 384,16
:DRAW 384,398:DRAW 0,398:DRAW 0,16
1850 RETURN
```

```
10 REM ######### CLEAR INPUT ########
20 CLS
30 PRINT "wprowadz znak z klawiatury"
40 FOR i=1 TO 5000
50 NEXT i
60 CLEAR INPUT
```

Ten słownik może się przydać wszystkim tym, którzy samodzielnie przedzierają się przez literaturę. Ci, którzy pierwsze trudności mają już za sobą wiedzą ile wysiłku wymaga zrozumienie obcych terminów technicznych i ich właściwe przetłumaczenie. Nasz słownik adresujemy do wszystkich tych, którzy odważyli się samodzielnie poszukiwać wiedzy. W następnych numerach „IKS-a", zawsze na tych samych stronach, drukować będziemy kolejne hasła. Wytrwali Czytelnicy będą mogli skompletować pokaźnych rozmiarów angielsko-polski słownik informatyczny.

A

ABSOLUTE – absolutny, bezwzględny,
ABSOLUTE ADDITION – dodawanie bezwzględne, dodawanie bez zwracania uwagi na znak algebraiczny,
ABSOLUTE ADDRESS – adres bezwzględny, adres komputera,
ABSOLUTE ADDRESSING – adresowanie absolutne, adresowanie bezwzględne,
ABSOLUTE CODE – kod komputera, kod wewnętrzny,
ABSOLUTE COORDINATES – współrzędne bezwzględne (odniesione do początku określonego systemu współrzędnych),
ABSOLUTE INSTRUCTION – rozkaz pełny,
ABSOLUTE LOADER – program ładujący absolutny,
ABSOLUTE MAGNITUDE – wartość bezwzględna, moduł (liczby),
ABSOLUTE NUMBER – liczba niemianowana, liczba bezwzględna,
ABSOLUTE PROGRAM – program o adresach rzeczywistych,
ABSOLUTE PROGRAMMING – programowanie w adresach rzeczywistych,
ABSOLUTE VALUE – wartość bezwzględna, moduł,
ABSOLUTE VALUE OF A COMPLEX NUMBER – moduł liczby zespolonej,
ABSOLUTE VALUE OF A REAL NUMBER – wartość bezwzględna liczby rzeczywistej,
ABSOLUTE VALUE OF A VECTOR – wartość bezwzględna wektora, miara bezwzględna wektora,
ABSOLUTE VECTOR – wektor bezwzględny, (dot. grafiki),
ABSORBED POWER – moc pobierana,
ABSTRACT – abstrakt, streszczenie, wyciąg, skrót, analiza dokumentacyjna,
ABSTRACT CARD – karta dokumentacyjna.,
ABSTRACT NUMBER – liczba niemianowana, liczba oderwana,
A.C. PLANT – urządzenie klimatyzacyjne, klimatyzator
ACCELERATE – przyspieszać,
ACCELERATION – przyspieszanie,
ACCEPT – akceptować, przyjmować, także: pobrać,
ACCEPT INSTRUCTIONS – przyjmować, realizować rozkazy
ACCEPTABLE RELIABILITY LEVEL – dopuszczalny poziom niezawodności,
ACCESS – dostęp, także: współpraca,
ACCESS ARM – ramię dostępu (mechanizm zapisu lub odczytu danych z dysku),
ACCESS CYCLE – cykl dostępu (do pamięci),
ACCESS METHOD – metoda dostępu,
ACCESS MODE – tryb dostępu (sposób adresowania zapisu w pliku),
ACCESS SPEED – szybkość dostępu,
ACCESS TIME – czas dostępu,
ACCESSIBILITY – dostępność,
ACCESSIBLE – dostępny,
ACCESSING FILE – dostęp do pliku (zbioru),
ACCESSORY – dodatkowy, pomocniczy,
ACCLIMATIZE – aklimatyzować,
ACCOMMODATE – ulokować, umieścić, zamieścić,
ACCOMPLISH – osiągać, realizować, uzyskiwać, wykonywać,
ACCOUNT – faktura, rachunek, także: konto,
ACCOUNTING – księgowanie,
ACCOUNTING DETAIL CARD – karta [illegible],
ACCOUNTING PROGRAM – program księgujący,
ACCUMULATE – gromadzić, zbierać (np. informacje), także: sumować, kumulować,
ACCUMULATED ERROR – błąd sumacyjny,
ACCUMULATING COUNTER – licznik dodający,
ACCUMULATION – dodawanie, gromadzenie (np. dokumentów), akumulacja,
ACCUMULATOR – akumulator,
ACCUMULATOR REGISTER – akumulator,
ACCURACY – dokładność, wierność (np. przeniesienia, przepisania), precyzja,
ACCURACY CONTROL CHARACTER – znak sprawdzenia dokładności,
ACCURACY OF MEASUREMENT – dokładność pomiarów,
ACCURACY OF READING – dokładność odczytu,
ACCURACY OF RECORDING – wierność zapisu,
ACCURATELY – wiernie,
ACHIEVE – osiągać,
ACKNOWLEDGE – potwierdzać,
ACKNOWLEDGE CHARACTER – znak potwierdzający,
ACKNOWLEDGEMENT – potwierdzenie,
ACOUSTIC COUPLER – akustyczne urządzenie WEJ/WYJ,
ACOUSTIC INDICATOR – wskaźnik akustyczny,
ACOUSTIC MEMORY – pamięć akustyczna (na liniach opóźniających),
ACOUSTIC MODEM – modem akustyczny,
ACOUSTIC OUTPUT – wyjście foniczne, wyjście akustyczne,
ACQUAINT – zapoznawać się (z czymś),
ACQUIRE – nabywać (np. wiedzę),
ACROSS – w poprzek (np. taśmy),
ACT – działać, spełniać funkcję, pełnić rolę,
ACTION – działanie, także: operacja, czynność,
ACTIVATE – uruchamiać (np. system, urządzenie),
ACTIVATE THE DEVICE – uruchamiać urządzenie,
ACTIVE CURRENT – prąd czynny,
ACTIVE PAGE – stronica dostępna (część bufora ekranu dostępna dla instrukcji zapisu),
ACTIVE-TIME – czas (data) ważności pliku (zbioru),
ACTIVITY – czynność (np. sumowania, sortowania), także: działalność,
ACTUAL – faktyczny, rzeczywisty,
ACTUAL PARAMETER – parametr aktualny,
ACTUALLY – faktycznie, rzeczywiście,
ACTUATE – uruchamiać, także: pobudzać,
ACTUATING ERROR – uchyb na wejściu członu wzmacniającego,
ACTUATING SIGNAL – sygnał uruchamiający,
ACU – patrz: AUTOMATIC CALLING UNIT,
A-D CONVENTER – patrz: ANALOG-TO-DIGITAL CONVENTER
A/D CONVERSION – przetwarzanie analogowo-cyfrowe,
ADAPT – dostosowywać,
ADAPTATION – adaptacja, dostosowanie,
ADAPTER – urządzenie dopasowywujące,
ADAPTING – dostosowywanie się, przystosowywanie się,
ADAPTIVE CONTROL – adaptacyjny układ sterowania,
ADD – dodawać, sumować,
ADD CARRY – przeniesienie przy dodawaniu,
ADD INSTRUCTION – rozkaz dodania,
ADD-ON DEVICE – urządzenie dodatkowe, także: element dyskretny wmontowany do hybrydowego ukł. scalonego
ADD-ON MEMORY – pamięć kumulowana,
ADD-SUBTRACT TIME – czas operacji dodawania lub odejmowania,
ADD UP – dodawać (liczby),
ADDEND – dodajnik, składnik dodawania,
ADDER – sumator, układ sumujący,
ADDER-ACCUMULATOR – akumulator przechowujący wynik dodawania,
ADDER-SUBTRACTOR – układ sumująco-odejmujący,
ADDING CIRCUIT – obwód sumujący,
ADDITION – dodawanie, dodatek, uzupełnienie,
ADDITION WITHOUT CARRY – dodawanie bez przeniesienia,
ADDITIONAL – dodatkowy,
ADDITIONAL CHARACTER – znak specjalny, tzn. nie litera i nie cyfra,
ADDITIONAL INSTRUCTION – rozkaz dodatkowy,
ADDRESABLE MATRIX – macierz adresowalna (określa wielkość obrazu),
ADDRESING – adresowanie,
ADDRESING CIRCUIT – układ wybierania adresu,
ADDRESING SYSTEM – system adresowania,
ADDRESS – adres, pozycja w pamięci,
ADDRESS ADJUSTMENT – przesunięcie adresu, korekta adresu,
ADDRESS ARITHMETIC – arytmetyka adresów,
ADDRESS ASSIGNMENT – przydzielanie adresu,
ADDRESS BUS – szyna adresowa,
ADDRESS COMPUTATION – obliczanie adresu,
ADDRESS CONSTANT – stała adresowa, adres podstawowy, adres początkowy,
ADDRESS CONVERSION – konwersja adresu, przekształcanie adresu,
ADDRESS COUNTER – rejestr (licznik) adresu,
ADDRESS DECODER – dekoder adresu,
ADDRESS INDICATOR – wskaźnik adresu,
ADDRESS LATCH ENABLE (ALE) – znacznik adresu (odróżniający adres od danych),
ADDRESS MODIFICATION – modyfikacja adresu,
ADDRESS NUMBERING SYSTEM – system obliczania adresów
ADDRESS PART – część adresowa (rozkazu),
ADDRESS REGISTER – rejestr adresu,
ADDRESS SELECTION OPERATION – operacja selekcji adresu,

ADDRESS SORTING - sortowanie adresów,
ADDRESS SUBSTITUTION - podstawienie adresu, przeadresowanie,
ADDRESS SYSTEM - system adresowy,
ADDRESS TABLE - lista adresów, tabela adresów,
ADDRESS TRACK - ścieżka adresowa,
ADDRESS TRANSLATION - przekształcenie adresu, konwersja adresu,
ADDRESSABILITY - adresowalność, obszar adresowy,
ADDRESSABLE - adresowalny, dający się adresować,
ADDRESSABLE POINT - punkt adresowalny,
ADDRESSING - adresowanie,
ADDRESSING CIRCUIT - układ wybierania adresu,
ADDRESSING METHOD - metoda adresowania,
ADDRESSING MODE - tryb adresowania,
ADDRESSING SYSTEM - system adresowania,
ADDRESSING TECHNIQUE - metoda adresowania,
ADDRESSLESS INSTRUCTION - rozkaz bezadresowy,
ADEQATE ACCURACY - wymagana dokładność, wierność,
ADEQUATE - adekwatny, odpowiedni, stosowny, wystarczający,
ADEQUATE TOOL - odpowiednie narzędzie,
ADJACENT - przyległy,
ADJACENT PITCH ERROR - błąd podziałki pojedyńczej,
ADJOINT - sprzężony,
ADJUGATE - sprzężony,
ADJUST - dobierać, korygować, regulować,
ADJUSTMENT - poprawianie, poprawka, porządkowanie, regulowanie,
ADLC - patrz: ADWANCED DATA LINK CONTROLLER,
ADMISSIBLE - dopuszczalny,
ADMISSIBLE DECISION - decyzja dopuszczalna,
ADMISSIBLE ERROR - błąd dopuszczalny,
ADMISSIBLE NUMBER - liczba dopuszczalna,
ADOPTION - przyjęcie (np. metody),
ADP - patrz: AUTOMATIC DATA PROCESSING
ADRESS CONSTANT - adres podstawowy, adres początkowy
ADVANCE - postęp, także: przesuwanie się papieru, karty (w maszynie),
ADVANCED - rozwinięty, udoskonalony, zaawansowany (np. system),
ADVANCING - przesuwający,
ADVENTURE GAME - gra przygodowa,
ADVICE - zawiadomienie, rada, także: wiadomość, informacja,
ADWANCED DATA LINK CONTROLLER (ADLC) - rozwinięty sterownik łącza danych,
ADWANCED FEATURE - także: wersja rozszerzona,
AFTER-IMAGE PROTECTION - zabezpieczenie obrazu końcowego (dot. zabezpieczania danych),
AFTERGLOW - poświata (ekranu luminescencyjnego),
AGEING - starzenie się (np. maszyny),
AGENT'S SET - stanowisko księgowania,
AGGREGATE - agregować, gromadzić, zbierać, także: globalny, łączny, sumaryczny,
AGGREGATE CHART - zestawienie, tabela (statystyczna)
AIKEN CODE - kod AIKENA,
AIMING CIRCLE - plamka kierunkowa (służy do dokładnego ustawienia pióra świetlnego),
AIMING FIELD - plamka kierunkowa (służy do dokładnego ustawienia pióra świetlnego),
AIMING SYMBOL - plamka kierunkowa (służy do dokładnego ustawienia pióra świetlnego),
AIR CONDITIONED - klimatyzowany,
AIR CONDITIONING - klimatyzacja,
AL - patrz: AUTOLOAD,
ALARM SIGNAL DEVICE - urządzenie alarmowe w komputerze,
ALCOR-CODE - kod ALCOR,
ALE - patrz: ADDRESS LATCH ENABLE,
ALGEBRAIC ADDER - sumator algebraiczny,
ALGEBRAICAL ADDITION - dodawanie algebraiczne,
ALGORITHM - algorytm,
ALGORITHMIZATION - algorytmizacja,
ALIAS - nazwa umowna, pseudonim,
ALIASING - utożsamianie,
ALIGN - wyrównywać (np. papier w maszynie), także: wyrównywać w pionie w stosunku do przecinka,
ALIGNMENT - wyrównanie,
ALIGNMENT OF RADIX POINTS - wyrównanie przecinków zgodnie z podstawą liczenia,
ALL-INSULATED - łącznik całkowicie izolowany,
ALL POINTS ADDRESSABLE MODE (APA) - tryb adresacji z dokładnością do punktu,
ALL-OR-NOTHING RELAY - przekaźnik dwupołożeniowy,
ALL-OR-PART SWITCH - przełącznik dwugrupowy,
ALLOCATE - przydzielać, przyporządkowywać, rezerwować,
ALLOCATION - przydział, przyporządkowanie, rezerwowanie,
ALLOCATION COUNTER - wskaźnik adresu,
ALLOT - przydzielać,
ALLOWABLE - dopuszczalny,
ALLOWANCE - przydział, także: przydział dodatkowy,
ALLOY JUNCTION - złącze stopowe,
ALPHABET - alfabet,
ALPHABETIC - alfabetyczny,
ALPHABETIC SYMBOL - symbol literowy,
ALPHANUMERIC - alfanumeryczny,
ALPHANUMERIC CARD PUNCH - dziurkarka kart alfanumeryczna,
ALPHANUMERIC CODE - kod alfanumeryczny,
ALPHANUMERIC DISPLAY - alfaskop,
ALPHANUMERIC LITERAL - literał alfanumeryczny,
ALTER - zmieniać, modyfikować (np. program),
ALTERING - dokonywanie zmian,
ALTERNATE - zmieniać,
ALTERNATE TRACK - ścieżka zapasowa (zapisu w pamięci magnetycznej),
ALTERNATING CURRENT - prąd przemienny,
ALTERNATING OPERATION - transmisja naprzemienna, transmisja półdupleksowa,
ALTERNATION - suma logiczna, dysjunkcja,
ALTERNATION OF THE LIST OF INSTUCTIONS - zmiana ciągu rozkazów,
ALTERNATIVE - alternatywa, możliwość, wariant,
ALU - patrz: ARITHMETIC-LOGIC UNIT,
AMBIGUITY - dwuznaczność, możliwość wyboru dwóch rozwiązań,
AMBIGUOUS FILE NAME - niejednoznaczna nazwa pliku (zbioru),
AMEND - poprawiać, wnosić poprawkę, poprawka,
AMENDMENT - poprawa, naprawa (błędu),
AMPERSAND - znak "AND" (handlowe I), znak &,
AMPLIFICATION - zwiększenie, wzmocnienie,
AMPLIFICATION FACTOR - współczynnik wzmocnienia,
AMPLIFIER - wzmacniacz, wzmacniak,
AMPLIFIER ELEMENT - człon wzmacniający,
AMPLIFIER GAIN - wzmocnienie wzmacniacza, współczynnik wzmocnienia,
AMPLIFIER STAGE - stopień wzmacniacza,
AMPLIFY - zwiększać, wzmacniać,
AMPLIFYING WINDING - uzwojenie wzmacniające,
AMPLITUDE ADDER - sumator amplitudowy,
AMPLITUDE OF A COMPLEX NUMBER - argument liczby zespolonej,
AMSTRAD DISC OPERATING SYSTEM - Dyskowy System Operacyjny firmy AMSTRAD,
ANALOG - analogowy,
ANALOG ADDER - sumator analogowy,
ANALOG COMPUTER - komputer analogowy,
ANALOG DATA - dane analogowe,
ANALOG-DIGITAL COMPUTER - komputer hybrydowy,
ANALOG-DIGITAL CONVERSION - konwersja analogowo-cyfrowa,
ANALOG PLOTTER - ploter analogowy,
ANALOG SIMULATION - modelowanie analogowe,
ANALOG TO DIGITAL CONVERSION - przetwarzanie analogowo-cyfrowe,
ANALOG TO DIGTAL - przetwarzanie sygnału analogowego na cyfrowy,
ANALOG-TO-DIGITAL CONVERTER - konwerter analogowo-cyfrowy, przetwornik analogowo-cyfrowy, konwerter A/C,
ANALOQUE - analogowy,
ANALYSING FILE ROUTINES - programy drukowania raportu o zawartości pliku (zbioru),
AND - operacja logiczna "I", operator boolowski,
AND CIRCUIT - układ "I", bramka "I", układ iloczynu logicznego,
AND-ELEMENT - element "I",
AND FUNCTION - funkcja "I", funkcja iloczynu logicznego,
AND GATE - bramka "I", układ "I", układ iloczynu logicznego,
AND OPERATION - iloczyn logiczny, koniunkcja,
ANIMATION - animacja,
ANNOTATION - uwaga, notatka,
ANSWERING - zgłoszenie,
ANTECEDENT - poprzednik,
ANTECEDENT DRIVEN - także: wnioskowanie uprzedzające (w systemach ekspertowych),
ANTI-INTERFERENCE DEVICE - urządzenie przeciwzakłóceniowe,
ANTICIPATING CONTROL - sterowanie wyprzedzające,
ANTIVALENCE - różnica symetryczna, "LUB" wykluczające,
APA - patrz: ALL POINTS ADDRESSABLE MODE,
APERTURE CARD - karta okienkowa, karta aperturowa, mikrokarta,
API - patrz: AUTOMATIC PROGRAM INTERRUPT,
APL (A PROGRAMMING LANGUAGE) - skrót nazwy języka programowania APL do zastosowań naukowych,
APOSTROPHE - apostrof,
APPARATUS - aparat, przyrząd, urządzenie,
APPARENT ERROR - błąd pozorny,
APPARENT VARIABLE - zmienna pozorna,
APPENDIX - aneks, dodatek,
APPLIANCE - przyrząd, urządzenie,

Pocztowa giełda

Rozwiązanie krzyżówki z nr. 4/87 „IKS-a". Hasło brzmi: ELWAT, ZAM, ODRA, KRAK — komputery

Bony pieniężne (1000 zł) wylosowali: Krzysztof Zieliński — Łódź, Zdzisław Ziółkowski — Łódź, Sławomir Sobczak — Częstochowa, Marian Śmigielski — Kęty, Adam Grodzki — Warszawa.

Nagrody książkowe otrzymują: Ryszard Leś — Gdańsk, Wojciech Wierzbicki — Katowice, Stanisław Błoński — Hyżne, Joanna Pasternak — Brzeg, Leszek Pilaczyński — Laski Wielkie.

Rozwiązanie krzyżówki z nr. 5/87 „IKS-a"

Hasło brzmi: „IKS" propagatorem wiedzy i techniki mikrokomputerowej

Bony pieniężne (1000 zł) wylosowali: Barbara Karłowska — Wrocław, Grzegorz Kowalski — Sandomierz, Robert Grochala — Rzeszów, Maria Gajewska — Szczecin — Piotr Jakubiec — Hrubieszów.

Nagrody książkowe otrzymują: Maria Konieczna — Leszno, Leszek Jerzewski — Łódź, Marek Olesz — Gdynia, Patrycja Błaszak — Częstochowa, Sławomir Bukowski — Pruszków, Michał Paśniewski — Stawiguda woj. olsztyńskie, Iwona Kostro — Białe Szczepanowice, Tomasz Tęcza — Jelenia Góra, Anna Wróbel — Sieradz, Elżbieta Pawelec — Przasnysz.

Rozwiązanie krzyżówki z nr. 6/87 „IKS-a". Hasło brzmi: W podróży z „IKS-em"

Bony pieniężne (1000 zł) wylosowali: Małgorzata Żylińska — Szczecin, Paweł Malarski — Warszawa, Piotr Hoppel — Poznań, Karol Klepacz — Łódź, Grażyna Kijowska — Szczecin.

Nagrody książkowe otrzymują: Marek Szymocha — Poręba, Wiesława Laskowska — Poznań, Elżbieta Marlewska — Gdańsk, Ewa Tiahybok — Kraków-N.Huta, Tomasz Smoleński — Szczecin, Andrzej Ryczkiewicz — Rawa Mazowiecka, Ireneusz Grzybek — Wałbrzych, Krzysztof Gajda — Kraków, Mariusz Gruszewski — Dębica, Tomasz Sobczak — Żyrardów.

Sharp MZ-800: wymienię programy, literaturę. Zbigniew Zając, ul. 1 Maja 13 37-500 Jarosław.

Atari, Commodore, Spectrum — sprzedaż, wymiana literatury, także po polsku. Informacje — znaczki 30 zł. Klub MIKROPOL, 86-300 Grudziądz 1, skryt- ...3 49

Kupię układ scalony ROM 8 kB, 2364 CT 2, 27-0152-03-01 rok produkcji 1983. Piotr Elwart. 80-287 Gdańsk-Wrzeszcz, ul. Migowska 18/2

Poszukuję zeszytów „Wszystko o komputerze" nr 3, 4, 5, 6. Artur Żabecki 33-300 Nowy Sącz, ul. Klasztorna 28/99

Wymienię programy na Commodore C-64. Bartosz Muszyński 75-519 Koszalin, ul. Hanki Sawickiej 166/3, tel 521-11

Mikrokomputer Atari 800 XL — ponad 100 programów, w tym bardzo ciekawe gry — wymienię lub tanio odstąpię Radosław Wójtowicz 55-035 Oborniki Śl ul F. Dzierżyńskiego 3/2

LIGA MYŚLĄCYCH

Zadanie 1

Nazwijmy liczbą symetryczną taką liczbę naturalną, której cyfry stojące na miejscach pierwszym i ostatnim; drugim i przedostatnim itd. są takie same np. 3553, 474. Ile jest liczb symetrycznych trzycyfrowych, a ile liczb symetrycznych czterocyfrowych?

Zadanie 2

Znajdź trzy kolejne liczby naturalne o tej własności, że jeżeli kwadrat największej z nich podzielimy przez liczbę najmniejszą i od otrzymanego ilorazu odejmiemy 6, to otrzymamy liczbę większą od pozostałej z szukanych liczb?

Zadanie 3

W jakim układzie liczenia prawdziwa jest równość:

$$300 - 233 = 1$$

Zadanie 4

W jednej ze szkół podstawowych odbyły się zawody sportowe. W rywalizacji indywidualnej każdy uczestnik przeprowadził z każdym z pozostałych po 3 próby sprawności. Wszystkich prób było 135. Ilu uczniów brało udział w zawodach?

Zadanie 5

Grupa członków Warszawskiego Towarzystwa Cyklistów wybrała się na wielodniową wycieczkę rowerową. Zaplanowano przebycie na rowerach trasy 360 km, jadąc codziennie taką samą liczbę kilometrów. Faktycznie jednak grupa przejeżdżała każdego dnia o 4 km mniej i z tego powodu wycieczka przedłużyła się o 3 dni. Ile dni miała trwać wycieczka?

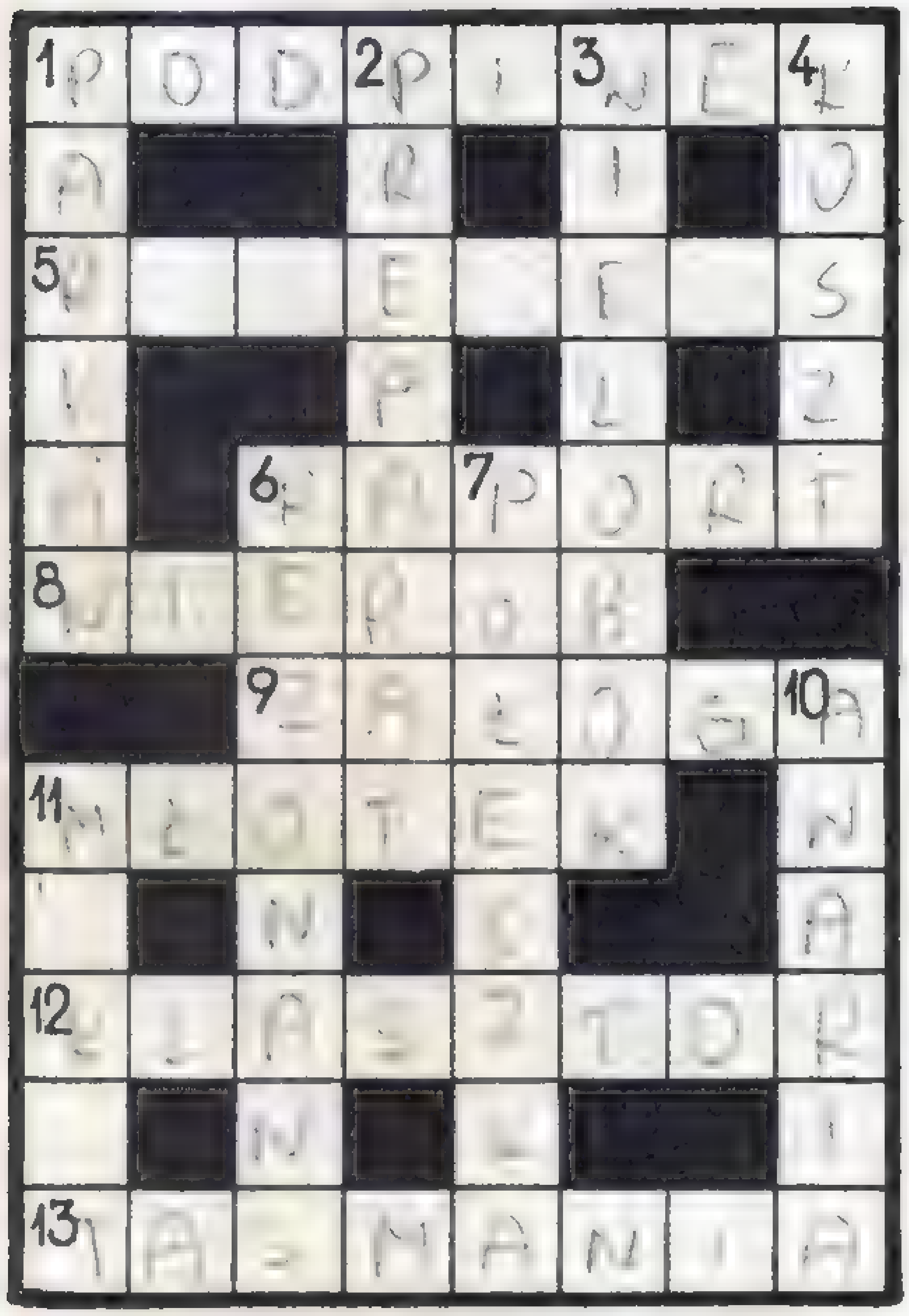
Rozwiązania zadań prosimy przesyłać do redakcji do końca listopada br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody-niespodzianki.

ZX — GEODEZJA

```
 100 INPUT PAPER 7; BRIGHT 1," M
AX.SZEROKOSC X ? ";ms: LET sks=2
52/ms
 110 INPUT PAPER 7; BRIGHT 1," M
AX.WYSOKOSC Y ? ",mw: LET skw=17
2/mw
 120 IF skw<=sks THEN LET skala=
skw
 130 IF skw>sks THEN LET skala=s
ks
 140 INPUT PAPER 7, BRIGHT 1," N
AZWA OBIEKTU ? "; LINE n$: PRINT
 TAB (32-LEN n$)/2, BRIGHT 1,n$
 150 FOR p=10 TO 250 STEP 10  PL
OT p,0  DRAW 0,2. NEXT p
 160 FOR r=10 TO 170 STEP 10: PL
OT 0,r: DRAW 2,0  NEXT r
 170 DIM x(50): DIM y(50)
 180 FOR n=1 TO 50
 190 INPUT PAPER 7,-BRIGHT 1,"X
<= ";INT (252/skala);" ";x(n)
 200 IF x(n)*skala>255 THEN BEEP
 1,-6: GO TO 190
 210 PRINT OVER 1,TAB 4;"X(",n,"
)=";  OVER 1;x(n);
 220 INPUT PAPER 7; BRIGHT 1;"Y
<= ";INT (172/skala);" ";y(n)
 230 IF y(n)*skala>175 THEN BEEP
 1,-6: GO TO 220
 240 PRINT OVER 1;TAB 20;"Y(",n;
")="; OVER 1;y(n);
 250 IF n=1 THEN PLOT 1+skala*x(
n),1+skala*y(n)
 260 IF n>1 THEN DRAW -skala*(x(
n-1)-x(n)),-skala*(y(n-1)-y(n))
 270 IF x(n)=x(1) AND y(n)=y(1)
AND n<>1 THEN GO TO 290
 280 NEXT n
 290 LET f=0: LET d=0
 310 FOR i=1 TO n-1
 320 LET f=f+(x(i)-x(i+1))*(y(i)
+y(i+1))/2
 330 LET d=d+SQR ((ABS (x(i)-x(i
+1))↑2)+(ABS (y(i)-y(i+1))↑2))
 340 NEXT i
 350 LET f=ABS f
 360 BEEP .3,26: PRINT OVER 1; I
NK 2'""     Pole wynosi ",f;" mA"
 370 PRINT OVER 1; INK 2;"    Obw
od wynosi ";d;" mb"
 380 FOR r=10 TO 170 STEP 10. PL
OT 0,r: DRAW 2,0: NEXT r
 390 INPUT PAPER 7; BRIGHT 1;" D
RUKOWAC T/N ? "; LINE d$: IF d$=
"t" OR d$="T" THEN COPY
 400 PRINT #0;AT 1,4; FLASH 1; B
RIGHT 1;"NACISNIJ DOWOLNY KLAWIS
Z" PAUSE 0  CLS : GO TO 0100
 410 REM [illegible]
 411 CLS : BEEP .2,14: BEEP .2,2
: BORDER 6: PLOT 26,20. DRAW 0,6
0: DRAW 130,-60  DRAW -130,0
 420 PAUSE 10: PLOT 240.165: DRA
W -60,0: DRAW 60,-155: DRAW 0,15
5
 430 PAUSE 10: PLOT 60,40: DRAW
140,0: DRAW 0,75  DRAW -140,-75
 440 PAUSE 10: PLOT 15,165: DRAW
 80,0: DRAW -80,-50: DRAW 0,50
 450 PAUSE 20: PRINT AT 7,4; INK
 3; BRIGHT 1; FLASH 1;" G E O D
E Z J A ": PAUSE 0: CLS
 460 FOR u=0 TO 7. READ v  POKE
USR "a"+u,v: NEXT u
 470 DATA 56.8,16,32.56,0,0,0
 480 REM a=A
 490 POKE 23609,25
 500 GO TO 100
9000 REM [illegible]
9001 SAVE "-Geodezja-" LINE 410:
 PRINT '"Verify  -Geodezja-". V
ERIFY "-Geodezja-"  GO TO 410
```

KUPON

KRZYŻÓWKA NR 9



POZIOMO: 1) piwo domowej roboty, 5) członek męskiego kwartetu wokalnego, 6) relacja, meldunek, 8) próżniak, wałkoń, 9) ogół pracowników zatrudnionych w jednostce produkcyjnej, 11) do wbijania gwoździ, 12) zgromadzenie zakonne; zakon, 13) górzysta wyspa koło Australii,

PIONOWO: 1) ogrodzenie z desek, 2) związek chemiczny lub ich mieszanina, przygotowane do jakiegoś specjalnego użytku, 3) wielokąt, 4) nakład pieniężny na określony cel, 6) fizyczne zjawisko związane z ruchem drgającym lub falowanym; współdrganie, 7) mała półka, 10) uszkodzenie, zepsucie się maszyny, urządzenia technicznego, 11) sportowa gra mieszana w tenisie.

Rozwiązania krzyżówki **na kartach pocztowych** należy nadsyłać pod adresem redakcji do końca listopada 1987 r., **naklejając kupon „IKS-a"**. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI



Kolejna wersja AMSTRADA

Amstrada PC 1640 ECD zbudowano na mikroprocesorze 8086. W porównaniu z poprzednikami ma znacznie zwiększone możliwości graficzne: dwa tryby rozdzielczości 320 na 200 lub 640 na 200 linii w 4 kolorach zapewniają dobrą jakość prac graficznych. Systemem operacyjnym PC 1640 jest MS DOS 3.2.



Rysunek z kolekcji GEM-u wykonany na zestawie firmy ICL:
komputer — DRS 300
drukarka mozaikowa ML 192

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk rotograwiurowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 9296. Nr ind. 361682.

K-97

INFORMATYKA KOMPUTERY SYSTEMY



Cena 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 11/1987 ISSN 0860—2794



— W szponach Atari
— Oszczędzajmy czas
— Mapa dysku

W NUMERZE:

Monitor zamiast bibliotekarki — str. 3
Mona Lisa i komputer — str. 4
W szponach Atari (9) — str. 5—9
Mikrokomputery IBM PC (5) — str. 13
Grafika na ekranie (2) — str. 18—19
Opowiadanie SF — str. 20
Pascal (7) — str. 24—25
Komputerowy test dla kierowców — str. 28
Liga Myślących — str. 31
Krzyżówka — str. 32

UWAGA. Brakujące numery „IKS-a" można jeszcze kupić w księgarniach Wydawnictwa MON, W-wa ul. Grzybowska 77.

W kioskach jest już 3 zeszyt programów. Tym razem przeznaczony jest dla wszystkich posiadaczy mikrokomputerów, które zbudowane zostały na mikroprocesorze Z80. Zeszyt zawiera konstrukcje programowe tego mikroprocesora i z pewnością przyda się tym, którzy chcą wykorzystać wszystkie właściwości swojego mikrokomputera.

Kolejny błąd ROM-u ZX Spectrum

Napiszemy program:

```
 1 CLEAR:REM kasowanie danych
10 LET X$ = „123"
20 PRINT X$
30 SAVE„proba": DATA x$ ()
50 LOAD „proba": DATA x$ ()
60 PRINT x$
```

W linii 60 wystąpi błąd. Spowodowany on jest pewnym przeoczeniem ROM-u ZX Spectrum

Na taśmie została zapisana zmienna tekstowa x$ jako tablica x$. Odczyt był identyczny. Spectrum potraktowało x$ jako tablicę. Ale przecież struktura zmiennej tekstowej różni się od struktury tablicy. Dlatego interpreter, odwołując się w linii 60 do tablicy x$, napotyka dziwne wymiany tablicy i uruchamia obsługę błędu.

Krzysztof POŹNIAK
Klub „HOBBYTE"

Czy przestajemy się interesować informatyką? Na ile stosunkowo łatwy dostęp do mikrokomputerów pozwolił na upowszechnienie dyscypliny, przed którą bez wątpienia jest jeszcze przyszłość?

Jestem gorącym zwolennikiem informatyki i wierzę, że dobrze oprogramowany komputer może pomóc nie tylko tam, gdzie trzeba przetworzyć ogromne ilości informacji, ale i tam, gdzie szybkość i niezawodność decydują o poprawności podjętej decyzji.

Nie łudźmy się jednak, że komputer pomoże na wszystko. To nieprawda. Komputer to urządzenie, które nie zrobi niczego, czego nie nauczył go człowiek. A jeśli nawet dobrze zaprojektowany system informacyjny działać będzie w instytucji, w której panuje organizacyjny bałagan, to efektem takiego eksperymentu będzie po prostu jeszcze większy bałagan. Wszystkim menedżerom przypominam, że jeśli trudno im utrzymać równowagę na rowerze, to z pewnością daleko nie pojadą motocyklem.

Niestety tego rodzaju próby już podejmowano, wydano bowiem kilka milionów, zaangażowano sporo pracowników do prac projektowych i ...wkrótce okazało się, że jest to zmarnowany wysiłek. Po nadziejach na postęp pozostał jedynie niesmak i pustka w kasie. Wielu młodych ludzi postąpiło podobnie (tylko Pewex sprzedał ponad sto tysięcy mikrokomputerów ATARI). Wydali spore przecież pieniądze na sprzęt, zainwestowali w programy, tymczasem efekty bywają czasem nikłe. Po okresie zabawy komputer wędruje na szafę, pudełko pokrywa warstwa kurzu i tylko czasami wraca do łask. Wiadomo jednak, że komputer nie tylko zmusza do nauki, ale także wymaga jednoznacznego formułowania problemów i kształci to, czego tak bardzo będzie potrzeba w dojrzałym już twórczym życiu zawodowym. I jeszcze raz powtórzmy: jeśli chcemy działać w przyszłości sprawnie, to nie odkładajmy komputerów na półki, nie rezygnujmy z niczego, co pomoże w przyszłości lepiej organizować nasze twórcze życie.

Wiesław CETERA

Trwa kolejna edycja Ligi Myślących. Miłośników zadań logicznych jest wielu — świadczą o tym nadchodzące odpowiedzi. Wiele z nich nosi znamię oryginalności, niektóre rozwiązywane są za pomocą komputerów. Najważniejszy jest jednak efekt.

Tymczasem w redakcji odbyło się wręczenie nagród dla zwycięzców pierwszej edycji Ligi. Dyplomy i nagrody dla najlepszych wręczył redaktor naczelny „Żołnierza Wolności" płk Zdzisław Janoś.

Zwycięzcą został Hubert Nowaczyk, kolejne miejsca zajęli Andrzej Erd, Antoni Goryl, Janusz Morbitzer i Andrzej Wrześniowski.



Monitor zamiast bibliotekarki

Według wielu specjalistów, społeczeństwo techniczne zostanie zastąpione społeczeństwem informatycznym. Produkcja i przetwarzanie informacji może wysunąć się na pierwsze miejsce przed produkcją towarów. W społeczeństwie amerykańskim blisko połowa czynnej zawodowo ludności pracuje w przemyśle informatycznym — wliczając w to ludzi produkujących komputery, pracujących przy „przenoszeniu" dysków i oczywiście informatyków. Powodem rozrostu przetwarzania danych stał się zalew informacji. Dziś nikt nie jest w stanie przeczytać samych tytułów artykułów ukazujących się w wąskiej dziedzinie. Szacuje się, że w samej medycynie rocznie ukazuje się ponad 20 tysięcy opracowań, pism specjalistycznych, popularnonaukowych, popularnych, biuletynów instytutów naukowych, prac doktorskich, habilitacyjnych, raportów rządowych, organizacji międzynarodowych itp. Nie ma mocnych, którzy byliby w stanie zapoznać się z tymi publikacjami, a przecież drukuje się je po to, by dzielić się doświadczeniami. Aby można było korzystać z już opublikowanego dorobku powstała nowa grupa zawodowa „przezuwacze wiedzy" — czyli specjaliści zajmujący się lekturą pism, książek itp., to co przeczytają (jeśli napisane jest na odpowiednim poziomie) opracowują pod kątem wprowadzenia do komputerowego banku danych.

O każdej wprowadzonej do banku pozycji przechowuje się kilkanaście bądź kilkadziesiąt informacji. Oprócz podstawowych: tytułu, autora, czasu, języka i miejsca publikacji, szczegółowo opisana jest jej tematyka. Tak, że użytkownik może najczęściej za pomocą odpowiedniego języka wyszukiwania informacji przeszukiwać bazę danych korzystając ze zdań składających się z kombinacji pojedynczych pytań. Daje to kolosalne oszczędności czasu, nie trzeba mozolnie przeszukiwać katalogu biblioteki. Na przykład przed napisaniem wypracowania o Mickiewiczu można sobie zażyczyć spisu opracowań o wieszczu (będzie składał się z kilkudziesięciu stron), z tego spisu (bibliografii) wybrać co lepszych autorów. Można sobie też zażyczyć bibliografię wypowiedzi Słowackiego o Mickiewiczu. Najlepsze efekty uzyskuje się w trakcie przeszukiwania bazy danych w systemie on-line (bezpośrednia konwersacja z komputerem). Użytkownik widzi na ekranie informacje przechowywane w bazie i może zmieniać kombinację pytań, dopóki nie otrzyma bibliografii, która go zadowoli. Problem polega między innymi na tym, że projektanci baz danych nie są w stanie przewidzieć wszystkich potrzeb użytkowników, w związku z czym, przy skomplikowanych i nietypowych kombinacjach pytań mogą powstać rozbieżności między oczekiwaną bibliografią a wydrukowaną przez system.

W pierwszym etapie objęto zautomatyzowanymi bazami danych pojedyncze biblioteki. Początkowo terminale stały gdzieś w holu, potem połączono je siecią transmisji danych z innymi bibliotekami, zaczęły powstawać specjalistyczne bazy danych. Jedna z baz chemicznych obejmuje ponad półtora miliona związków chemicznych nazw, wzorów i właściwości, baza uzupełniana jest co dwa tygodnie. Czy wyobrażacie sobie podobne wydawnictwo robione w sposób tradycyjny? Aktualnie istniejące bazy danych dostępne są za pośrednictwem sieci teletransmisji ze wszystkich kontynentów. W Polsce pierwszą biblioteką korzystającą ze zautomatyzowanego systemu informacji bibliotecznej była Główna Biblioteka Lekarska (GBL) w Warszawie. GBL podłączona jest do amerykańskiego systemu **Medline** przez europejski ośrodek systemu w Sztokholmie. Lekarze mogą korzystać z sześciu baz danych między innymi o onkologii, toksykologii (danych toksykologicznych ponad 55 tys. środków chemicznych). Z systemu korzystają pracownicy służby zdrowia — naukowcy i praktycy. Początkowe lęki przed nową techniką szybko mijają, gdy pozna się korzyści. System może wydrukować np. bibliografię wydawnictw w językach: polskim, niemieckim, rosyjskim na temat leczenia odmrożeń oraz oparzeń osób w wieku 75—80 lat, których przyczyną jest benzyna lotnicza (np. ofiar katastrofy lotniczej w górach czy w terenie podbiegunowym). Komputer odszuka wszystkie opracowania, w których wspomniano o podobnych problemach. Z tego spisu wybiera się następnie interesujące pozycje — książki można wypożyczyć, krótsze artykuły skopiować. Lekarz w najmniejszej miejscowości może za pośrednictwem GBL korzystać z opracowań publikowanych na świecie. Teoretycznie nie ma przeszkód w śledzeniu publikacji na całym globie, niestety korzysta się z systemu odpłatnie i to jest jedynym ograniczeniem.

Osoby, które miały kontakt z komputerem zdają sobie sprawę, że przy zarządzaniu tak dużą bazą danych musi pracować komputer o dużej szybkości działania, z olbrzymią pamięcią masową, że muszą być zainstalowane sprawne łącza telekomunikacyjne. W naszych warunkach są to rzeczy nieosiągalne, możemy jedynie stosować rozwiązania cząstkowe, wykorzystujące istniejący sprzęt i doświadczenie ludzi. Duże biblioteki wprowadzają zautomatyzowane systemy informacji bibliotecznej. W GBL dzięki automatyzacji szybciej opracowuje się Polską Bibliografię Lekarską, w Bibliotece Politechniki Warszawskiej opracowany został system przechowujący informację o materiałach z konferencji i sympozjów naukowych (dlatego, że najnowsze odkrycia omawiane są na konferencjach, dopiero potem trafiają do pism specjalistycznych). Biblioteka Uniwersytetu Warszawskiego przygotowuje się do zautomatyzowania informacji o księgozbiorze dydaktycznym, kilka systemów uruchomiono w Bibliotece Politechniki Wrocławskiej, również Centralna Biblioteka Wojskowa zamierza objąć swoje zbiory systemem informatycznym.

Czytelnikom pozostaje nadzieja, że nie trzeba będzie długo czekać na terminale w bibliotekach, że katalogi zostaną zastąpione przez monitory.

Dariusz OTTO

Dysk optyczny do IBM PC

Po latach badań nad stosowaniem nowych technologii w informatyce, na półkach sklepowych pojawiły się dyski optyczne. Jak podaje lipcowy „Byte" za 2795 dolarów użytkownik IBM PC i kompatybilnych z nim mikrokomputerów może kupić stację dysków optycznych WC 525 produkowanych przez Information Storage Inc. wielkości pudełka do butów (męskich sandałów). O dwieście dolarów tańszy jest model montowany wewnątrz komputera o rozmiarach stacji miękkich dysków.

Obecnie produkowane dyski optyczne dzielą się na trzy rodzaje: jednokrotnego zapisu, CD-ROM oraz dyski wielokrotnego zapisu. Wszystkie one mają wspólną cechę — odczyt znaków czy wgłębień dokonywany jest przez laser tak precyzyjnie ustawiony, że każde zgrubienie powierzchni, każdy pyłek utrudnia odczyt.

Jak do tej pory bariera cen powoduje, że przeciętni użytkownicy muszą zadowolić się dyskami jednokrotnego zapisu, takim jest też produkt ISI. Ten typ dysków nazywany jest WORM (write once, read many — raz zapisujesz, odczytujesz wiele razy), niekasowany, OPROM (optical programmable read only memory — pamięć optyczna zapisana, wielokrotnie odczytywana.

Dyski jednokrotnego zapisu produkowane są w rozmiarach od 5,25 cala do 14 cali średnicy, przechowują od 100 megabajtów od 1,8 gigabajtów na stronie. Ceny stacji dyskowych współpracujących z mikrokomputerami wahają się od 3000 do 30 tysięcy dolarów. WC 525 pracuje na dyskach o średnicy 5,25 cala, na jednej stronie mieści 115 megabajtów. Sprzedawane są również dyskietki 115-megabajtowe jednostronne za 100 dolarów oraz dwustronne o dwukrotnie większej pojemności za 125 dolarów. Dostęp do drugiej powierzchni uzyskuje się po odwróceniu dyskietki.

Zapis informacji następuje w wyniku wypalenia laserem otworów w powierzchni dyskietki zawierającej tellur, jest to proces nieodwracalny. Według ISI ten sposób przechowywania informacji zapewnia jej trwałość przez 20 lat! Inne firmy produkujące dyski optyczne przyjęły odmienne rozwiązania, w wyniku tego nie można stosować zamiennie dyskietek ISI, IBM czy Optotech. Dziennikarz „Byte'a", który przez dwa miesiące testował WC 525, stwierdził, że jest to dobry produkt, nie miał jednak możliwości sprawdzenia, czy dyskietki zapisywane w jednej stacji dysków mogą być odczytywane w innej.

Przepisywanie zbiorów z dysków magnetycznych na optyczne również wymaga wcześniejszej modyfikacji, resztę wykonują instrukcje dostarczone przez ISI. Producent zadbał o to, by użytkownik kupując WC 525 mógł korzystać z najpopularniejszego systemu operacyjnego na IBM PC — DOS.

Na podstawie „Byte'a"

opracował D. O.

Mona Lisa i komputer

Mówiąc najprościej, przetwarzanie obrazów oznacza „manipulowanie" na obrazie, czego przykładem jest np. zmiana kontrastu na ekranie odbiornika telewizyjnego. Gdy dysponujemy cyfrowym odwzorowaniem postaci obrazu — komputery dostarczają nam niemalże nieograniczonych możliwości zmiany tej postaci informacji. Wykorzystanie do tego celu maszyn uwarunkowane jest dostępem do technologii pozwalającej „zamienić" obraz na wspomniane „cyfrowe odwzorowanie" — szereg liczb reprezentujących stopień szarości, barwę, jej nasycenie w każdym wyróżnianym przez komputer umownym punkcie obrazu (pikselu). Manipulowanie na tych informacjach (w pamięci maszyny) pozwala „udoskonalać" obrazy, odkrywać w nich nowe informacje, a nawet wskazywać (sugerować) poprawną ich interpretację.

Jest to pierwszy krok w rozwoju maszyn, by mogły „widzieć". Początkowo przetwarzanie obrazów było marginesem praktycznych zastosowań informatyki — dużych wówczas i drogich komputerów. Obecnie — w dobie mikrokomputerów — okazało się to bardzo ciekawym przedmiotem zainteresowań twórców nowoczesnej informatyki. Podobnie jak przetwarzanie liczb i przetwarzanie tekstów — przetwarzanie obrazów, to kolejny etap w upowszechnianiu zastosowań komputerów. Wielu uważa ten etap za najbardziej znaczący w historii elektronicznego przetwarzania informacji.

Bardzo ciekawym kierunkiem tych nowych możliwości komputerów jest wspomaganie pracy konserwatorów dzieł sztuki.

Większość artystycznej spuścizny, którą ulokowano w muzeach, to „sztuka wizualna" typu: malowidła, rysunki, druki i pismo klinowe oraz trójwymiarowe obiekty. Niestety, bardzo mało z tych dzieł wygląda dzisiaj tak jak wyglądało bezpośrednio po ich stworzeniu. Temperatura i światło spowodowały wyblaknięcie farb i ściemnienie lakierów. Swój niszczycielski wpływ ma także powietrze, które na skutek reakcji chemicznych powoduje często nieodwracalne zmiany obrazów, rzeźb i innych dzieł sztuki — doświadcza tego na co dzień nasz Kraków. Dzieła zniszczenia dopełnia przypadkowa a czasem i rozmyślna działalność człowieka. Często na przykład artyści lub konserwatorzy obrazów pokrywali istniejące obrazy „nowym dziełem" lub je „twórczo" poprawiali.

Rozwój techniki ostatnich kilku dekad zrewolucjonizował czynności związane z konserwacją dzieł sztuki, będących przedmiotem zainteresowań archeologów, historyków sztuki i antropologów. Szczególnie przydatne okazały się, nie niszczące badanego obiektu promienie Roentgena (x). Uznanie zdobyło także określanie wieku dzieła na podstawie okresu rozpadu węgla, zawartego w tworzących obiekt składnikach (np. w farbie). Techniki te szybko wzbogacono o wiele innych — między innymi o fotoakustyczną spektroskopię, termoluminescencję, badania laserowe. W efekcie można uzyskać bardzo dużą liczbę obiektywnych danych o konserwowanym bądź badanym obiekcie.

W restauracji dzieł sztuki innowacje te pomogły między innymi odróżniać autentyczne fragmenty przedmiotów artystycznych (by je zachować) od materiałów konserwatorskich (by je usunąć). Metody te są źródłem obiektywnych danych, mówiących o postaci dzieła, o materiale, z jakiego jest stworzone, bez względu na jego formę i treść. Najczęściej jednak, właśnie forma i treść okazują się jedynym kluczem do poprawnej analizy całego dzieła. Ocena i zrozumienie tych bardzo „esencjonalnych" charakterystyk decyduje często o poprawności zabiegów konserwatorskich.

Restauracja obrazów opiera się na ludzkich oczach — wykorzystuje mikroskop optyczny, kamerę — częściej niż promienie X. Stąd też wizualna ocena konserwowanego dzieła przesądza o powodzeniu tego typu prac.

Wspomniany na wstępie dostęp do cyfrowej postaci obrazów znacznie rozszerzył zakres zastosowań sprzętu informatycznego — od astronomii przez diagnostykę medyczną do konserwacji dzieł sztuki.

Niespodziewanym „pośrednikiem" między nowoczesnymi technologiami komputerowymi a sztuką stał się Leonardo da Vinci. Wśród bardzo wielu ciekawych zdarzeń w jego bogatym życiu były także takie, w których nieporozumienia i zatargi ze zleceniodawcą nie pozwalały mu kończyć swych prac. Działo się to w wypadku „Bitwy pod Anghiari". Giorgio Vasari zamiast tego obrazu (nie dokończonego) stworzył własne dzieło o podobnej treści. Dzisiaj powstał problem, czy można odtworzyć to, czego dokonał Mistrz Leonardo?

W 1975 roku temat ten podjął J. F. Asmus — zdecydowały o tym wiarygodne przesłanki o niezniszczeniu (tylko zamalowaniu) przez Vasariego fragmentów malowidła Leonardo da Vinci. Sprawę komplikował olbrzymi wymiar dzieła, którego zaledwie kilka niewielkich fragmentów wykonał sam Mistrz. W pracach tych badacze wykorzystali termowizję, promienie Roentgena, aktywność neutronową, przewodnictwo elektryczne i mikrofale. Ostatecznie wybrano ultradźwiękową echolokację (urządzenie podobne do sonaru — hydrolokatora). Technika ta spełniła oczekiwania badaczy i pozwoliła wyróżnić fragmenty wykonane przez Leonarda da Vinci.

Duża część zabiegów konserwatorskich polega na sprawdzaniu i interpretacji fotografii i radiografów. Często promienie X są wykorzystywane do odsłonięcia wcześniejszych kompozycji ukrytych pod aktualną powłoką — innym obrazem. Generalnie obrazy widziane tą techniką bywają mieszaniną powierzchniowego obrazu, „poprzedniego" obrazu oraz struktury nośnika (drewno lub płótno). Po wielu badaniach okazało się, że „czyszczenie" tak otrzymanych zdjęć jest możliwe dzięki pewnym operacjom matematycznym na cyfrowych odwzorowaniach obrazów (stosując „filtrowanie" danych o obrazie szybką transformatą Fouriera). Usunięto w ten sposób interferencję tła, które daje nośnik obrazu — płótno lub drewno. Rezultaty okazały się bardzo dobre — „szarość" zamazująca obraz została wyeliminowana.

Przedstawione uwagi dotyczą rycin „kreskówek" lub obrazów monochromatycznych. W konserwacji obrazów jest to zaledwie szczyt góry lodowej, zdecydowana większość dzieł zgromadzonych w muzeach jest bowiem polichromatyczna. Obraz tworzą nie tylko trzy składowe (czerwień, zieleń i kolor niebieski), ale także ich nasycenie, odcień. Duże znaczenie ma również wizualna percepcja obrazu, której mechanizmy są niezmiernie złożone — decydują o tym, że zauważamy piękno lub kiczowatość obrazu.



Prawdopodobnie jedynym z najsłynniejszych obrazów jest Mona Lisa (La Gioconda) Leonarda da Vinci (eksponowana w paryskim Luwrze). Obecnie obraz ten jest ledwo cieniem swej oryginalnej postaci. Przyczyna? Czas i... niestety, zabiegi wielu konserwatorów. Przystępując do konserwacji obrazu jego wysokiej jakości fotografia została „zamieniona" („digityzowana") na liczby (dla każdego z 6 milionów wyróżnionych punktów i dla każdego z trzech podstawowych kolorów).

Wstępne przetwarzanie danych odzwierciedlających postać obrazu zmieniło kolor nieba z brązowego na niebieski, skóry niewiasty z żółtego na alabastrowy, a z sukni bohaterki dzieła wydobyto odcień głębokiej, pięknej zieleni.

Do dalszych, bardziej złożonych badań, wykorzystano zdjęcia (10 • 12,5 cm — piksele o wymiarze 0,02 mm²) fragmentów obrazu. W efekcie otrzymano trzy zbiory danych dla każdego z podstawowych kolorów (RGB). Każdy zbiór zawierał dane o 2600 liniach, po 2200 pikseli w jednej linii. Każdy piksel opisywany był wartością od 0 do 255 — w zależności od jasności każdego z trzech podstawowych kolorów. Zero prezentowało czerń, a 255 maksymalną jasność koloru. Przeprowadzone następnie obliczenia pozwoliły na dokonywanie korekt kolorów. W efekcie każdy niebieski piksel był „dzielony" przez 0,4, czerwony przez 0,2, a czerwony przez 1,0.

Większość profesjonalnych artystów, oglądających wynik tej „korekty", zgodziło się z jej rezultatami. Uznali „poprawione" kolory za bardziej naturalne. Wiek obrazu został „skompensowany".

Zwrócono także uwagę na odblaski („świecenie"), które zredukowano (stosując także szybką transformatę Fouriera).

Po tych komputerowych zabiegach dostrzeżono wiele nowych detali obrazu Mony Lisy, na przykład fragmenty naszyjnika? Nasunęło to wątpliwości, czy są one dziełem Leonarda, restauratorów, czy po prostu są to zniszczenia?

Dokonano jeszcze wielu innych obliczeń eliminujących lub wskazujących krytyczne miejsca obrazu. Stanowiło to między innymi wskazówkę (naszyjnik) prawdopodobnego pierwowzoru Mony Lisy — La Muta Raphaela.

Coraz częściej w muzeach świata wykorzystywane są komputery do wspomagania zabiegów konserwatorskich, tym bardziej że umożliwia to powszechnie dostępny sprzęt nawet już klasy IBM PC XT (rozdzielczość 1024x1024).

Kończąc warto podkreślić problemy psychologicznych oporów przy stosowaniu maszyn w procesach konserwatorskich. Dotyczy to zarówno profesjonalistów, jak i generalnie publiczności zwiedzającej muzea.

Po przedstawieniu omawianych efektów „korekty" Mony Lisy, pewna mistrzyni „sztuki konserwatorskiej", stwierdziła, że „nigdy mnie nie przekonacie, że Leonardo namalował te drzewa **na błękitnym tle**". Malowidła są w rzeczywistości górami, zniekształcenia kolorów mylą bowiem nawet wybitnych profesjonalistów w poprawnej interpretacji obrazów.

W.G.

Opracowano na podstawie:
J. F. Asmus, Digital Image Processing in Art Conservation, BYTE, march 1987.

Atari i dźwięk

Tomasz MROWIEC
Ludwik PIELA

Komputery Atari serii XL i XE, jak wszystkie komputery domowe, mają możliwość generowania dźwięku. Dysponują czterema niezależnie sterowanymi kanałami, które mogą być używane jednocześnie. Instrukcja SOUND języka BASIC, za pomocą której wybieramy numer kanału, wysokość dźwięku, brzmienie i głośność, w niewielkim tylko stopniu umożliwia wykorzystanie wszystkich zalet układu generowania dźwięku. Niezbędne jest dokładne poznanie tego układu i różnych sposobów jego wysterowania.

W poniższym artykule spróbujemy zapoznać Czytelników z możliwościami układu generowania dźwięku oraz sposobami sterowania nim. Zaprezentujemy programy umożliwiające praktyczne zbadanie większości tych możliwości. Należy jednak pamiętać, że sterowanie dźwiękiem jest skomplikowane, trudne do badania i wymaga konstruowania dosyć rozbudowanych programów.

Do generowania dźwięku przeznaczony jest układ scalony POKEY, który obsługuje także szynę szeregową wejścia/wyjścia oraz klawiaturę. Ma on kilka rejestrów (w pamięci RAM), które są używane podczas generowania dźwięku. Ich nazwy, adresy oraz przeznaczenie są następujące:

AUDF 1 (53760) — rejestr częstotliwości kanału 1
AUDC1 (53761) — rejestr sterujący kanału 1
AUDF2 (53762) — rejestr częstotliwości kanału 2
AUDC2 (53763) — rejestr sterujący kanału 2
AUDF3 (53764) — rejestr częstotliwości kanału 3
AUDC3 (53765) — rejestr sterujący kanału 3
AUDF4 (53766) — rejestr częstotliwości kanału 4
AUDC4 (53767) — rejestr sterujący kanału 4
AUDCTL (53768) — rejestr sterujący trybem działania

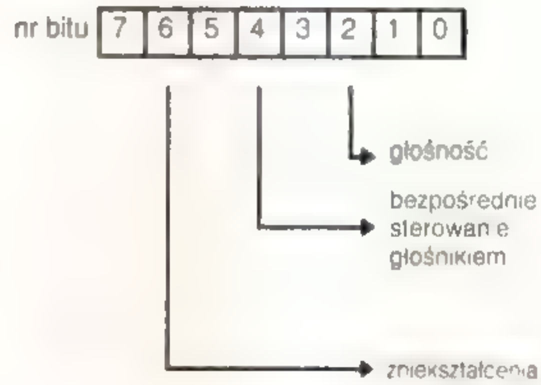
Aby poprawnie działał układ POKEY musi być zainicjowany. Wymagane to jest po dowolnej operacji na szynie szeregowej (współpraca z magnetofonem, stacją dysków lub drukarką). W celu zainicjowania POKEY w BASIC'u należy wykonać zerową instrukcję SOUND, to jest SOUND 0,0,0,0. W języku maszynowym należy zapamiętać 0 w AUDCTL ($D208 = 53768) i 3 w SKCTL ($D207 = 53775).

Każdy kanał dźwiękowy ma odpowiadający mu rejestr częstotliwości AUDF1—4, który steruje wysokością generowanego dźwięku. Rejestr częstotliwości, o długości 1 bajta, zawiera liczbę N używaną do sterowania układami dzielników częstotliwości, w celu wytworzenia różnych częstotliwości wyjściowych. Im większa jest wartość N, tym rzadsze są impulsy wyjściowe i tym niższa częstotliwość dźwięku, która może być określona następującym wzorem:

$$fwy = fz/2 \times (N + 1)$$

gdzie: fz — jest częstotliwością zegara, a N zawartością rejestru częstotliwości. Normalną częstotliwością zegara 64 kHz (a dokładniej 63921 Hz) dysponujemy zakresem częstotliwości od 125 Hz do 32 kHz.

Każdy kanał ma także rejestr sterujący AUDC1—4 umożliwiający ustawienie głośności (cztery młodsze bity) i brzmienia dźwięku (zawartości zniekształceń).
Przydział bitów tego rejestru jest następujący:



Sterowanie głośnością każdego kanału dźwiękowego jest proste. Młodsze bity rejestru sterującego zawierają liczbę czterobitową, która określa głośność dźwięku. Istnieje zatem 16 różnych ustawień głośności. Zera na wszystkich bitach oznaczają głośność zerową, a 15 oznacza maksymalną. Suma głośności czterech kanałów nie powinna przewyższać 32, ponieważ może spowodować przesterowanie wyjścia dźwiękowego.

Każdy kanał dźwiękowy ma w swoim rejestrze sterującym wyróżnione trzy bity. Służą one do sterowania zniekształceniem dźwięku używanym do tworzenia specjalnych efektów akustycznych. Dzięki tej możliwości łatwo jest uzyskać ogromną różnorodność dźwięków, od huków, szczęków i skrzeków do trzasków i szmerów.

Zniekształcenie nie polega na zmianie kształtu fali (jest ona zawsze prostokątna), lecz na usuwaniu z ciągu wybranych impulsów. Aby w pełni wyjaśnić na czym polega zniekształcenie dźwięku, musimy zrozumieć działanie tak zwanych liczników wielomianowych (*polynomial counters*). Są one używane jako źródło impulsów losowych stosowanych w generowaniu zakłóceń. Liczniki te wykorzystują rejestry przesuwne pracujące z częstotliwością 1,79 MHz. Zawartość rejestru jest przesuwana w prawo, każdy wychodzący bit poddawany jest pewnej obróbce i wprowadzany ponownie do rejestru. Wytwarza to niby- losowy ciąg bitów na wyjściu rejestru przesuwnego.

Liczniki te nie są w pełni losowe, ponieważ po pewnym czasie powtarzają sekwencje bitów. Jak można oczekiwać, okres powtarzania zależy od długości rejestru przesuwnego, to znaczy im dłuższy rejestr, tym dłuższy okres powtarzania.

Zakłócenie dźwięku w komputerach Atari uzyskuje się poprzez wprowadzenie impulsów z liczników wielomianowych i impulsów podstawowych do układu wyboru. W rzeczywistości jest to komparator cyfrowy, który przepuszcza na wyjście tylko te sygnały, które są zgodne z impulsami losowymi. Efekt końcowy polega na losowym eliminowaniu pewnych impulsów z układu dzielnika częstotliwości.

Jak wspomnieliśmy impulsy wyjściowe liczników wielomianowych mają charakter cykliczny. W

Nuta	Oktawa																	
	1		2		3		4		5		6		7		8		9	
	16-b	8-b	16-b	8-b	16 b	8-b	16-b	8-b	16-b	8-b	16-b.	8-b	16-b	8-b	16-b	8-b	16-b	8-b
C	27357		13675		6834	243	3414	121	1703	60	848	30	421	14	207		100	
C	25821		12907		6450	230	3222	114	1607	57	800	28	397		195		94	
D	24372		12182		6088	217	3040	108	1517	53	755	26	374		183		88	
D	23003		11498		5746	204	2869	102	1431	50	712	25	353		173		83	
E	21712		10852		5423	193	2708	96	1350	47	672	23	332		163		78	
F	20493		10243		5118	182	2555	91	1274	45	634	22	313		153		73	
F	19342		9668		4830	172	2412	85	1202	42	598	21	295		144		69	
G	18256		9125		4559	162	2276	81	1134	40	564	19	278		136		64	
G	17231		8612		4303	153	2148	76	1070	37	532	18	262		128		60	
A	16264		8128		4061	144	2027	72	1010	35	501	17	247		120		57	
A	15351		7672		3832	136	1913	68	953	33	473	16	233		113		53	
H	14489		7241		3617	128	1805	64	899	31	446	15	219		106		50	

Uwaga: 1 — kontra, 2 — wielka, 3 — mała, 4 — razkreślna, 5 — dwukreślna, 6 — trzykreślna, 7 — czterokreślna, 8 — pięciokreślna, 9 — sześciokreślna

rezultacie zakłócony dźwięk także będzie miał taką samą postać powtarzania. Umożliwia to układowe generowanie takich dźwięków jak brzęczenie, warkot silnika i podobnych, mających charakter cykliczny.

Komputery Atari wyposażone są w trzy liczniki wielomianowe, o różnych długościach, które mogą być łączone na wiele sposobów w celu wytworzenia interesujących efektów dźwiękowych. Krótsze liczniki (o długości 4 i 5 bitów) powtarzają się wystarczająco często, aby wytworzyć dźwięki naśladujące warkot, które tak szybko narastają i opadają, podczas gdy dłuższy licznik (o długości 17 bitów) tak rzadko się powtarza, że żaden wzór zakłócenia nie może być wyraźnie rozpoznany. Licznik 17-bitowy może być używany do naśladowania dźwięków eksplozji i krzyku. Jest on na tyle nieregularny, że może służyć do generowania białego szumu.

Każdy kanał dźwiękowy oferuje sześć różnych kombinacji trzech liczników wielomianowych. Sterują tym trzy bity rejestrów AUDC 1—4 następująco

7	6	5	4	3	2	1	0

Bity 7 6 5	
0 0 0	5-bitowy i 17-bitowy
0 X 1	5-bitowy
0 1 0	5-bitowy i 4-bitowy
1 0 0	17-bitowy
1 X 1	bez liczników
1 1 0	4-bitowy

Każda kombinacja liczników oferuje unikalny dźwięk. Ponadto zniekształcone dźwięki mogą brzmieć różnie przy różnych częstotliwościach. Z tego powodu niezbędne jest wykonanie wielu prób w celu znalezienia kombinacji zniekształceń i częstotliwości, które dadzą pożądany efekt dźwiękowy. Poniżej prezentujemy tablicę, która ułatwi nam rozpoczęcie prób.

Bit 4 rejestruje AUDC 1—4 określa specjalny tryb pracy. Gdy jest on ustawiony, wartość głośności (bity 0—3 AUDC 1—4) wysyłana jest bezpośrednio do głośnika, nie jest on modulowany częstotliwością w rejestratorach AUDF 1—4.

Dla pełnego zrozumienia użycia tego trybu działania, musimy przypomnieć sobie jak pracuje głośnik i co się z nim dzieje po odebraniu impulsu. Każdy głośnik ma membranę, która przesuwa się. Pozycja membrany w dowolnej chwili jest wprost proporcjonalna do napięcia, które przychodzi z komputera. Jeśli wysyłane napięcie jest zerowe, wtedy głośnik jest w pozycji spoczynkowej. Kiedy membrana zmienia pozycję, wtedy przesuwa powietrze, co jest odbierane przez nasze ucho jako dźwięk.

Z definicji impulsu wiemy, że składa się on z narastającego napięcia, a następnie opadającego. Jeśli do głośnika wyślemy pojedynczy impuls, powstanie fala powietrza, która może być wykryta przez ucho jako trzask. Przykładem może być realizacja następujących instrukcji

POKE 53761,31 **POKE** 53761,16

Ciąg impulsów (lub fala) może wprawić membranę w stały ruch i słyszane będzie ciągłe brzęczenie lub dźwięk. Im szybciej wysyłane są impulsy, tym wyższy dźwięk. W taki sposób komputer wytwarza dźwięk w głośniku.

Należy zaznaczyć, że w trybie bezpośredniego sterowania głośnikiem, wysłana wartość głośności nie wraca automatycznie do zera, lecz pozostaje stała do czasu, aż zmieni ją program. Dla wytworzenia dźwięku głośność powinna być modulowana wystarczająco często. Spróbujemy teraz wykonać następujące instrukcje, uważnie nasłuchując po każdej

POKE 53761,31
POKE 53761,31

Za pierwszym razem usłyszymy trzask zgodnie z oczekiwaniem. Głośnik wysunie się i przesunie powietrze. Lecz za drugim razem nic się nie stanie. Membrana głośnika była już w pozycji wysuniętej, kolejny rozkaz wysunięcia niczego nie zmieni, zatem nic nie usłyszymy. Spróbujmy teraz

POKE 53761,16
POKE 53761,16

Tak jak poprzednio, za pierwszym razem usłyszymy trzask, gdy membrana przesuwa się do pozycji spoczynkowej, za drugim razem nic nie usłyszymy, ponieważ głośnik był już w pozycji spoczynkowej.

Jak widać, w tym trybie pracy program ma pełne sterowanie pozycją membrany głośnika w dowolnym czasie. Chociaż powyższe przykłady są tylko binarne (włączone lub wyłączone), nie jesteśmy niczym ograniczeni do tylko tego typu modulacji głośnika. Możemy ustawić membranę w dowolnej z 16 różnych pozycji.

Na przykład prosta fala trójkątna (podobna do fal wytwarzanych przez instrumenty dęte) może być utworzona poprzez wysłanie wartości głośności 8, a następnie 9, 10, 11, 10, 9, 8, 7, 6, 5, 6, 7, i znowu 8 i powtarzanie tego ciągu bardzo szybko. Poprzez wystarczająco szybką zmianę głośności może być utworzona dowolna postać fali. Przy pomocy tej techniki można łatwo uzyskać syntezę mowy. Wymaga to użycia języka asemblera.

Oprócz rejestrów niezależnego sterowania kanałami (AUDC1—4), istnieje bajt (AUDCTL) dotyczący wszystkich czterech kanałów. Każdy bit w AUDCTL przydzielony jest do określonej funkcji AUDCTL ($D208 = 53768).

Przed przystąpieniem do wyjaśniania opcji AUDCTL musimy wyjaśnić inne pojęcie taktowanie (ang. clocking). Do synchronizacji milionów operacji wewnętrznych, realizowanych w każdej sekundzie w komputerze, używany jest ciąg impulsów. Jest on generowany przez centralny zegar działający z częstotliwością 1,79 MHz. Istnieje także kilka zegarów pomocniczych o niższych częstotliwościach. Rejestr AUDCTL umożliwia nam wybranie częstotliwości impulsów wejściowych układu dzielenia przez N. Poprzez zmianę zegara drastycznie możemy zmienić częstotliwość wyjściową. Można to łatwo zilustrować prostym wzorem podawanym w niektórych podręcznikach (np. De Re Atari)

$$f_{wy} = \frac{f_{zeg}}{N}$$

gdzie: f_{wy} — częstotliwość wyjściowa
f_{zeg} — częstotliwość używanego zegara
N — zawartość rejestru AUDF1—4

W rzeczywistości zależność częstotliwości wyjściowej od częstotliwości zegara i zawartości rejestru AUDF1—4 ma nieco inną postać

$$f_{wy} = f_{zeg}/2 \times (N + 1)$$

Ustawienie bitu 1 rejestru AUDCTL przełącza zegar 64 kHz na 15 kHz. Podobnie poprzez ustawienie bitów 5 lub 6 możemy taktować, odpowiednio kanały 3 lub 1, częstotliwością 1,79 MHz. Otrzymamy wtedy dźwięk znacznie wyższy, co można zilustrować następującym przykładem

SOUND 0,255,10,8 — włączenie kanału, niski ton

POKE 53768,64 — ustawienie bitu 6 AUDCTL

Osiem bitów w rejestrach sterujących częstotliwością wydaje się oferować więcej niż wystarczającą rozdzielczość dla wybrania dowolnej częstotliwości. Jednak wprawne ucho może zauważyć, że generowane dźwięki, odpowiadające na przykład gamie muzycznej, nie są zbyt dostrojone i nie mogą być. Niech w rejestrze częstotliwości będzie liczba N = 72 (która według instrukcji daje nutę A), wtedy uzyskamy częstotliwość 437,8 Hz. Dla N = 71 otrzymamy 443,9 Hz. A częstotliwość nuty A jest 440 Hz i takiej częstotliwości nie możemy uzyskać.

Istnieją również sytuacje, w których każde ucho może usłyszeć, że ośmiobitowy rejestr częstotliwości jest niewystarczający. Spróbujmy wykonać następujące instrukcje

FOR I = 255 **TO** 0 **STEP** -1: **SOUND** 0,I,10,8: **NEXT** I

Dźwięk początkowo podnosi się łagodnie, lecz gdy zbliża się do końca swojego zakresu zauważalne są znaczne zmiany częstotliwości. Nie może być inaczej, jeśli dzielimy częstotliwość zegara przez coraz mniejsze liczby. 15 kHz podzielone przez 255 ma zbliżoną wartość, jak 15 kHz podzielone przez 254, lecz 15 kHz podzielone przez 2 znacznie różni się od 15 kHz podzielonych przez 1.

Z tego powodu efektywny zakres dla muzyki ograniczony jest do niepełnych czterech oktaw, ponieważ dokładność dostrojenia odtwarzanych nut staje się postępująco gorsza, gdy częstotliwość staje się wyższa. Przyczyną niedokładności dostrojenia jest ograniczona liczba wartości możliwych do użycia jako dzielniki (256 unikalnych wartości).

Na szczęście Atari dostarcza rozwiązanie tego problemu, dzielniki 16-bitowe, które umożliwiają wybranie 65536 różnych częstotliwości wyjściowych. Lecz dokładniejsze dostrojenie nie przychodzi za darmo. Szesnastobitowe dzielniki uzyskuje się poprzez połączenie rejestrów częstotliwości dwóch kanałów dźwiękowych (AUDF1 z

Głos	Instrukcja SOUND	Równoważne instrukcje POKE
1	SOUND 0, N, Z, G	POKE 53760, N: POKE 53761, 16 • Z + G
2	SOUND 1, N, Z, G	POKE 53762, N: POKE 53763, 16 • Z + G
3	SOUND 2, N, Z, G	POKE 53764, N: POKE 53765, 16 • Z + G
4	SOUND 3, N, Z, G	POKE 53766, N: POKE 53767, 16 • Z + G

```
FK 1 REM ****************
WZ 2 REM * Prosram nr 1 *
FM 3 REM ****************
AE 4 GRAPHICS 0:? :? :? :? "OD JA
   KIEJ CZESTOTLIWOSCI CHCESZ":?
   "ZACZAC ";:INPUT CZEST
AG 5 AUDF3=INT(63921/(2*CZEST)-1+
   0.5)
ZY 6 AUDF2=INT((1789790/(2*CZEST)
   -7)/256)
DG 7 AUDF1=INT((1789790/(2*CZEST)
   )-7-256*AUDF2+0.5)
NO 8 IF CZEST>125 AND CZEST<16000
    THEN 10
XQ 9 ? "Dopuszczalny zakres 125 d
   o 16000.":? "Sprobuj znowu.";:
   INPUT CZEST:GOTO 5
NJ 10 GRAPHICS 0:? "Manipulator #
   1 steruje czestotliwos-"
GZ 15 ? "cia o wysokiej rozdzielc
   zosci.":? "Nacisnij przycisk,
   aby otrzymac":? "dzwiek."
AZ 20 ? "Przesun w gore lub dol d
   la wolnej":? "zmiany czestotli
   wosci lub w lewo/":? "prawo dl
   a szybkiej zmiany."
LA 30 ? "Manipulator #2 steruje n
   iska roz-":? "dzielczoscia, pr
   zycisk daje dzwiek "
OO 40 ? "lub gora/dol dla zmiany.
   "
FH 55 SOUND 0,0,0,0:SOUND 1,0,0,0
   :SOUND 2,0,0,0:SOUND 3,0,0,0
JF 100 POKE 752,1
FZ 110 POKE 53768,80
UQ 130 POKE 53763,10*16+15
VU 140 POKE 53765,10*16+15
BO 160 IF STICK(0)=13 THEN AUDF1=
   AUDF1+1:IF AUDF1>255 THEN AUDF
   2=AUDF2+1:AUDF1=0
NY 170 IF STICK(0)=14 THEN AUDF1=
   AUDF1-1:IF AUDF1<0 THEN AUDF2=
   AUDF2-1:AUDF1=255
PK 180 IF STICK(0)=11 THEN AUDF2=
   AUDF2+1
UE 190 IF STICK(0)=7 THEN AUDF2=A
   UDF2-1
TJ 220 IF STICK(1)=13 THEN AUDF3=
   AUDF3+1
WX 230 IF STICK(1)=14 THEN AUDF3=
   AUDF3-1
XD 240 IF STRIG(0)=0 THEN POKE 53
   760,AUDF1:POKE 53762,AUDF2:GOT
   O 260
MY 250 POKE 53760,0:POKE 53762,0
XL 260 IF STRIG(1)=0 THEN POKE 53
   764,AUDF3:GOTO 280
UA 270 POKE 53764,0
UP 280 CZWY2=1789790/(2*(AUDF2*25
   6+AUDF1+7))
VP 290 CZWY3=63921/(2*(AUDF3+1))
ML 300 POSITION 1,10
ZO 310 ? :? "WYSOKA ROZDZIELCZOSC
   :":?
OR 320 ? "CZEST = 1789790/(2*(";A
   UDF2;"*256+";AUDF1;"+7))    ":?
UG 325 ? "          = ";CZWY2
SE 330 ? :? :? "NISKA ROZDZIELCZO
   SC:":?
VD 340 ? "CZEST = 63921/(2*(";AUD
   F3;"+1))       ":? :? "       =
   ";CZWY3
LK 500 GOTO 100
```

```
FL 2 REM ****************
XW 5 REM * Prosram nr 2 *
FP 6 REM ****************
HR 100 DATA 520,584,655,694,779,8
   74,982,1040
IF 110 DIM DZWIEK(8),A$(1)
TL 120 FOR I=1 TO 8:READ X:DZWIEK
   (I)=X:NEXT I
AE 150 GRAPHICS 0:? :? "   NORMAL
   NA           WYSOKA ROZDZIEL.":?
    " --------------         --------
   -------"
KY 160 ? "  P       NUTA          P1
    P2     NUTA":? " --.       ----
      --  --      -----"
JT 170 POKE 752,1
ZJ 200 REM GENEROWANIE 'NORMALNEG
   O' DZWIEKU ATARI
PV 210 FOR J=1 TO 8
GV 220 P=INT(63921/(2*DZWIEK(J))-
   1)
ZT 230 SOUND 2,P,10,8
DT 240 DZWIEK=63921/(2*(P+1))
LW 250 POSITION 3,J+6:? P;"   ";DZ
   WIEK
ZI 260 FOR W=1 TO 200:NEXT W:NEXT
    J
TG 270 SOUND 2,0,0,0:FOR W=1 TO 5
   00:NEXT W
YH 300 REM DZWIEK O WYSOKIEJ ROZD
   ZIELCZOSCI
GO 305 POKE 53768,80
PW 310 FOR J=1 TO 8
AN 320 P2=INT((1789790/(2*DZWIEK(
   J))-7)/256)
FL 330 P1=INT(1789790/(2*DZWIEK(J
   ))-7-256*P2+0.5)
RE 340 POKE 53760,P1:POKE 53762,P
   2:POKE 53763,(16*10)+8
XD 350 DZWIEK=1789790/(2*(256*P2+
   P1+7))
ZU 360 POSITION 21,J+6:? P1:POSIT
   ION 25,J+6:? P2;"   ";DZWIEK
CC 370 FOR W=1 TO 150:NEXT W:NEXT
    J
NF 380 POKE 53760,0:POKE 53762,0
WK 400 POSITION 2,19:? " CZY ZNOW
   U GRAC (T LUB N)";:INPUT A$:IF
    A$="T" THEN 150
DJ 410 ? "WPROWADZ 8 NOWYCH WARTO
   SCI DZWIEKU":? "POJEDYNCZO."
OB 420 FOR I=1 TO 8:INPUT X:DZWIE
   K(I)=X:NEXT I:GOTO 150
```

```
XP 20000 REM ****************
WJ 20010 REM * Prosram nr 3 *
XX 20020 REM ****************
BT 20040 REM
KJ 20050 REM X=USR(ADR(HF1$),N1,G
   1,N2,G2,N3,G3)
CB 20060 REM
FU 20100 SOUND 0,0,0,0:X=64+16:PO
   KE 53768,X
PY 20110 DIM HF1$(56):RESTORE 201
   40
QF 20120 FOR I=1 TO 56:READ X:HF1
   $(I,I)=CHR$(X):NEXT I
DN 20130 RETURN
XH 20140 DATA 104,170,104,141,2,2
   10,104,141,0,210,104,104,41,15
   ,9,160,141,3,210
AH 20150 DATA 224,2,240,32,104,10
   4,141,4,210,104,104,41,15,9,16
   0,141,5,210
DI 20160 DATA 224,4,240,14,104,10
   4,141,6,210,104,104,41,15,9,16
   0,141,7,210,96
```

```
XP 20000 REM ****************
XH 20010 REM * Prosram nr 4 *
XX 20020 REM ****************
BT 20040 REM
TP 20050 REM X=USR(ADR(HF2$),N1,G
   1,N2,G2)
CB 20060 REM
XO 20100 SOUND 0,0,0,0:X=64+16+32
   +8:POKE 53768,X
MO 20110 DIM HF2$(41):RESTORE 201
   40
NA 20120 FOR I=1 TO 41:READ X:HF2
   $(I,I)=CHR$(X):NEXT I
DN 20130 RETURN
XH 20140 DATA 104,170,104,141,2,2
   10,104,141,0,210,104,104,41,15
   ,9,160,141,3,210
SE 20150 DATA 224,2,240,17
FM 20160 DATA 104,141,6,210,104,1
   41,4,210,104,104,41,15,9,160,1
   41,7,210,96
```

```
FK 1 REM ****************
AB 2 REM * Prosram nr 5 *
FM 3 REM ****************
NJ 4 REM
LW 10 SOUND 0,0,0,0
QZ 80 DIM S(5,8),IN$(50)
EH 90 FOR I=0 TO 8:FOR J=0 TO 5:S
   (J,I)=0:NEXT J:NEXT I
HE 100 REJ=5000:ZNI=5100:CZE=5200
   :WYW=5300:WYL=5400
NI 102 CLD=5900:CLX=6000:GLO=6100
   :POKAUD=6200:STA=6300:START=64
   00:REJWYS=6500:BUZZ=6600
WN 104 PWYK=6700:SWYK=6800:EWYS=6
   900
XA 1000 REM WYSWIETLENIE
OJ 1002 GRAPHICS 0:POKE 752,1
WW 1008 POSITION 2,0:? "(REJ)ESTR
    AUDCTL:"
CW 1010 POSITION 2,1:? "    wielom
   ian 9 bitowy:(B7):"
PC 1020 POSITION 2,2:? "   1.79 MH
   z w kanale 0:(B6):"
RI 1030 POSITION 2,3:? "   1.79 MH
   z w kanale 2:(B5):"
CO 1040 POSITION 2,4:? "     kanal
    1 + kanal 0:(B4):"
HG 1050 POSITION 2,5:? "     kanal
    3 + kanal 2:(B3):"
LS 1060 POSITION 2,6:? "filtr gor
   noprzep.k0/2:(B2):"
NW 1070 POSITION 2,7:? "filtr gor
   noprzep.k1/3:(B1):"
CR 1080 POSITION 2,8:? "
          15 kHz:(B0):"
VC 1090 POSITION 2,9:? "(REJ)ESTR
    DZWIEKU:"
LS 1100 POSITION 2,10:? " (ZNI)EK
   SZTALCENIE:"
DC 1110 POSITION 2,11:? "    (CZE)
   STOTLIWOSC:"
YM 1120 POSITION 2,12:? " WYMUSZO
   NE WYJSCIE:"
JJ 1126 POSITION 2,13:? "
   (GLO)SNOSC:"
ED 1128 POSITION 2,14:? "X:"
MA 1130 POSITION 2,15:? "D:"
BO 1140 POSITION 2,16:? "REJ ZNI
   CZE WYW GLO"
DS 1150 POSITION 2,17:? "WYL STA
   PWYK SWYK"
GX 1500 GOSUB START
NO 2000 REM TABLICA SKOKOW
VR 2008 FOR ZZZ=1 TO 2 STEP 0
HV 2010 POSITION 5,15:POKE 752,0:
   INPUT IN$:POKE 752,1
JK 2020 TRAP 2040:A=VAL(IN$):TRAP
    40000
RI 2030 POSITION 5,14:? A:GOSUB C
   LD
OO 2040 IF IN$="REJ" THEN GOSUB R
   EJ
QM 2042 IF IN$="ZNI" THEN GOSUB Z
   NI
RS 2044 IF IN$="CZE" THEN GOSUB C
   ZE
FI 2046 IF IN$="WYW" THEN GOSUB W
   YW
LI 2048 IF IN$="WYL" THEN GOSUB W
   YL
BO 2049 IF IN$="STA" THEN GOSUB S
   TA
SX 2058 IF IN$="GLO" THEN GOSUB G
   LO
QI 2060 IF IN$="B7" THEN S(4,7)=
   NOT (S(4,7)):POSITION 30,1:? S
   (4,7):GOSUB CLD
NA 2061 IF IN$="B6" THEN S(4,6)=
   NOT (S(4,6)):POSITION 30,2:? S
   (4,6):GOSUB CLD
JS 2062 IF IN$="B5" THEN S(4,5)=
   NOT (S(4,5)):POSITION 30,3:? S
   (4,5):GOSUB CLD
GK 2063 IF IN$="B4" THEN S(4,4)=
   NOT (S(4,4)):POSITION 30,4:? S
   (4,4):GOSUB CLD
DC 2064 IF IN$="B3" THEN S(4,3)=
   NOT (S(4,3)):POSITION 30,5:? S
   (4,3):GOSUB CLD
ZU 2065 IF IN$="B2" THEN S(4,2)=
   NOT (S(4,2)):POSITION 30,6:? S
   (4,2):GOSUB CLD
WM 2066 IF IN$="B1" THEN S(4,1)=
   NOT (S(4,1)):POSITION 30,7:? S
   (4,1):GOSUB CLD
TE 2067 IF IN$="B0" THEN S(4,0)=
   NOT (S(4,0)):POSITION 30,8:? S
   (4,0):GOSUB CLD
ZL 2070 IF IN$="PWYK" THEN GOSUB
   PWYK
EY 2072 IF IN$="SWYK" THEN GOSUB
   SWYK
```

```
AE 2980 IF BLAD=1 THEN GOSUB BUZZ
US 2989 BLAD=0
WC 2990 NEXT ZZZ
HC 5000 REM REG REGISTER SET
KY 5010 IF A<0 OR A>3 THEN BLAD=1
II 5020 IF A>0 OR A<4 THEN POSITI
   ON 24,9:? A
GV 5030 C=A:REM S(C,B)
MG 5040 GOSUB REJWYS
OF 5088 GOSUB CLD:GOSUB CLX
BE 5090 RETURN
LV 5100 REM ZNI POZIOM ZNIEKSZTAL
   CEN
KZ 5110 IF A<0 OR A>14 THEN BLAD=
   0:GOTO 5180
IK 5112 IF INT(A/2)-A/2<>0 THEN B
   LAD=1:GOTO 5180

BC 5120 D1=A*16
GS 5130 POSITION 21,10:? "   "
LH 5135 POSITION 21,10:? A
QC 5140 S(C,1)=D1:S(C,5)=A
JA 5170 S(C,8)=A
NB 5180 GOSUB CLD:GOSUB CLX
BG 5190 RETURN
RQ 5200 IF A<0 OR A>255 THEN BLAD
   =1
LN 5210 POSITION 21,11:? "
   "
LF 5220 POSITION 21,11:? A
GI 5230 S(C,2)=A
ND 5280 GOSUB CLD:GOSUB CLX
BI 5290 RETURN
EE 5300 REM FRC SET FORCE BIT
XU 5310 A=INT(A)
FB 5315 IF A<0 OR A>3 THEN BLAD=1
   :GOTO 5380

LK 5320 S(A,3)= NOT S(A,3)
KT 5350 POSITION 21,12:? S(C,3)
DB 5380 GOSUB CLD
BK 5390 RETURN
YV 5400 REM WYL WYLACZENIE DZWIEK
   U
PS 5410 POKE 53761,0:POKE 53763,0
   :POKE 53 35,0:POKE 53767,0
DD 5480 GOSUB CLC
BM 5490 RETURN
MC 5900 REM CLD KASOWANIE POZYCJI
   D
WD 5910 POSITION 5,15:? "
                         "
BW 5990 RETURN
TH 6000 REM CLX KASOWANIE POZYCJI
   X
VH 6010 POSITION 5,14:? "
                       ":A=0
BF 6090 RETURN
MX 6100 REM GLO USTAWIENIE GLOSNO
   SCI
OK 6110 IF A<0 OR A>15 THEN BLAD=
   1:GOTO 6180
ZZ 6120 POSITION 21,13:? "
   "
MY 6122 POSITION 21,13:? A
HB 6130 S(C,4)=A
NC 6180 GOSUB CLD:GOSUB CLX

BH 6190 RETURN
BW 6200 REM POKAUD POKE WARTOSC A
   UDCTL
BO 6208 SUM=0
ZJ 6210 FOR I=0 TO 7
UC 6220 IF S(4,I)=1 THEN SUM=SUM+
   2^I
FH 6230 NEXT I
NU 6240 POKE 53768,SUM
8J 6290 RETURN
ZW 6300 REM STA WLACZENIE KANALU
   DZWIEKOWEGO

RV 6310 GOSUB POKAUD
AU 6320 IF A=0 THEN POKE 53761,S(
   0,1)+S(0,4):POKE 53760,S(0,2)
MK 6322 IF A=1 THEN POKE 53763,S(
   1,1)+S(1,4):POKE 53762,S(1,2)
YA 6324 IF A=2 THEN POKE 53765,S(
   2,1)+S(2,4):POKE 53764,S(2,2)
JQ 6326 IF A=3 THEN POKE 53767,S(
   3,1)+S(3,4):POKE 53766,S(3,2)
MP 6380 GOSUB CLX:GOSUB CLD:GOSUB
   REJWYS
BL 6390 RETURN
VD 6400 REM START
AQ 6410 FOR I=1 TO 8:POSITION 30,
   I:? "0":NEXT I
BN 6490 RETURN
ZC 6500 REM REJWYS WYSWIETLENIE R
   EJESTRU
IY 6505 POSITION 21,12:? "   "
LH 6506 POSITION 21,12:? S(C,3)
KW 6510 POSITION 21,11:? "       "
IW 6511 POSITION 21,11:? S(C,2)
KG 6520 POSITION 21,10:? "       "
MV 6521 POSITION 21,10
CQ 6522 ? S(C,1)/16
MO 6530 POSITION 21,13:? "       "
MQ 6531 POSITION 21,13:? S(C,4)
BP 6590 RETURN
LM 6600 REM BUZZ
HO 6610 ? CHR$(253)
BR 6690 RETURN
IT 6700 REM PWYS WYSWIETLENIE DAN
   YCH POKE
QW 6705 GOSUB EWYS
RN 6710 POSITION 20,18:? "POKE 53
   768, ";SUM
XC 6720 POSITION 2,19:? "POKE 537
   61, ";S(0,1)+S(0,4):POSITION 2
   0,19:? "POKE 53760, ";S(0,2)
MX 6730 POSITION 2,20:? "POKE 537
   63, ";S(1,1)+S(1,4):POSITION 2
   0,20:? "POKE 53762, ";S(1,2)
FE 6740 POSITION 2,21:? "POKE 537
   65, ";S(2,1)+S(2,4):POSITION 2
   0,21:? "POKE 53764, ";S(2,2)
XL 6750 POSITION 2,22:? "POKE 537
   67, ";S(3,1)+S(3,4):POSITION 2
   0,22:? "POKE 53766, ";S(3,2)
DK 6780 GOSUB CLD
BT 6790 RETURN
NK 6800 REM SWYS WYSWIETLENIE DAN
   YCH SOUND
QY 6805 GOSUB EWYS
XF 6810 FOR I=0 TO 3
NS 6820 POSITION 2,19+I:? "SOUND
   ";I;", ";S(I,2);", ";S(I,8);",
   ";S(I,4)

FT 6830 NEXT I
DM 6880 GOSUB CLD
BV 6890 RETURN
WG 6900 REM EWYS KASOWANIE PWYS &
   SWYS
XY 6910 FOR I=0 TO 4
MR 6920 POSITION 0,18:? CHR$(156)
FY 6940 NEXT I
BX 6990 RETURN
ET 7000 END
```

AUDF2 lub AUDF3 z AUDF4). Umożliwia to wybór jednego głosu 16-bitowego i dwóch ośmiobitowych, lub dwóch głosów szesnastobitowych. Za to możliwe jest uzyskanie częstotliwości wyjściowej mniejszej niż 1 Hz.

W takim przypadku nie możemy już używać instrukcji SOUND, która niszczy ustawienia dla dźwięku 16-bitowego. Musimy zatem poznać inny sposób sterowania dźwiękiem. Trzeba używać instrukcji POKE zamiast SOUND. Nie jest to żadnym problemem, ponieważ możemy zdefiniować dokładną równoważność do SOUND następująco.

Do łączenia rejestrów częstotliwości wykorzystujemy odpowiednie ustawienia bitów rejestru AUDCTL, w szczególności bitów 3 i 4. Ustawienie bitu 4 (POKE 53768,16) łączy rejestry częstotliwości AUDF1 i AUDF2, a ustawienie bitu 3 (POKE 53768,8) — rejestry AUDF3 i AUDF4. Dodatkowo dla uzyskania pełnego zakresu częstotliwości wyjściowych, niezbędne jest włączenie zegara o częstotliwości 1 79 MHz (a dokładniej 1 78979 MHz), a zatem POKE 53768,80 dla połączenia kanałów 1 i 2 oraz POKE 53768,120 dla połączenia kanałów 1 i 2 oraz 3 i 4.

Zmienia się również nieco wzór dla określenia częstotliwości wyjściowej

fwy = fzeg/(2 × (256 × N2 + N1 + 7))

gdzie: N2 i N1 są zawartościami rejestrów częstotliwości, drugi rejestr pary jest młodszym bajtem, AUDF2 lub AUDF4.

Nowymi głosami możemy sterować następująco

Głos 1 + 2

POKE 53760,N1 **POKE** 53762,N2 **POKE** 53763 16 • Z + G

Głos 3 + 4

POKE 53764,N1 **POKE** 53766,N2 **POKE** 53767 16 • Z + G

Odwrotnie, jeżeli chcemy określić jakie wartości należy wstawić do rejestrów częstotliwości w celu wygenerowania dźwięku o wybranej częstotliwości, możemy rozwiązać (lub zlecić komputerowi) następujące równania

N2=**INT**((fzeg/(2×fwy)—7)/256)
N2=**INT**((fzeg/(2×fwy)—7)—256×N2+0 5)

W celu wygenerowania nuty o częstotliwości 440 Hz otrzymujemy N2 = 7 i N1 = 235 i wstawiamy je do rejestrów. W rezultacie uzyskujemy częstotliwość 439,97 Hz — bardzo dobre dostrojenie.

Rejestr sterujący niższego kanału nie jest używany i powinien być ustawiony na zero. Głośność sterowana jest zawartością drugiego rejestru sterującego (AUDC2 lub AUDC4).

Program nr 1 demonstruje użycie dwóch manipulatorów do sterowania częstotliwością jednego głosu o wysokiej rozdzielczości i jednego „normalnego". Wyświetla on na ekranie odpowiednie równania i aktualizuje na bieżąco zmieniane w nich parametry. Można zaobserwować, jakim zakresem sterowania dźwiękiem dysponujemy w każdym przypadku.

Program nr 2 umożliwia nam eksperymentowanie z różnymi skalami nut — używając najlepszej normalnej aproksymacji, a następnie wykorzystując sterowanie dźwiękiem o wysokiej rozdzielczości.

Instrukcje obsługi programów wyświetlane są na ekranie.

Programy nr 3 i 4 są procedurami w języku maszynowym, które ułatwiają korzystanie z dźwięku o wysokiej rozdzielczości z BASIC-u. Pierwszy z nich wykorzystuje jeden głos 16-bitowy i dwa ośmiobitowe, a drugi dwa głosy szesnastobitowe. **SOUND** 0,0,0,0 na początku każdej procedury inicjuje układ **POKEY** dla dźwięku. **POKE** 53768 X ustawia zawartość rejestru AUDCTL dla dźwięku 16-bitowego, jednego lub dwóch głosów. Pamiętajmy, że dowolna instrukcja SOUND, wykonana później, wyzeruje ten rejestr.

W celu użycia tych procedur, należy skopiować jedną z nich dla własnego programu i wykonać GOSUB 20100 (raz, na początku programu). Później, dla uzyskania dźwięku, należy wywołać procedurę w języku maszynowym

X=**USR**(ADR(HF1$),N1,G1,N2,G2,N3,G3)

lub

X=**USR**(ADR(HF2$),N1,G1,N2,G2)

Nx jest określeniem częstotliwości dźwięku, a Gx — głośnością. N1 jest głosem 16-bitowym w procedurze trzygłosowej. Nie musimy przekazywać parametrów dla nieużywanych głosów. Na przykład, jeśli chcemy usłyszeć tylko dźwięk 16-bitowy w procedurze trzygłosowej, możemy użyć X = **USR**(ADR(HF1$),N1,G1), lecz dla użycia tylko głosu ośmiobitowego musimy dać: X = **USR** (ADR(HF1$),0,0,N2,G2)

Zamieszczona poniżej tablica prezentuje najbardziej dokładne wartości nut dla czterech oktaw ośmiobitowych i dziewięciu oktaw 16-bitowych.

Praktycznie pierwsza oktawa nut 16-bitowych nie nadaje się do wykorzystania, ponieważ powstają pewne dźwięki harmoniczne, które maskują odtwarzaną aktualnie nutę.

Bity 1 i 2 AUDCTL sterują filtrami górnoprzepustowymi w kanałach 2 i 1 odpowiednio. Filtr górnoprzepustowy umożliwia przejście tylko wyższym częstotliwościom. W przypadku tych filtrów wyższe częstotliwości zdefiniowane są jako będące nieco wyższe niż wyjście innego kanału wybranego przez kombinację bitów AUDCTL. Na przykład, jeśli dźwięk w kanale 3 naśladuje ryk krowy i ustawiony jest drugi bit AUDCTL, wtedy tylko dźwięki o częstotliwościach wyższych od tego ryku będą

słyszane w kanale 1 (niższe częstotliwości będą odfiltrowane)

Filtr może być programowany na bieżąco ponieważ kanał filtrujący może być zmieniany „w locie". Daje to programiście duże możliwości. Filtry używane są głównie do tworzenia efektów specjalnych. Dla przykładu sprawdźmy następujące instrukcje

SOUND 0,0,0,0
POKE 53768,4
POKE 53761,168 **POKE** 53765,168
POKE 53760,254 **POKE** 53764,127

Bit 7 AUDCTL przełącza licznik 17-bitowy na 9--bitowy. Im krótszy licznik, tym częściej powtarza się jego wzór zakłóceń lub jest bardziej dostrzegalny. Zatem zmieniając licznik 17-bitowy uczynimy wzór zniekształceń bardziej powtarzalnym lub bardziej dostrzegalnym. Spróbujmy wykonać następujące instrukcje, słuchając uważnie kiedy wykonywana jest instrukcja **POKE:**

SOUND 0,80,8,8 użycie licznika 17-bitowego
POKE 53768,128 zmiana na licznik 9-bitowy

Podsumowanie

Dźwięk jest jedną z najciekawszych i jednocześnie najtrudniejszych do poznania możliwości komputera Atari. Istnieją cztery niezależne kanały dźwiękowe, których częstotliwość, zniekształcenie i głośność mogą być sterowane niezależnie. Są one adresowane przez instrukcję SOUND za pomocą liczb 0 do 3. Dane o dźwięku mogą być nieza eznie wstawiane (POKE) do rejestrów o adresach 53760 do 53767. Komórki nieparzyste sterują głośnością i zniekształceniem a parzyste sterują częstotliwością. Rejestr 53768 (AUDCTL) steruje wszystkimi kanałami, dźwiękowymi w określony sposób. Jeśli użyjemy instrukcji SOUND, dowolne zmiany które zostały zrobione w AUDCTL są zerowane — AUDCTL ustawiany jest na zero. Zatem instrukcja SOUND nie zapewnia pełnego wykorzystania możliwości dźwiękowych komputera

Program nr 5 przeznaczony jest do demonstracji przedstawionych powyżej możliwości. Próbuje dać łatwe sterowanie wszystkimi parametrami. Wymaga nieco wprawy w obsłudze, lecz pozwala na totalne sterowanie wszystkimi rejestrami plus AUDCTL, włącza kanały indywidualnie i wyłącza je wszystkie naraz. Kiedy otrzymane dźwięki odpowiadają nam, możemy wyświetlić odpowiednie instrukcje BASIC-u w postaci POKE albo SOUND

Po uruchomieniu programu na ekranie pojawia się szereg linii. Pierwsze osiem z nich, oznaczone B7 do B0, reprezentują zawartości bitów rejestru AUDCTL. W celu zmiany siódmego bitu na 1, należy wprowadzić B7 i RETURN. Dla zmiany z powrotem na zero, wprowadzamy B7 i RETURN. Są to zmiany techniczne, które nie dają wskazań, jak będzie brzmiał nowy dźwięk. Najlepiej eksperymentować.

Dla omówienia następnych pięciu linii musimy przejść w dół, do linii oznaczonych D i X. Program przyjmuje dwa typy danych komendy i liczby. Jeśli chcemy mieć liczbę, wprowadzamy ją i naciskamy RETURN. Jeśli komendę, to postępujemy podobnie. W celu wybrania zerowego kanału dźwiękowego wprowadzamy następujący ciąg: 0, RETURN REJ, RETURN. Zero pojawi się na ekranie po REJ ESTR DŹWIĘKU. Dla uzyskania czystego tonu wprowadźmy 10,RETURN, ZNI RETURN 10 pojawi się po ZNIEKSZTAŁCENIE. Podobnie 100 RETURN, CZE, RETURN, 8 RETURN GLO RETURN, wypełni pozostałe linie

Dla usłyszenia dźwięku musimy wprowadzić, 0 RETURN STA, RETURN a dla wyłączenia WYL RETURN. Aby zobaczyć wartości POKE dla tego dźwięku, należy wprowadzić PWYK, RETURN i na ekranie pojawi się wykaz dziewięciu instrukcji POKE. Możemy skopiować je do naszego programu, aby odtworzyć dźwięk, który usłyszeliśmy. Prawy górny POKE dotyczy rejestru AUDCTL. Następne cztery wiersze są kanałami 0 do 3 - lewa kolumna dotyczy zniekształcenia i głośności, a prawa — częstotliwości każdego kanału

Jeśli zawartość rejestru AUDCTL jest zerowa - zerowe wartości bitów B0 do B7 — wtedy mogą być używane instrukcje SOUND. Aby je zobaczyć wprowadźmy SWYK RETURN

Wyjście „wymuszone" występuje w rejestrach nieparzystych wytwarza trzask w głośniku. Jest on włączany i wyłączany za pomocą WYW, RETURN. Kanały dźwiękowe muszą być włączane indywidualnie za pomocą komendy STA WYL wyłącza wszystkie kanały. Jeśli dokonamy zmiany chcemy ją usłyszeć, wprowadzamy numer kanału i STA

Tomasz MROWIEC
Ludwik PIELA

Oszczędzajmy czas — *„ZX Spectrum"*

Zadanie było proste: ułożyć z dwudziestu znaków graficznych UDG komputera Spectrum wzór kobierca. W celu uzyskania symetrii należało podzielić zasoby znaków UDG na 4 grupy po 5 znaków, symetrycznych względem osi poziomej i pionowej. Wystarczyło zdefiniować tylko 5 znaków po 8 bajtów (64 punkty), a następnie powtórzyć je w odbiciach zwierciadlanych względem obu osi symetrii

ABCDE	JIHGF
ONMLK	PQRST

Odwrócenie w stosunku do osi poziomej nie było trudne — wystarczyło wpisać 40 bajtów odpowiadających znakom ABCDE w odwrotnej kolejności. Trudniej było z odwróceniem w stosunku do osi pionowej — tu należało rozbić bajty na poszczególne bity, a następnie złożyć z nich odwrócone „antybajty". Przypominam, że wartość liczbowa bitów w bajcie odpowiada kolejnym potęgom liczby 2.

W pierwszej próbie (wiersze 1000—1020) rozbito generowane przez generator liczb losowych RND bajty o wartości 0—255 na poszczególne bity metodą rachunkową dzieląc je przez kolejne potęgi liczby 2, a następnie złożono w odwrotnej kolejności, mnożąc przez kolejne potęgi dwójki. W końcu wprowadzono obliczone antybajty przez POKE w odpowiednie komórki pamięci UDG. Pozostało tylko wydrukowanie (wiersz 800)

Program jest napisany poprawnie, ale wskutek użycia pętli **FOR—NEXT** i podnoszenia dwójki do kolejnych potęg trwał nieznośnie długo — ponad dwie minuty

W drugiej próbie (w. 2000—2020) wygenerowano od razu bity (zera lub jedynki) i złożono z nich kolejne bajty i antybajty. Przyspieszyło to działanie programu o 28 sekund, ale tak pozostały jeszcze 92 sekundy — o wiele za długo, jak na tak krótki program

W trzeciej próbie (wiersze 3000—3020) ułożono na początku tabelę kolejnych potęg dwójki. Dało to od razu 4-krotne przyspieszenie programu. Program wraz z generacją tabeli trwał 23 sekundy, powtórzenia o sekundę krócej

Skoro tabela potęg dała tak dobre wyniki, należało spróbować stworzyć tabelę antybajtów (wersja trzecia). Obliczanie takiej tabeli trwało ponad dwie minuty, ale potem kolejne mutacje kobierca powstawały po prawie 3 sekundach. W ten sposób **(wiersze 4000—4030)** wyczerpaliśmy możliwości metod rachunkowych BASIC-a. Szybciej można by w asemblerze, ale to już zupełnie inna historia

W piątej wersji (wiersze 5000—5020) zastosowano trick graficzny: kolejne bajty wprowadzono do pamięci ekranu pod adresem 22432 (lewe dolne naroże ekranu), a następnie odczytywano punkt po punkcie przez POINT (x, y) i sumowano w antybajty wprost bez użycia pętli. Dzięki temu skrócono generację zestawu znaków do 4 sekund

Ale to jeszcze nie koniec

Dzięki omawianemu już trickowi graficznemu w ciągu 17 sekund stworzono tablicę antybajtów i zapisano w pamięci pod adresem 60000—60255. Ta wersja programu **(szósta — wiersze 6000—6030)** tworzy kolejne mutacje kobierca co 2,65 sek. Chyba nieźle? A może uda się jeszcze komuś urwać kawałek sekundy? Czekamy na pomysły Czytelników

W wierszach 9000—9020 znajduje się prosty program „wyplatający" wzór kobierca. Wzór ten składa się z dziewięciu pasów po 64 znaki (a/$) i z dwóch pasów skrajnych (c/$). Przez kombinację tych pasów uzyskuje się 704-znakowy wzór d/$. Oczywiście zawartość a/$ i c/$ wprowadza się w trybie graficznym (kursor "G"). Wzór ten w postaci pierwotnej-literowej można zobaczyć tylko raz, bezpośrednio po wgraniu programu z kasety. Potem znaki UDG zamieniają się w mozaikę punktów. Na wszelki wypadek jest w programie wiersz 400, przywracający w razie potrzeby pierwotną postać znaków UDG. Uruchamia się go przez **GO TO 400**

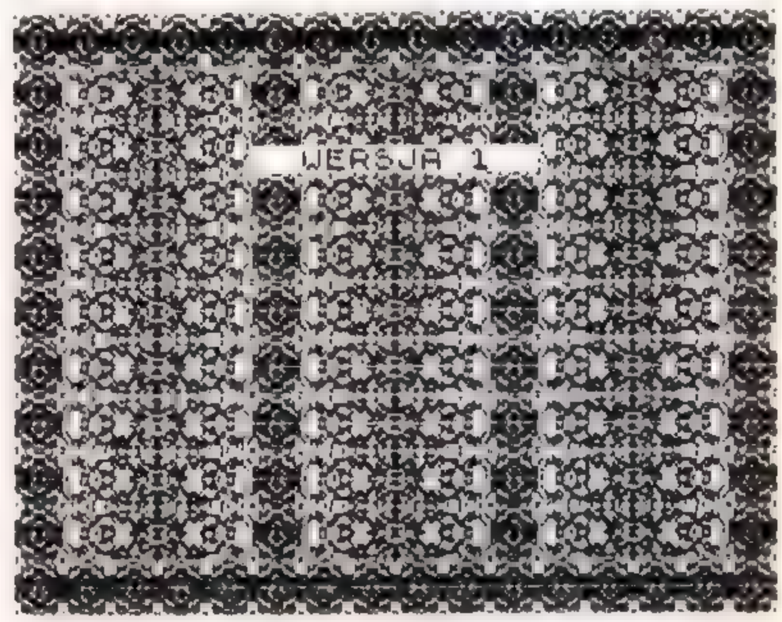
Pozostaje jeszcze wiersz 9999 zapisujący program na taśmę i sprawdzający poprawność zapisu

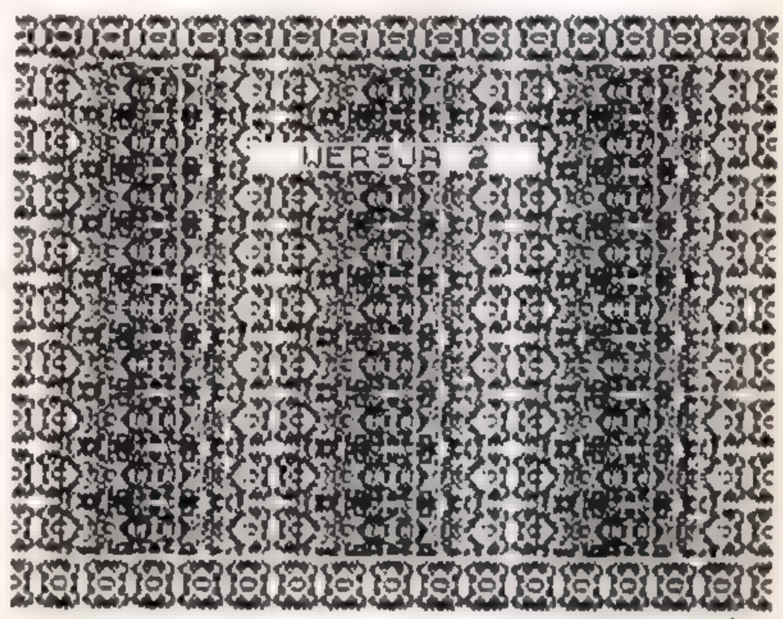
Jeszcze jedna propozycja dla Czytelników: Kto wymyśli prosty sposób, jak każdej czwórce znaków UDG (np. AFOT lub EJKP) przyporządkować jeden z pięciu kolorów?

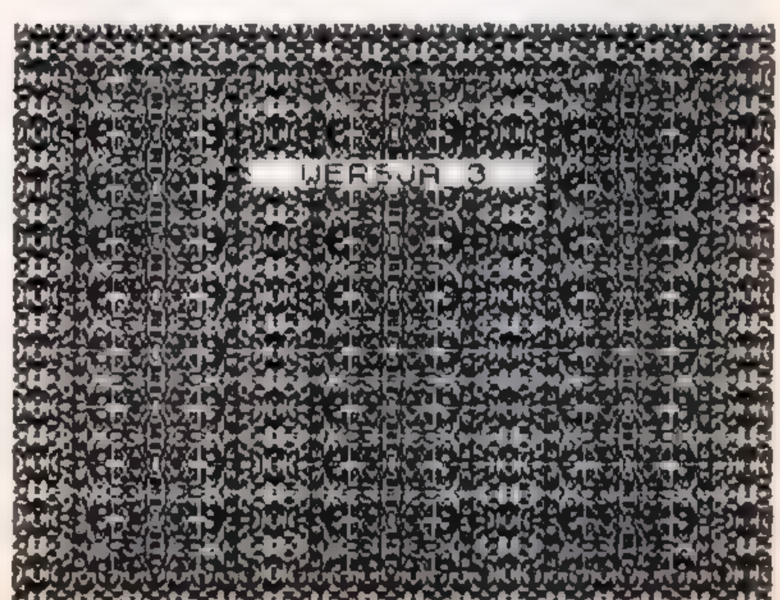
Jan RUBINOWICZ

```
   0 REM      Jan Rubinowicz 1987
   1 REM
   2 REM
   3 REM
 100 REM KOBIERZEC SAVE-RUN 9999
 200 GO SUB 9000: GO SUB 800
 210 LET we=1: GO SUB 500: GO SU
B 1000: GO SUB 800
 220 LET we=2: GO SUB 500: GO SU
B 2000: GO SUB 800
 230 LET we=3: GO SUB 500: GO SU
B 3000: GO SUB 800
 235 BEEP .02,14: FOR r=1 TO 5:
GO SUB 3010: GO SUB 800: GO SUB
800: NEXT r
 240 LET we=4: GO SUB 500: GO SU
```

```
B 4000: GO SUB 300
 245 BEEP .02,14: FOR r=1 TO 5:
GO SUB 4020: GO SUB 800: GO SUB
600: NEXT r
 250 LET we=5: GO SUB 500: GO SU
B 5000: GO SUB 300
 255 BEEP .02,14: FOR r=1 TO 5:
GO SUB 5000: GO SUB 800: GO SUB
600: NEXT r
 250 LET we=6: GO SUB 500: GO SU
B 6000: GO SUB 300
 255 BEEP .02,14: FOR r=1 TO 5:
GO SUB 5020: GO SUB 800: GO SUB
600: NEXT r
 270 GO TO 210
 400 FOR n=0 TO 167: POKE 65368+
n,PEEK (15880+n): NEXT n: GO TO
800
 500 BEEP .02,14: PRINT AT 5,10;
 BRIGHT 1; FLASH 1;"  WERSJA ";w
e;"  ": RETURN
 600 PRINT BRIGHT 1;AT 21,0;r: R
ETURN
 800 PRINT AT 0,0; BRIGHT 1;d$:
RETURN
 999 REM
 wersja 1 - odwracanie bajtow
1000 FOR m=0 TO 39: LET bt=INT (
RND*256): LET abt=0
1010 FOR b=1 TO 8: LET p=(bt/2↑b
-INT (bt/2↑b))>.5: LET abt=abt+2
*p*2↑(7-b): NEXT b
1020 POKE (65368+m),bt: POKE (65
408+m),abt: POKE (65487-m),bt: P
OKE (65527-m),abt: NEXT m
1030 RETURN
1999 REM
 wersja 2 - skladanie z bitow
2000 FOR m=0 TO 39: LET bt=0: LE
T abt=0:
2010 FOR b=0 TO 7: LET a=INT (RN
D*2): LET bit=(2↑b)*a: LET bt=bt
+bit: LET abit=(2↑(7-b))*a: LET
abt=abt+abit: NEXT b
2020 POKE (65368+m),bt: POKE (65
408+m),abt: POKE (65487-m),bt: P
OKE (65527-m),abt: NEXT m
2030 RETURN
2999 REM
 wersja 3 - tabela poteg
3000 DIM k(8): FOR n=1 TO 8: LET
 k(n)=2↑(n-1): NEXT n
3010 FOR m=0 TO 39: LET bt=0: LE
T abt=0: FOR b=1 TO 8: LET a=INT
 (RND*2): LET bit=k(b)*a: LET bt
=bt+bit: LET abit=k(9-b)*a: LET
abt=abt+abit: NEXT b
3020 POKE (65368+m),bt: POKE (65
408+m),abt: POKE (65487-m),bt: P
OKE (65527-m),abt: NEXT m
3030 RETURN
3999 REM
 wersja 4 - tabela antybajtow
4000 DIM k(8): FOR n=1 TO 8: LET
 k(n)=2↑(n): NEXT n
4010 DIM t(256): DIM p(8): FOR a
=1 TO 256: LET t(a)=0: FOR b=1 T
O 8: LET p(b)=((a-1)/k(b)-INT ((
a-1)/k(b)))>.5: LET t(a)=t(a)+(.
5*p(b)*k(9-b)): NEXT b: NEXT a
4020 FOR m=0 TO 39: LET bt=INT (
RND*256): LET abt=t(bt+1)
4030 POKE (65368+m),bt: POKE (65
408+m),abt: POKE (65487-m),bt: P
OKE (65527-m),abt: NEXT m
4040 RETURN
4999 REM
 wersja 5 graf.odwracanie bajtow
5000 FOR m=0 TO 39: LET bt=INT (
RND*256): POKE 22432,bt
5010 LET abt=POINT (0,0)+2*POINT
 (1,0)+4*POINT (2,0)+8*POINT (3,
0)+16*POINT (4,0)+32*POINT (5,0)
+64*POINT (6,0)+128*POINT (7,0)
5020 POKE (65368+m),bt: POKE (65
408+m),abt: POKE (65487-m),bt: P
OKE (65527-m),abt: NEXT m
5030 RETURN
5999 REM
 wersja 6 graf.tabela antybajtow
6000 FOR n=0 TO 255: POKE 22432,
n
6010 LET t=POINT (0,0)+2*POINT (
1,0)+4*POINT (2,0)+8*POINT (3,0)
+16*POINT (4,0)+32*POINT (5,0)+6
4*POINT (6,0)+128*POINT (7,0): P
OKE 60000+n,t: NEXT n
6020 FOR m=0 TO 39: LET bt=INT (
RND*256): LET abt=PEEK (60000+bt
)
6030 POKE (65368+m),bt: POKE (65
408+m),abt: POKE (65487-m),bt: P
OKE (65527-m),abt: NEXT m
6040 RETURN
8999 REM
 wyplatanie wzoru
9000 LET a$="
FABCDEJIHGFABCDEJIHGFABCDEJIHGFA
TONMLKPQRSTONMLKPQRSTONMLKPQRSTO
"
9010 LET b$="": FOR m=1 TO 9: LE
T b$=b$+a$: NEXT m
9020 LET c$="
FAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFA
TOTOTOTOTOTOTOTOTOTOTOTOTOTOTOTO
": LET d$=c$+b$+c$
9030 RETURN
9999 CLS : SAVE "KOBIERZEC" LINE
 100: BEEP 1,2: PRINT '"Verify.
 KOBIERZEC": VERIFY "KOBIERZEC":
 GO TO 100
```







Arytmetyka modulo liczba całkowita (1)

Mówiąc o arytmetyce mamy zwykle na myśli umiejętność wykonywania podstawowych działań dodawania i mnożenia na dowolnych liczbach. *Arytmetyka modulo liczba całkowita* zajmuje się działaniami na pewnej skończonej grupie liczb. Przykładowo arytmetyka modulo 7 dotyczy liczb 0, 1, 2, 3, 4, 5, 6, a arytmetyka modulo 11 liczb 0,1,2,...,10. Łatwo zauważyć, że są to wszystkie możliwe reszty otrzymane z dzielenia dowolnej liczby całkowitej przez liczbę wybraną jako moduł. Wynikiem dodawania bądź mnożenia liczb modulo liczba całkowita np. 7, jest reszta z dzielenia sumy bądź iloczynu przez 7, np. (2 + 8) mod 7 = 3, (2 * 4) mod 7 = 1. Z arytmetyką modulo 24 mamy do czynienia przy odczytywaniu godzin, gdzie dopuszczalne są wyniki z przedziału [0,23]. Dni liczymy modulo 7, gdzie zerowym dniem jest niedziela, pierwszym poniedziałek, a szóstym sobota.

Program nasz podaje przykłady dodawania i mnożenia modulo liczba całkowita, sprawdza umiejętności operatora programu i ocenia, czy może on przejść do programu matematycznie bardziej skomplikowanego, tj. do twierdzenia Fermata.

A.T.

```
10  REM arytmetvka modulo
20 INPUT "liczymy (t/n)";x$
30 IF x$="t" THEN GOTO 60
40 IF x$<>"n" THEN GOTO 20
50 GOTO 420
60 PRINT "podac przyklady obliczen (t/n) ?"
70 DEF FN mp(x)=x-INT(x/n1)*n1
80 DEF FN los(n)=INT(RND(1)*9)+n
90 INPUT x$
100 IF x$="t" THEN GOTO 130
110 IF x$<>"n" THEN GOTO 60
120 GOTO 220
130 n1=FNlos(2)
140 PRINT "arytmetyka modulo ";n1
150 n2=FNlos(1)
160 n3=FNlos(1)
170 x=n2+n3
180 y=n2*n3
190 PRINT n2;"  +  ";n3;" mod ";n1;" = ";FNmp(x)
200 PRINT n2;" * ";n3;" mod ";n1;" = ";FNmp(y)
210 GOTO 60
220 PRINT "sprawdze czy tez potrafisz "
230 licz=0
240 FOR i=1 TO 5
250 n1=FNlos(2)
260 n2=FNlos(1)
270 n3=FNlos(1)
280 PRINT "liczymy modulo  ";n1
290 PRINT n2;" + ";n3;" mod ";n1;" = "
300 x=n2+n3
310 v=n2*n3
320 INPUT a
330 IF a=FNmp(x) THEN licz=licz+1:PRINT "dobrze"
340 IF a<>FNmp(x) THEN PRINT "zle - sprobuj to :"
350 PRINT n2;" * ";n3;" mod ";n1;" = ?"
360 INPUT a
370 IF a=FNmp(y) THEN licz=licz+1: PRINT "dobrze"
380 IF a<>FNmp(y) THEN PRINT "zle"
390 NEXT i
400 IF licz<6 THEN PRINT "musisz jeszcze potrenowac":GOTO 20
410 PRINT"mozesz przejsc do programu :TWIERDZENIE FERMATA"
420 END
```







Nasz pan profesor ma komputerową głowę.

Fortran w CP/M

Mimo że od wielu lat PASCAL wypiera z użytkowania język FORTRAN, to jednak w wielu centrach obliczeniowych wyposażonych w komputery ODRA, RIAD itp. ze względu na bogate oprogramowanie w tym języku, wiele programów pisze się nadal w języku FORTRAN.

Przy dużym obciążeniu tych maszyn i wysokim koszcie użytkowania, dostępność do nich jest ograniczona. Warto więc sprawdzić poprawność działania opracowywanego programu na bardziej dostępnym Personal Computerze.

Istnieje kompilator tego języka na komputery pracujące w systemie CP/M o nazwie FORTRAN-80. Jak wiele innych programów tak i ten dostępny jest na giełdach jednakże najczęściej bez opisu. Oto w skrócie podstawowe informacje o sposobie uruchamiania i użytkowania tego kompilatora.

Pakiet oprogramowania FORTRAN-80 zawiera bloki o nazwach:

F80	.COM	L80	.COM
FORLIB	.REL		

Pakiet ten porównywalny jest z kompilatorami tego języka na dużych maszynach cyfrowych i minikomputerach. Posiada on pewne ograniczenia np. brak typu COMPLEX, ograniczenia w stosowaniu tablic itd. Mimo to wydaje się być pomocny w uruchamianiu nawet dużych programów.

Pisanie i kompilowanie programów:

Przystępując do pracy z kompilatorem radzę zrobić jego dodatkową kopię roboczą na innym dysku. Wszystkie powstające w kolejnych etapach kompilacji zbiory należy zapisać na ten dysk, gdyż kompilator musi komunikować się z podprogramami zawartymi w pliku. Kopia zabezpieczy Cię przed przypadkowym skasowaniem któregoś ze zbiorów kompilatora, a co się z tym wiąże, z jego utratą.

Program w języku FORTRAN piszemy w dowolnym wordprocesorze dającym na dysku blok znaków kodu ASCII (znakomicie do tego celu nadaje się edytor języka TURBO PASCAL. Edytor tekstów TASWORD lub inne).

W stosowaniu musisz pamiętać, aby nazwę programu podać z rozszerzeniem .FOR. oraz zachować właściwy FORTRAN-owi format wiersza.

Kompilator pracuje w systemie CP/M., dlatego przed dalszymi czynnościami należy wprowadzić ten system do komputera.

Kompilacja przebiega dwuetapowo

W pierwszym etapie przy użyciu kompilatora F80 tekst programu przekształcany jest w blok typu .REL.

Przykładowym zleceniem dla kompilatora jest:

F80 = PROG.FOR

gdzie: PROG oznacza nazwę kompilowanego programu.

Nakazuje ono: wczytaj kompilator F80, następnie program zawarty w bloku PROG.FOR, po czym skompiluj ten program, wypisz znalezione błędy składniowe i zapisz postać wynikową na dysku w blok o nazwie PROG.REL.

Znalezione błędy opisywane są komunikatami w języku angielskim podającym określenie błędu, numer linii, w której wystąpił oraz część tej linii.

Jeśli w pierwszej części kompilacji nie wykryto błędów, można przystąpić do drugiego etapu.

Teraz będziemy używać programu L80. Ma on za zadanie wyszukać wszystkie funkcje i podprogramy zewnętrzne, jakie zostały użyte w programie (należą do nich funkcje standardowe FORTRAN-u procedury drukowania itp.), dołączyć je do kompilowanego bloku, a następnie całość zapisać na dysku w bloku typu .COM. Przykładowym zleceniem jest:

L80 PROG.PROG/N.PROG/S/E

które nakazuje:
wczytaj z dysku program L80, następnie blok PROG.REL., potrzebne procedury, utwórz w systemie CP/M blok o nazwie PROG.COM i zapamiętaj go na dysku.

Czas przetwarzania pliku programem L80 jest stosunkowo długi, dlatego nie należy się zrażać faktem długiej pracy disc drive'u.

W trakcie tej części kompilacji mogą pojawić się informacje o niezdefiniowanych zmiennych globalnych. Oznacza to, że na dysku nie znaleziono funkcji lub podprogramu, do którego odwoływano się w kompilowanym programie. Tak skompilowany program uruchamia się w systemie CP/M zleceniem:

PROG.

Podane tutaj zlecenia są zleceniami przykładowymi. Dokładniejsze informacje zarówno o kompilatorze, jego ograniczeniach i rozszerzeniach w stosunku do standardu ANSI FORTRAN X3.9—1966, jak i różnych postaciach zleceń, można znaleźć w DOKUMENTACJI UŻYTKOWEJ KOMPILATORA FORTRANU I MAKROASEMBLERA MACRO-80 DLA SYSTEMU CP/M dostarczanej wraz z kompilatorem przy zakupie od firmy zajmującej się sprzedażą oprogramowania.

Ceny giełdowe kompilatora są w granicach 1000—2000 zł (bez dokumentacji użytkowej).

(Opracowano na podstawie „Dokumentacji użytkowej kompilatora FORTRAN dla systemu CP/M.)

Cezary DOBROWOLSKI

Pojazd kosmiczny — CPC 6128

```
10 ' Pojazd kosmiczny
20 INPUT"ILE RAZY POWTORZYC GLOS POJAZDU KOSMI-  CZNEGO";I
30 FOR L=1 TO I
40 ENV 1,1,5,1,1,-5,1
50 FOR k = 7 TO 0 STEP -1
60 ENT -1,10,-1,1,9,1,0.5
70 SOUND 1,100,-100,k,1,1
80 NEXT
90 NEXT
10 ' Kukulka
20 INPUT"ILE RAZY POWTORZYC GLOS KUKULKI";I
30 FOR L=1 TO I
40 FOR k = 1 TO 2
50 ENV 1,20,1,2
60 SOUND 1,63,0,2,1
70 SOUND 1,75,0,2,1
80 SOUND 1,0,70
90 NEXT
100 NEXT
```

Helikopter — CPC 6128

```
10 ' Helikopter
20 INPUT "ILE RAZY POWTORZYC GLOS HELIKOPTERA";I
30 FOR L=1 TO I
40 ENV 1,5,1,0.5,1,-5,0.5,5,3,0.5,1,-15,0.5
50 'ENT -1,1,20,50
60 ENV 2,5,1,100,1,0,255,5,-1,100
70 ENT -2,10,1,0.5,10,-1,0.5
80 g=1
90 SOUND 10,758,-3,7,2,2,1
100 SOUND 17,50,-7,g,1,0,1
110 FOR k=1 TO 50
120 IF k<15 AND g<15 THEN g=g+1
130 IF k>35 AND g>0 THEN g=g-1
140 SOUND 1,50,-7,g,1,0,1
150 NEXT
160 NEXT
```

Mikrokomputery IBM PC (5)

IBM Personal System

Sześć lat temu firma IBM wprowadziła na rynek komputerowy model Personal Computer, wyposażony w mikroprocesor INTEL 8088 i pamięć zapisywalną RAM 64 KB — mikrokomputer, który stał się standardem światowym, powielanym przez setki firm zachodnioeuropejskich i azjatyckich. W ciągu kolejnych lat pojawiły się modele XT, AT, RT oraz klony tych modeli (kompatybilne z nimi), wyposażone w: mikroprocesory rodziny iAPX 86 (8088, 8086, 80186, 80286, 80386) i koprocesory: pamięć zapisywalną od 640 KB do 16 MB; dyski stałe o pojemności od 10 do 210 MB; stacje dysków elastycznych 5 1/4 cala jednostronne 160/180 KB, dwustronne 320/360 KB i dużej pojemności 1,2 MB; karty graficzne Color Graphics Card (CGA) o rozdzielczości 320 × 200/640 × 200 punktów adresowalnych bezpośrednio (pixeli) przy 4 kolorach, Monochrome Graphics Card (MA) o rozdzielczości 720 × 348 pixeli, Enhanced Graphics Card (EGA) o rozdzielczości 640 × 350 pixeli przy 16 kolorach, Enhanced Graphics Card Plus (EGA Plus) o rozdzielczości 1280 × 1024 pixeli przy 256 kolorach; moduły i porty wejścia/-wyjścia umożliwiające współpracę z różnego typu urządzeniami wejścia/-wyjścia m.in. drukarkami, plotterami, digitizerami oraz pracę w sieci lokalnej (LAN — Local Area Network).

Rok 1987 i modele IBM Personal System/2, to prawdopodobnie kolejny krok w historii mikrokomputerów. Oto opis modeli 30, 50, 60 i 80.

Wymienione cztery modele mają kilka wspólnych cech. Firma IBM zdecydowała się na wprowadzenie mniejszych i trwalszych dyskietek 3 1/2 cala, umożliwiających zapisywanie większej ilości informacji niż na dyskietkach 5 1/4 cala. Zapis może mieć pojemność 720 KB lub 1,44 MB.

Mikrokomputery wyposażono w pięć portów wejścia/wyjścia: dla klawiatury (typu mini-DIN), dla myszki (typu ditto), dla monitora kolorowego (typu DIN) oraz równoległy CENTRONICS i szeregowy RS-232 C.

Klawiatura mikrokomputerów posiada 102 — klawisze.

Modele 50, 60 i 80 wyposażone są w nowe układy scalone b. dużej skali integracji VLSI, wśród nich: VGA (Video Graphics Array) z pamięcią 256 KB; 16-kanałowy sterownik bezpośredniego dostępu do pamięci (DMA). Układ VGA umożliwia grafikę z rozdzielczością 320 × 200/640 × 480 pixeli w 256 kolorach lub monochromatycznie. Po kolejnym rozwiązaniu sprzętowym można będzie uzyskać rozdzielczość 1024 × 768 pixeli w 256 kolorach.

Modele te wyposażone są w pamięć stałą o pojemności 128 KB, zawierającą CBIOS i ABIOS (w nich m.in. procedury BIOS-u Basic Input Output System). Dla wymienionych modeli zostanie opracowany OS/2 (Operating System/2), przystosowany specjalnie do możliwości mikroprocesora 80286.

Wszystkie modele mają możliwość dołączenia zewnętrznej stacji dysków elastycznych 5 1/4 cala i stacji z zapisem laserowym WORM (Write-Once Read-Many) o pojemności 200 MB.

Kolejną wspólną cechą począwszy od modelu 30 to zastosowanie dysków stałych 3 1/2 cala. W mikrokomputerach IBM PS/2 wykorzystano PC DOS 3.30.

Model 30 wyposażono w mikroprocesor 8086 pracujący z częstotliwością zegara 8 MHz. Pamięć zapisywalna standardowo ma pojemność 640 KB. W pamięci stałej 64 KB zapisane są procedury CBIOS (kompatybilne z BIOS). Model 30 ma dwie wersje: 30—002 z dwiema stacjami dysków elastycznych 3 1/2 cala o pojemności 720 KB oraz 30—021 z jedną stacją dysków elastycznych i dyskiem stałym 3 1/2 cala o pojemności 20 MB. Posiada on układ MCGA (Multi-Colour Graphics Array) umożliwiający pracę w trybie graficznym z rozdzielczością 640 × 480 pixeli monochromatycznie i 320 × 200 pixeli w 256 kolorach.

Konfigurację systemu można rozbudować wykorzystując trzy łączówki 8-bitowej magistrali, zamontowane w płycie systemowej. Model 30 jest krokiem w ewolucji IBM PC/XT. Cena wersji 30 — 002 £1106.

Model 50 wyposażony jest w mikroprocesor 80286 pracujący z częstotliwością zegara 10 MHz. Pojemność pamięci zapisywalnej o czasie dostępu 150 ns standardowo 1 MB może być rozszerzona do 7 MB. Pamięć stała 128 KB zawiera CBIOS i ABIOS. Model może posiadać jedną lub dwie stacje dysków elastycznych 3 1/2 cala o pojemności 1,44 MB oraz dysk stały 20 MB. Model 50 posiada układ VGA. System może być rozbudowany o trzy moduły dołączone do łączówek 16-bitowej magistrali mikrokomputera. Cena modelu z 1 stacją dysków elastycznych i dyskiem stałym 20 MB£2658. Model 50 pozwoli na pewno lepiej wykorzystać możliwości mikroprocesora 80286 niż w IBM PC/AT.

Model 60 różni się od modelu 50 tym, że posiada większą pojemność pamięci zapisywalnej i masowej oraz lepszą możliwość rozbudowy.

Pojemność pamięci zapisywalnej może wynosić 15 MB, mikrokomputer może posiadać dysk stały 44 MB (wersja 60-041) lub 70 MB (wersja 60-071) oraz konfiguracja może być rozszerzona o siedem modułów. Jednostka centralna modelu 60 umieszczona jest w obudowie typu „wieża", stawianej pionowo na podłodze. Cena wersji 60-071 £ 4464.

Model 80 posiada trzy wersje: 80-041 i 80-071 z mikroprocesorem 80386 pracującym z częstotliwością zegara 16 MHz oraz 80-111 z mikroprocesorem 80386 pracującym z częstotliwością 20 MHz. Pamięć zapisywalna o czasie dostępu 80 ns, standardowo o pojemności 2 MB, może być rozszerzona do 16 MB. Mikrokomputer w wersji 80-041 posiada dysk stały 44 MB, w wersji 80-071 dysk 70 MB, a w wersji 80-111 dysk o pojemności 115 MB. Konfiguracja systemu może być rozbudowana o trzy moduły — dołączone do 32-bitowej magistrali i cztery — dołączone do 16-bitowej magistrali, poprzez łączówki umieszczone w płycie systemowej. Cena wersji 80-041 £ 4916, wersji 80-111 £ 7056.

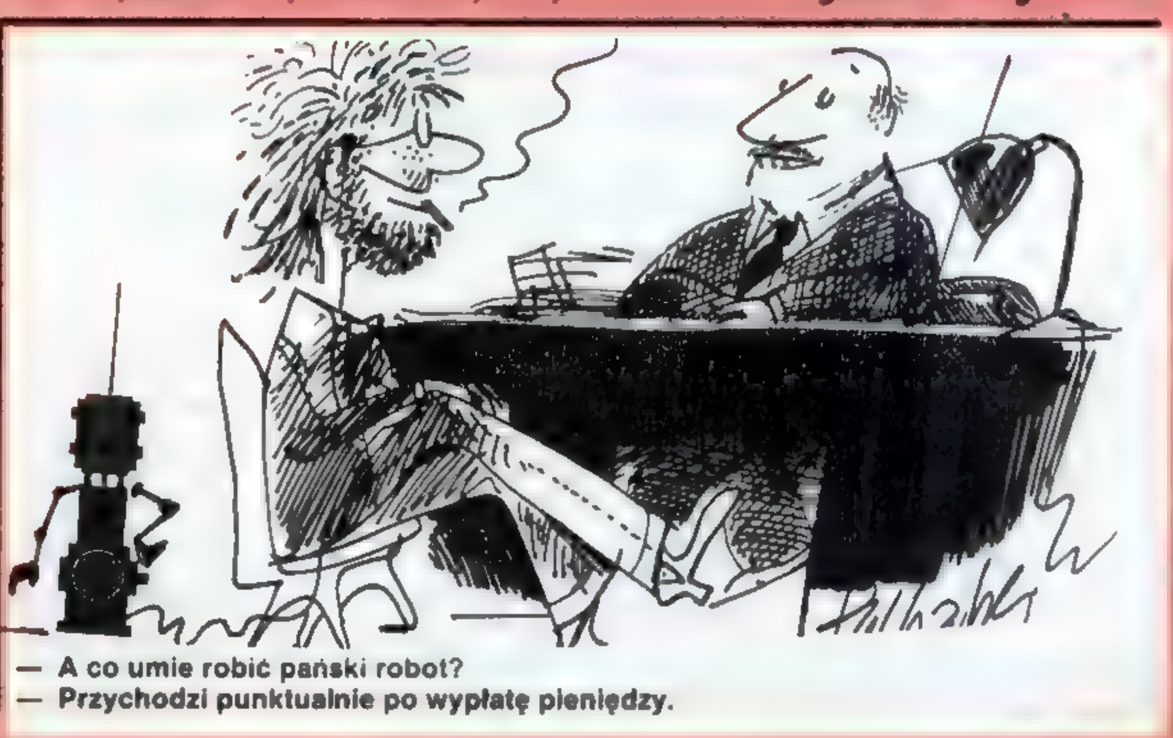
Z przedstawionymi wyżej modelami mogą pracować monitory kolorowe CD 8513 o przekątnej 12 cali lub CD 8512 o przekątnej 14 cali.

Wśród nowych drukarek, przeznaczonych do współpracy z IBM PS/2, można wymienić: IBM Proprinter II, X24, XL24 lub IBM Quietwriter III Printer.

Mikrokomputery IBM PS/2 to kolejna, nie ostatnia propozycja firmy. Rozwiązania modeli z uwagi na parametry są atrakcyjne i na pewno zadowolą użytkowników profesjonalnych, którym wystarczy proponowana rozdzielczość 1024 × 768 pixeli, pojemność pamięci operacyjnej i masowej oraz szybkość procesora i niezbyt długi czas dostępu do informacji.

Mikrokomputery będą chyba z powodzeniem wykorzystywane do prac inżynierskich w małych i średnich systemach CAD, CAE i CAM, do prac biurowych i innych, których wymienić nie jest łatwo.

Jacek WOJTALA



— A co umie robić pański robot?
— Przychodzi punktualnie po wypłatę pieniędzy.

Procedury ROM SPECTRUM (2)

Podstawową procedurą wydruku znaków na ekranie (drukarce) jest 16. Można ją wywoływać przez CALL 16, lecz warto pamiętać, że specyfika mikroprocesora Z80 pozwala na stosowanie krótszych, 1-bajtowych rozkazów RST.

4 PRINT	16	#10
CEL: wydruk znaku na ekranie/drukarce		
WE: A — kod znaku 0..31 — kody kontrolne 32..127 — kody znaków drukowalnych ASCII 128..143 — semigrafika 144..164 — znaki użytkownika, 165..255 — słowa kluczowe (tokens).		
WY: BC, DE HL, IX przechowane, używane rejestry pomocnicze (HL' przechowane)		

Procedura 4 uwzględnia aktualny kanał (odczyt CURCHL), atrybuty ekranu (ATTR-T MASK-T), tryby ekranowe (PFLAG), bada zmienne CHARS i UDG. Semigrafiki tworzone są w obszarze pamięci kalkulatora „na gorąco". Znaki o kodach 0..5, 7, 9..12, 14..15, 24..31 są drukowane jako znak zapytania.

5 PRINT OUT	2548	#09F4
CEL: jak w proc. 4,		
WE: jak w proc. 4;		
WY: IX przechowane, używane AF'.		

Procedura 5 może zastępować procedurę 4 wszędzie tam, gdzie nie jest wskazane używanie rejestrów pomocniczych. Nie odczytuje również zmiennej CURCHL.

6 POSITION FETCH	2819	#0B03
CEL: odczyt współrzędnych pozycji wydruku znaku na ekranie/drukarce.		
WE: nieistotne		
WY: dla ekranu głównego lub edycyjnego: B — rząd fizyczny, C — kolumna fizyczna, HL — adres pola wydruku w pamięci ekranu (górny z ośmiu bajtów) dla drukarki C — pozycja w buforze drukarki wskazująca ilość znaków pozostałych do wydruku, HL — adres w buforze zawierający bajt do wydruku; flaga Z = 1 dla ekranu głównego, flaga Z = 0 dla ekranu edycyjnego i drukarki, A, DE, IX przechowane		

Procedura 6 dokonuje odczytu odpowiednich dla używanego kanału zmiennych systemowych
- — dla ekranu głównego: S-POSN i DF-CC,
- — dla ekranu edycyjnego: S-POSNL i DF-CCL,
- — dla drukarki: P-POSN i PR-CC,

7 POSITION STORE	2780	#0ADC
CEL: ustawienie pozycji wydruku znaku na ekranie lub drukarce,		
WE: dla ekranu B — rząd fizyczny; C — kolumna fizyczna, HL — adres górnego pola bajtu w pamięci ekranu; dla drukarki C — pozycja fizyczna w buforze; HL — adres w buforze.		
WY: A, BC, DE, HL, IX przechowane;		

Procedury 6 i 7 stosują nieco inną konwencję współrzędnych ekranu niż użytkownik w BASIC'u. Przechodzenie jest stosunkowo proste:

rząd fizyczny = 24 — rząd logiczny (użytkownika)
kolumna fizyczna = 33 — kolumna logiczna

Przykład rozwieje ewentualne nieporozumienia: chcąc ustawić kursor w pozycji logicznej 4,5 (rząd, kolumna) dokonuje się następujących operacji:

```
LD      B,4
LD      C,5
LD      A,24
SUB     B
LD      B,A
LD      A,33
SUB     C
LD      C,A
```

WY: B,C — współrzędne fizyczne

Ustawiając pozycję należy podać również adres, co może być czasem dużym utrudnieniem, które pomoże pokonać Procedura 8.

8 C—ADDR	3739	#0E9B
CEL: obliczenie adresu pola znakowego dla podanego rzędu fizycznego		
WE: B — rząd fizyczny		
WY: HL — adres najwyższego z ośmiu bajtów pola wydruku flaga C = 0, BC, E, IX przechowane.		

Obliczenie adresu dla współrzędnych logicznych 4,5 wygląda następująco:

```
LD      B,4
LD      C,5
PUSH    BC
LD      A,24
SUB     B
LD      B,A
CALL    CL ADDR
LD      B,0
ADD     HL,BC
POP     BC
CALL    POS STORE
```

Ostatnia linia ustawia pozycję kursora według zawartości rejestrów BC (współrzędne) i HL (adres). Jeśli chcemy posługiwać się procedurą 8 podając współrzędną logiczną, to należy ją wywoływać pod adresem #0E9E i wprowadzać rząd w akumulatorze.

9 AT	8202	#200A
CEL: ustawienie pozycji wydruku znaku, WE: B — kolumna logiczna (brak konsekwencji), C — rząd logiczny;		
WY: jak w proc. 4		

10 TAB	8220	#201C
CEL: ustawienie kolumny wydruku znaku,		
WE: B — kolumna logiczna;		
WY: jak w proc. 4		

Procedury 8 i 9 pozwalają przesuwać kursor ekranu tekstowego wedle zasad przyjętych w BASIC'u. Obie używają procedury 4, więc są nieco wolniejsze od 7. **TAB** wydrukuje spacje od kolumny ostatniej pozycji do podanej.

11 PRINT CR	2639	#0A4F
CEL: wysłanie kodu 13 (#0D) na aktualny kanał;		
WE: nieistotne.		
WY: jak w proc. 4.		

Na ekranie będzie to nowa linia, a w drukarce powrót głowicy i przesunięcie papieru o jedną linię (line feed). Jeśli bufor był częściowo zapełniony, nastąpi jego wydruk.

12 PRINT QUEST	2665	#0A69
CEL: wydruk znaku zapytania		
WE: nieistotne		
WY: jak w proc. 4.		
13 PRINT COMMA		#0A5F
CEL: przeniesienie kolumny wydruku do następnej strefy (czyli do kolumny 0 lub 16) połączone z dodrukiem spacji.		
WE: nieistotne;		
WY: E, IX przechowane; używane rejestry pomocnicze (HL przechowane).		

Zamiennikiem instrukcji BASIC'a

PRINT PAPER 4; **INK** 1; **AT** 5,2; „A", „B".
będzie następująca sekwencja instrukcji w języku wewnętrznym

```
LD      A,33
LD      (ATTR_T), A
LD      B,2
LD      C,5
CALL    AT
LD      A, „A"
RST     16
CALL    PR COMMA
LD      A, „B"
RST     16
CALL    PRINT CR
```

14 CL—SET	3545	# 0DD9
CEL: obliczenie adresu ekranu dla fizycznych wartości rzędu i kolumny oraz uaktualnienie pozycji wydruku.		

WE: dla ekranu głównego:
bit 1, (FLAGS) = 0;
bit 0, (TVFLAG = 0;
B— rząd fizyczny; C— kolumna fizyczna;
dla ekranu edycyjnego;
bit 0, (TVFLAG) = 1;
B i C — jw.
dla drukarki
bit 1, (FLAGS) = 1;
C— pozycja fiz. w buforze drukarki;		

WY: przez proc. 7. BC, HL, IX przechowane.		
15 PRINT BACK	2595	# 0A23
CEL: przesunięcie pozycji wydruku na ekranie/drukarce o jedną pozycję w lewo;		
WE: B— rząd fizyczny; C— kolumna fizyczna.		
WY: przez proc. 14.		

Procedura 15, powodująca cofnięcie pozycji, wydaje się pomyłką, gdyż trudno inaczej wytłumaczyć szkolne błędy, jakie zawiera. Brakuje możliwości cofania z pozycji (1,0) do (0,31), przeoczony został także szczególny przypadek cofnięcia w pozycji (0,0). Po uruchomieniu programu

```
10 CLS: PRINT CHR$ 8;
20 PRINT PEEK 23688
```

pozycja wydruku znajduje się w obszarze atrybutów!

Poniższy program rozwiązuje ten problem.

```
 10 ;  ** BACKSPACE '86
 20 ; GENS         3M21    assembler
 30 FETCH          EQU     # 0B03
 40 CL_SET         EQU     # 0DD9
 45 ;
 50                ORG     60000
 60 START          CALL    FETCH
 70                LD      A, 33
 80                CP      C
 90                JR      NZ, NZERO
100                LD      A, 24
110                CP      B
120                RET     Z
130 NZERO          INC     C
140                LD      A, 34
150                CP      C
160                JR      NZ, WYJ
170                BIT     1, (IY + 1)
180 ;       (IY + 1) = (FLAGS)
190                JR      NZ, NLINIA
200                INC     B
210                LD      C, 2
220                LD      A, 25
230                CP      B
240                JR      NZ, WYJ
250                DEC     B
260 NLINIA         LD      C, 33
270 WYJ            JP      CL_SET
```

Linie 70...120 eliminują cofnięcie z pozycji (0,0). Program można uruchamiać z poziomu BASIC'a, zastępując instrukcję
PRINT CHR$ 8

K. MAMCARZ



Dasz radę wykołować go, tak jak kołujemy majstra?



TMMT '87

Techniczna
młod

W Łodzi w dniach 27
resująca wystawa, na k
zentowali swoje o
Główna część ekspoz
łazków i nowatorskich
propozycje dla rodzim
cej pomysłów dotycz
sów i operacji produk
przemysłowych), a ta
nologii (światłowody)
oczekiwanych na ryn
cznych (magnetowidy,
firm komputerowych i
towało swe najnowsz
komputer BOSMAN z
Ekspozycję informat
ska klubów mikrokomp
na wystawie również
Swój dorobek prezento
czych z Poznania, Kł
jednostki Warszawskie
wy Klub Mikrokompute
cy przy Ośrodku Obli
WSOWŁ.
Imprezę ubarwiały
muzyki młodzieżowej
kaset wideo.





a twórczość
dzieży

X—5 XI odbyła się inte-
której młodzi twórcy pre-
iągnięcia innowacyjne.
ycji to ponad 150 wyna-
rozwiązań stanowiących
nego przemysłu. Najwię-
ło automatyzacji proce-
cyjnych (modele robotów
ze nowoczesnych tech-
Przedstawiono projekty
nku urządzeń elektroni-
plotery). Kilka uznanych
elektronicznych prezen-
ze produkty (np. mikro-
UNIMOR-u)
tyczną wzbogacały stoi-
puterowych. Nie zabrakło
przedstawicieli wojska
owały Klub Wojsk Lotni-
ub Mikrokomputerowy z
ego OW oraz Garnizono-
erowy TELEBIT działają-
czeniowym zegrzyńskiej

pokazy mody, koncerty
oraz projekcje filmów z

K. M.





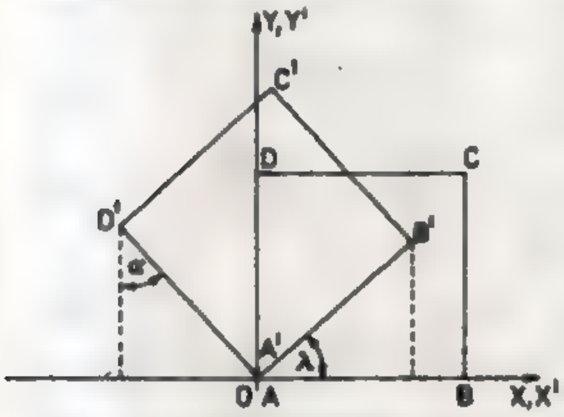
Grafika na ekranie (2)

OBRÓT

Z. MAJCHRZAK

Istotnym przekształceniem, wykorzystywanym w grafice komputerowej, jest obrót względem wybranego punktu o żądany kąt. Przekształcenie to może być wykorzystywane do obrotu zdefiniowanych figur lub całych obrazów. Jednym z punktów, wokół którego wykonywany jest obrót, może być początek układu współrzędnych.

W numerze tym omówimy obrót na płaszczyźnie względem początku układu współrzędnych z wykorzystaniem rachunku macierzowego. Uogólnioną macierz 2 × 2, która pozwala realizować omawianą funkcję można wyznaczyć, rozpatrując obrót jednostkowego kwadratu (rys. 1.) w kierunku dodatnim, tzn. przeciwnym do kierunku obrotu wskazówek zegara.



Rys. 1. Obrót jednostkowego kwadratu o kąt α

Jak widać na rysunku, punkt B, mający przed obrotem współrzędne (1,0) przekształcił się w punkt B' o współrzędnych ($x' = (1) \cdot \cos\alpha$, $y' = (1) \cdot \sin\alpha$), a punkt D o współrzędnych (0,1) w punkt D' o współrzędnych ($x' = -(1) \cdot \sin\alpha$, $y' = (1) \cos\alpha$). Po zastosowaniu ogólnego przekształcenia na kwadracie jednostkowym ABCD otrzymujemy:

$$\begin{matrix} A \rightarrow \\ B \rightarrow \\ C \rightarrow \\ D \rightarrow \end{matrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ a & b \\ a+c & b+d \\ c & d \end{bmatrix} \begin{matrix} \rightarrow A' \\ \rightarrow B' \\ \rightarrow C' \\ \rightarrow D' \end{matrix}$$

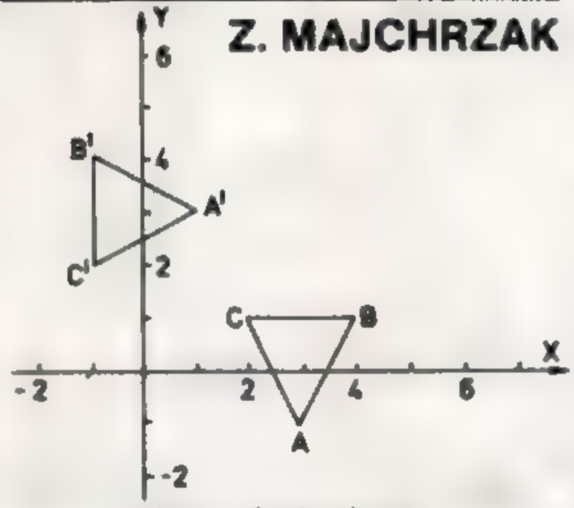
Z powyższego wynika, że współrzędne punktu B' określa się pierwszym wierszem macierzy transformacji, a współrzędne punktu D' drugim.

Z zestawienia powyższych wyników otrzymujemy postać poszukiwanej macierzy transformacji.

$$\begin{bmatrix} \cos & \sin \\ -\sin & \cos \end{bmatrix}$$

Otrzymana macierz transformacji nie powoduje zmiany skali i kształtu figury, lecz zmienia położenie na płaszczyźnie. Dla zilustrowania problemu rozpatrzymy trójkąt ABC, pokazany na rys. 2.

Obrót trójkąta ABC o 90° względem początku układu współrzędnych w kierunku



Rys. 2 Obrót trójkąta

przeciwnym do kierunku obrotu wskazówek zegara otrzymujemy poprzez transformację wierzchołków macierzą

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

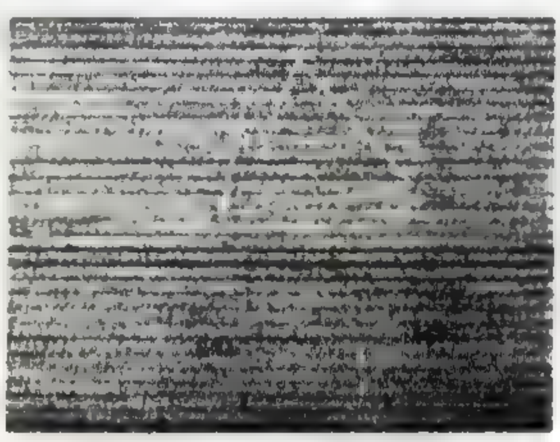
Otrzymujemy nowe współrzędne wierzchołków

$$\begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 4 \\ -1 & 2 \end{bmatrix}$$

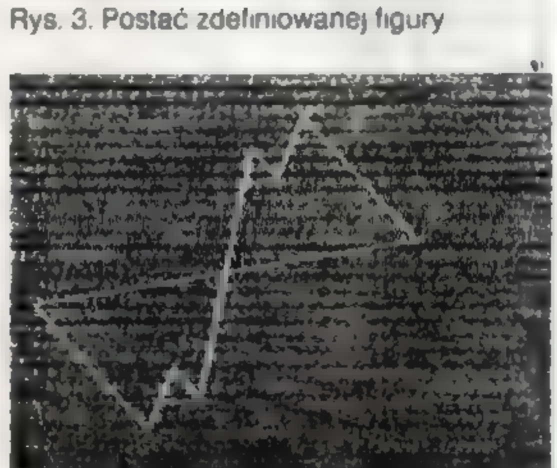
ków trójkąta ABC, które tworzą trójkąt A'B'C'. Obrót o 180° w kierunku dodatnim uzyskujemy dzięki macierzy

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \text{ a o 270° macierzy } \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Przedstawiona tematyka została zastosowana w prezentowanym programie realizującym obroty figur płaskich wokół początku układu współrzędnych. Program realizuje obroty jednokrotne o zadany kąt oraz wielokrotne w określony przyrost kąta.



Rys. 3. Postać zdefiniowanej figury



Rys. 4. Obraz powstały w wyniku obrotu figury o kąt 180°



Rys. 5. Obraz powstały w wyniku obrotu figury w 3 krokach o kąt 90°



Rys. 6. Obraz powstały w wyniku obrotu figury w 50 krokach o kąt 43°

Gdy zadana wartość kąta jest dodatnia mamy obrót przeciwny do kierunku obrotu wskazówek zegara i zgodny, gdy wartość kąta jest ujemna.

Otoczenie i sposób obsługi programu zostały ustalone w podobny sposób, jak w prezentowanym poprzednio, w ramach tego cyklu programie skalowania.

Obraz jest definiowany poprzez wprowadzenie współrzędnych (x,y) punktów wyznaczających kolejno kreślone odcinki. Współrzędne wprowadza się na zaproszenie „podaj wsp punktu (x,y)". W celu zakończenia definiowania figury należy wprowadzić punkt o współrzędnej x > 998 oraz dowolnej wartości o współrzędnej y.

Po zakończeniu definiowania figury pojawia się komunikat „pojedynczy — j, wielo — w", pozwalający wybrać rodzaj obrotu. Wprowadzenie litery „j" oznacza obrót jednokrotny o kąt, który należy wprowadzić po komunikacie: „podaj kąt". Wprowadzenie litery „w" powoduje obrót figury w określonej ilości kroków o zadany przyrost kąta w jednym kroku.

Wartości te wprowadzamy po ukazaniu się komunikatów: „podaj krok", „podaj ilość kroków".

Po wykonaniu przez program obrotu pojawiają się następne komunikaty określające operacje, które możemy wykonać na otrzymanym obrazie.

Wyboru operacji dokonujemy przez wprowadzenie określonej litery, podanej w treści komunikatu.

Dla zilustrowania działania programu zamieszczamy przykładowe wyniki uzyskane na figurze zdefiniowanej punktami: (0,0), (20,120); (40,90), (60,150); (150,30); (0,0).

```
30 REM Obroty
60 DIM wsp(101,2),mat(2,2),nwsp(101,2)
70 MODE 1:INK 0,6:INK 1,13:INK 2,1:INK 3,24
80 BORDER 0:PEN 1:PAPER 2
90 CLS
100 PRINT "Rysowanie figur-poligonow"
110 PRINT"Obrot w jednym kroku lub"
120 PRINT"obrot w dowolnej ilosci krokow"
130 INPUT "Start gdy wcisniesz t"; nk$
140 IF nk$="t" THEN 160
150 GOTO 130
160 GRAPHICS PAPER 1:REM Tlo grafiki
170 ORIGIN 320,208,0,640,16,400
180 CLEAR INPUT
190 WINDOW 1,40,25,25
200 GOSUB 1220:REM Rysowanie ukladu wspol.
210 i=0 :REM  Licznik punktow
220 INPUT "podaj wsp. punktu x,y";x,y
230 IF x>998 THEN 300
240 i=i+1
250 IF i>100 THEN GOTO 300
260 wsp(i,1)=x
270 wsp(i,2)=y
280 GOSUB 830:REM Rysowanie odcinka
290 GOTO 220
300 INPUT "pojedynczy-j,wielo-w";jk$
310 IF jk$="j" THEN 340
320 IF jk$="w" THEN 650
330 GOTO 300
340 INPUT "podaj kat";ak
350 ak=ak*PI/180
360 GOSUB 900 :REM Ustaw macierz przekszta1.
370 GOSUB 1020:GOSUB 1130:GOTO 380
380 INPUT "poczatek-r,koniec-k";nk$
390 IF nk$="r" THEN 420
400 IF nk$="k" THEN 1420
410 GOTO 380
420 INPUT "kasowanie obrazu-k,rysowanie-r";nk$
430 IF nk$="k" THEN 460
440 IF nk$="r" THEN 730
450 GOTO 420
460 INPUT"nowego obrazu-n,starego-s";nk$
470 IF nk$="n" THEN 610
480 IF nk$="s" THEN 500
490 GOTO 460
500 REM Kasowanie starego obrazu
510 GOSUB 1220:REM Rys. ukladu wspolrzednych
520 FOR k=1 TO i
530 wsp(k,1)=nwsp(k,1):wsp(k,2)=nwsp(k,2)
540 NEXT k
550 GRAPHICS PEN 3
560 GOSUB 1320
570 INPUT "obrot-o,dorysowanie-d";nk$
580 IF nk$="o" THEN 300
590 IF nk$="d" THEN 220
600 GOTO 570
610 REM Kasowanie nowego obrazu
620 GOSUB 1220
630 GRAPHICS PEN 3
640 GOSUB 1320:GOTO 570
650 INPUT "podaj krok";kr
660 INPUT "podaj ilosc krokow";nk
670 kr=kr*PI/180
680 FOR ii=1 TO nk
690 ak=ii*kr
700 GOSUB 900:GOSUB 1020:GOSUB 1130
710 NEXT ii
720 GOTO 380
730 FOR n=1 TO i:REM Rysowanie od poczatku
740 FOR k=1 TO 2
750 wsp(n,k)=0:nwsp(n,k)=0
760 NEXT k
770 NEXT n
780 GOTO 200
800 REM   PODPROGRAMY
820 REM
830 REM Rysowanie punktu lub odcinka
840 REM
850 GRAPHICS PEN 3
860 IF i=1 THEN PLOT wsp(i,1),wsp(i,2)
870 IF i>1 THEN DRAW wsp(i,1),wsp(i,2)
880 RETURN
890 REM
900 REM Ustawianie macierzy
910 REM
920 sk=SIN(ak):ck=COS(ak)
930 mat(1,1)=ck:mat(1,2)=sk
940 mat(2,1)=-sk:mat(2,2)=ck
950 FOR k=1  TO i:REM Zerowanie nwsp
960 FOR l=1 TO 2
970 nwsp(k,l)=0
980 NEXT l
990 NEXT k
1000 RETURN
1010 REM
1020 REM Mnozenie macierzy
1030 REM
1040 FOR n=1 TO i
1050 FOR k=1 TO 2
1060 FOR m=1 TO 2
1070 nwsp(n,k)=nwsp(n,k)+wsp(n,m)*mat(m,k)
1080 NEXT m
1090 NEXT k
1100 NEXT n
1110 RETURN
1120 REM
1130 REM Rysowanie nowej figury
1140 REM
1150 GRAPHICS PEN 0:REM nowe wspolrzedne
1160 PLOT nwsp(1,1),nwsp(1,2)
1170 FOR n=2 TO i
1180 DRAW nwsp(n,1),nwsp(n,2)
1190 NEXT
1200 RETURN
1210 REM
1220 REM Rysowanie ukladu wspolrzednych
1230 REM
1240 CLG 1
1250 GRAPHICS PEN 2:REM Pioro wspolrzednych
1260 PLOT 0,-208
1270 DRAW 0,208
1280 PLOT -320,0
1290 DRAW 320,0
1300 RETURN
1310 REM
1320 REM Rysowanie starej figury
1330 REM
1340 PLOT wsp(1,1),wsp(1,2)
1350 FOR n=2 TO i
1360 DRAW wsp(n,1),wsp(n,2)
1370 NEXT n
1380 RETURN
1390 REM
1400 REM Zakonczenie
1410 REM
1420 WINDOW 1,40,1,25
1430 CLS
1440 CLG 1
1450 END
```

MICHAŁ URSUS

Gdy wyłączą prąd...

— Rozpoczynamy przesłuchanie Edgara Quaill, podejrzanego o zamordowanie swojej żony. Głos ma oskarżony!

— Wysoki sądzie! W dobie postępującej komputeryzacji i automatyzacji życia nasze dotychczasowe powłoki zaczynają być bezużyteczne, często hamując postęp naszej wspaniałej cywilizacji. Nasze ciało ma takie wygórowane wymagania, jak określona temperatura otoczenia, oddychanie tlenem, zapotrzebowanie na znaczną w skali globalnej ilość materii organicznej jako pokarmu; że nie wspomnę o takich utrudnieniach, jak golenie czy mycie. Już kilkanaście lat temu odkryłem możliwość przeniesienia informacji z mózgu żyjącego organizmu do pamięci dowolnego komputera. Jest to wspaniała szansa dla naszego świata!!! Wyobraźmy sobie...

— Niech oskarżony powie jeszcze sądowi o skutkach ubocznych...

— Hm, no tak... Podczas kopiowania w organiźmie dawcy następują nieodwracalne zmiany, spowodowane przez... zresztą, nieważne.. W każdym razie zmiany te mogą doprowadzić do śmierci, ale jedynie w b i o l o g i c z n y m tego słowa znaczeniu! Umysł nadal żyje, a w dodatku obdarzony jest nowymi, wspaniałymi możliwościami! Wyobraźmy sobie, że zamiast tego zlepku białek mamy poprzez nasz elektroniczny mózg dostęp, i to bezpośredni, do wszystkich wynalazków techniki. Możemy na przykład sterować sondami, badającymi przestrzeń kosmiczną, i czuć się częścią ich orga... przepraszam, mechanizmu. W ten sposób każdy będzie mógł zrealizować swoje dziecięce marzenia o podróżach kosmicznych, nie narażając przy tym swojego istnienia na żadne ryzyko, zostawiwszy duplikat swojego mózgu na przykład w Banku Pamięci.

Każdy wie, że jednym z odwiecznych marzeń człowieka było przedłużenie swojego życia. Dobrze funkcjonujące mechanizmy dadzą nam tę możliwość — mamy szansę stać się nieśmiertelni!

— Czyżby żona oskarżonego stała się rzeczywiście nieśmiertelna?

— To nie była moja wina! Skąd mogłem wiedzieć, że akurat wtedy elektrownia wyłączy mi dopływ prądu?! To był tylko tragiczny wypadek, że cała zawartość mózgu mojej żony została skasowana, zanim zapisałem ją na dyskietkę. Mój komputer jest modelem starszego typu i nie ma własnego, awaryjnego źródła napięcia...



Charakterystyka zestawu podstawowego

I Jednostka centralna
- mikroprocesor: Intel 8088-2/NEC V20
- zegar procesora 4.77/8 MHz
- koprocesor arytmetyczny 8087 (opcjonalnie)
- pamięć operacyjna 256KB do 1024KB
- pamięć stała ROM 8KB do 64KB (BIOS i Basic)
- pamięć masowa: 2 jednostki dysków elastycznych 360KB, do 2 jednostek dysków typu Winchester 20MB, szybka pamięć taśmowa 20 do 60 MB
- pakiet wizyjny: O tryb tekstowy — max 80 znaków w 25 wierszach, O tryb graficzny — rozdzielczość 640 x 200 (czarno-biały), 320 x 200 (kolor), O wyjścia: RGB i sygnał zespolony
- interfejsy: O równoległy Centronics (drukarka), O szeregowy RS232
- zegar czasu rzeczywistego z podtrzymaniem bateryjnym
- 8 gniazd do podłączenia dodatkowych pakietów (grafika, specjalizowane kontrolery, karty prototypowe itp.)
- zasilacz 150 W o wysokiej sprawności

Sumy kontrolne

W numerze czwartym "X" (1987) opublikowano program pt „Edytor Basic-a". Celem jego było ułatwienie bezbłędnego wprowadzenia wydrukowanych programów, a w razie wystąpienia błędu łatwe jego odszukanie. Ten sam cel przyświecał opracowaniu poniższego programu. Otrzymywane za jego pomocą sumy kontrolne są identyczne jak sumy uzyskane dzięki powyżej wspomnianemu programowi.

Po uruchomieniu programu (RUN), ukazuje się nam menu. Mamy do wyboru cztery możliwości: wyświetlanie sum kontrolnych na ekranie, drukowanie ich na drukarce, powrót programu do Basic-a i dla posiadaczy stacji dysków — wywołanie zawartości dyskietki. W przypadku wybrania opcji drukowanie lub wyświetlanie należy: gdy wprowadzamy program z dyskietki, wpisać nazwę programu, a gdy z magnetofonu, po poleceniu wpisania nazwy programu, trzeba wcisnąć klawisz RETURN. Program kolejno wypisuje etykiety oraz przyporządkowane im sumy kontrolne w sposób, jak na załączonym przykładzie.

Na koniec jeszcze jedna uwaga. Przed przystąpieniem do sprawdzania, powinniśmy dany program zapisać na taśmie lub dyskietce instrukcją LIST.

Janusz W. JANIEC

```
100 REM ************************
105 REM *                      *
110 REM *   SUMY   KONTROLNE   *
115 REM *                      *
120 REM ************************
125 DIM A$(130),B$(130),C$(6),D$
(10),A(5),N$(15),AN$(14),SUM$(2)
130 DIM ME$(18):N$="D:":B=1:E=0
135 OPEN #3,8,0,"E:":SETCOLOR 2,
0,0:SETCOLOR 1,0,10:POKE 752,1
140 REM **** OPCJE PROGRAMU ****
:POKE 82,4:? CHR$(125):POKE 752,
1:SETCOLOR 2,0,0:SETCOLOR 1,0,10
145 POSITION 2,1:? "SUMY KONTROL
NE":POSITION 2,4:? CHR$(177);" W
YSWIETLANIE SUMY KONTROLNEJ"
150 POSITION 2,6:? CHR$(178);" D
RUKOWANIE SUMY KONTROLNEJ":POSIT
ION 2,8:? CHR$(179);" POWROT DO
BASIC-a"
155 POSITION 2,10:? CHR$(180);"
MENU DYSKIETKI":POSITION 9,13:?
"     >?<";
160 CLOSE #1:OPEN #1,4,0,"K:":GE
T #1,K:? CHR$(30);CHR$(30);CHR$(
K+128):IF K<49 OR K>52 THEN 160
165 IF K=49 THEN 190
170 IF K=51 THEN 490
172 IF K=52 THEN 500
175 REM ** WYDRUK NA DRUKARCE **
180 TRAP 450:CLOSE #3:OPEN #3,8,
0,"P:"
185 REM *** OTWARCIE  ZBIORU ***
190 POSITION 2,15:? "Podaj nazwe
 zbioru do sprawdzenia"
191 POSITION 7,16:? CHR$(17);:FO
R L1=1 TO 15:? CHR$(18);:NEXT L1
:? CHR$(5)
195 POSITION 7,17:? "|D:
     |":POSITION 7,18:? CHR$(26)
;:FOR L1=1 TO 15:? CHR$(18);:NEX
T L1:? CHR$(3)
200 POSITION 10,17:? ;:INPUT AN$
205 POKE 752,1
210 IF AN$(1,1)=" " THEN N$="C:"
:POSITION 8,17:? "C:":GOTO 225
215 SK=LEN(AN$):N$(3,SK+2)=AN$
220 TRAP 235
225 CLOSE #1:OPEN #1,4,0,N$
230 GOTO 240
235 ? :? "Nie ma takiego zbioru
na dyskietce !":GOSUB 460:RUN
240 POKE 82,0:P=0:B=1:E=0:? CHR$
(125);" TWORZENIE SUM KONTROLNYC
H...":?
265 REM ********* OPIS *********
270 A$="     * SUMA KONTROLNA /
ETYKIETA *":? #3;A$
275 A$=" ----------------------
--------------":? #3;A$
280 REM ** POBRANIE  ETYKIETY **
285 C=1:TRAP 335
290 GET #1,Z
295 IF Z=32 THEN C$(C,C)=CHR$(32
):GOTO 310
300 C$(C,C)=CHR$(Z):C=C+1:GOTO 2
90
310 IF E=1 THEN 435
315 REM **** POBRANIE LINII ****
320 B$="":B$=C$:T=C+1:C=0
325 GET #1,Z:IF Z=155 THEN 340
330 B$(T+C,T+C)=CHR$(Z):C=C+1:GO
TO 325
335 E=1:GOTO 310:REM * KONIEC  *
340 REM ****** SUMA LINII ******
345 C=0:ANS=0
350 FOR A=1 TO LEN(B$)
355 C=C+1:ANS=ANS+(C*ASC(B$(A,A)
)):NEXT A
360 KOD=INT(ANS/676):KOD=ANS-(KO
```

```
D*676)
365 HKOD=INT(KOD/26):LKOD=KOD-(H
KOD*26)+65
370 HKOD=HKOD+65
375 SUM$(1,1)=CHR$(HKOD):SUM$(2,
2)=CHR$(LKOD)
380 REM *** FORMAT  ETYKIETY ***
385 D$="     ":L=VAL(C$):A=LEN(ST
R$(L)):D$(6-A)=STR$(L):GOTO 410
395 ? CHR$(125);"BLEDNE DANE !!!
";CHR$(125):END
400 REM ** DRUKOWANIE  WYNIKU **
405 REM ***** NA EKRANIE *****
410 IF P=0 THEN CC=1:X=2:P=1:A$=
" ":A$(38)=" ":A$(2)=A$:GOTO 425
415 X=X+10:IF X>=39 THEN X=2:Y=Y
+1:? #3;A$:A$=" ":A$(38)=" ":A$(
2)=A$
425 A$(X,38)=SUM$:A$(X+2,38)=CHR
$(32):A$(X+3,39)=D$
430 FOR LI=10 TO 30 STEP 10:A$(L
I,LI)="|":NEXT LI:GOTO 280
435 ? #3;A$
440 A$=" -------------------------
------ --------- ":? #3;A$
445 CLOSE #1:GOSUB 460:RUN
450 ? :? :? "Drukarka nie dziala
 !":? :? :GOSUB 460:RUN
455 REM *** WCISNIJ  KLAWISZ ***
460 CLOSE #1:OPEN #1,4,0,"K:"
465 ? "      ";CHR$(17);:FOR L1=1
 TO 23:? CHR$(18);:NEXT L1:? CHR
$(5)
470 ? "      |Wcisnij dowolny kla
wisz|"
475 ? "      ";CHR$(26);:FOR L1=1
 TO 23:? CHR$(18);:NEXT L1:? CHR
$(3)
480 GET #1,Z:CLOSE #1:RETURN
485 REM ** POWROT  DO BASIC-a **
490 GRAPHICS 0:END :REM *******
500 REM **** MENU DYSKIETKI ****
505 TRAP 525
510 CLOSE #3:OPEN #3,6,0,"D:*.*"
:FOR KO=0 TO 64:INPUT #3,ME$
520 ? ME$:NEXT KO
525 CLOSE #3
530 GOSUB 455:GOTO 135:REM *****
```

* SUMA KONTROLNA / ETYKIETA *

PC	100	YF	105	XS	110	YH	115
PG	120	YB	125	OI	130	YM	135
TS	140	LV	145	UJ	150	RB	155
ND	160	SZ	165	RO	170	MI	172
TF	175	EM	180	HJ	185	RY	190
KF	191	UE	195	HK	200	JV	205
UF	210	BN	215	NU	220	ES	225
NP	230	LJ	235	YJ	240	MK	265
VD	270	JY	275	KY	280	TV	285
NY	290	ML	295	WG	300	YK	310
GS	315	TK	320	SP	325	XI	330
CK	335	WJ	340	GN	345	IG	350
JJ	355	WD	360	QE	365	PY	370
OZ	375	YC	380	DS	385	OO	395
SV	400	FP	405	LQ	410	OH	415
IF	425	XI	430	SG	435	GY	440
QQ	445	IR	450	PW	455	XK	460
PZ	465	SK	470	KW	475	JI	480
AV	485	TQ	490	NJ	500	PB	505
YJ	510	SY	520	MS	525	KJ	530

Mapa dysku

Program DYSKMAP tworzy i wyświetla na ekranie monitora mapę dysku. Pod pojęciem tym kryje się wykaz wszystkich sektorów dysku z zaznaczeniem, który z nich jest wolny (niezapisany), a który zajęty (zapisany). Program umożliwia tworzenie mapy dysku sformatowanego z pojedynczą gęstością zapisu (DOS 2.0—720 sektorów) oraz z podwójną gęstością (DOS 2.5—1040 sektorów)

Program po uruchomieniu żąda podania liczby sektorów zawartych na dyskietce. Jezeli nie jesteśmy pewni, z jaką gęstością zapisana jest dyskietka, to podajemy 1040 Program jest tak skonstruowany, że gdy wprowadzimy liczbę sektorów równą 1040, to sprawdza, z jaką gęstością jest rzeczywiście sformatowana dyskietka. Informacja o tym zapisana jest w sektorze o numerze 360, w bajcie pierwszym i drugim (licząc od zera), zwanym tabelą zawartości wolumenu (dyskietki) — ang. Volume Table of Contents. Jeżeli w wyniku sprawdzenia okaże się, że na dyskietce jest 720 sektorów, to program sam poprawi wprowadzoną przez nas liczbę Wprowadzenie liczby, na przykład 100, spowoduje, ze program utworzy i wyświetli mapę zawierającą informacje o pierwszych stu sektorach dyskietki

Mapa dysku zawierającego 720 sektorów wyświetlana jest na ekranie w postaci 20-wierszowej tablicy zawierającej po 36 znaków w każdym wierszu. Każdy element (znak) z tej tablicy opisuje stan jednego sektora Numer sektora, któremu odpowiada znak z tablicy, odczytuje się przez dodanie liczby zapisanej na początku wiersza do liczby zapisanej na dole kolumny, zawierających dany znak. Znaczenie znaków z tablicy jest następujące

Gwiazdka — oznacza sektor zapisany, kropka — oznacza sektor wolny, wykrzyknik — oznacza sektor uszkodzony, tzn. taki, który przy próbkie odczytu generuje błąd

Tę ostatnią cechę programu mozna wykorzystać do wykrywania i oznaczania tak zwanych bad sektorów, stosowanych przez producentów oprogramowania do jego zabezpieczenia przed kopiowaniem

Mapa dysku sformatowanego z podwójną gęstością zapisu (1040 sektorów) składa się z dwóch części Część pierwsza opisuje sektory od 1 do 520, a część druga sektory od 521 do 1040 Każda z części jest również tablicą, analogiczną do opisanej poprzednio, zawierającą 20 wierszy 26-znakowych. Klawisz OPTION przełącza wyświetlanie części pierwszej na drugą i odwrotnie Po przeanalizowaniu utworzonej mapy mozna program uruchomić ponownie, wciskając klawisz START

Program DYSKMAP zawiera kilka procedur napisanych w kodzie maszynowym (linie DATA programu). Szczególną uwagę proponuję zwrócić na procedurę umieszczoną w 40-bajtowej stałej tekstowej o nazwie S — linie 802, 806 Jej funkcją jest odczyt lub zapis sektora o dowolnym numerze. Zwracam uwagę na tę procedurę, poniewaz odczyt i zapis sektora za pomocą instrukcji języka BASIC (NOTE, POINT, PUT, GET) jest dosyć kłopotliwy (wymaga skonstruowania pętli) i trwa znacznie dłużej. Procedurę tę mozna wykorzystać w każdym programie w języku BASIC Sposób jej wywołania jest następujący

X = USR(PAR1,PAR2,PAR3,PAR4,PAR5)

gdzie

PAR1 — adres pamięci RAM, od którego umieszczona jest procedura,

PAR2 — numer napędu dyskowego (1—4),

PAR3 — numer sektora do odczytu lub zapisu,

PAR4 — adres 128-bajtowego bufora, do którego mają być

wczytane informacje z sektora, a przy zapisie informacje zawarte w tym buforze zostaną zapisane do sektora

PAR5 — kod funkcji do wykonania, 82 — odczyt sektora, 87 — zapis sektora

Przykładowo, jeżeli procedurę umieściliśmy w tablicy PROC$ i chcemy wczytać sektor o numerze 360 do tablicy BUF$, to proce-

X = USR(ADR(PROC$),1,360,ADR(BUF$),82)

Odczyt nastąpi ze stacji dysków o numerze 1

Po wykonaniu procedury zmienna X zawierać będzie informacje o tym, czy operacja odczytu bądź zapisu zakończyła się pomyślnie. Wartość zmiennej X różna od jedności oznacza, że przy odczycie lub zapisie wystąpił błąd

H. KRASUSKI

```
5 ? CHR$(125):? "Prosze chwile p
oczekac..."
6 GOTO 100
10 ? CHR$(125);"        *** DYSKMA
P (C) 1987 ***"
20 ? :? :? "Wloz dyskietke do st
acji dyskow"
30 ? "Nacisnij [RETURN]"
31 ? "[SPACJA] Wyjscie z program
u"
34 CLOSE #1:OPEN #1,4,0,"K:"
35 GET #1,K
39 IF K=32 THEN POKE 752,0:END
40 IF K<>155 THEN 35
45 RETURN
100 DIM LO$(19),DMAP$(117),MAP$(
1041),OKNO$(107),L$(60),L2$(60),
L3$(80),SE$(128),S$(40)
110 RESTORE 200
120 FOR I=1 TO 19:READ A:LO$(I)=
CHR$(A):NEXT I:GOTO 300
200 DATA 104,104,133,204,104,133
,203,160,0,169,0,145,203,200,192
,240,208,249,96
300 RESTORE 350
310 FOR I=1 TO 117:READ A:DMAP$(
I)=CHR$(A):NEXT I:GOTO 400
350 DATA 104,104,133,204,104,133
,203,104,104,141,1,3,104,133,207
,104,133,206,169,4
352 DATA 141,5,3,169,0,141,4,3,1
69,82,141,2,3,160,1,162,0,169,1,
141,10,3,142,11,3,138,72,152,72,
32
354 DATA 83,228,160,0,173,3,3,20
1,1,208,24,190,0,4,208,12,200,19
2,128,208,246,104,168,169,14,56
356 DATA 176,11,104,168,169,10,5
6,176,4,104,168,169,1,145
358 DATA 203,104,170,200,208,3,2
32,230,204,228,207,208,4,196,206
,240,9,140,10,3,142,11,3,56,176,
185,96
400 RESTORE 450
410 FOR I=1 TO 107:READ A:OKNO$(
I)=CHR$(A):NEXT I:GOTO 500
450 DATA 104,104,133,206,104,133
,205,104,133,118,104,133,117,104
,133,208,104,133,207,104,104
452 DATA 133,209,104,133,120,104
,133,119,165,88,133,203,165,89,1
33,204,165,203,101,117,133,203,1
65,204
454 DATA 101,118,133,204,160,0,1
77,203,170,177,205,145,203,138,1
45,205,200,196,209,208,241,169,0
,198,207
456 DATA 197,207,208,6,197,208,2
40,28,198,208,24,152,101,205,133
,205,144,3,230,206
458 DATA 24,165,203,101,119,133,
203,165,204,101,120,133,204,24,1
44,199,96
500 RESTORE 550
510 FOR I=1 TO 60:READ A:L$(I)=C
HR$(A):NEXT I:GOTO 600
550 DATA 0,0,16,0,19,22,0,23,18,
17,16,24,17,20,20,17,24,16,18,1,
,22,18,21,18,18,24,24,19,18,20,1
9,22,16,19
552 DATA 25,22,20,19,18,20,22,24
,21,16,20,21,19,16,21,23,22,22,1
7,18,22,20,24,22,24,20
600 RESTORE 650
610 FOR I=1 TO 60:READ A:L2$(I)=
CHR$(A):NEXT I:GOTO 700
650 DATA 0,0,16,0,18,22,0,21,18,
0,23,24,17,16,20,17,19,16,17,21,
22,17,24,18,18,16,24,18,19,20,18
,22,16,18
652 DATA 24,22,19,17,18,19,19,24
,19,22,20,19,25,16,20,17,22,20,2
0,18,20,22,24,20,25,20
700 RESTORE 750
710 FOR I=1 TO 80:READ A:L3$(I)=
CHR$(A):NEXT I:GOTO 800
750 DATA 0,21,18,16,0,21,20,22,0
,21,23,18,0,21,25,24,0,22,18,20,
0,22,21,16,0,22,23,22,0,23,16,18
,0,23,18,24
752 DATA 0,23,21,20,0,23,24,16,0
24,16,22,0,24,19,18,0,24,21,24,
0,24,24,20,0,25,17,16,0,25,19,22
,0,25,22,18
754 DATA 0,25,24,24,17,16,17,20
800 SE$(1)=CHR$(0):SE$(128)=SE$:
SE$(2)=SE$
802 RESTORE 805:FOR I=1 TO 40:RE
AD A:S$(I)=CHR$(A):NEXT I:GOTO 8
10
805 DATA 104,104,104,141,1,3,104
,141,11,3,104,141,10,3,104,141,5
,3,104,141,4,3,104,104,141,2,3,3
2,83,228
806 DATA 173,3,3,133,212,169,0,1
33,213,96
810 GOSUB 10:? :? "Podaj liczbe
sektorow [720] lub [1040] "
820 INPUT DO
830 IF DO=1040 THEN GOTO 900
840 X=USR(ADR(LO$),40000+17*40):
GOTO 5510
900 SEK=360:X=USR(ADR(S$),1,SEK,
ADR(SE$),82):ALA=ADR(SE$):LICZS=
PEEK(ALA+1)+256*PEEK(ALA+2)
901 IF LICZS=1010 THEN GOTO 840
902 ? :? "Dyskietka o pojedyncze
j gestosci":? "Liczba sektorow w
ynosi 720":DO=720:GOTO 840
5510 ? :? :? "Czytanie dysku "
5520 ? :? "Czekaj..."
5530 MAP$="":MAP$(DO,DO)=CHR$(0)
5540 X=USR(ADR(DMAP$),ADR(MAP$)-
1,1,DO+1)
5550 IF DO=1040 THEN 5700
5560 POKE 752,1:? CHR$(125):POSI
TION 0,20:? CHR$(30);CHR$(31);
5570 X=USR(ADR(OKNO$),ADR(MAP$),
4,20,36,40):X=USR(ADR(OKNO$),ADR
(L$),0,20,3,40)
5580 POSITION 4,20:? "000000001
111111111222222222233333333  1234
56789012345678901234567890123456
"
5590 POSITION 15,23:? "[START] =
 NOWA MAPA";
5600 IF PEEK(53279)=6 THEN 5620
5610 GOTO 5600
5620 X=USR(ADR(OKNO$),ADR(L$),0,
20,3,40):GOTO 810
5700 POKE 752,1:? CHR$(125):POSI
TION 0,20:? CHR$(30);CHR$(31);
5710 POSITION 9,20:? "000000001
1111111112222222                1234
5678901234567890123456"
5715 X=USR(ADR(OKNO$),ADR(L2$),5
,20,3,40)
5720 X=USR(ADR(OKNO$),ADR(MAP$),
9,20,26,40)
5730 POSITION 3,23:? "[START]= N
OWA MAPA  [OPTION]= 2 CZESC";
5740 IF PEEK(53279)=6 THEN 5770
5750 IF PEEK(53279)=3 THEN 5800
5760 GOTO 5740
5770 X=USR(ADR(OKNO$),ADR(L2$),5
,20,3,40):GOTO 810
5800 X=USR(ADR(OKNO$),ADR(MAP$),
9,20,26,40)
5802 X=USR(ADR(OKNO$),ADR(L2$),5
,20,3,40)
5805 X=USR(ADR(OKNO$),ADR(L3$),4
,20,4,40)
5810 X=USR(ADR(OKNO$),ADR(MAP$)+
520,9,20,26,40)
5820 POSITION 3,23:? "[START]= N
OWA MAPA  [OPTION]= 1 CZESC";
5830 IF PEEK(53279)=6 THEN 5860
5840 IF PEEK(53279)<>3 THEN 5830
5845 X=USR(ADR(OKNO$),ADR(L3$),4
,20,4,40)
5850 X=USR(ADR(OKNO$),ADR(MAP$)+
520,9,20,26,40):GOTO 5715
5860 X=USR(ADR(OKNO$),ADR(L3$),4
,20,4,40):GOTO 810
```

* SUMA KONTROLNA / ETYKIETA *

CK	5	DB	6	EA	10	MZ	20
GB	30	SS	31	[illegible]	34	II	35
YX	39	VT	40	MM	45	PR	100
RK	110	WQ	120	OL	200	UP	300
PP	310	AJ	350	DB	352	XV	354
XY	356	EI	358	VD	400	NT	410
DI	450	GQ	452	YA	454	RI	456
ME	458	VR	500	JA	510	GZ	550
JC	552	WF	600	XZ	610	EL	650
UP	652	WT	700	CI	710	DB	750
XO	752	JQ	754	JN	800	NI	802
ML	805	IF	806	UU	810	UV	820
TE	830	TT	840	CO	900	SW	901
SI	902	VH	5510	IH	5520	JW	5530
GE	5540	WB	5550	AF	5560	DM	5570
IA	5580	YD	5590	FE	5600	RT	5610
BE	5620	ZR	5700	BG	5710	OE	5715
AP	5720	LI	5730	MP	5740	DI	5750
UW	5760	BY	5770	AL	5800	NR	5802
PK	5805	BJ	5810	JG	5820	MN	5830

JANUSZ MILLER
SHARP PC-1500
R E F L E K S

```
500:"A":RANDOM :
    PAUSE "POMIAR
    REFLEKSU"
510:N=0:WAIT 0:FOR
    K=1TO 30
520:A$=CHR$ (64+
    RND 26):BEEP 1
530:B$=INKEY$ :N=N
    +1
540:CLS :PRINT " *
    ";N;" * ";A$
550:IF A$<>B$GOTO
    530
560:NEXT K:CLS :
    WAIT
570:PRINT "CZAS=";
    STR$ N;" JEDNO
    STEK"
580:INPUT "JESZCZE
     RAZ ? T=1 N=0
     ";T:IF T=1
    GOTO 510
590:END
```

Pascal (7)

Grafika komputerowa jest bardzo ważną dziedziną informatyki, rozwijaną w wielu liczących się ośrodkach. Jej celem jest wypracowanie efektywnych pakietów programowych (standardów) umożliwiających łatwe i szybkie generowanie obrazów figur i brył z uwzględnieniem punktu widzenia i napotykanych przeszkód.

W tym kursie nie jest możliwe przedstawienie żadnego znanego standardu graficznego. Ograniczymy się wyłącznie do zaprezentowania koncepcji kreślenia kolorowych obrazów w komputerze AMSTRAD CPC 6128 na podstawie pakietu graficznego CPC 6128 — Grafik Version 1.0 firmy Heimsoeth Software Munich.

Aby włączyć pakiet do poszczególnych programów, wykorzystano specjalną dyrektywę komputera

{$ I *nazwa zbioru*}

Umożliwia ona włączenie do tekstu programu, w miejsce wskazane dyrektywą, przygotowanego wcześniej tekstu programu przechowywanego w zbiorze o nazwie podanej w dyrektywie. Należy pamiętać, że tekst całego programu (tzn. z włączonymi zbiorami) musi mieć strukturę akceptowaną przez kompilator.

Ekran monitora dzieli się na dwie odrębnie wysterowywane i niezależne od siebie części:

— obszar główny, wykorzystywany do zapisu tekstów i kreśleń obrazów, zwany kartką, — pobocze.

Pobocze nie jest wykorzystywane do zapisu tekstów i kreślenia obrazów, można natomiast zmieniać jego kolor. Kolor pobocza jest tłem dla kartki. Do zmiany koloru pobocza wykorzystywana jest procedura

border (*i, j*)

której parametrami są identyfikatory kolorów. Programista dysponuje dwudziestomasiedmioma kolorami, oznaczonymi numerami od 0 do 26. Kolory przyporządkowane poszczególnym numerom zamieszczono w tabeli 1.

TABELA 1

numer kałamarza	kolor	numer kałamarza	kolor	numer kałamarza	kolor
0	czarny	9	zielony	18	jasnoziel
1	niebieski	10	cyjanowy	19	morski ziel
2	jasnonieb	11	kolor nieba	20	jasnocyjan
3	czerwony	12	żółty	21	zieleń wapna
4	karmazynowy	13	biały	22	zieleń pastel
5	lila-róż	14	nieb. pastel	23	cyjan pastel
6	jasnoczerw	15	pomarańcz	24	jasnożółty
7	purpurowy	16	różowy	25	żółty pastel
8	jasnopurp	17	purp pastel	26	jasnobiały

W procedurze border podaje się numery dwóch kolorów zmienianych naprzemiennie kilka razy w ciągu sekundy. Jeśli oba numery są jednakowe, to efekt migotania nie jest dostrzegalny. Program umożliwiający zmianę koloru pobocza (bez migotania) zamieszczono w przykładzie 22.

Przykład 22

Program ten zmienia kolor pobocza na odpowiadający kolejnemu numerowi, po każdorazowym naciśnięciu dowolnego klawisza. Do sprawdzenia, czy wciśnięto klawisz, wykorzystano funkcję standardową **keypressed** przyjmującą wartość **true** po wciśnięciu klawisza i **false** po odczytaniu kodu wciśniętego klawisza (READ (*kbd...*) Po wyczerpaniu wszystkich możliwych kolorów, kolor pobocza jest zmieniany na niebieski (przyjmowany standardowo przez system). Efekt migotania można uzyskać realizując program z przykładu 23, w którym kolejne pary kolorów pobocza są ustalone przez użytkownika.

```
PROGRAM prog21;
 VAR
  i:integer;
  ch:char;
  {$i grafik3.inc}
 BEGIN
  FOR i:=0 TO 26 DO
   BEGIN
    border(i.i);
    REPEAT UNTIL keypressed;
    READ(kbd.ch);
   END;
  border(1.1);
 END.
```

Przykład 23

Bardziej skomplikowane są zasady wykorzystywania kartki. Na kartce można zapisać teksty **piórem tekstowym** oraz kreślić obrazy **piórem graficznym**. Przed użyciem pióra należy wskazać **kałamarz**, z którego czerpany będzie tusz. W każdym kałamarzu można umieścić jednocześnie — w dwóch odrębnych jego częściach — dwa tusze (różnokolorowe lub tego samego koloru). Jeżeli tusz będzie czerpany z kałamarza, w którym umieszczono różnokolorowe tusze, uzyskamy efekt migotania, jak w przypadku pobocza.

```
PROGRAM prog22;
 VAR
  k1,k2:integer;
  ch:char;
  {$i grafik3.inc}
 BEGIN
  REPEAT
   WRITELN('zmiana koloru ramki (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('podaj nr pierwszego koloru');
     READLN(k1);
     WRITELN('podaj nr drugiego koloru');
     READLN(k2);
     border(k1,k2);
    END;
   UNTIL (ch='N') OR (ch='n');
  border(1,1);
 END.
```

Liczba dostępnych kałamarzy uzależniona jest od trybu wykorzystania ekranu:

— 16, w trybie 0 (numerowanych od 0 do 15),

— 4, w trybie 1 (numerowanych od 0 do 3),

— 2, w trybie 2 (numerowanych od 0 do 1).

W miarę wzrostu numeru trybu wykorzystania ekranu poprawia się jakość kreślonego obrazu (rozdzielczość) kosztem zmniejszającej się liczby równocześnie możliwych do wykorzystania kałamarzy, a tym samym — liczby kolorów.

Do napełniania kałamarzy wykorzystuje się procedurę

ink *(nrk, i, j)*

gdzie:

nrk — numer kałamarza (który powinien uwzględnić tryb wykorzystania ekranu),

i, j — numery kolorów tuszu (wg tabeli 1).

W przykładzie 24 zamieszczono procedurę napełniania kałamarzy wg żądań użytkownika. Procedura ta wypełnia tylko tyle kałamarzy, ile wynika z trybu wykorzystania ekranu.

Przykład 24

Jak już podano, do zapisu tekstów oraz kreślenia obrazów wykorzystywane są dwa różne pióra: tekstowe i graficzne. Z każdym z nich związane są oddzielne procedury. Ich omawianie rozpoczniemy od procedur związanych z piórem tekstowym.

Użycie pióra tekstowego powinno być poprzedzone wyborem kałamarza, którego tusze będą wykorzystywane do zapisu tekstu. Służy do tego procedura

pen (nrk)

gdzie: *nrk* — numer kałamarza.

Wszystkie litery zapisywane są na tle, którego

```
PROCEDURE napelnij(tryb:integer);
 VAR
  il,k1,k2,
  i:integer;
  ch:char;
 BEGIN
  IF tryb=0 THEN il:=16
  ELSE
   IF tryb=1 THEN il:=4
   ELSE il:=2;
  i:=0;
  REPEAT
   WRITELN('zmiana tuszu w ');
   WRITELN('kalamarzu nr',i);
   WRITELN('(T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('nr pierwszego koloru');
     READLN(k1);
     WRITELN('nr drugiego koloru');
     READLN(k2);
     ink(i,k1,k2);
    END;
    i:=i+1;
   UNTIL i=il;
END;
```

kolor wybierany jest przez wykonanie procedury.

paper (nrk)

gdzie: nrk — numer kałamarza.

Jeżeli procedury te nie zostaną użyte przed wykorzystaniem pióra, to dla tła przydzielany jest standardowo kałamarz O, a dla pióra — kałamarz I.

Dla ilustracji działania procedur **paper** i **pen** w przykładzie 25 po ustaleniu trybu wykorzystania ekranu i napełnieniu kałamarzy dla tła, a w przykładzie 26 — po czynnościach wstępnych — następuje wybieranie kolejnych kałamarzy dla pióra (przyjęto, że procedura uzupełniania kałamarzy znajduje się w zbiorze PROCNAP. PAS).

Przykład 25

```
PROGRAM prog23;
 VAR
  tryb,ilosc,nrk:integer;
  ch:char;
  {$i grafik3.inc}
  {$i procnap.pas}
 BEGIN
  WRITELN('podaj tryb');
  READLN(tryb);
  napelnij(tryb);
  IF tryb=0 THEN ilosc:=16
  ELSE
   IF tryb=1 THEN ilosc:=4
   ELSE ilosc:=2;
  mode(tryb);
  FOR nrk:=0 TO ilosc-1 DO
   BEGIN
    paper(nrk);
    WRITELN;
    WRITELN('01234567890123456789');
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
   END;
  mode(2);
  ink(0,1,1);
  paper(0);
  ink(1,26,26);
  pen(1);
 END.
```

```
PROGRAM prog24;
 VAR
  tryb,ilosc,nrk:integer;
  ch:char;
  {$i grafik3.inc}
  {$i procnap.pas}
 BEGIN
  WRITELN('podaj tryb');
  READLN(tryb);
  napelnij(tryb);
  IF tryb=0 THEN ilosc:=16
  ELSE
   IF tryb=1 THEN ilosc:=4
   ELSE ilosc:=2;
  mode(tryb);
  FOR nrk:=0 TO ilosc-1 DO
   BEGIN
    pen(nrk);
    WRITELN;
    WRITELN('01234567890123456789');
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
   END;
  mode(2);
  ink(0,1,1);
  paper(0);
  ink(1,26,26);
  pen(1);
 END.
```

Przykład 26

W obu podanych przykładach do ustalenia żądanego trybu wykorzystania ekranu zastosowano procedurę.

modele (nrtryb)

gdzie:

nrtryb — przyjmuje wartość 0,1,2.

Należy tutaj wyjaśnić, że z trybem wykorzystania ekranu wiąże się również liczba znaków tekstu, które mogą być wyświetlone w jednym wierszu.

Wynosi ona:

— 20, dla trybu O,
— 40, dla trybu 1,
— 80, dla trybu 2.

Przed zakończeniem każdego z programów przywracany jest standardowy tryb wykorzystania ekranu (2), standardowe numery kałamarzy tła (0) i pióra (1) oraz standardowe numery tuszów w tych kałamarzach.

Program z przykładu 27 umożliwia prześledzenie wszystkich przedstawionych właściwości pióra tekstowego

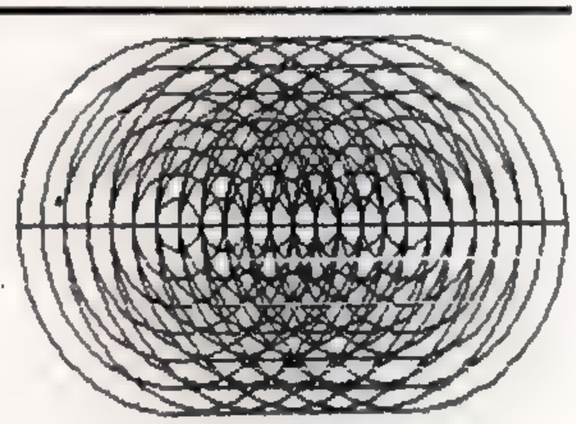
Przykład 27

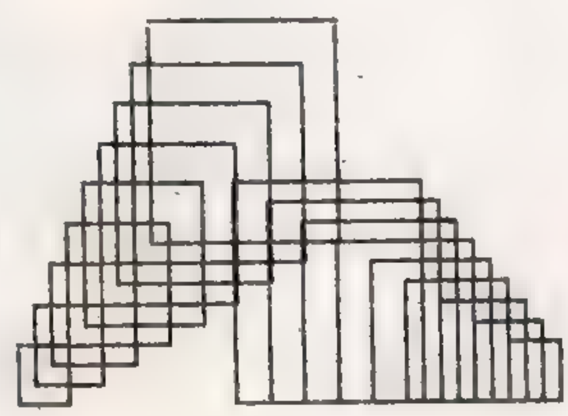
```
PROGRAM prog25;
 VAR
  tryb,nrk:integer;
  ch:char;
  {$i grafik3.inc}
  {$i procnap.pas}
 BEGIN
  WRITELN('podaj tryb');
  READLN(tryb);
  mode(tryb);
  WRITELN;
  REPEAT
   WRITELN('zmiana tuszu w ');
   WRITELN('kalamarzach (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
     napelnij(tryb);
   WRITELN('zmiana tla (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('podaj nr kalamarza');
     READLN(nrk);
     paper(nrk);
    END;
   WRITELN('zmiana piora (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('podaj nr kalamarza');
     READLN(nrk);
     pen(nrk);
    END;
  UNTIL ch=' ';
  mode(2);
  ink(0,1,1);
  paper(0);
  ink(1,26,26);
  pen(1);
 END.
```

W kolejnym odcinku zajmiemy się piórem graficznym.

Stefan ROZMUS

```
 10 FOR x=0 TO 70 STEP 8
 30 PLOT x,x
 40 DRAW 24+x,0
 50 DRAW 0,24+x
 60 DRAW -(24+x),0
 70 DRAW 0,-(24+x)
 80 NEXT x
110 FOR x=0 TO 70 STEP 8
120 PLOT 255-x,0
130 DRAW 0,24+x
140 DRAW -(24+x),0
150 DRAW 0,-(24+x)
160 DRAW 24+x,0
170 NEXT x
```





```
  5 PAPER 4: INK 0
 10 PLOT 0,85
 20 DRAW 240,0
 50 LET x=70
 55 LET y=85
 56 FOR j=1 TO 5
 60 FOR i=10 TO 70 STEP 10
 70 CIRCLE x,y,i
 80 NEXT i
 90 LET x=x+20
100 NEXT j
```

ZW

Komputerowy test dla kierowców

Ten program może się przydać wszystkim, którzy swój refleks i manualną sprawność muszą utrzymywać w stałej kondycji. Dokładny opis działania znajduje się w programie.

Program powstał w Wyższej Szkole Oficerskiej Wojsk Pancernych w Poznaniu

```
10 INPUT "Czy wynik testu na drukarce ? ( t - tak/
 n -nie )";druk$:IF druk$<>"T" AND druk$<>"t" AND
druk$<>"N" AND druk$<>"n" THEN 10
20 ilbl(1)=2:ilbl(2)=2:ilbl(3)=4:ilbl(4)=5:oceprs=
"              TEST NIE ZALICZONY":ocezw$=ocepr$
30 kt=0:IF druk$="t" OR druk$="T" THEN PRINT #8,CH
R$(27) CHR$(64)
40 DIM tt(10),c$(10),t(20),blad(10)
50 GOSUB 2630:CLS:MODE 1:BORDER 26:INK 0,26:INK 1,
6:INK 2,2:INK 3,0:PAPER 0:PEN 1
60 NA=10:ZLY=6:CZAS=11:ZA=13:GOTO 90
70 LOAD "scr"
80 IF INKEY$="" THEN 80
90 czesc=1:czesctestu=1
100 '
110 '  ekran 1
120 '
130 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:I
NK 3,11:PAPER 2:CLS:PEN 0
140 FOR e=1 TO 4:blad(e)=0:NEXT e:czesc=1:czesctes
tu=1:IF druk$="T" OR druk$="t" THEN INPUT "Podaj i
mi[ i nazwisko";imie$
150 ON czesctestu GOTO 160,400,480,580,730

160 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:I
NK 3,11:PAPER 2:CLS:PEN 0:PRINT "Program jest prze
znaczony do sprawdzaniarefleksu kierowc5w wojskowy
ch pojazd5w  mechanicznych.Czas reakcji jest zlicz
a-"
170 PRINT "ny przy 2 rodzajach bodjc5w:"
180 PRINT "     - dla oczu (zmysl wzroku)"
190 PRINT "     - dla uszu (zmysl sluchu)"
200 PRINT "Program sklada si[ z 4 cz[7ci,sekwency-
jnie sprawdzaj!cych czas reakcji testo- wanego ki
erowcy:"
210 PRINT " 1.Test oczu - reakcja sygnalizowana do
-   woln! r[k!."
220 PRINT " 2.Test uszu - reakcja sygnalizowana do
-   woln! r[k!.
230 PRINT " 3.Test oczu - reakcja lew! lub praw!
  r[k! w zale9no7ci od koloru."
240 PRINT " 4.Losowa kombinacja 3 wy9ej wymienio-
  nych test5w."

250 PRINT "Ka9da cz[7" sklada si[ z 10 pr5b i jest
 zako3czona przedstawieniem 7redniego    czasu rea
kcji."
260 PRINT "Po cz[7ci 4 testu nast[puje podsumowani
e! ocena badanego kierowcy wojskowego."
270 PRINT " "
280 PEN 1:PRINT " klawisz  COPY  dla uruchomienia
testu"
290 IF INKEY(9)<>0 THEN 290
300 PEN 1
310 CLS
320 PRINT " ";PRINT "          CZ08] 1 - TEST OCZU"
:PRINT " ";PEN 0
330 PRINT " Istot! tej cz[7ci testu jest zbadanie
  czasu reakcji na bodjce odbierane       przez zm
ysl wzroku.Po odebraniu bodjca  prawidlowa reakcja
 polega na naci7ni[-  ciu klawisza SPACJA.Bodziec
polega na"
340 PRINT " wy7wietleniu 951tego kola na
  niebieskim polu roboczym,w dowolnym     miejscu
ekranu.W dolnej cz[7ci ekranu   ukazuje si[ czas r
ee ... na bodziec.    Zbyt wczesna reakcja  przed
ukazaniem   si[ kola) powoduje powt5rn! pr5b[."
350 GOSUB 2590
360 IF INKEY(9)<>0 THEN 360
370 FOR i=1 TO 10:GOSUB 1010:t(I)=tb:NEXT i
380 GOSUB 1310
390 czesctestu=2:czesc=2:GOTO 150
400 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:I
NK 3,11:PAPER 2:CLS:PEN 0
410 PRINT " ": PEN 1:PRINT "          CZ08] 2 - TE
ST USZU":PRINT " ";PEN 0

420 PRINT " Istot! tej cz[7ci testu  jest zbadanie
  czasu reakcji na bodjce odbierane      przez zm
ysl sluchu.Po odebraniu bodjca  prawidlowa reakcja
 polega na naci7ni[-  ciu klawisza SPACJA.Bodziec
polega na"
430 PRINT " wygenerowaniu przez mikrokomputer sy-
  gnalu dwi[kowego.W dolnej cz[7ci      ekranu u
kazuje si[ czas reakcji na bo-  dziec.Zbyt wczesna
 reakcja (przed      uslyszeniem sygnalu dwi[kow
ego) powo-  duje powt5rn! pr5b[."
440 GOSUB 2590
450 IF INKEY(9)<>0 THEN 450
460 FOR i=1 TO 10:GOSUB 1420:t(i)=tb:NEXT i:GOSUB
1310
470 czesctestu=3:czesc=3:GOTO 150
480 GOTO 490
490 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:I
NK 3,11:PAPER 2:CLS:PEN 0
500 CLS:PEN 1:PRINT "         CZ08] 3 - TEST OCZU"
:PRINT " ";PEN 0
510 PRINT "Istot! tej cz[7ci testu jest zbadanie
 czasu reakcji na bodjce odbierane przez zmysl wzr
oku.Bodziec polega na wy7wie-  tleniu czerwonego l
ub zielonego kola    na niebieskim polu roboczym w
 dowo-     lnym miejscu ekranu.Prawidlowa reakcja"
520 PRINT "polega na:
  -naci7ni[ciu lew! r[k! klawisza 1      po poka
zaniu si[ koloru czerwonego    -naci7ni[ciu praw!
 r[k! klawisza 0       po pokazaniu si[ koloru zie
lonego"

530 PRINT "W dolnej cz[7ci ekranu ukazuje si[
 czas reakcji na bodziec.Zbyt wczesna   reakcja (
przed  ukazaniem si[ kola) lub niewla7ciwa (naci7n
i[cie nieodpowiednie-go do koloru klawisza) powodu
je"
540 PRINT "powt5rn! pr5b[."
550 GOSUB 2590
560 IF INKEY(9)<> 0 THEN 560
570 FOR i=1 TO 10:GOSUB 1010:t(i)=tb:NEXT i:GOSUB
1310:czesctestu=4:czesc=4
580 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:I
NK 3,11:PAPER 2:CLS:PEN 0
590 CLS:PEN 1:PRINT " ":PRINT "              CZO
8] 4                     KOMBINACJA TEST6W 1,2
,3":PEN 0:PRINT " "
600 PRINT "Istot! tej cz[7ci jest zbadanie czasu
 reakcji na dowolny z przedtem przedsta- wionych b
odjc5w.Obowi!zuj! dotychczasowezasady:"
610 PRINT "1.Prawidlowe reakcje:
  sygnal dwi[kowy   - klawisz SPACJA    951ta
plamka      - klawisz SPACJA    czerwona plamka
  - klawisz '1'      zielona plamka     - klawi
sz '0'"
620 PRINT "2.Zbyt wczesna reakcja powoduje powt5rn
! pr5b[.  /                         3.Bodjce
s! generowane losowo,a wi[c      nie mo9esz przewi
dzie*,jaki b[dzie     prezentowany przez mikrokom
puter."
630 GOSUB 2590
640 IF INKEY(9)<> 0 THEN 640
650 FOR I=1 TO 10
660 xx=INT(RND(1)*35)+1:yy=INT(RND(1)*15)+1
670 xxx=xx MOD 2:yyy=yy MOD 2
680 IF xxx=0 AND yyy=0 THEN czesctestu=1:GOSUB 101
0:t(i)=tb
690 IF xxx=0 AND yyy=1 THEN czesctestu=2:GOSUB 142
0:t(i)=tb
700 IF xxx=1 THEN czesctestu=3:GOSUB 1010:t(i)=tb
710 NEXT i
720 GOSUB 1310
730 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:I
NK 3,11:PAPER 2:CLS:PEN 0
740 CLS:PEN 1:PRINT " ":PRINT " ":PRINT "
 PODSUMOWANIE TESTU":PEN 0:PRINT " ":PRINT " W pos
zczeg5lnych cz[7ciach testu      uzyskale7 nast[
puj!ce 7rednie czasy    reakcji:":PRINT " "
750 pr=(tt(1)+tt(2))/2:zw=(tt(3)+tt(4))/2
760 pr=(ROUND(pr*100))/100:zw=(ROUND(zw*100))/100
770 IF blad(1)=1 THEN 790
780 PRINT "  CZ08] 1 - TEST OCZU  - ";USING "##.##
";tt(1):GOTO 800
790 PRINT "  CZ08] 1 - TEST OCZU  - NIE ZALICZONA"
800 IF blad(2)=1 THEN 820
810 PRINT "  CZ08] 2 - TEST USZU  - ";USING "##.##
";tt(2):GOTO 830
820 PRINT "  CZ08] 2 - TEST USZU  - NIE ZALICZONA"
830 IF blad(3)=1 THEN 850
840 PRINT "  CZ08] 3 - TEST OCZU  - ";USING "##.##
";tt(3):GOTO 860
850 PRINT "  CZ08] 3 - TEST OC  .   NIE ZALICZONA"
860 IF blad(4)=1 THEN 880
870 PRINT "  CZ08] 4 - KOMBINACJA - ";USING "##.##
";tt(4):GOTO 890

880 PRINT "  CZ08] 4 - KOMBINACJA - NIE ZALICZONA"
890 IF blad(1)=1 OR blad(2)=1 THEN 910 ELSE 930
900 IF blad(3)=1 OR blad(4)=1 THEN 920 ELSE 940
910 PRINT " ":PRINT " BADANIE REAKCJI PROSTEJ - NI
E ZALICZONE":GOTO 900
920 PRINT " BADANIE REAKCJI Z WYBOREM - NIE ZALI-
  CZONE":GOTO 950
930 PRINT " ":PRINT " SR. CZAS REAKCJI PROSTEJ:
 ";USING "##.##";pr
940 PRINT " SR. CZAS REAKCJI Z WYBOREM: ";USING "#
#.##";zw
950 IF druk$="T" OR druk$="t" THEN GOSUB 2060
960 LOCATE 1,20:PRINT " Je9eli chcesz uzyska* ocen
[ testu naci- 7nij klawisz COPY,je7li nie - ENTER"
:IF INKEY(9)<>0 AND INKEY(6)<>0 THEN 960
970 IF INKEY(9)=0 THEN 1660

980 LOCATE 1,20:PRINT " Je9eli chcesz powt5rzy* te
st - naci7nij klawisz COPY,je7li nie - ENTER
  ":IF INKEY(6)<>0 AND INKEY(9)<>0 THEN 980
990 IF INKEY(9)=0 THEN 90
1000 END
1010 CLS #0:CLS #1:BORDER 9:INK 0,11:INK 1,24:INK
2,6:INK 3,9:PAPER #0,2:PEN #0,1:PAPER #1,0:PEN #1,
1
1020 WINDOW #0,1,40,20,25:WINDOW #1,1,40,1,19
1030 '            plamka
1040 a$=CHR$(143)+CHR$(144)+CHR$(10)+CHR$(8)+CHR$(
8)+CHR$(145)+CHR$(146)
1050 CLS
1060 'randomize - czas od 1 do 11 sek.
1070 g=INT(RND(1)*10)+1:g1=TIME
1080 g2=TIME:j$=INKEY$
1090 IF INKEY(47)=0 OR INKEY(32)=0 OR INKEY(64)=0
THEN LOCATE #0,ZA,2:PRINT #0,"Za wcze7nie !!!":FO
R ii=1 TO 1000:NEXT:CLS #0:bledy=bledy+1:CLEAR INP
UT:IF bledy > ilbl(czesc) THEN 1610 ELSE 1070
1100 IF ((g2-g1)/300)<g THEN 1080
1110 x=INT(RND(1)*35)+1:y=INT(RND(1)*15)+1
1120 IF czesctestu<3 THEN 1170
```

```
1130 steri=x MOD 2
1140 IF steri = 0 THEN 1150 ELSE 1160
1150 PEN #1,3:GOTO 1170
1160 PEN #1,2
1170 LOCATE #1,x,y:CLEAR INPUT:PRINT#1,A$;:t=TIME

1180 IF czesctestu<3 THEN 1230
1190 IF INKEY$="" THEN 1190
1200 IF INKEY(32)=0 AND steri=0 THEN 1250
1210 IF INKEY(64)=0 AND steri=1 THEN 1250
1220 LOCATE #0,NA,2:PRINT #0,"Nacisn:le7 zly klawi
sz '''''":FOR ii=1 TO 1000:NEXT:CLS #0:CLS #1:bled
=bledy+1:CLEAR INPUT:IF bledy > ilbl(czesc) THEN :

610 ELSE 1070
1230 IF INKEY$="" THEN 1230
1240 IF INKEY(47) <>0 THEN 1220
1250 tb=(TIME-t)/300
1260 LOCATE #0,CZAS,2:PRINT "CZAS REAKCJI:"; USING
 "##,.##";tb
1270 LOCATE #0,NA,6:PRINT #0,"Naci7nij klawisz COP
Y"
1280 IF INKEY(9)<>0 THEN 1280
1290 CLS#1
1300 CLEAR INPUT:RETURN


1310 CLS:REM    OBLICZANIE  SREDNIEGO CZASU REAKCJ
I
1320 tijd=0
1330 FOR x=1 TO 10:tijd=t(x)+tijd:NEXT x
1340 tt(czesc)=tijd/10:ON CZESC GOTO 1350,1360,137
0,1380
1350 LOCATE #0,1,2:PRINT " SREDNI CZAS REAKCJI TWO
ICH OCZU:":GOTO 1390
1360 LOCATE #0,1,2:PRINT " SREDNI CZAS REAKCJI TWO
ICH USZU:":GOTO 1390
1370 LOCATE #0,1,2:PRINT " SREDNI CZAS REAKCJI TWO
ICH OCZU:":GOTO 1390
1380 LOCATE #0,1,2:PRINT " SREDNI CZAS REAKCJI OCZ
U I USZU:"
1390 LOCATE #0,35,2:PRINT USING "##.##";tt(czesc):
LOCATE #0,NA,6:PRINT #0,"Naci7nij klawisz COPY"
1400 IF INKEY(9)<>0 THEN 1400
1410 RETURN
1420 CLS #0:CLS #1:BORDER 9:INK 0,11:INK 1,24:INK
2,6:INK 3,9:PAPER #0,2:PEN #0,1:PAPER #1,0:PEN #1.
1
1430 WINDOW #0,1,40,20,25:WINDOW #1,1,40,1,19
1440 CLS
1450 g=INT(RND(1)*10)+1:g1=TIME
1460 g2=TIME:IF INKEY(47)=0 THEN LOCATE #0,ZA,2:PR
INT "Za wcze7nie''''":FOR ii=1 TO 1000:NEXT:CLS #
:bledy=bledy+1:CLEAR INPUT:IF bledy > ilbl(czesc)
THEN 1610 ELSE 1450

1470 IF((g2-g1)/300)<g THEN 1460
1480 CLEAR INPUT:PRINT#1,CHR$(7):t=TIME
1490 j$=INKEY$:IF j$="" THEN 1490
1500 IF INKEY(47)=0 THEN 1530
1510 LOCATE #0,NA,2:PRINT #0,"Nacisn:le7 zly klaw:
sz '''''":FOR ii=1 TO 1000:NEXT:CLS #0:CLS #1:bledy
=bledy+1:CLEAR INPUT:IF bledy > ilbl(czesc) THEN 1
610 ELSE 1450
1520 CLS #1:GOTO 1450
1530 tb=(TIME-t)/300
1540 LOCATE #0,CZAS,2:PRINT "CZAS REAKCJI:"; USING
"##,.##";tb
1550 LOCATE #0,NA,6:PRINT #0,"Naci7nij klawisz COP
Y"
1560 IF INKEY(9)<>0 THEN 1560
1570 CLS
1580 CLEAR INPUT:RETURN
1590 PRINT "JESZCZE RAZ ?  ...TO NACISKAJ KLUCZ"
1600 IF INKEY$="" THEN 1600 ELSE RUN
1610 ' OBSZUGA BLEDOW BADANEGO
1620 blad(czesc)=1:tt(czesc)=0:czesctestu=czesctes
tu+1:czesc=czesc+1
1630 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:
INK 3,11:PAPER 2:CLS:PEN 0
1640 LOCATE 2,11:PRINT "    POPE2NI2ES ZA DU:O B20
DOW !'
                 PRoBA NIEZALICZONA":FOR I=1 TO 1700:
NEXT
1650 bledy=0:IF czesc > 4 THEN 730 ELSE 150
1660 REM OCENA WYNIKOW  TESTU
1670 IF blad(1)=1 OR blad(2)=1 THEN ocpr=0:GOTO 17
30
1680 IF pr >= 0.27 THEN ocenapr=2
1690 IF pr < 0.27 THEN ocenapr=3
1700 IF pr > 0.32 THEN ocenapr=1
1710 IF pr < 0.23 THEN ocenapr=4
1720 ocpr=1
1730 IF blad(3)=1 OR blad(4)=1 THEN oczw=0:GOTO 17
90
1740 IF zw >= 0.52 THEN ocenazw=2
1750 IF zw < 0.52 THEN ocenazw=3
1760 IF zw > 0.59 THEN ocenazw=1
1770 IF zw < 0.47 THEN ocenazw=4
1780 oczw=1
1790 CLS:MODE 1:BORDER 4:INK 0,26:INK 1,6:INK 2,9:
INK 3,11:PAPER 2:CLS:PEN 1
1800 PRINT " ":PRINT "    **** OCENA REAKCJI PROST
EJ ****":PRINT " ":PEN 0:IF ocpr=0 THEN PRINT ocep
r$:GOTO 1930
1810 PEN 0:ON ocenapr GOSUB 1830,1860,1880,1910
1820 GOTO 1930
1830 PRINT " TWOJE WYNIKI S) BARDZO S2ABE.
   CZAS REAKCJI PROSTEJ DYSKWALIFIKUJE     CIO JAK
O KIEROWCO WOJSKOWEGO.ZDECYDOWA- NIE  MUSISZ JWICZ
YJ CZAS REAKCJI          PROSTEJ NA BODKCE WZROKOWE
I S2UCHOWE."
1840 PRINT " UZYSKA2ES OCENO : 2"
1850 RETURN
1860 PRINT " TWOJE WYNIKI S) S2ABE.CZAS REAKCJI
   PROSTEJ KWALIFIKUJE CIO JAKO KIEROWCO   WOJSKOW
EGO,ALE MUSISZ ZDECYDOWANIE       JWICZYJ CZAS REAK
CJI PROSTEJ NA BODKCE  WZROKOWE I S2UCHOWE.
               UZYSKA2ES OCENO : 3"
1870 RETURN
1880 PRINT " TWOJE WYNIKI S) DOBRE.
   CZAS REAKCJI PROSTEJ KWALIFIKUJE CIO    JAKO KI
EROWCO WOJSKOWEGO.NIE ZASZKODZI  JEDNAK JAK BODZIE
SZ JWICZY2 CZAS REA-   KCJI PROSTEJ NA BODKCE WZRO
KOWE I      S2UCHOWE."
1890 PRINT " UZYSKA2ES OCENO : 4"
1900 RETURN
1910 PRINT " TWOJE WYNIKI S) BARDZO DOBRE.
   CZAS REAKCJI PROSTEJ KWALIFIKUJE CIO    JAKO KI
EROWCO WOJSKOWEGO.NIE MUSISZ     JWICZYJ CZASU REA
KCJI PROSTEJ NA        BODKCE WZROKOWE I S2UCHOWE.
              UZYSKA2ES OCENO :
1920 RETURN
1930 PEN 1:PRINT " ":PRINT "   **** OCENA REAKCJI
Z WYBOREM ****":PRINT " ":PEN 0:IF oczw=0 THEN PRI
NT ocepr$:GOTO 1950
1940 ON ocenazw GOSUB 1960,1990,2010,2040
1950 IF druk$="T" OR druk$="t" THEN 2310 ELSE 960
1960 PRINT " TWOJE WYNIKI S) BARDZO S2ABE.
   CZAS REAKCJI Z WYBOREM DYSKWALIFIKUJE   CIO JAK
O KIEROWCO WOJSKOWEGO.ZDECYDOWA- NIE  MUSISZ JWICZ
YJ CZAS REAKCJI         Z WYBOREM NA BODKCE WZROKOW
E I S2UCHOWE"
1970 PRINT " UZYSKA2ES OCENO : 2"
1980 RETURN
1990 PRINT " TWOJE WYNIKI S) S2ABE.CZAS REAKCJI
   Z WYBOREM KWALIFIKUJE CIO JAKO KIEROWCO WOJSKOW
EGO,ALE MUSISZ ZDECYDOWANIE       JWICZYJ CZAS REAK
CJI Z WYBOREM NA       BODKCE  WZROKOWE I S2UCHOWE
              UZYSKA2ES OCENO : 3"
2000 RETURN
2010 PRINT " TWOJE WYNIKI S) DOBRE.
   CZAS REAKCJI Z WYBOREM KWALIFIKUJE CIO  JAKO KI
EROWCO WOJSKOWEGO.NIE ZASZKODZI  JEDNAK JAK BODZIE
SZ JWICZY2 CZAS REA-   KCJI Z WYBOREM NA BODKCE WZ
ROKOWE I     S2UCHOWE."
2020 PRINT " UZYSKA2ES OCENO : 4"
2030 RETURN
2040 PRINT " TWOJE WYNIKI S) BARDZO DOBRE.
   CZAS REAKCJI Z WYBOREM KWALIFIKUJE CIO  JAKO KI
EROWCO WOJSKOWEGO.NIE MUSISZ     JWICZYJ CZASU REA
KCJI Z WYBOREM NA      BODKCE WZROKOWE I S2UCHOWE.

              UZYSKA2ES OCENO : 5"
2050 RETURN
2060 '  wyjscie na drukarke
2070 kt=kt+1:PRINT #8," ":PRINT #8," ":PRINT #8,CH
R$(27) CHR$(87) CHR$(1) CHR$(27) CHR$(71):IF kt=1
THEN PRINT #8,"      WYNIKI TESTOWANIA REAKCJI"

2080 IF kt=1 THEN PRINT #8,"             PROSTEJ I Z
WYBOREM":PRINT #8,CHR$(27) CHR$(72)
2090 PRINT #8,kt;".";imie$
2100 PRINT #8,CHR$(27) CHR$(87) CHR$(0)
2110 IF blad(1)=1 THEN 2130
2120 PRINT #8,"          CZESC 1 - TEST OCZU  - ";US
ING "##.##";tt(1):GOTO 2140
2130 PRINT #8,"          CZESC 1 - TEST OCZU  - NIE
ZALICZONA"
2140 IF blad(2)=1 THEN 2160
2150 PRINT #8,"          CZESC 2 - TEST USZU  - ";US
ING "##.##";tt(2):GOTO 2170
2160 PRINT #8,"          CZESC 2 - TEST USZU  - NIE
ZALICZONA"
```

7 6 5 4 3 2 1 0

Uwaga Atari!

AUDC 1 - 4

	niskie częstotliwości	średnie częstotliwości	wysokie częstotliwości
0 0 0	licznik Geigera	szalejący ogień	prąd powietrza, wrzenie
0 X 1	karabin maszynowy	stojący samochód	silnik elektryczny, transformator mocy
0 1 0	spokojny ogień	jadący samochód	
1 0 0	zawalenie budynku	dudnienie radiowe	wodospad
1 X 1	czyste tony		
1 0 0	samolot	strzyżenie trawnika	elektryczna maszynka do golenia

Odpowiednio programując rejestry niezależnego sterowania kanałami AUDC 1—4 otrzymać można takie oto efekty dźwiękowe

```
2170 IF blad(3)=1 THEN 2190
2180 PRINT #8,"          CZESC 3 - TEST OCZU  - ";USING "##.##";tt(3):GOTO 2200
2190 PRINT #8,"          CZESC 3 - TEST OCZU  - NIE ZALICZONA"
2200 IF blad(4)=1 THEN 2220
2210 PRINT #8,"          CZESC 4 - KOMBINACJA - ";USING "##.##";tt(4):GOTO 2230
2220 PRINT #8,"          CZESC 4 - KOMBINACJA - NIE ZALICZONA"
2230 IF blad(1)=1 OR blad(2)=1 THEN 2250 ELSE 2280

2240 IF blad(3)=1 OR blad(4)=1 THEN 2270 ELSE 2290
2250 PRINT #8," "
2260 PRINT #8,"       BADANIE REAKCJI PROSTEJ  - NIE ZALICZONE":GOTO 2240
2270 PRINT #8,"       BADANIE REAKCJI Z WYBOREM - NIE ZALICZONE":GOTO 2300
2280 PRINT #8," ":PRINT #8,"       SR. CZAS REAKCJI PROSTEJ:   ";USING "##.##";(tt(1)+tt(2))/2:GOTO 2240
2290 PRINT #8,"       SR. CZAS REAKCJI Z WYBOREM: ";USING "##.##";(tt(3)+tt(4))/2
2300 RETURN
2310 PRINT #8," ":PRINT #8,"              ##### OCENA REAKCJI PROSTEJ #####":PRINT #8," ":IF ocpr=0 THEN PRINT #8,ocezw$:GOTO 2450
2320 ON ocenapr GOSUB 2340,2370,2390,2420
2330 GOTO 2450
2340 PRINT #8,"        TWOJE WYNIKI SA BARDZO SLABE.CZAS REAKCJI PROSTEJ DYSKWALIFIKUJE":PRINT #8,"      CIE JAKO KIEROWCE WOJSKOWEGO.ZDECYDOWANIE MUSISZ CWICZYC CZAS":PRINT #8,"        REAKCJI PROSTEJ NA BODZCE WZROKOWE I SLUCHOWE."
2350 PRINT #8,"        UZYSKALES OCENE : 2"
2360 RETURN
2370 PRINT #8,"        TWOJE WYNIKI SA SLABE.CZAS REAKCJI PROSTEJ KWALIFIKUJE CIE":PRINT #8," JAKO KIEROWCE WOJSKOWEGO,ALE MUSISZ ZDECYDOWANIE CWICZYC CZAS"
2380 PRINT #8,"        REAKCJI PROSTEJ NA BODZCE WZROKOWE I SLUCHOWE":PRINT #8,"        UZYSKALES OCENE : 3":RETURN
2390 PRINT #8,"        TWOJE WYNIKI SA DOBRE.CZAS REAKCJI PROSTEJ KWALIFIKUJE CIE JAKO":PRINT #8,"      KIEROWCE WOJSKOWEGO.NIE ZASZKODZI  JEDNAK JAK BEDZIESZ CWICZYL":PRINT #8,"       CZAS REAKCJI PROSTEJ NA BODZCE WZROKOWE I SLUCHOWE."
2400 PRINT #8,"        UZYSKALES OCENE : 4"
2410 RETURN
2420 PRINT #8,"        TWOJE WYNIKI SA BARDZO DOBRE.CZAS REAKCJI PROSTEJ KWALIFIKUJE CIE":PRINT #8,"      JAKO KIEROWCE WOJSKOWEGO.NIE MUSISZ CWICZYC CZASU REAKCJI PROSTEJ":PRINT #8,"        NA BODZCE WZROKOWE I SLUCHOWE."
2430 PRINT #8,"        UZYSKAZES OCENE : 5"
2440 RETURN
2450 PRINT #8," ":PRINT #8,"              ##### OCENA REAKCJI Z WYBOREM #####":PRINT #8," ":IF oczw=0 THEN PRINT #8,ocezw$:GOTO 980
2460 ON ocenazw GOSUB 2480,2510,2530,2560
2470 GOTO 980
2480 PRINT #8,"        TWOJE WYNIKI SA BARDZO SLABE.CZAS REAKCJI Z WYBOREM DYSKWALIFIKUJE":PRINT #8,"      CIE JAKO KIEROWCE WOJSKOWEGO.ZDECYDOWANIE MUSISZ CWICZYC CZAS":PRINT #8,"        REAKCJI Z WYBOREM NA BODZCE WZROKOWE I SLUCHOWE."
2490 PRINT #8,"        UZYSKALES OCENE : 2"
2500 RETURN
2510 PRINT #8,"        TWOJE WYNIKI SA SLABE.CZAS REAKCJI Z WYBOREM KWALIFIKUJE CIE":PRINT #8,"  JAKO KIEROWCE WOJSKOWEGO,ALE MUSISZ ZDECYDOWANIE CWICZYC CZAS":PRINT #8,"        REAKCJI Z WYBOREM NA BODZCE  WZROKOWE I SLUCHOWE."
2520 PRINT #8,"        UZYSKALES OCENE : 3":RETURN
2530 PRINT #8,"        TWOJE WYNIKI SA DOBRE.CZAS REAKCJI Z WYBOREM KWALIFIKUJE CIE":PRINT #8,"   JAKO KIEROWCE WOJSKOWEGO.NIE ZASZKODZI JEDNAK JAK BEDZIESZ":PRINT #8,"        CWICZYL CZAS REAKCJI Z WYBOREM NA BODZCE WZROKOWE I SLUCHOWE."
2540 PRINT #8,"        UZYSKALES OCENE : 4"

2550 RETURN
2560 PRINT #8,"        TWOJE WYNIKI SA BARDZO DOBRE.CZAS REAKCJI Z WYBOREM KWALIFIKUJE":PRINT #8,"      CIE JAKO KIEROWCE WOJSKOWEGO.NIE MUSISZ CWICZYC CZASU REAKCJI":PRINT #8;"     Z WYBOREM NA BODZCE WZROKOWE I SLUCHOWE. "
2570 PRINT #8,"        UZYSKALES OCENE : 5"
2580 RETURN
2590 PRINT " Ta cz[7" testu sklada si[ z 10 pr5b,   na jej zako3czenie wy7wietli si[ 7re-   dni czas reakcji.                  Uwaga:                        Gdy jeste7 got5w naci7nij klawisz COPY"
2600 RETURN
2610 IF INKEY$="" THEN 2610
2620 GOTO 130
2630 SYMBOL AFTER 142
2640 SYMBOL 171,0,0,120,12,124,204,118,3
2650 SYMBOL 172,24,60,102,102,126,102,102,3
2660 SYMBOL 173,24,0,60,102,96,102,60,0
2670 SYMBOL 174,24,60,102,192,192,102,60,0
2680 SYMBOL 175,0,0,60,102,126,96,60,6
2690 SYMBOL 176,254,98,104,120,104,98,254,3
2700 SYMBOL 177,56,24,28,24,56,24,60,0
2710 SYMBOL 178,240,96,112,96,226,102,254,0
2720 SYMBOL 179,24,0,220,102,102,102,102,0
2730 SYMBOL 180,24,198,230,246,222,206,198,0
2740 SYMBOL 181,24,0,60,102,102,102,60,0
2750 SYMBOL 182,12,56,108,198,198,108,56,0
2760 SYMBOL 183,24,0,60,96,60,6,124,0
2770 SYMBOL 184,24,60,96,60,6,102,60,0
2780 SYMBOL 185,24,0,126,76,24,48,126,0
2790 SYMBOL 186,254,198,140,126,48,102,254,0
2800 SYMBOL 187,12,24,126,76,24,48,126,0
2810 SYMBOL 188,48,254,172,24,50,102,254,0
2820 KEY DEF 10,1,55,171,172
2830 KEY DEF 11,1,56,173,174
2840 KEY DEF 3,1,57,175,176
2850 KEY DEF 20,1,52,177,178
2860 KEY DEF 12,1,53,179,180
2870 KEY DEF 4,1,54,181,182
2880 KEY DEF 13,1,49,183,184
2890 KEY DEF 14,1,50,185,186
2900 KEY DEF 5,1,51,187,188
2910 SYMBOL 143,3,31,63,127,127,127,255,255:SYMBOL 144,192,248,252,254,254,254,255,255:SYMBOL 145,255,255,127,127,127,63,31,3:SYMBOL 146,255,255,254,254,254,252,248,192
2920 RETURN
```



APPLICATION - wersja użytkowa, zastosowanie, także: sposób użycia,
APPLICATION LAYER - warstwa aplikacyjna,
APPLICATION PROGRAM - program aplikacyjny (realizujący określone zadanie), program użytkowy,
APPLICATION SOFTWARE - oprogramowanie użytkowe,
APPLICATION SYSTEM ADMINISTRATOR - administrator zastosowań,
APPLICATIONS ENGINEERING - projektowanie zastosowania, projektowanie systemu informatycznego,
APPLY - zastosować,
APPROACH - metoda, podejście, rozwiązanie, sposób,
APPROPRIATE - odpowiedni, stosowny,
APPROXIMATE - przybliżony,
APPROXIMATE COMPUTATION - rachunek przybliżony,
APPROXIMATELY - w przybliżeniu,
APPROXIMATION - aproksymacja, przybliżenie,
APPROXIMATION ERROR - błąd przybliżenia,
APT - patrz: AUTOMATIC PROGRAMMING FOR TOOLS,
ARABIC NOTATIONS - cyfry arabskie,
ARABIC NUMBERS - cyfry arabskie,
ARABIC NUMERALS - cyfry arabskie,
ARBITRARY ACCESS - dostęp dowolny,
ARBITRARY FORMAT - format dowolny,
ARBITRARY SEQUENCE COMPUTER - komputer sekwencyjny z dowolnym wykonywaniem rozkazów,
ARCADE GAME - gra rozrywkowa,
ARCTANGENT - arcus tangens,
AREA - obszar,
AREA DEFINITION - definiowanie obszaru,
AREA SEARCH - przeszukiwanie obszaru (pamięci),
ARGUMENT - argument, parametr, zmienna niezależna,
ARITHMETIC CHECK - kontrola arytmetyczna,
ARITHMETIC INSTRUCTION - rozkaz arytmetyczny,
ARITHMETIC-LOGIC UNIT (ALU) - jednostka arytmetyczno-logiczna,
ARITHMETIC REGISTER - rejestr arytmometru,
ARITHMETIC SECTION - moduł arytmometru,
ARITHMETIC SHIFT - przesunięcie arytmetyczne,
ARITHMETIC UNIT - arytmometr, jednostka arytmetyczno-logiczna,
ARITHMETICAL COMPUTATION - działania arytmetyczne,
ARITHMETICAL PROGRESSION - postęp arytmetyczny,
ARQ - patrz: AUTOMATIC REQUEST FOR REPEATS,
ARRANGE IN ORDER OF... - uszeregować w kolejności... ,
ARRANGEMENT - rozmieszczenie, układ, także: projektowanie,
ARRAY - tablica (zbiór zmiennych o wspólnej nazwie),
ARRAY ELEMENT - element tablicy,
ARRAY OF CORES - matryca rdzeni,
ARRAY PROCESSOR - procesor macierzowy,
ARROW - patrz: ARROWHEAD,
ARROWHEAD - strzałka wskazująca (np. na wykresie),
ARTIFICAL INTELLIGENCE - sztuczna inteligencja,
ARTIFICIAL DATA - dane sztuczne, dane zmyślone,
ASA FONT - pismo ASA (pismo stylizowane dostosowane do odczytywania optycznego),
ASCENDING - rosnący, wstępujący, wzrastający,
ASCERTAIN - ustalić, upewnić się, stwierdzić dokładnie,
ASPECT RATIO - współczynnik kształtu obrazu (stosunek długości osi pionowej do poziomej obrazu),
ASSEMBLER - program assembler, translator języka symbolicznego,
ASSEMBLER LANGUAGE - język typu assembler, język assemblerowy,
ASSEMBLY - tłumaczenie języka symbolicznego,
ASSEMBLY INSTRUCTION - instrukcja assemblera,
ASSEMBLY LANGUAGE - język symboliczny, język adresów symbolicznych,
ASSEMBLY LINE ROBOTS - roboty montażowe,
ASSEMBLY PROGRAM - translator języka symbolicznego,
ASSEMBLY ROUTINE - translator języka symbolicznego,
ASSESS - oceniać, szacować,
ASSESSMENT - ocena, oszacowanie,
ASSIGN - przyporządkować, przydzielić, także: określić,
ASSIGN PRIORITIES - wyznaczyć pierwszeństwo, określić priorytet,
ASSIGNMENT - przydział, wyznaczenie, przyporządkowanie,
ASSIGNMENT PROBLEM - zagadnienie przydziału, problem przyporządkowania,
ASSIGNMENT STATEMENT [illegible] przydziału,
ASSOCIATE - dołączony, sprzężony, stowarzyszony,
ASSOCIATED DESCRIPTOR - deskryptor skojarzeniowy,
ASSOCIATED GROUP - grupa stowarzyszona,
ASSOCIATED MATRIX OF LINEAR MAP - macierz odpowiadająca odwzorowaniu liniowemu,
ASSOCIATIVE MEMORY - pamięć asocjacyjna, pamięć skojarzeniowa,
ASSOCIATIVE STORAGE - pamięć asocjacyjna,
ASSOCIATOR - urządzenie kojarzące (elementy podobne)
ASSORT - sortować, segregować, dobierać,
ASSORTMENT - dobór, zestaw, także: sortowanie, segregowanie,
ASSUMED DECIMAL POINT - przecinek obliczeniowy,
ASSUMED MEAN - średnia dowolnie przyjęta,
ASSUMPTION - założenie, przypuszczenie,
ASSURANCE - pewność,
ASTERISK - gwiazdka jako znak graficzny, odsyłacz,
ASYNCHRONOUS - asynchroniczny,
ASYNCHRONOUS COMPUTER - komputer asynchroniczny,
ASYNCHRONOUS METHOD OF OPERATION - praca w trybie asynchronicznym, praca asynchroniczna,
ASYNCHRONOUS OPERATION - praca w trybie asynchronicznym, praca asynchroniczna,
ASYNCHRONOUS SERIAL PROTOCOL - asynchroniczny protokół szeregowy,
ASYNCHRONOUS TRANSMISSION - transmisja asynchroniczna (każdy znak poprzedza bit startu a po znaku występuje bit stopu),
AT DIFFERENT LEVELS - na różnych poziomach,
ATTACHAMENT - urządzenie dodatkowe (dołączane), uzupełnienie,
ATTACHMENT UNIT INTERFACE - sprzężenie z jednostką połączeniową tj. kabel, złącza i odpowiednia elektronika,
ATTENTION - uwaga,
ATTENUATION - tłumienie, osłabienie,
ATTENUATION BAND - pasmo tłumienia,
ATTENUATION COEFFICIENT - współczynnik tłumienia,
ATTENUATION CONSTANT - stała tłumienia, tłumienność jednostkowa,
ATTENUATION DIAGRAM - charakterystyka tłumienia, także: charakterystyka amplitudowa,
ATTENUATION DISTORTION - zniekształcenie tłumieniowe (sygnału),
ATTENUATION EQUALIZER - korektor tłumieniowy,
ATTENUATION RATE - współczynnik tłumienia,
ATTRIBUTE - artrybut, cecha, także: cecha elementu obrazu, np.wykrywalność, poziom intensywności świecenia, itd.,
AU - patrz: ARITHMETIC UNIT,
AUDIBLE ALARM - wskaźnik akustyczny,
AUDIBLE SIGNAL - wskaźnik akustyczny,
AUDIO RESPONSE DEVICE - wyjście foniczne, wyjście akustyczne,
AUGMENT - zwiększać,
AUGMENTER - wartość dodawana,
AUTO-SHUTDOWN - automatyczne wyłączanie systemu,
AUTOCODE - autokod, język autokodowy, język automatycznego programowania,
AUTOINDEXING - indeksowanie automatyczne,
AUTOLOAD - ładowanie automatyczne,
AUTOMATA THEORY - teoria automatów,
AUTOMATE - zautomatyzować,
AUTOMATIC - automatyczny,
AUTOMATIC AMPLITUDE CONTROL - automatyczna regulacja amplitudy,
AUTOMATIC ASSEMBLY RECORDING SYSTEM - system zapisu z automatycznym łączeniem,
AUTOMATIC CALLING UNIT - autowzywak,
AUTOMATIC CARD PUNCH - automatyczna dziurkarka kart,
AUTOMATIC CARRIAGE - przesuw automatyczny, karetka automatyczna,
AUTOMATIC CHARACTER RECOGNITION - automatyczne czytanie pisma,
AUTOMATIC CODING SYSTEM - system automatycznego kodowania,
AUTOMATIC CONTROL - sterowanie automatyczne,
AUTOMATIC CONTROL SYSTEM - układ automatycznej regulacji,
AUTOMATIC DATA PROCESSING - automatyczne przetwarzanie danych,
AUTOMATIC NEED-TO-REPAIR INDICATOR - automatyczny wskaźnik konieczności wykonania naprawy,
AUTOMATIC PROCESSING - przetwarzanie automatyczne,
AUTOMATIC PROGRAM INTERRUPT - automatyczne przerwanie realizacji programu,
AUTOMATIC PROGRAMMING - programowanie automatyczne,
AUTOMATIC PROGRAMMING FOR TOOLS - język programowania do sterowania numerycznego obrabiarek,
AUTOMATIC REQUEST FOR REPEATS - system z odpowiedzią zwrotną (umożliwia ochronę przed błędami),
AUTOMATIC RESPONSE FILE - plik (zbiór) przygotowanych odpowiedzi,
AUTOMATIC SCAN - przeglądanie automatyczne,
AUTOMATIC STORAGE ALLOCATION - automatyczny przydział pamięci,
AUTOMATIC STROBE PULSE - bramka śledząca,
AUTOMATIC TYPEWRITER - automat piszący,
AUTOMATICS - automatyka (nauka),
AUTOMATION - automatyzacja,
AUTOMATIZATION - automatyzacja,
AUTOMATON - automat, także: komputer z oprogramowaniem,
AUTONOMOUS WORKING - działanie niezależne,
AUTOPLOTTER - pisak X-Y,
AUTOPOLLING - samoczynne zbieranie danych (poprzez przeglądanie),
AUXILIARY CARRY - przeniesienie pomocnicze,
AUXILIARY CONTROL AREA - obszar zapisów zniekształconych, także: obszar dodatkowy,
AUXILIARY MEMORY - pomocnicza pamięć wewnętrzna,
AUXILIARY STORAGE - pamięć uzupełniająca, pamięć pomocnicza, pamięć wspomagająca,
AUXILIARY STORE - pamięć pomocnicza,
AVAILABILITY - dyspozycyjność, dostępność, zapas,
AVERAGE - wartość przeciętna, wartość średnia,
AVERAGE DEVIATION - odchylenie średnie,
AVERAGING DEVICE - układ uśredniający,
AVERAGING TIME - czas uśredniania,

AVOIDANCE - unikanie (np. zbędnych danych),
AX (ACCUMULATOR) - akumulator ,
AXIS - oś,
AXIS OF ABSCISSAE - oś odciętych,
AXIS OF ORDINATES - oś rzędnych,

B

BACK BONE - wspólna oś,
BACK CIRCUIT - układ sprzężenia zwrotnego,
BACK SLASH - odwrócony ukośnik,
BACK-UP MEMORY - pamięć rezerwowa,
BACKGROUND - tło,
BACKGROUND COLOR - kolor tła (obszaru otaczającego przedmiot),
BACKGROUND IMAGE - tło (podstawa) obrazu,
BACKGROUND PROGRAM - program drugoplanowy,
BACKING STORAGE - pamięć uzupełniająca, pamięć pomocnicza, pamięć wspomagająca,
BACKSPACE - cofacz,
BACKSPACE CHARAKTER - znak cofania, cofacz,
BACKSPACE KEY - klawisz cofający, cofacz,
BACKSPACING - cofanie (np. taśmy),
BACKTALK - przekazywanie informacji z komputera rezerwowego do komputera czynnego,
BACKUP - składowanie, żelazna rezerwa,
BACKUS NORMAL FORM - notacja BACKUSA-NAURA,
BACKWARD CHAINING - wnioskowanie wsteczne (w systemach ekspertowych),
BACKWARD COUNTER - licznik odejmujący,
BALANCE - bilans, pozostałość, równowaga, saldo, także: uzgadniać, zestawiać, równoważyć,
BALANCED ERROR - błąd zrównoważony,
BALANCING - uzgadnianie (np. wyników uzyskanych na maszynie), także: sprowadzanie do stanu zerowego,
BALL PRINTING HEAD - głowica pisząca, głowica czcionkowa, głowica kulista,
BALLOT BOX - kratka na formularzu dla zamieszczenia w niej np. krzyżyka lub kreski,
BAND - pasmo (np. częstotliwości), strefa, zakres,
BANDWIDTH - szerokość pasma,
BANK - bank, także: górna lub dolna połowa karty 90-kolumnowej,
BAR - cecha, także: kreska, linia,
BAR CODE - kod kreskowy, kod paskowy,
BAR-CODE READER - pióro świetlne do odczytywania kodu kreskowego,
BAR-CODE SCANNER - pióro świetlne do odczytywania kodu kreskowego,
BAR CODING - kodowanie kreskowe (wyrobów),
BAR DISPLAY - wskaźnik słupkowy, wskaźnik kolumnowy,
BAR PRINTER - drukarka dźwigniowa,
BARREL DISTORSION - zniekształcenia beczkowate (niedoskonałość kineskopu),
BASE - baza, także: podstawa liczenia,
BASE ADDRESS - adres bazowy, adres początkowy, adres podstawowy,
BASE ADDRESS REGISTER - rejestr adresu bazowego,
BASE NOTATION - oznaczanie podstawy liczenia,
BASE NUMBER - podstawa systemu liczenia,
BASE OF FLOATING POINT NUMBER - podstawa liczby zmiennoprzecinkowej,
BASE OF THE COUNTING SYSTEM - podstawa systemu liczenia,
BASE PAGE - strona bazowa.
BASE POINT - punkt bazowy,
BASE SPACE - przestrzeń bazowa,
BASEBAND COAXIAL SYSTEM [illegible]

BASIC - język instrukcji symbolicznych ogólnego zastosowania dla początkujących (wysokopoziomowy język programowania),
BASIC CIRCUIT - układ podstawowy,
BASIC DISC OPERATING SYSTEM (BDOS) - podstawowy dyskowy system operacyjny,
BASIC ELEMENT - element podstawowy,
BASIC INDISSOUBLE INFORMATION UNIT (BIT) - jednostka miary określająca zawartość informacyjną wiadomości,
BASIC INPUT/OUTPUT SYSTEM (BIOS) - podstawowy system WEJŚCIA/WYJŚCIA,
BASIC MACHINE CYCLE - cykl podstawowy,
BASIC OPERATING SYSTEM (BOS) - podstawowy system operacyjny,
BASIC SET - zbiór bazowy,
BATCH - grupa, partia, seria, wsad, także: grupować, gromadzić, paczkować (np. dokumenty),
BATCH DATA PROCESSING - wsadowe (partiowe) przetwarzanie danych,
BATCH OF DATA - partia danych,
BATCH PROCESSING - przetwarzanie wsadowe (partiowe),
BATCH STATION - stacja przetwarzania wsadowego (partiowego),
BATCHING - grupowanie programów we wsady (w partie),
BAUD - bod, jednostka miary szybkości transmisji danych cyfrowych,
BAUD RATE - szybkość transmisji danych cyfrowych (bitów/sekundę),
B-BOX - rejestr B, rejestr modyfikacji,
BCD - patrz: BINARY CODED DECIMAL,
BCD - patrz: BLANK COLUMN DETECTION,
BEAM - promień światła, pęczek promieni,
BEAM ACCESSIBLE MEMORY - pamięć z dostępem strumieniowym,
BEAM LEAD - wyprowadzenie belkowe (w układach scalonych),
BEAM-LEAD BONDING - mikromontaż metodą wyprowadzeń belkowych,
BEEP - sygnał akustyczny modulowany,
BEFORE-IMAGE PROTECTION - zabezpieczenie obrazu pierwotnego,
BEGINING FLAG - znacznik początku,
BEGINING OF THE TAPE (BOT) - znacznik początku taśmy,
BEGINNERS ALL-PURPOSE SYMBOLIC INSTRUCTION CODE - patrz: BASIC,
BEGINNING OF THE FILE - początek zbioru,
BEGINNING OF THE FILE AREA - początek obszaru pliku (zbioru),
BEGINNING OF THE SEEK AREA - początek obszaru przeszukiwania,
BELIEVED CORRECTNESS - "wiarygodność poprawności",
BELL CHARACTER - znak wywołujący sygnał akustyczny,
BENCHMARK - standardowe zadanie rozwiązywane przez komputer w celach porównawczych,
BENCHMARK PROBLEM - problem wzorcowy (dot. porównywania wydajności różnych typów komputerów),
BENCHMARKING - komparatystyka,
BEND - wygięcie (np. karty),
BER - patrz: BIT ERROR RATE,
BIAS - polaryzacja stałą liczbą, zwiększyć o stałą wartość, wartość odchylenia, obciążenie (estymatora), podkład,
BIASED BY... - zwiększone o wartość ...

BIASED NUMBER - liczba nieskalowana (przesunięta),
BIAX - biaks, także: element ferrytowy pamięciowy lub logiczny,
BIDIRECTIONAL - dwukierunkowy,
BIDIRECTIONAL BUS - magistrala dwukierunkowa.
BIFURCATION - rozgałęzienie, rozwidlenie,
BILATERAL - dwustronny ,
BILLI - przedrostek oznaczający miliard (w nomenklaturze amerykańskiej),
BILLIBIT - miliard bitów (kilomegabit),
BILLICYCLE - miliard cykli, gigacykl,
BILLION - bilion (w Wielkiej Brytanii) także: miliard (w USA),
BIMAG CORE - pamięciowy rdzeń magnetyczny mający dwa stany namagnesowania,
BIN - zbiornik (dla kart), pojemnik na taśmę,
BINARY - binarny, także: system dwójkowy,
BINARY ARITHMETIC - arytmetyka binarna.
BINARY CARRY - przeniesienie dwójkowe,
BINARY CELL - komórka dwójkowa (dla przechowywania jednego znaku w zapisie dwójkowym), także: urządzenie dwustanowe,
BINARY CODE - kod binarny, kod dwójkowy,
BINARY CODED DECIMAL (BCD) - system dziesiętny kodowany dwójkowo, kod dwójkowo-dziesiętny,
BINARY COMPLEMENT - uzupełnienie do dwóch, uzupełnienie binarne,
BINARY COUNTER - licznik binarny, licznik dwójkowy,
BINARY DATA - dane binarne,
BINARY DIGIT - cyfra binarna,
BINARY DIGITAL VARIABLE - zmienna binarna,
BINARY ELEMENT - element dwójkowy,
BINARY ERROR CORRECTING CODE - dwójkowy kod poprawiania błędów,
BINARY ERROR DETECTING CODE - dwójkowy kod wykrywający błąd,
BINARY LOGIC - logika dwójkowa, logika dwuwartościowa,
BINARY LOGICAL FUNCTION - funkcja logiczna (dwuwartościowa),
BINARY NOTATION - zapis dwójkowy,
BINARY NUMBER - liczba dwójkowa,
BINARY NUMBER SYSTEM - układ zero-jedynkowy,
BINARY POINT - przecinek dwójkowy,
BINARY PROGRAM - program binarny,
BINARY REPRESENTATION - reprezentacja dwójkowa,
BINARY SCALER - przelicznik dwójkowy,
BINARY SEARCH - przeszukiwanie metodą połowienia,
BINARY-DECIMAL - dwójkowo-dziesiętny,
BINARY-HEXADECIMAL - dwójkowo-szestnastkowy,
BINARY-OCTAL - dwójkowo-ósemkowy,
BINARY-QUATERNARY - dwójkowo-czwórkowy,
BINARY-TO-DECIMAL CONVERSION - konwersja dwójkowo--dziesiętna,
BINDER - urządzenie do oprawiania (np. tabulogramów),
BINNING - sortowanie struktur półprzewodnikowych,
BINOMIAL - dwuczłonowy, dwumienny,
BIOS - moduł zawierający podstawowe procedury obsługi urządzeń zewnętrznych,
BIPOLAR - dwubiegunowy,
BIPOLAR JUNCTION TRANSISTOR - tranzystor bipolarny,
BIQUINARY - dwupiątkowy,

GIEŁDA POMYSŁÓW

Tylko dla szczęściarzy

Przysłowia są mądrością narodów. Jedno z nich mówi, że nadzieja jest matką głupich. Jest jednak drugie, które zapewnia, ze matka swoich dzieci nie opuszcza. W tym drugim należy chyba upatrywać tak dużej popularności gry w „Toto Lotka".

Z grą tą związane są co tydzień emocje dwojakiego rodzaju. Najpierw osoba, która zdecydowała się w nią zagrać przeżywa rozterkę związaną z wyborem liczb. Później gracz doprowadza się do stanów przedzawałowych, tkwiąc przed ekranem telewizora w niecierpliwym oczekiwaniu na wyniki losowania.

Prezentowany tu program pozbawia graczy pierwszego z tych stresów. Wystarczy tylko, zgodnie z poleceniem wyświetlonym na ekranie, wybrać jakiś wariant „lotka", chwilkę poczekać, po czym przepisać słupki liczb na kupon.

Na wszelki wypadek, radzę przygotować sobie odpowiedź na pytanie: „Co zrobić z tak dużą ilością wygranych pieniędzy?" Ży-
[illegible]

```
10 CLS : RANDOMIZE : DIM a(2,3): DIM t$(3,15): DIM o(5,7)
20 LET t$(1)="Express-lotek": LET t$(2)="Super-lotek": LET t$(3)="Duzy-lotek"
30 LET a(1,1)=5: LET a(1,2)=7: LET a(1,3)=6: LET a(2,1)=42: LET a(2,2)=49: LET
a(2,3)=49
40 PRINT "Mozesz zagrac w Toto-lotka"''"Ktory z wariantow wybierasz?    (wcisni
odpowiedni klawisz)"''"     e - express-lotek"''"     s - super-lotek"''"     d -
duzy-lotek"''"Wcisniecie innego klawisza powoduje zakonczenie pracy"
50 IF INKEY$="" THEN GO TO 50
60 IF INKEY$="e" THEN LET i=1: GO TO 200
70 IF INKEY$="s" THEN LET i=2: GO TO 200
80 IF INKEY$="d" THEN LET i=3: GO TO 200
90 PRINT "Odpowiedz spoza repertuaru e/s/d"''"Koniec pracy": STOP
200 PRINT ': FLASH 1;"Poczekaj": FLASH 0
210 FOR k=1 TO 5
220 GO SUB 1000: LET o(k,1)=x
230 FOR l=2 TO a(1,i)
240 GO SUB 1000
250 LET n=1
260 IF o(k,n)=x THEN GO TO 240
270 IF n<l-1 THEN LET n=n+1: GO TO 260
280 LET o(k,l)=x
290 NEXT l
300 FOR n=2 TO a(1,i)
310 LET l=n-1: LET x=o(k,n)
320 IF x>o(k,l) THEN LET o(k,l+1)=x: GO TO 360
330 LET o(k,l+1)=o(k,l)
340 IF l>1 THEN LET l=l-1: GO TO 320
350 LET o(k,1)=x
360 NEXT n: NEXT k
400 CLS : PRINT TAB 5;t$(i): PRINT
410 FOR l=1 TO a(1,i)
420 FOR k=1 TO 5
430 IF o(k,l)<10 THEN PRINT " ";
440 PRINT "  ";o(k,l);
450 NEXT k: PRINT : NEXT l: GO TO 40
1000 LET x=INT (RND*(a(2,i)+1))
1010 IF x=0 OR x>a(2,i) THEN GO TO 1000
1020 RETURN
```

Refleks

„...Jestem stałym czytelnikiem „IKS-a" i bardzo sobie cenię Wasze pismo, w szczególności za umieszczanie w nim programów komputerowych.

Obecnie jestem użytkownikiem mikrokomputera COMMODORE 128, lecz kilka lat temu miałem SHARP-a...

...Gdyby Państwo byli zainteresowani moimi programami, bardzo proszę o odpowiedź!..."

Rafał PŁUTECKI

CZEKAMY! „IKS" JEST WASZ!

Nadesłany program REFLEKS ma dwie cenne zalety: jest *krótki* oraz *użyteczny* (pomaga w opanowaniu „ślepego zapisu").

Po wywołaniu RUN 500 ukazuje się napis POMIAR REFLEKSU, a zaraz potem generator liczb losowych rozpoczyna „zręcznościówkę": wyświetla jedną z liter (od A do Z) — wybraną przypadkowo. Należy SZYBKO nacisnąć WŁAŚCIWY klawisz. Wtedy pokaże się kolejna. Tak „palcujemy" trzydzieści liter.

```
PG 500 REM *************************
YA 502 REM *                       *
HX 504 REM *    POMIAR REFLEKSU    *
QC 506 REM *      *  Atari  *      *
YS 508 REM *                       *
PI 510 REM *************************
PI 520 OPEN #1,4,0,"K:"
JL 522 DIM A$(1),B$(1)
UY 526 PRINT CHR$(125)
IL 528 ? :? "POMIAR REFLEKSU"
GB 530 GOSUB 556:M=N
TQ 532 FOR K=1 TO 31
CQ 533 B=INT(RND(0)*27)
IX 534 A$=CHR$(64+B):B=B+64
YX 536 FOR C=0 TO 10:SOUND 0,100,10
   ,10:NEXT C:SOUND 0,0,0,0
CU 540 PRINT CHR$(125):POKE 752,1
AT 541 ? :? " * ";N-M;" * ";A$
UB 542 IF K>30 THEN 548
HL 543 GET #1,Z:Z=PEEK(763)
YK 544 GOSUB 556
EA 545 IF Z<>B THEN 540
HP 546 NEXT K
MR 548 ? :? "CZAS=";N-M;" JEDNOSTEK
   "
DJ 550 ? :? "JESZCZE RAZ ";CHR$(212
   );"/";CHR$(206);:INPUT A$
CI 552 IF A$=CHR$(84) THEN 530
GB 554 END :REM ******************
   *
FQ 556 N=PEEK(18)*65536+PEEK(19)*25
   6+PEEK(20):N=INT(N/10)
AJ 558 RETURN
```

LIGA MYŚLĄCYCH

Zadanie 1

Kierowca samochodu przebył drogę w ciągu 5 godzin przy przeciętnej prędkości p. Dwie piąte drogi przejechał z prędkością o 8 km/h mniejszą, a resztę z prędkością o 5 km/h większą. Do celu podróży dotarł z opóźnieniem 10 min. Z jaką przeciętną prędkością P przebył drogę kierowca samochodu?

Zadanie 2

Podczas trwania I wojny światowej wojska oblegały stary zamek. Podczas oblężenia kula z armaty uszkodziła posąg, stojący na dziedzińcu zamkowym. Dzień ataku był ostatnim dniem miesiąca, a liczba lat stojącego na dziedzińcu posągu i liczba lat dowódcy wojsk oblegających zamek były liczbami parzystymi. Liczba, wyrażająca dzień miesiąca, w którym nastąpił atak, pomnożona przez numer miesiąca ataku, pomnożona przez podaną w stopach długość piki posągu, pomnożona przez połowę wieku dowódcy wojsk oblegających zamek, pomnożona przez połowę czasu stania posągu daje w iloczynie 451 066. Należy podać rok, w którym wzniesiono posąg na placu zamkowym. Odpowiedź proszę uzasadnić.

Zadanie 3

Połowa pasażerów, wchodząca na przystanku początkowym do autobusu zajęła miejsca siedzące. Na kolejnym przystanku do autobusu wsiadło 8 proc. dotychczasowej liczby pasażerów. Ile osób usiadło na początkowym przystanku, jeżeli wiadomo, że autobus pomieści nie więcej niż 70 osób?

Zadanie 4

Ile zer na końcu ma liczba, którą otrzymamy mnożąc 1000 kolejnych liczb naturalnych tzn. 1 • 2 • 3 • ... • 998 • 999 • 1000? Odpowiedź proszę uzasadnić.

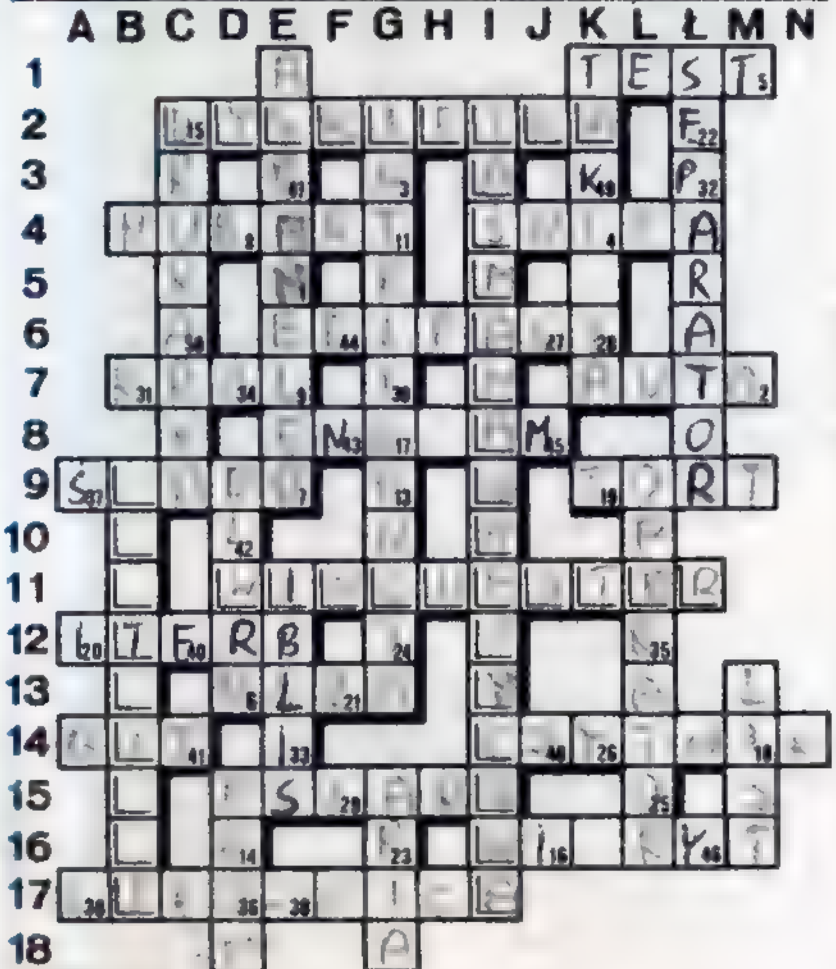
Zadanie 5

Mamy nierzetelną wagę dwuramienną (tzn. ramiona wagi nie są równe). Chcąc odważyć 2 kg cukru postępujemy w następujący sposób. Kładziemy kilogramowy odważnik na lewej szalce, a na prawą sypiemy cukier aż do zrównoważenia. Opróżniwszy obie szalki wagi, kładziemy kilogramowy odważnik na prawej szalce, a na lewą sypiemy cukier aż do zrównoważenia. Czy łączny ciężar odważonego w ten sposób cukru jest równy 2 kg?

Rozwiązania zadań prosimy przesyłać do redakcji do końca grudnia br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody-niespodzianki.

KUPON ▶

KRZYŻÓWKA NR 10



POZIOMO:
1K) instrukcja pozwalająca na sprawdzenie, czy określony punkt ekranu świeci się, czy nie lub... próba, rodzaj sprawdzianu, 4B) rzadkie imię męskie, 4I) asysta, orszak, 6E) stolica Irlandii Pn., 7B) Bolesław Chrobry lub Stefan Batory, 7K) samochód, 8E) ślad pamięciowy, 9A) półprodukt do wyrobu papieru, otrzymany w wyniku mechanicznego rozwłókniania drewna, 9K) urodzinowe ciasto, 12A) lantanowiec o liczbie atomowej 70, 13D) przepływa przez Cieszyn, 14A) 0 lub 1, 14I) urządzenie wejściowe komputera do przyjmowania informacji z taśm lub kart dziurkowanych, 15D) miłośnik psów, 16I) saletra potasowa, saletra indyjska, 17A) pomocnicza lunetka, używana zwłaszcza przy fotografowaniu

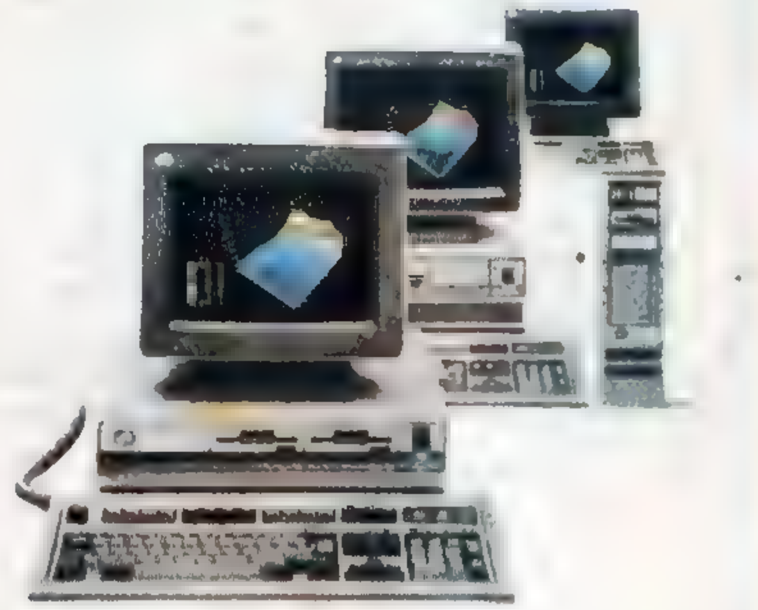
PIONOWO:
C2) np. MPS 801, Seikosha GP500, Star Gemini 10X, D9) polski mikrokomputer z Wrocławia, D15) programowana pamięć tylko do odczytu lub... na trasie Ystad—Świnoujście, E1) język programowania, E11) diabeł, zły duch w wierzeniach muzułmanów, G2) brak jej niestety komputerom, G15) operowa pieśń solowa, K1) tyczka lekkoatlety, L9) pracownik obsługujący komputer, Ł1) średnik, spacja, dwukropek jako oddzielacz, M13) instrukcja wyświetlenia tekstu programu lub... korespondencja

Oznaczone pola (od 2C i 11D poziomo oraz B9 i I2 pionowo) — rodzaje pamięci zewnętrznych mikrokomputerów.
Jako rozwiązanie należy podać cztery nazwy pamięci wpisane w oznaczone pola diagramu krzyżówki oraz część nazw urządzeń we-wy, które otrzymamy po uszeregowaniu liter z pól oznaczonych w prawym dolnym rogu liczbami od 1 do 50.

Rozwiązania krzyżówki **na kartach pocztowych** należy nadsyłać pod adresem redakcji do końca grudnia 1987 r., **naklejając kupon „IKS-a"**. Wśród czytelników rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI

IBM Personal System/2



„IKS" — dodatek „Żołnierza Wolności". Redaguje: Wiesław Cetera (kierownik zespołu); stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 9332. Nr ind. 361682.
K-96

INFORMATYKA KOMPUTERY SYSTEMY

Cena 80 zł

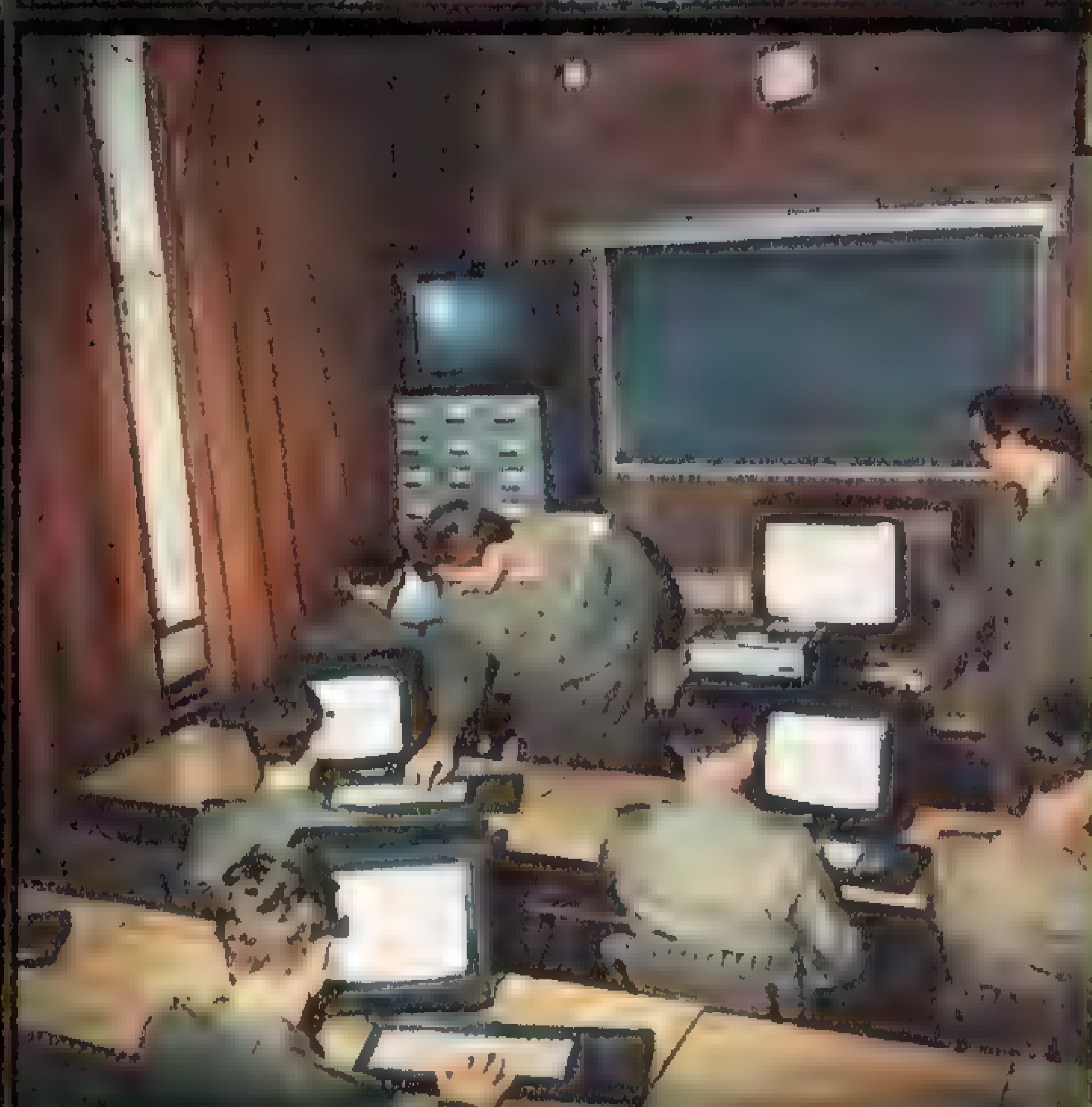
DODATEK „ŻOŁNIERZA WOLNOŚCI" Nr 12/1987 ISSN 0860—2794

Czytelnikom „IKS-a"
i sympatykom
informatyki
życzymy pomyślności
w NOWYM ROKU

REDAKCJA







Za sobą mamy już drugi rocznik naszego miesięcznika. Czas szybko mija. Po okresie ogromnego zainteresowania mikrokomputerami przyszedł czas na „chudszy rok". Miejmy nadzieję, że jest on już za nami. Po kilku latach dyskusji ruszyła wreszcie produkcja komputera dla szkół. Z wrocławskiej ELWRO do klas trafi zapowiadany od dawna Junior, i choć nie zaspokoi on sprzętowego „głodu", to jednak zapowiada poprawę.

Bez wątpienia jesteśmy świadkami informatycznej rewolucji. O tym, co daje komputeryzacja już się nie dyskutuje, choć jeszcze często komputer wzbudza obawy, a co bardziej wrażliwych straszy utratą dotychczasowych kompetencji. Komputery po prostu są i basta. Sprawą informatyków, projektantów i programistów jest ich właściwe zastosowanie. To kolejny krok. Jakże często w tym właśnie momencie przeżywa się pierwsze rozczarowanie. Bo przecież wydano miliony na sprzęt i... nic. Otóż komputer, nawet ten najlepszy, to przecież dopiero początek drogi. I to nie on będzie najkosztowniejszy — najdroższa była i pozostanie ludzka myśl. To właśnie człowiek decyduje o sprawności tego urządzenia, to właśnie twórcza praca szarych komórek decyduje o pracy elektronicznych układów.

To wszystko przed nami. Czeka nas wiele pracy. I wierzmy, że przyniesie ona sukcesy. Porażki, bo i takie się przecież zdarzają w twórczym działaniu, nie mogą nas zniechęcać.

Zatem w nowym roku życzymy Czytelnikom wielu sukcesów, uśmiechu i radości na co dzień, odwagi w działaniu w którym pragniemy Wam zawsze towarzyszyć

REDAKCJA

W NUMERZE:

Kosmos, medycyna i komputerowe obrazy .... — str. 4
W szponach Atari (10) .... — str. 5
Rozkład Gaussa w Asemblerze .... — str. 11
Labirynt .... — str. 14
Wyścig po energię .... — str. 16
Procedury ROM Spectrum (3) .... — str. 18
Edytor kodu maszynowego .... — str. 23
Mikrokomputery IBM PC .... — str. 26
Giełda pomysłów .... — str. 28
Tajemnice matematyki (2) .... — str. 31
Na skróty .... — str. 31

DEEP BLUE C
Kompilator języka C dla mikrokomputera ATARI

Jednym z kompilatorow jezyka C opracowanych dla mikrokomputerów ATARI serii XL/XE jest kompilator DEEP BLUE C. Umozliwia on tworzenie programow w jezyku C o roznej wielkosci, w tym programów duzych – na ktore składa się więcej niż 100 wierszy kodu

Programy napisane w języku C dla tych mikrokomputerów, mimo iż działają nieco wojniej niż napisane w assemblerze, są duzo łatwiejsze w pisaniu i analizowaniu. Takie elementy języka C jak wskaźniki, funkcje rekurencyjne i wysokopoziomowe struktury sterujące czynią skomplikowane oprogramowanie łatwym w projektowaniu, implementowaniu i testowaniu. Co więcej, C jest faktycznie językiem oprogramowania systemowego nowej generacji komputerów. W przeciwieństwie do programów napisanych w jezykach assemblerowych, programy napisane w C będą mogły zostac przeniesione na inne komputery (również na te nie oparte na mikroprocesorze 6502) kosztem nieznacznych modyfikacji.

Do poprawnej pracy kompilatora DEEP BLUE C wymagane są nastepujące akcesoria sprzętowo – programowe:

- 48 kB RAM;
- Stacja dyskow ATARI;
- Edytor tekstowy niezorientowany wierszowo.

Ponadto dla pełnego wykorzystania mozliwosci oferowanych przez kompilator DEEP BLUE C (włączenie do programu tworzonego w języku C procedur w języku assemblera) może okazac się przydatnym ATARI Macro Assembler (MAC65) lub ATARI Assembler Editor.

Z oczywistych powodów DEEP BLUE C nie posiada zaimplementowanej pełnej wersji języka C zdefiniowanej przez Kernighana i Ritchie'a.

W szczególności DEEP BLUE C nie zapewnia realizacji:

1) struktur, unii;
2) tablic wielowymiarowych;
3) liczb zmiennoprzecinkowych;
4) funkcji zwracających inne wartości niż int;
5) operatora jednoargumentowego "sizeof";
6) operatora konwersji typow "typecasting";

Ponadto DEEP BLUE C posiada następujące niestandardowe cechy:

1) Ostatni punkt instrukcji "switch", "case" lub "default" musi być zakończony instrukcją "break", "continue" lub "return".

2) Usunięto starą konstrukcję =<op>. Zamiast niej należy używać <op>=.

3) Znaki (zmienne typu char) nie mają znaku. Przyjmują one wartości od 0 do 255.

4) Łancuchy nie mogą być wydłużone do następnej linii logicznej.

5) Linie kodu źródłowego mogą zawierać maksymalnie 79 znaków.

6) Funkcje mogą mieć maksymalnie 126 argumentów.

DEEP BLUE C zapewnia realizację następujących elementów standardowego języka C:

1) typy danych: char, int, pointer;

2) tablice jednowymiarowe;

3) operatory unarne:

+, -, *, &, ++, --, !, $- (tylda);

4) operatory binarne:

+, -, *, /, %, |, ^, &, ==, !=, <, >, <=, >=, <<, >>, <op>=, &&, ||, ?, przecinek;

5) instrukcje /słowa kluczowe języka C/:

if, else, while, break, continue, return, for, do, switch, case, default;

6) dyrektywy kompilatora:

define, #include;

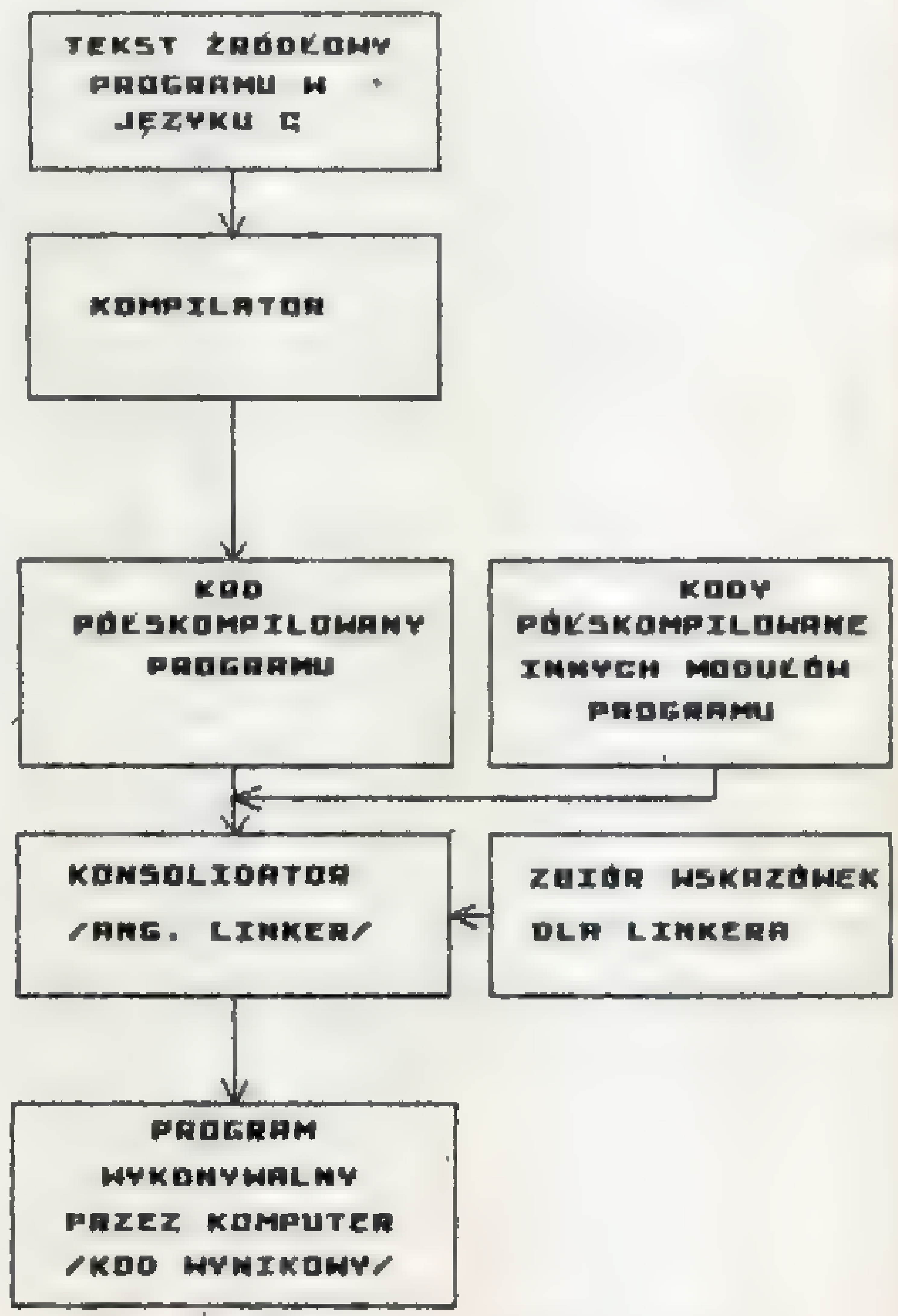
7) linker relokacyjny,

8) stałe: szesnastkowe, ósemkowe, "backslash".

Standardowy język "C" wykorzystuje kilka znaków ASCII niedostępnych w komputerach ATARI. I tak: nawiasy klamrowe zastąpiono kombinacjami $(i $), a znak tyldy zastąpiono parą znaków "$-". Ponieważ znak $ nie jest w standardowym "C" używany, to można przy pomocy funkcji edytora "znajdź i podmień" programy napisane w standardowym "C" zamienić na format akceptowalny przez DEEP BLUE C.

Pisanie programu jest "przekazywaniem" komputerowi sposobu (algorytmu) postępowania niezbędnego do rozwiązania określonego problemu, przy czym forma przekazu jest ściśle określona przez język programowania. Język C, podobnie jak każdy inny język programowania, posiada ściśle określone reguły syntaktyczne (gramatyczne) i semantyczne (znaczeniowe) - zbliżone do reguł rządzących naturalnym językiem, którym porozumiewają się ludzie. Program w języku C nie może być "zrozumiały" przez komputer w formie takiej, w jakiej został napisany - postać ta nazywana będzie dalej kodem źródłowym programu. Musi on być przetłumaczony przez odpowiedni program tłumaczący do postaci, którą komputer "zrozumie" i wykona. Taki program tłumaczący nazywa się kompilatorem. Kompilator języka C tłumaczy kod źródłowy programu na postać przejściową (jeszcze niewykonywalną przez komputer, a potrzebną tylko do przeprowadzenia pełnego procesu kompilacji), zwaną kodem półskompilowanym. Aby kod półskompilowany był wykonywalny przez komputer, musi być najpierw połączony z kodami półskompilowanymi procedur użytkowych, stanowiących część kompilatora, zwaną biblioteką języka C lub biblioteką kompilatora C. Dopiero w wyniku konsolidacji powstaje program, który może być wykonany przez mikrokomputer. Taka postać programu nazywana jest kodem wynikowym. Proces konsolidacji (łączenia) wykonywany jest również przez odpowiedni program zwany konsolidatorem lub, z angielskiego - linkerem. Proces konsolidacji dla programisty przyzwyczajonego do języków takich jak PASCAL czy BASIC może początkowo wydać się niepotrzebnym i niezrozumiałym. I chociaż jest on dodatkowym przebiegiem w procesie kompilacji programu w języku C, to właśnie dzięki niemu język ten ma wiele zalet w porównaniu z innymi językami. Praktyką stosowaną powszechnie przez programistów w języku C jest pisanie programu "po kawałku", oddzielne kompilowanie i testowanie każdego fragmentu programu i na końcu łączenie takich kawałków w jeden cały program. Każdy napisany, przetestowany i skompilowany fragment jakiegoś programu może być w postaci półskompilowanej dołączony do biblioteki języka i wykorzystany w innym programie bez konieczności tworzenia go od nowa. Dzięki takiej technice proces tworzenia programów w języku C jest znacznie szybszy niż w innych językach wysokopoziomowych.

Proces kompilacji można zobrazować schematem przedstawionym na rys.1.



Reasumując: aby przekształcić program napisany w języku C na papierze na realizowalny program w języku maszynowym, należy wykonać cztery kroki:

1. Z wykorzystaniem edytora przygotować program jako jeden lub więcej zbiorów z tekstem źródłowym.

2. Przekształcić zbiory źródłowe w zbiory półskompilowane poprzez skompilowanie ich przy pomocy kompilatora DEEP BLUE C.

3. Stworzyć zbiór linkujący. Zbiór linkujący powinien zawierać nazwy wszystkich zbiorów półskompilowanych, które są częściami programu i jest wykorzystywany przez linker (w kroku 4) do połączenia razem wszystkich części programu.

4. Przy pomocy linkera DEEP BLUE C następuje połączenie wszystkich poszczególnych zbiorów półskompilowanych w kod wynikowy kompletnego programu.

KOSMOS, MEDYCYNA I KOMPUTEROWE OBRAZY

Jako początek ery kosmicznej przyjmuje się rok 1957, gdy radziecki Sputnik I — pierwszy zbudowany przez człowieka obiekt stał się satelitą Ziemi. Krótko potem Sputnik II wzniósł w kosmos pierwszą żywą istotę — pieska Łajkę. Łuna I okrążyła Księżyc, a Łuna II dotknęła jego powierzchni. Dzięki Łunie III człowiek mógł zobaczyć drugą stronę Księżyca. Statek ten przesłał na Ziemię, drogą radiową, informacje odzwierciedlające obraz wiecznie odwróconej od nas części Księżyca.

W tym czasie Amerykanie mieli wielkie kłopoty ze zbudowaniem rakiety zdolnej pokonać grawitację ziemską. Dwa Rangery — zbudowane, by powtórzyć sukces Łuny II — przekazanie obrazu Księżyca na Ziemię — nie mogły jej niestety opuścić, trzeci rozbił się o Księżyc, a dwa kolejne minęły go o tysiące kilometrów. Dopiero Ranger 6 zdołał przesłać na Ziemię jeden obraz Księżyca. Ranger 7 był źródłem już 4300 obrazów — pełnych sygnałów wideo.

Jednym z głównych celów wspomnianych „eksperymentów kosmicznych" były zdjęcia naszego Księżyca. Stąd też bardzo ubolewano nad kiepską jakością uzyskiwanych w ten sposób zdjęć. W najlepszym wypadku oceniane były one jako dostateczne. Brak ostrości, odpowiedniego kontrastu oraz wiele innych zniekształceń zdecydowały o gorączkowych poszukiwaniach nowych metod i technik mogących udoskonalić tworzenie, transmisję oraz ewentualną korektę obrazów uzyskiwanych z kosmosu.

Dr Robert Nathan uznał, że przyczyny kiepskiej jakości odbieranych obrazów należy głównie upatrywać w technice przesyłania obrazów ze statków kosmicznych. Dotychczas stosowano bowiem wyłącznie technikę analogową. Oznacza to, że informacje o obrazie wysyłane i odbierane były w postaci ciągłego sygnału elektrycznego (oczywiście za pośrednictwem fal radiowych), którego amplituda mówiła o tym, czy określony „kawałek" obrazu jest mniej lub bardziej czarny. Sygnał taki jest w dużym stopniu podatny na zakłócenia — stąd wspomniane narzekania na słabą jakość zdjęć z kosmosu. Pan Nathan zaproponował rewolucyjne, jak na owe czasy, rozwiązanie — zamieniać analogowy sygnał wideo, otrzymany z kamer sputnika, na postać cyfrową i dopiero w takiej formie przesyłać go na Ziemię. Innymi słowy, każdy wyróżniony punkt obrazu zamieniano na liczbę (np.: od 0 do 255). Odbiorem tak uformowanych informacji obciążono komputer, który gromadził je nie tylko w celu ich odtworzenia w postaci obrazu na monitorze, ale także w celu ewentualnej korekty zniekształceń. Zabieg ten polega na usuwaniu zniekształceń geometrycznych, których źródłem jest sama kamera, braku fotometrycznej linearności kamery (różne fragmenty tej samej szarości obrazu kamera zamienia na różne wartości sygnału elektrycznego) i szeregu innych zakłóceń. Wspomniany brak linearności usuwano przez „odejmowanie" od uzyskanego z kosmosu obrazu — zakłóceń wprowadzanych przez kamerę, które komputer wcześniej zmierzył na Ziemi. Na przykład kamera zwiększa czerń obrazów w lewym górnym rogu — komputer „wiedząc" o tym, odejmuje od liczb odzwierciedlających tę część obrazu (z kosmosu) wartość 17. Powoduje to „rozjaśnianie" tego zaciemnionego (przez kamerę) obrazu — czyniąc go bliższym oryginałowi. Podobnie dokonano wielu innych korekt obrazów z kosmosu. Efekty przerosły oczekiwania twórców metody. Cyfrowa reprezentacja obrazu stała się standardem.

Po raz pierwszy „cyfrowe obrazy" wykorzystano w przesyłaniu danych o obrazach ze statku Mariner 4, wędrującego do Marsa i Wenus. Informacje te przebyły niemal 80 mln kilometrów. Dane o każdym pikselu (wyróżnianym elemencie obrazu) — 8 bitów — przesyłane były aż jedną sekundę! Odebranie pierwszego obrazu z Marsa trwało osiem godzin. Niestety, okazało się, że jedna połowa tego obrazu jest czarna, a druga „sterylnie" biała. Przyczyną okazała się kiepska pogoda na Marsie — było pochmurnie... Następne zdjęcia, dzięki poprawie pogody na Marsie, okazały się lepsze — Mariner 4 przesłał pierwsze „bliskie" zdjęcie Czerwonej Planety.

Dzięki cyfrowej reprezentacji zdjęć przesyłanych przez kolejne Marinery — nawet znaczne zakłócenia okazały się łatwe (dzięki komputerom) do usunięcia. Wykorzystano do tego celu między innymi dwuwymiarową transformatę Fouriera.

Podczas „wędrówek" Marinerów i Voyagerów, zaczęto stosować wykorzystaną tam technikę cyfrowej „obróbki" obrazów w badaniach biomedycznych.

Wydaje się, że warto w tym miejscu przypomnieć, iż już w 1895 roku Wilhelm Roentgen odkrył promienie X — elektromagnetyczne promieniowanie o bardzo dużej częstotliwości, które tworzyło obraz wnętrza prześwietlanego ciała. Promienie te, przechodząc przez ciało, są różnie absorbowane przez tkankę, kości i organy wewnętrzne. „Wychodząc" z ciała tworzą na błonie fotograficznej obraz jego wnętrza — mocniej lub słabiej naświetlając różne fragmenty kliszy. Zważywszy jednak na zamazania uzyskiwanych w ten sposób obrazów — wynikające z ruchu pacjenta, nakładania się konturów prześwietlanych organów — poprawne odczytanie tak uzyskanego zdjęcia wymaga odpowiednich kwalifikacji.

Późniejsze odkrycia pozwoliły wykorzystać inne formy energii, by tworzyć obraz wnętrza ludzkiego ciała. Ultrasonografia wykorzystuje fale dźwiękowe, które odbijając się od organów, są odbierane przez specjalne urządzenia i prezentowane na monitorze TV. Medycyna nuklearna natomiast wykorzystuje takie kamery, które wykrywają emisję promieni gamma z szybko rozpadających się radioaktywnych substancji wprowadzonych do ciała. Technika ta pozwala na zobrazowanie organów, dróg przepływu cieczy przez narządy wewnętrzne i obserwację wielu różnych procesów metabolicznych.

Rewolucja komputerowa przyczyniła się do tego, że badacze zaadaptowali technikę „cyfrowych obrazów" do potrzeb medycyny.



AGPOL ma dobrą rękę. Po mikrokomputerach przyszła kolej na video. W Viktorii pełna gala. Zainteresowanie duże, klientów nieco mniej.

(Foto: J. Zelman)



We wspomnianych technikach wykorzystywania „cyfrowych obrazów" w medycynie jednym z podstawowych elementów elektronicznych jest szybki konwerter, zamieniający analogowe sygnały — odzwierciedlające obraz (promienie X, dźwięk, promienie gamma) na wartości cyfrowe. Pozwala to wykorzystywać tę samą co w obrazach z kosmosu technikę, by obrazy „poprawiać", eliminować zakłócenia, nawet wynikające z ruchu pacjenta.

Powszechnie znanym już przykładem stosowania techniki komputerowej w oglądaniu wnętrza organizmu jest tomografia komputerowa (pisaliśmy o niej w poprzednim numerze). Umożliwia ona obserwowanie trójwymiarowych obrazów wnętrza ciała pacjenta. Wykorzystywany do tego celu specjalny sprzęt pozwala „obracać" trójwymiarowy obraz tak, że lekarz może np. oglądać nasze serce z każdej strony. Uzyskano to dzięki zapamiętywaniu wielu „warstw" prześwietleń w pamięci komputera. Obecnie komputery te są bardzo szybkie — 10 mln instrukcji/sek. Bufor takiego obrazu zapamiętuje 48 bitów na każdy piksel. 12 bitów na każdy z 3 kolorów — czerwony, zielony i niebieski oraz 12 bitów odzwierciedlających stopień przezroczystości prześwietlanego obiektu.

W omawianych pracach — komputerowe przetwarzanie obrazów — wąskim gardłem okazała się szybkość komputerów. Wykorzystano do tego celu równoległą pracę bardzo szybkich układów scalonych (VLSI). W efekcie uzyskano od 50- do 1000-krotnie większą szybkość działania komputerów. Układy te po raz pierwszy zamontowano w statku Voyager 2, który przesłał w 1986 roku między innymi obrazy z Mirandy — najmniejszego księżyca Urana.

Kończąc należy zwrócić uwagę, że przetwarzanie obrazów jeszcze „ręcznie" wykonywano już podczas II wojny światowej do „czyszczenia" obrazów uzyskiwanych ze stacji radiolokacyjnych. Obecnie dzięki udoskonaleniu technik przetwarzania informacji w ogóle — poprzez wykorzystanie komputerów — z powodzeniem oglądamy zarówno wnętrze naszego ciała, jak i odległe obiekty Systemu Słonecznego.

W. GOGOŁEK

Opracowane na podstawie: K. Sheldon, PROBING SPACE BY CAMERA, Byte, marzec'87

CPC 6128

Samolot

```
10 ' Samolot
15 INPUT "ILE RAZY POWTORZYC
   GLOS SAMOLOTU";
17 FOR L=1 TO I
20 ENV 3.5.1,100
30 SOUND 10,758.500,5,3.3,3
40 SOUND 17.20,-1,1
50 ENV 2.10,-1.100
60 SOUND 10,758.1000.11.2,2.3
70 SOUND 17.20,-1,1
80 NEXT
```

W szponach Atari (10)

Programowe techniki generowania dźwięku

Po zapoznaniu się z możliwościami układów generowania dźwięku (przedstawionymi w poprzednim odcinku) wypada przystąpić do następnych prób, których celem będzie uzyskanie interesujących efektów dźwiękowych, służących jako „przerywniki" w opracowywanych przez nas programach.

Najprostszym sposobem, tzw. statycznym, jest włączenie jednego lub więcej generatorów dźwięku, przejście do innej działalności, a następnie wyłączenie ich. Tworzone w taki sposób efekty akustyczne ograniczają się do pojedynczych tonów, trzasków i brzęczeń. Możliwe jest jednak uzyskanie interesujących efektów, na przykład przez wykorzystanie interferencji fal dźwiękowych. Wprowadźmy następujące instrukcje:

SOUND 0,255,10,8
SOUND 1,254,10,8

W wyniku nakładania się fal dźwiękowych otrzymujemy w pewnych momentach zwiększenie głośności, a w innych — zmniejszenie (do całkowitego zaniku włącznie). Efekt ten można zobaczyć na wykresie sumy funkcji opisujących fale. Można zauważyć, że im mniejsza różnica częstotliwości obu fal, tym dłuższy okres powtarzania wzoru dźwięku wynikowego. W miarę wzrostu różnicy częstotliwości krótszy staje się okres powtarzania.

Równie ciekawe efekty można uzyskać, nakładając dźwięki z trzech i czterech kanałów. Jako przykład zbadajmy wynik działania następujących instrukcji:

SOUND 0,255,10,8
SOUND 1,254,10,8
SOUND 2,253,10,8
SOUND 3,252,10,8

W prosty sposób można uzyskać inny efekt, zmieniając częstotliwość zegara z 64 kHz na 15 kHz za pomocą instrukcji:

POKE 53768,1

Interesującym pomysłem może być napisanie programu kreślącego na ekranie wzór wzajemnego oddziaływania dwóch, trzech i czterech kanałów.

Uzyskanie bardziej skomplikowanych efektów akustycznych wymaga użycia dynamicznego sposobu generowania dźwięku. Najwygodniej byłoby wykorzystać do tego język BASIC. Ma on ograniczone możliwości tworzenia dźwięku (instrukcja SOUND niszczy specjalne ustawienia rejestru AUDCTL) oraz małą szybkość. Jeśli program nie jest całkowicie przeznaczony do generowania dźwięku, rzadko kiedy mamy wystarczającą ilość czasu procesora na uzyskanie czegoś więcej niż dźwięk statyczny lub wątpliwej jakości dźwięk dynamiczny. Jedynym rozwiązaniem jest chwilowe zatrzymanie przetwarzania podczas tworzenia dźwięku. Ponadto mogą wystąpić zauważalne opóźnienia włączania poszczególnych generatorów podczas używania kilku kanałów do tworzenia muzyki.

Najlepszym sposobem wyeliminowania tych wad byłoby użycie do generowania dźwięku języka asemblera. Jak wiemy, nie jest to takie proste. **Dlatego lepsze jest rozwiązanie kompromisowe: dostarczenie gotowych procedur w języku maszynowym, możliwych do wykorzystania z poziomu BASIC-a, które pomogą wyeliminować chociaż część wad tego języka.**

W celu zademonstrowania możliwości generowania dźwięku w BASIC-u, proponujemy wprowadzić i uruchomić program nr 1. Jest on przeznaczony do poszukiwania ciekawych efektów akustycznych dzięki odpowiednio dobranym parametrom zmian głośności i częstotliwości dźwięku oraz zniekształceń. Wykorzystuje dwie pętle **FOR-NEXT**. Użytkownik musi podać szczegółowe dane dotyczące wartości początkowych, końcowych i kroków w obu pętlach oraz początkowej wysokości dźwięku i zniekształcenia. Aby „poczuć", jakie wartości należy podawać, proszę wprowadzić na próbę następujące wartości: ZNIEKSZT = 10, DŹWIĘK = 20, L1 od 15, L1 do 0,L1 krok — 0.5, L2 od 3,L2 do 0,L2 krok — 1. Zauważmy, że dźwięk wydaje się wibrować w miarę zanikania. Jeśli chcemy usłyszeć go ponownie, musimy nacisnąć klawisz OPTION. Możemy eksperymentować dowolnie.

Może się zdarzyć, że podczas prób uzyskamy dźwięk, który zechcemy użyć w którymś z naszych programów. W tym celu u góry ekranu wyświetlany jest podprogram, który umożliwia uzyskanie takiego efektu. Wystarczy go zanotować. Gdy jesteśmy gotowi do dalszych poszukiwań, możemy nacisnąć klawisz START.

Kolejny program, (nr 2) umożliwia szybkie ustawianie parametrów dla czterech kanałów dźwiękowych. Służy do tego krótki podprogram w języku maszynowym, przechowywany w zmiennej SIMUL$. Dzięki tej procedurze program w BASIC-u może bardzo szybko manipulować wszystkimi kanałami. Wystarczy wywołać podprogram, wraz z odpowiednimi parametrami określającymi wartości rejestrów dźwiękowych, za pomocą funkcji USR — jak w linii 50 programu nr 2. Parametry powinny być uporządkowane jak w przykładzie, najpierw zawartość rejestru częstotliwości, a następnie zawartość rejestru sterującego, i tak dla każdego kanału, który chcemy używać. Prezentowana procedura umożliwia także manipulowanie mniejszą liczbą kanałów: to jest 1,2 i 3 lub 1 i 2 lub tylko 1. Po prostu nie wprowadzamy parametrów dla nieużywanych kanałów do wnętrza funkcji USR.

Procedura SIMUL$ ma jeszcze jedną zaletę dla programisty; używanie jej nie zmienia zawartości rejestru AUDCTL — jest zatem lepsza i wygodniejsza w użyciu od instrukcji SOUND.

Początkujący programiści pytają nas często o sposób napisania sekwencji instrukcji umożliwiających wygenerowanie kilkudziesięciu

taktów melodii, które mogłyby służyć do ilustracji dźwiękowej programu. Rozwiązanie jest bardzo proste, jak widać na wydruku programu nr 3. Kłopoty zaczynają się dopiero wtedy, gdy musimy wprowadzić dane o poszczególnych dźwiękach: wysokości nuty i czasie jej trwania. Podając czas trwania dźwięku, należy pamiętać tylko o zachowaniu właściwych proporcji między poszczególnymi nutami. Czas trwania całej nuty (i odpowiednio pozostałych) możemy dobrać eksperymentalnie, przez wybranie właściwego tempa.

Podanie wysokości dźwięku wymaga nieco trudu. Z zapisu na pięciolinii musimy odczytać wartość nuty, a następnie korzystając z odpowiedniej tabeli (na przykład tej, która została zamieszczona w poprzednim odcinku) zamienić ją na liczbę. Przy dłuższych fragmentach muzycznych jest to czynność żmudna i czasochłonna, łatwo wtedy o pomyłkę.

Dane o dźwięku umieszcza się w programie w liniach DATA. Dzięki temu tę samą sekwencję instrukcji można wykorzystać do wygrywania różnych melodii, zmieniając zawartość linii DATA albo numer linii, od której rozpoczyna się pobieranie danych.

Prezentowany program generuje dźwięk tylko w jednym kanale. W celu użycia większej ilości kanałów należy go nieco zmodyfikować, ponieważ dźwięki w różnych kanałach mogą mieć różne długości. W tym celu musimy przyjąć pewne założenie: najkrótszą używaną przez nas nutą będzie szesnastka. Długość dowolnej nuty podajemy przez kilkakrotne umieszczenie jej wysokości w linii DATA, tzn. dla ósemki — 2 razy, dla ćwiartki — 4 razy, a dla półnuty — 8 razy. Jest to dosyć żmudne, lecz umożliwia zmianę częstotliwości generatorów (w zasadzie dowolnej liczby) po każdej szesnastce. Ten sposób prezentowany jest w programie nr 4, gdzie jeden z kanałów udaje dźwięk perkusji.

Kolejny program, **„Muzykant"**, przeznaczony jest do tworzenia przerywników muzycznych w różnorodnych grach pisanych w BASIC-u. Tym się różni od poprzednich, że dane o nutach możemy wpisywać bezpośrednio z zapisu nutowego (instrukcją działania jest program nr 6).

Na przykład, dla odegrania nut C, D, E, F, G, A, B — w takiej kolejności — musimy zapisać ciąg „CDEFGAB" w zmiennej PLAY$. Resztę wykona „Muzykant".

Postać instrukcji USR (330 w programie 6) służącej do wywołania procedury, jest następująca:

X=USR (ADR (MUSIC$), ADR (PLAY$), ADR (NT$), ADR (TIME$), GLOS)

Zmienna MUSIC$ przechowuje rozkazy procedury „Muzykant" i ma długość 159 bajtów. NT$ zawiera wartości dziesiętne wszystkich nut, a TIME$ przechowuje różne czasy trwania nut (MUSIC$, NT$ i TIME$ dostarczane są wraz z procedurą, zatem nie potrzebujemy martwić się o nie).

GLOS ma takie samo znaczenie, jak czwarty parametr instrukcji SOUND i musi być między 0 a 15.

Zmienna PLAY$ zawiera aktualne nuty, które będą grane i może zawierać opcjonalnie informację o oktawie, tempie i czasie trwania nut. Poniżej podane są dopuszczalne znaki, jakie mogą wystąpić w ciągu PLAY$.

Nuty *A* do *G* wykonują dokładnie to, co na standardowej skali muzycznej. Krzyżyki reprezentowane są przez #, jak w A#. Bemole oznaczane są przez —, tak jak A—.

On określa, która oktawa będzie używana, *O* jest wielką literą „O", nie zerem, a *n* jest

UWAGA!

Dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną linii dla Edytora BASIC'u (IKS nr 4 z br.). Nie wprowadzamy ich do komputera.

Występujące w programach podkreślone znaki należy wprowadzić w negatywie.

```
FK 1 REM ***************
WZ 2 REM * Program nr 1 *
FM 3 REM ***************
NJ 4 REM
FW 10 GRAPHICS 0:DIM X$(1),BL$(30
   ):BL$=" ":BL$(30)=BL$:BL$(2)=B
   L$
JE 20 ? :? "DZWIEK = dowolna licz
   ba od 0 do 255,":? "wstawimy j
   a do P2."
ZM 30 ? :? "P1 = petla (zewnetrzn
   a) glosnosci."
AE 40 ? "Wprowadz dowolna liczbe
   od 0 do 15,":? "odpowiednio dl
   a od, do i krok."
UZ 50 ? :? "P2 = Petla 2 (wewnetr
   zna) dzwieku.":? "Wprowadz dow
   olna liczbe od 0 do 255,"
NR 60 ? "odpowiednio dla od, do i
   krok."
YT 70 ? :? "Nacisnij <RETURN>";:I
   NPUT X$
DE 80 GRAPHICS 0:? :? :? :? " TES
   T DZWIEKU "
VK 90 POKE 752,1:C=RND(0)*16:POWT
   =0
GM 100 SETCOLOR 1,C,2:SETCOLOR 2,
   C,8:SETCOLOR 4,C,2
XC 110 POSITION 2,3:? "WPROWADZ Z
   NIEKSZT ",:TRAP 130:INPUT D:TR
   AP 40000
BL 120 IF D=0 OR D=2 OR D=4 OR D=
   6 OR D=8 OR D=10 OR D=12 OR D=
   14 THEN 140
MG 130 POSITION 2,3:? BL$:GOTO 11
   0
NF 140 POSITION 2,5:? "WPROWADZ D
   ZWIEK",:TRAP 160:INPUT P:TRAP
   40000
IM 150 IF P<256 THEN 170
RE 160 POSITION 2,5:? BL$:GOTO 14
   0
IZ 170 POSITION 2,7:? "WPROWADZ L
   1 OD",:TRAP 190:INPUT F1:TRAP
   40000
ZI 180 IF F1<16 THEN 200
WC 190 POSITION 2,7:? BL$:GOTO 17
   0
CM 200 POSITION 2,9:? "WPROWADZ L
   1 DO",:TRAP 220:INPUT T1:TRAP
   40000
FL 210 IF T1<16 THEN 230
PW 220 POSITION 2,9:? BL$:GOTO 20
   0
FS 230 POSITION 2,11:? "WPROWADZ
   L1 KROK",:TRAP 250:INPUT S1:TR
   AP 40000
HR 240 IF S1<16 THEN 260
CQ 250 POSITION 2,11:? BL$:GOTO 2
   30
DD 260 POSITION 2,13:? "WPROWADZ
   L2 OD",:TRAP 280:INPUT F2:TRAP
   40000
BP 270 IF F2<256 THEN 290
HT 280 POSITION 2,13:? BL$:GOTO 2
   60
XT 290 POSITION 2,15:? "WPROWADZ
   L2 DO",:TRAP 310:INPUT T2:TRAP
   40000
AN 300 IF T2<256 THEN 320
MD 310 POSITION 2,15:? BL$:GOTO 2
   90
MZ 320 POSITION 2,17:? "WPROWADZ
   L2 KROK",:TRAP 340:INPUT S2:TR
   AP 40000
CH 330 IF S2<256 THEN 350
GM 340 POSITION 2,17:? BL$:GOTO 3
   20
MT 350 IF PONT>0 THEN GOSUB 530:G
   OTO 480
VS 360 GOSUB 530:SOUND 0,0,0,0:?
   CHR$(125)
PJ 370 ? :? " PODPROGRAM GENERUJA
   CY DZWIEKU "
WE 380 ? :? "100 FOR L1=";F1;" TO
   ";T1;" STEP ";S1
KM 390 ? "110 FOR L2=";F2;" TO ";
   T2;" STEP ";S2
YH 400 ? "120 SOUND 0,";P;"-L2,";
   D;",L1"
PP 410 ? "130 NEXT L2":? "140 NEX
   T L1":? "150 RETURN"
HG 420 FOR OPO=1 TO 300:NEXT OPO
II 430 FOR CZ=1 TO 5:POSITION 2,2
   0:? " <START> RESTART":?
   " OPTION POWTORZENIE "
UN 440 FOR OPO=1 TO 20:NEXT OPO
UF 450 POSITION 2,20:? " <START>
   RESTART":? " OPTION
   POWTORZENIE "
HP 460 FOR OPO=1 TO 20:NEXT OPO
IN 470 NEXT CZ
NE 480 SOUND 0,0,0,0
GM 490 IF PEEK(53279)=6 THEN 80
FI 500 IF PEEK(53279)=3 THEN 520
IN 510 GOTO 480
TH 520 PONT=PONT+1:GOTO 350
PH 530 FOR L1=F1 TO T1 STEP S1
SH 540 FOR L2=F2 TO T2 STEP S2
HH 550 TRAP 570:SOUND 0,P-L2,D,L1
   :TRAP 40000
PE 560 NEXT L2:NEXT L1:RETURN
IT 570 ? :? " ZLE PARAMETRY, SPR
   OBUJ JESZCZE RAZ":SOUND 0,0,0,
   0
LI 580 FOR OPO=1 TO 250:NEXT OPO
```

```
FK 1 REM ***************
XT 2 REM * Program nr 2 *
FM 3 REM ***************
NJ 4 REM
NF 10 SOUND 0,0,0,0:DIM SIMUL$(16
   )
RK 20 RESTORE 9999:X=1
CE 25 READ Q:IF Q<>-1 THEN SIMUL$
   (X)=CHR$(Q):X=X+1:GOTO 25
JT 27 RESTORE 100
GM 30 READ F1,C1,F2,C2,F3,C3,F4,C
   4
BX 40 IF F1=-1 THEN END
WE 50 X=USR(ADR(SIMUL$),F1,C1,F2,
   C2,F3,C3,F4,C4)
FF 55 FOR X=0 TO 150:NEXT X
SF 60 GOTO 30
BP 100 DATA 182,168,0,0,0,0,0,0
JB 110 DATA 162,168,182,166,0,0,0
   ,0
KJ 120 DATA 144,168,162,166,35,16
   6,0,0
HD 130 DATA 128,168,144,166,40,16
   6,35,166
FN 140 DATA 121,168,128,166,45,16
   6,40,166
LF 150 DATA 108,168,121,166,47,16
   6,45,166
WJ 160 DATA 96,168,108,166,53,166
   ,47,166
AK 170 DATA 91,168,96,166,60,166,
   53,166
YR 999 DATA -1,0,0,0,0,0,0,0
IE 9000 REM
JH 9010 REM
LI 9020 REM te instrukcje DATA za
   wieraja
WB 9030 REM program w jezyku masz
   ynowym
RM 9999 DATA 104,133,203,162,0,10
   4,104,157,0,210,232,228,203,20
   8,246,96,-1
```

```
FI 1 REM ***************
YH 2 REM * Program nr 3 *
FM 3 REM ***************
NJ 4 REM
WS 10 PRINT "PODAJ TEMPO "
OD 20 INPUT T
HG 30 RESTORE
PU 40 READ A,B
DO 50 IF A=-1 THEN 10
HO 60 SOUND 0,A,10,10
PI 70 FOR I=1 TO B*T
```

```
JK 70 NEXT I
SF 90 GOTO 40
HU 200 DATA [illegible]
PT 210 DATA [illegible]
   162,16,0,8,162,8,0,16,-1,-1
```

```
FK 1 REM ****************
ZH 2 REM * Program nr 4 *
FM 3 REM ****************
NJ 4 REM
HE 10 RESTORE
KC 20 FOR J=1 TO 16
PT 30 READ A,B
FH 40 SOUND 0,B,10,8
WG 50 FOR I=A TO 0 STEP -A/14
GP 60 SOUND 1,1,8,I
IY 70 NEXT I
GC 80 FOR D=0 TO 20:NEXT D
JG 90 NEXT J
PC 100 GOTO 10
UX 110 DATA 8,243,4,243,4,0,8,243
   ,4,162,4,162,8,0,4,162,8,0,1,0
   ,8,0,1,0,8,0,1,0,8,0,1,0
```

```
FK 1 REM ****************
HB 2 REM * Program nr 5 *
FM 3 REM ****************
NJ 4 REM
WJ 30 REM Program tworzy linie
CG 35 REM 10-16 dla programu
GO 40 REM MUZYKANT
AU 50 DIM FN$(20),TEMP$(20),AR$(93
   )
HO 60 DPL=PEEK(10592):POKE 10592,
   255
ZH 70 FN$="C:":REM Nazwa tworzone
   go zbioru
DO 80 GRAPHICS 0:POSITION 8,3:? "
   Program z czasopisma ANTIC"
XG 90 POSITION 10,4:? "napisal: M
   ATT LOVELESS"
PW 100 POKE 10592,DPL:TRAP 170
JO 110 ? :? :? "Tworzenie ";FN$:?
    "...prosze czekac."
LQ 120 RESTORE :READ LN:LM=LN:DIM
    A$(LN):C=1
BK 130 AR$="":READ AR$
XW 140 FOR X=1 TO LEN(AR$) STEP 3
   :POKE 752,255
DT 150 LM=LM-1:POSITION 10,10:? "
   (Odliczanie...T-";INT(LM/10);"
   )  "
UY 160 A$(C,C)=CHR$(VAL(AR$(X,X+2
   ))):C=C+1:NEXT X:GOTO 130
ZB 170 IF PEEK(195)=5 THEN ? :? :
   ? CHR$(125);"ZBYT DUZO LINII D
   ATA!":? "NIE MOGE UTWORZYC ZBI
   ORU!":END
WP 180 IF C<LN+1 THEN ? :? CHR$(1
   25);"ZBYT DUZO LINII DATA!":?
   "NIE MOGE UTWORZYC ZBIORU!":EN
   D
AL 200 OPEN #1,8,0,FN$
PP 210 POKE 766,1:? #1;A$;:POKE 7
   66,0
OT 220 CLOSE #1:GRAPHICS 0:? "KO
   NIEC "
LM 1000 DATA 378
HN 1010 DATA 0490480032068073077032
   0780684036040053054041050078008
   403606103414407203500015307603
   7000128064031
FF 1020 DATA 000136068033000024312
   106002300000000000021710805300
   023011405700019309604700007710
   2050000182091
FS 1030 DATA 045000000000000000016
   208104000017308504200003415504
   904903206807307703208407307706
   9036040049050
RR 1040 DATA 041058084073077069003
   606103400200400801600501002004
   001503006012003415504905103206
   8073077032080
```

```
GU 1050 DATA [illegible]
KZ 1060 DATA [illegible]
   [illegible]
   32191041332181041332211041332
   [illegible]
TH 1070 DATA 1332[illegible]
   [illegible]
JN 1080 DATA 2240[illegible]21332162
   [illegible]
   [illegible]
ER 1090 DATA [illegible]
   [illegible]
HF 1100 DATA [illegible]
   [illegible]
FT 1110 DATA 04007[illegible]
   [illegible]
   40[illegible]1320452080030241050
   120101514400[illegible]
EF 1120 DATA 16[illegible]01001002410121
   5162177220141000210165214024 10
   1216188177222160000613202013702
   0176252140000
UI 1130 DATA 210024144131034155
```

```
FK 1 REM ****************
HY 2 REM * Program nr 6 *
FM 3 REM ****************
NJ 4 REM
KD 9 DIM BL$(30):BL$=" ":BL$(30)=
   " ":BL$(2)=BL$
AY 10 REM
BH 11 REM
BL 12 REM
GU 13 REM Linie 10-16 utworzone
UM 14 REM beda przez program nr 5
BJ 15 REM
BE 16 REM
BZ 17 PLAY$="T3 O4 L2 G,A,F; O3 F
   ; O4 L3 C,,."
AF 18 GRAPHICS 0:POKE 82,2:? :POK
   E 752,1:SETCOLOR 2,5,6:SETCOLO
   R 1,0,0
VD 19 POSITION 13,2:? "MUZYKANT"
GM 20 POSITION 13,3:? "========"
MT 200 POSITION 0,20:? BL$
PH 205 POSITION 2,5:? "DOSTEPNE P
   OLECENIA" ;
UL 210 ? "  Tx      TEMPO x=(1-3)"
WG 220 ? "  Lx      DLUGOSC x=(1-4
   )"
YB 230 ? "  Ox      OKTAWE x=(3-6)
   "
UW 240 ? "  A-G     NUTA"
OZ 250 ? "  #       KRZYZYK"
NG 260 ? "  -       BEMOL"
MF 270 ? "  ,       PAUZA L1"
BD 280 ? "  ;       1/12 S PRZERWA
   "
TH 290 ? :?
PO 300 POKE 752,0:? "WPROWADZ NUT
   Y I POLECENIA"
RA 310 INPUT PLAY$:PLAY$(LEN(PLAY
   $)+1)=",":? :IF PLAY$="." THEN
    END
NK 320 TRAP 320:? "GLOSNOSC",:INP
   UT GLO:TRAP 40000
AJ 330 X=USR(ADR(MUSIC$),ADR(PLAY
   $),ADR(NT$),ADR(TIME$),GLO)
YF 335 POSITION 0,18:? BL$:POSITI
   ON 0,17:? BL$
SB 340 POKE 752,1:POSITION 2,20:?
    "INNY DZWIEK";:INPUT PLAY$:IF
    LEN(PLAY$)=0 THEN 360
TP 350 IF PLAY$(1,1)="T" THEN 200
KU 360 GRAPHICS 0:END
```

liczbą między 3 a 6. Środkową oktawą może być 03.

T*n* określa tempo, *n* jest między 1 a 3. Nie jest to aktualne tempo, lecz liczba, która odpowiada innej znalezionej w tablicy wcześniej przechowywanych wartości tempa (patrz poniższa tablica).

L*n* określa, jak długo brzmi nuta, *n* jest między 1 a 4. L1 reprezentuje ósemkę, L2 ćwierćnutę, L3 półnutę, a L4 nutę.

Poniższa tablica pokazuje aktualne czasy trwania nut podane w jednostkach równych 1/50s (0.02s) w zależności od **L*n*** i **T*n***.

	L1	L2	L3	L4
T1	2	4	8	16
T2	5	10	20	40
T3	15	30	60	120

Przecinek [,] wstawia przerwę lub pauzę do muzyki powodując, że „Muzykant" odczekuje najkrótszy czas dopuszczalny w tym tempie — dwie jednostki w T1, pięć w T2 i 15 w T3.

Średnik [;] wstawia opóźnienie jednej dziesiątej sekundy między nutami — wystarczająco długie dla rozdzielenia nut, podobne do zwolnienia klawisza fortepianu.

Kropka [.] kończy działanie „Muzykanta" i zwraca sterowanie do programu w BASIC-u.

„Muzykant" ignoruje wszystkie inne znaki, lecz zakłada, że jakiekolwiek wartości przez nas podane są poprawne i mieszczą się w podanych granicach. Jeśli popełnimy błąd, możemy usłyszeć fałszywe dźwięki.

Program nr 5 przeznaczony jest do utworzenia procedury „Muzykant", zawierającej niezbędne stałe i zmienne, i przechowania jej na taśmie magnetycznej jako linii programu BASIC-u o numerach 10 do 16. W tym celu należy wprowadzić ten program do pamięci komputera, najlepiej za pomocą Edytora BASIC-u (patrz „IKS" nr 4) i przechować jego kopię w pamięci zewnętrznej. Po komendzie RUN utworzy on linie procedury „Muzykant" i przechowa je na taśmie magnetycznej.

Program nr 6 jest krótkim programem demonstrującym wykorzystanie procedury „Muzykant". Można go uruchomić po uprzednim wprowadzeniu do komputera, utworzeniu kopii w pamięci zewnętrznej i dołączeniu procedury „Muzykant" (za pomocą ENTER). Jak już wspomnieliśmy, w poprzednim odcinku, fala tworzona przez generatory dźwięku Atari ma kształt prostokątny. Oznacza to, że głośność prawie natychmiast osiąga wartość maksymalną, która trwa przez pół okresu, a następnie szybko spada do zera. Dla takiego impulsu charakterystyczne są: wartość maksymalna i czas trwania. Taki kształt fali dźwiękowej można uzyskać tylko za pomocą urządzeń elektronicznych. Dźwięki wytwarzane przez tradycyjne instrumenty muzyczne mają impulsy o innym kształcie. Można w nich wyróżnić trzy fazy:

— nabrzmiewanie, czyli narastanie dźwięku;

— trwanie, czyli faza ustalona, nazywana również podtrzymywaniem, w której dźwięk nie ulega zmianom lub zmienia się nieznacznie;

— wybrzmiewanie, czyli opadanie dźwięku, w której następuje spadek natężenia dźwięku aż do całkowitego jego zaniku.

W odróżnieniu od dźwięków generowanych przez komputer, dźwięki instrumentów mogą ➡

```
FK 1 REM ****************
BP 2 REM * Program nr 7 *
FM 3 REM ****************
NJ 4 REM
EF 10 POKE 731,0
RH 100 DATA 104,104,104,141,0,210
   ,104,104,133,204,104,104,133,2
   05,104,104,133,206
MF 110 DATA 169,160,141,1,210,166
   ,204,32,65,6,24,105,1,201,176,
   208,241,169,14,166,205,32,65,6
   ,56,233,1,208,246
CZ 120 DATA 169,175,141,1,210,166
   ,206,32,65,6,56,233,1,201,159,
   208,241,96,160,76,136,208,253,
   202,208,248,96
AT 130 FOR I=1536 TO 1609:READ A
   POKE I,A:NEXT I
NO 140 GRAPHICS 8:SETCOLOR 2,0,0
JA 150 Z88=PEEK(88):Z89=PEEK(89)
   N=100:T=100:O=100
CX 155 PK=0
BM 160 POKE 88,PEEK(660):POKE 89,
   PEEK(661):POKE 87,0:POKE 752,1
PK 170 POSITION 9,0:? #6,"Narasta
   nie ";N*5;" ms.          "
CB 180 POSITION 9,1:? #6;"Trwanie
      ";T*5;" ms.          "
WR 190 POSITION 9,2:? #6,"Opadani
   e    ";O*5;" ms.          "
HS 210 S=STICK(0)
FA 212 NS=N:OS=O:TS=T
KU 215 IF S=7 OR S=11 THEN GOSUB
   1000
QS 220 IF S=14 THEN PK=PK-1*(PK>0
   )
OR 230 IF S=13 THEN PK=PK+1*(PK<2
   )
OO 240 FOR I=0 TO 2:POSITION 5,I
   ? #6;CHR$(31+1*(I<>PK))
GC 250 NEXT I
ZY 260 POKE 88,Z88:POKE 89,Z89:PO
   KE 87,8
XY 270 COLOR 1
XL 280 PLOT 32,100:DRAWTO 32+N/3,
   0:DRAWTO 32+T/3+N/3,0:DRAWTO 3
   2+T/3+N/3+O/3,100
QO 290 ON (S<>7 AND S<>11) GOTO 3
   10
YG 295 COLOR 0
JV 300 PLOT 32,100:DRAWTO 32+NS/3
   ,0:DRAWTO 32+TS/3+NS/3,0:DRAWT
   O 32+TS/3+NS/3+OS/3,100
DY 310 IF PEEK(764)=255 THEN 160
EI 320 A=USR(1536,120,N,T,O)
WP 330 POKE 764,255
OE 340 GOTO 160
QO 1000 ON PK+1 GOTO 1100,1200,13
   00
AF 1020 RETURN
SQ 1100 X=N:GOSUB 2000:N=X
AE 1110 RETURN
ZK 1200 X=T:GOSUB 2000:T=X
AG 1210 RETURN
TX 1300 X=O:GOSUB 2000:O=X
AI 1310 RETURN
KT 2000 X=X+(1*(S=7)-1*(S=11))*(1
   +10*( NOT STRIG(0)))
YC 2010 IF X<1 THEN X=1
IM 2020 IF X>255 THEN X=255
AJ 2030 RETURN
```

```
?K 1 REM ****************
?J 2 REM * Program nr 8 *
FM 3 REM ****************
NJ 4 REM
ES 10 POKE 731,1
DO 16 GRAPHICS 0:? "Wybierz rodza
   j dzwieku:":? :? "1 STRZAL"
BG 18 ? "2 WYBUCH":? "3 SYRENA":?
    "4 KARETKA POGOTOWIA"
LA 20 ? "5 BOMBA":? "6 PILOWANIE
   DRZEWA":? "7 URUCHAMIANIE SAMO
   CHODU":? "8 MORZE"
WN 25 ? "9 PTAKI":? "0 MLOT PNEUM
   ATYCZNY"
SZ 30 POKE 752,0:POKE 77,0:POSITI
   ON 0,15:? "Co wybierasz", INPU
   T A
MD 35 POKE 752,1:? :POKE 77,128
HB 40 ON A GOSUB 100,300,500,700,
   800,1000,1200,1400,1600
OX 45 IF A=0 THEN GOSUB 1700
SE 50 GOTO 30
DA 100 REM STRZAL
YH 110 DLG=3
WK 120 DZW=20:GOSUB 160
EW 130 SOUND 1,0,0,0:SOUND 2,0,0,
   0
RF 140 SOUND 0,0,0,0:RETURN
CF 160 SOUND 2,75,8,15
OL 170 PRZ=0.62+DLG/100
CW 180 G1=15:G2=15:G3=15
WP 190 SOUND 0,DZW,8,G1
OE 200 SOUND 1,DZW+20,8,G2
TN 210 SOUND 2,DZW+50,8,G3
VH 220 G1=G1*PRZ
CM 230 G2=G2*(PRZ+0.05)
FI 240 G3=G3*(PRZ+0.08)
OD 250 IF G3>1 THEN 190
RK 260 SOUND 0,0,0,0:RETURN
WE 300 REM WYBUCH
ZT 310 DLG=7
YX 320 DZW=50:GOSUB 360
EY 330 SOUND 1,0,0,0:SOUND 2,0,0,
   0
FH 340 SOUND 0,0,0,0:RETURN
CH 360 SOUND 2,75,8,15
HZ 370 PRZ=0.8+DLG/100
CY 380 G1=15:G2=15:G3=15
WT 390 SOUND 0,DZW,8,G1
UG 400 SOUND 1,DZW+20,8,G2
TP 410 SOUND 2,DZW+50,8,G3
VJ 420 G1=G1*PRZ
CO 430 G2=G2*(PRZ+0.05)
FK 440 G3=G3*(PRZ+0.08)
PP 450 IF G3>1 THEN 390
RM 460 SOUND 0,0,0,0:RETURN
VZ 500 REM SYRENA
QF 510 DLG=10
JX 520 DOL=50:GOR=35:KROK=-0.5
YA 530 FOR CZAS=1 TO DLG

XD 540 FOR DZW=DOL TO GOR STEP KR
   OK
QZ 550 SOUND 0,DZW,10,14
SJ 560 FOR CZE=1 TO 5:NEXT CZE
ER 570 NEXT DZW
PF 580 XX=DOL:DOL=GOR:GOR=XX:KROK
   =-KROK
JU 590 NEXT CZAS
RC 600 SOUND 0,0,0,0:RETURN
MV 700 REM KARETKA POGOTOWIA
ZF 710 DLG=5
GV 720 DOL=57:GOR=45:DZW=GOR
ZI 730 FOR CZAS=0 TO DLG*2
QZ 740 SOUND 0,DZW,10,14
GL 750 FOR CZE=1 TO 100:NEXT CZE
BZ 760 DZW=DOL:DOL=GOR:GOR=DZW
JS 770 NEXT CZAS
RT 780 SOUND 0,0,0,0:RETURN

TJ 800 REM BOMBA
ZG 810 DLG=5
VV 820 G1=4:FOR DZW=30 TO 75
FL 830 SOUND 0,DZW,10,G1
RG 840 SOUND 1,DZW+3,10,G1*0.7
BS 850 FOR CZE=1 TO DLG*3:NEXT CZ
   E
MC 860 G1=G1*1.03:NEXT DZW
YJ 870 SOUND 2,35,8,12
DD 880 G1=15:G2=15:G3=15
QX 890 DZW=DLG+5:PRZ=0.8+DLG/100
WH 900 SOUND 0,DZW,8,G1
QN 910 SOUND 1,DZW+20,8,G2
TW 920 SOUND 2,DZW+50,8,G3
VQ 930 G1=G1*PRZ
CV 940 G2=G2*(PRZ+0.05)
FR 950 G3=G3*(PRZ+0.08)
NL 960 IF G3>1 THEN 900
JI 970 SOUND 0,0,0,0:SOUND 1,0,0,
   0:SOUND 2,0,0,0:RETURN
YY 1000 REM PILOWANIE DRZEWA
VL 1010 DLG=8
XO 1020 FOR CZAS=1 TO DLG
GU 1030 KR=16:GL=12:GOSUB 1060
HB 1040 KR=18:GL=8:GOSUB 1060
CT 1050 NEXT CZAS:RETURN
OD 1060 FOR DZW=KR+5 TO KR STEP -
   1
LZ 1070 GOSUB 1130:NEXT DZW
FU 1080 FOR DZW=KR TO KR+5
MT 1090 GOSUB 1140:NEXT DZW
TV 1100 SOUND 0,0,0,0:SOUND 1,0,0
   ,0
RU 1110 FOR CZE=1 TO 25:NEXT CZE
AH 1120 RETURN
LN 1130 SOUND 0,DZW,4,GL
GF 1140 SOUND 1,DZW,8,GL*0.7
UW 1150 FOR CZE=1 TO 3:NEXT CZE:R
   ETURN
ZB 1200 REM URUCHAMIANIE SAMOCHOD
   U
WY 1210 FOR CZS=1 TO 3

VK 1220 FOR CYKL=1 TO 5
BL 1230 SOUND 0,200-CYKL*20,0,14-
   CYKL
BF 1240 FOR W=1 TO 25:NEXT W:NEXT
    CYKL
XH 1250 FOR CYKL=1 TO 7
OV 1260 SOUND 0,CYKL*20,0,14-CYKL
BO 1270 FOR W=1 TO 25:NEXT W:NEXT
    CYKL
BA 1280 SOUND 0,0,0,0:FOR W=1 TO
   100:NEXT W
IH 1290 NEXT CZS
JY 1300 SOUND 0,100,4,8
TH 1310 FOR W=1 TO 2000:NEXT W
JE 1320 FOR W=200 TO 50 STEP -1

BT 1330 SOUND 0,W,2,10:FOR WA=1 T
   O 10:NEXT WA
XK 1340 NEXT W:SOUND 0,50,2,8
DG 1350 FOR WA=1 TO 2000:NEXT WA
XY 1360 SOUND 0,0,0,0:RETURN
JY 1400 DLG=5:REM MORZE
JY 1410 FOR L=1 TO DLG
IO 1420 FOR J=0 TO 45
DF 1430 SOUND 0,J,8,4
MH 1440 FOR K=1 TO 20+RND(0)*10:N
   EXT K
FX 1450 NEXT J
YR 1460 FOR J=45 TO 0 STEP -1
DR 1470 SOUND 0,J,8,4
OS 1480 FOR K=1 TO 50+RND(0)*10:N
   EXT K

FK 1490 NEXT J:FOR K=1 TO 300+RND
   (0)*300:NEXT K
IW 1500 NEXT L:SOUND 0,0,0,0
AM 1510 RETURN
KN 1600 REM PTAKI
NN 1610 DLG=10
GO 1620 FOR J=1 TO DLG*5
FE 1630 FOR K=3 TO 10
```

Prezentujemy krótki, ale ciekawy program graficzny, który można wykorzystać przy tworzeniu stron tytułowych innych, większych programów. Program został napisany na komputerze ZX Spectrum, ale może być po niewielkich przeróbkach uruchomiony na dowolnym komputerze.

```
10 LET a=INT (RND*4)+3
20 INK a
30 BORDER 0: PAPER 0: CLS
40 FOR f=0 TO 255 STEP 5
50 PLOT 0,10: DRAW f,165
60 NEXT f
70 FOR f=0 TO 255 STEP 5
80 PLOT 255,175: DRAW -f,-165
90 NEXT f
```

Sekcja NKP Informatyki przy SPR WSOWRiArt. w Toruniu

```
LS 1640 SOUND 0,K,10,8
YM 1650 NEXT K NEXT J
LE 1660 SOUND 0,0,0,0 RETURN
DS 1700 REM DLOT
UV 1710 DLG=5
JN 1720 FOR J=1 TO DLG
BR 1730 SOUND 0,130+RND(0)*2,2,15
JF 1740 FOR K=1 TO 140+RND(0)*300
      NEXT K
IQ 1745 SOUND 0,0,0,0
HE 1750 FOR K=1 TO 140+RND(0)*100
      NEXT K
GG 1760 NEXT J
BI 1770 RETURN
```

nie mieć fazy ustalonej. Dźwięki strunowych instrumentów szarpanych i młoteczkowych mają krótkie fazy nabrzmiewania i dość długie fazy wybrzmiewania. A akordeon ma długie nabrzmiewanie i krótkie wybrzmiewanie.

Czas trwania nabrzmiewania zależy od sposobu pobudzenia i elementu drgającego. Najkrótszy czas nabrzmiewania (poniżej 20 ms mają instrumenty strunowe szarpane oraz dęte blaszane. Instrumenty dęte stroikowe nabrzmiewają w ciągu 35—70 ms, a smyczkowe w ciągu 40—120 ms. Czas nabrzmiewania fletów wynosi 100—500 ms, a piszczałek organowych, w najniższych rejestrach — 1 sekundę i dłużej.

Czas trwania wybrzmiewania bywa znacznie dłuższy. Dla instrumentów dętych drewnianych wynosi około 100 ms, dla blaszanych — 200 ms, a dla smyczkowych aż 2—4 sekundy.

Z tego krótkiego przeglądu wynika, że w celu naśladowania dźwięków wytwarzanych przez instrumenty muzyczne, konieczne jest dobranie odpowiednich czasów nabrzmiewania, trwania i wybrzmiewania dźwięku. Czynność tę nazywamy kształtowaniem obwiedni dźwięku. Niektóre komputery, np. Amstrad CPC 464 lub CPC 6128 mają do tego specjalne instrukcje języka BASIC. Dla Atari musimy to robić sami (albo w języku BASIC, lecz trwa to zbyt długo i nie daje pożądanych efektów, albo w języku maszynowym, jak demonstrujemy w programie nr 7). Jest on przeznaczony do prezentowania różnic w brzmieniu dźwięków o różnym kształcie obwiedni. Aby uniknąć nadmiernego rozbudowania programu oraz lepiej pokazać różnice, generowany jest jeden dźwięk o wysokości środkowego C. Za pomocą ruchów manipulatora w przód i w tył wybieramy odpowiednią fazę, a ruchy w lewo lub w prawo zmieniają czasy trwania wybranej fazy. W celu przyspieszenia zmiany długości fazy należy nacisnąć czerwony klawisz manipulatora. Można sprawdzić brzmienie dźwięków o różnym kształcie obwiedni, wykorzystując podane wcześniej przybliżone czasy poszczególnych faz.

Uznaliśmy, że w artykule traktującym o dźwięku nie może zabraknąć gotowego programu dla tych czytelników, którzy nie mają ochoty lub czasu na eksperymentowanie. W tym celu przygotowaliśmy program nr 8, który demonstruje dziesięć interesujących efektów dźwiękowych. Pewne z nich są dosyć skomplikowane i zadziwiająco realistyczne. Uzyskaliśmy je przez szybkie zmiany częstotliwości, zniekształceń i głośności w instrukcjach SOUND.

Tomasz MROWIEC
Ludwik PIELA

PROGRAM ATARI 33

CYKLOTRON

[illegible]

JAN GOLLA

Oto lista "poprawek" umożliwiających uzyskanie nieśmiertelności w niektórych grach na komputer ZX Spectrum. Poniższa lista dotyczy starszych gier, które są na tyle atrakcyjne, że jeszcze dziś mogą wielu zainteresować. Niektóre pozycje z tej listy były już prawdopodobnie publikowane, chcąc jednak przedstawić dorobek naszej sekcji, podajemy wszystkie, które udało nam się złamać.

Sekcja NKP Informatyki
przy SPR WSOWRiArt. w Toruniu

```
ALCHEMIST          POKE 47340,0
                   POKE 49745,195
ARCADIA            POKE 25776,0
ATIC ATAC          POKE 35519,0
BRUCE LEE          POKE 31795,0
COOKIE             POKE 28698,0
HERO               POKE 44322,182
JACK AND THE
    BEANSTALK      POKE 42404,255
JET PACK           POKE 25020,0
KOKOTONI W         POKE 43742,0
MANIC MINER        POKE 35136,0
MONTY MOLE         POKE 37512,0
                   POKE 47640,0
PSSST              POKE 24984,0
PSYTRON            POKE 28625,0
                   POKE 41099,32
PYRAMID            POKE 44685,0
TRANS AM           POKE 25446,0
DEFENDA            POKE 35730,52
HUNCHBACK          POKE 26888,0
MOON ALERT         POKE 39754,0
ZAXXON             POKE 48825,255
MR.WIMPY           POKE 33693,0
ALIEN 8            POKE 51736,0
CHUCKIE EGG        POKE 42508,3
THE WAY OF THE EXPLODING FIST
                   10 LOAD "" SCREEN$: LOAD "" CODE
                     POKE 44793,255 RANDOMIZE USR 39982
```

PROGRAM

ZX SPECTRUM

34

RUCHOMY TEKST

Ten program umożliwia przesuwanie w dolnej części ekranu tekstu o dowolnej długości. Szybkość przesuwania reguluje instrukcja PAUSE. Przerywanie przesuwania następuje po wciśnięciu spacji

```
  1 REM ruchomy tekst
 10 INPUT "podaj dowolny tekst
 "; LINE n$
 20 DIM k$(64*(LEN n$<32)+2*LEN
n$*(LEN n$>=32))
 30 IF LEN n$<32 THEN LET k$=n$
: LET k$(33 TO )=n$
 40 IF LEN n$>=32 THEN LET k$=n
$+n$
 50 FOR k=1 TO LEN k$/2
 60 PRINT #1;AT 0,0;k$(k TO k+3
1)
 70 IF INKEY$=" " THEN GO TO 10
 80 PAUSE 10
 90 NEXT k
100 GO TO 50
```

PROGRAM

ZX SPECTRUM

35

BASIC PRZYSPIESZA BASIC

Szybkość programu napisanego w Basicu jest piętą achillesową każdego komputera, a na pewno już ZX Spectrum. Znawcy doradzają, aby pisać jak najwięcej instrukcji w jednej linii, a podprogramy trwające najdłużej wprowadzać na początek programu. Nie ulega wątpliwości, że pierwsze linie pojawiają się szybciej niż ostatnie (a ściślej mówiąc czas ich odszukania jest znacznie krótszy). Kłopot dla programisty zaczyna się wtedy, kiedy tych ważnych i długich podprogramów jest kilka.

Mechanizm szukania linii jest następujący: program zaczyna się od komórki pamięci adresowanej przez zmienną systemową (PROG = 256 • PEEK 23636 + PEEK 23635). Od tego miejsca interpreter „przeskakuje z linii na linię", aż znajdzie szukaną. Odbywa się tak dla każdej instrukcji programu. Jest to konieczność związana z konwersacyjnym trybem obsługi ZX Spectrum: przecież użytkownik w każdej chwili może wprowadzić zmiany w programie.

Nasze „oszustwo" polega na tym, że zmieniamy adres początku programu. Po uruchomieniu procedury należy podać numer linii i wstawić wypisane POKE. Od tej chwili program zaczyna się od tej linii (nawet podczas listowania). Dlaczego konieczne jest zapamiętanie rzeczywistej wartości PROG: podajemy procedurze numer pierwszej linii programu. Tę wartość należy przywrócić, gdy chcemy mieć dostęp do całego programu.

Przeprowadziłem eksperyment licząc czas wykonywania najprostszej pętli. Trwało to 2,5 razy wolniej, jeśli poprzedzała ją setka innych linii, niż przy tej samej pętli na początku programu.

Zastosowanie tej procedury zapewni w obu wypadkach ten sam czas.

Działanie procedury polega na tym, że znajdzie ona w pamięci adres początku podanej linii (pod warunkiem że ta linia istnieje). Możemy w ten sposób dowolnie przenumerowywać linie: np. wprowadzić nr Ø — takiej linii nie można edytować. Zamieńmy naszą linię na zerową. Po wywołaniu procedury należy podać „1" (numer naszej linii). Otrzymamy adres 23755 (ten adres jest najczęściej początkiem programu). Wykonajmy POKE 23755 + 1, Ø. Od tej chwili edycja procedury jest niemożliwa.

Krzysztof POŹNIAK

```
  1 PRINT "procedura uzywa zmie
nnych "'"qqq1 do qqq5"'"n"
: LET qqq1=PEEK 23635+256*PEEK
23636: LET qqq2=PEEK 23627+256*P
EEK 23628: INPUT "wprowadz nr li
nii ";qqq4: LET qqq5=INT (qqq4/2
56): LET qqq4=qqq4-qqq5*256: LET
 qqq3=0: FOR n=qqq1 TO qqq2: LET
 qqq3=(PEEK n=qqq5 AND PEEK (n+1
)=qqq4)*n+qqq3: LET n=n+3+PEEK (
n+2)+PEEK (n+3)*256: NEXT n: PRI
NT "linia ";qqq4'"adres ";qqq3
'"wstaw POKE 23635,";qqq3-256*I
NT (qqq3/256)'"wstaw POKE 23636
,";INT (qqq3/256): STOP
```

ROZKŁAD GAUSSA W ASEMBLERZE

Proponowany program, choć na pierwszy rzut oka wydaje się bardzo długi, jest znacznie szybszy od jego odpowiednika w Basicu.

Umożliwia obliczanie dowolnego układu równań liniowych. Korzysta z „kalkulatora" Spectrum, dzięki czemu zachowuje pełną kompatybilność z pozostałą częścią programu oraz zapewnia identyczną dokładność numeryczną. Program jest całkowicie relokowalny i całkowicie zabezpiecza przed błędami. Napotkany błąd sygnalizowany jest odpowiednim komunikatem i awaryjnym przerwaniem programu. Procedurę uruchamia się przez **FN** (funkcja użytkownika). Przesyła się w niej trzy parametry (FN znajduje się w linii 2000). Tablica tekstowa n$ zawiera kod maszynowy procedury. Wyjaśnienia na ten temat zamieszczone są na końcu artykułu. Tekst „n()" określa nazwę tablicy numerycznej przeznaczonej do rozkładu (nawiasy po literze są konieczne). Zmienna sterująca „s" może zmieniać się od 0 do 255 włącznie, czyli można ją traktować jako zestaw ośmiu niezależnych bitów, wprowadzanych np. przez instrukcję BIN. Bity te mają następujące znaczenie:

bit 7 = 0 — ustawianie elementu głównego wg wektorów przemieszczeń podanych z poziomu Basica.
= 1 — automatyczne ustawianie elementu głównego.
bit 6 = 0 — kolumny nie są przestawiane.
= 1 — przestawianie kolumn.
bit 5 = 0, = 1 } jak bit 6, ale dotyczy wierszy.
bit 4 = 0 — macierz L nie jest obliczana.
= 1 — macierz L jest obliczana.
bit 3 = 0, = 1 } jak bit 4, ale dotyczy macierzy U.
bit 2 = 0, = 1 } jak bit 4, ale dotyczy macierzy B.
bit 1 = 0, = 1 } jak bit 4, ale dotyczy macierzy X.
bit 0 = 0 — liczenie odbywa się na stosie kalkulatora, zostają zachowane macierze A i B.
= 1 — liczenie jawne, odbywa się na przesłanej tablicy numerycznej.

Struktura tablicy przesyłanej do procedury jest następująca:

$a_{11}, \ldots, a_{1n}$	b_1	x_1	k_1	w_1
[A]	.	.	.	.
$a_{n1}, \ldots, a_{nn}$	b_n	x_n	k_n	w_n
	[B]	[X]	[Y_w]	[X_w]

Ukrywa ona taki oto układ równań:

$$\begin{matrix} a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + \ldots + a_{nn}x_n = b_n \end{matrix} \quad ; [A] \cdot [X] = [B]$$

gdzie: Y_w — wektor przestawień kolumn
X_w — wektor przestawień wierszy
A — macierz współczynników
B — wektor wyrazów wolnych
X — wektor niewiadomych, wektor szukanych

Podczas wprowadzania danych do tablicy $x_1,\ldots,x_n$ mogą przyjąć dowolne wartości, podobnie $k_1,\ldots, k_n$ i $w_1,\ldots, w_n$, ale pod warunkiem, że bit 7 = 1. Gdy bit 7 = 0, muszą one przyjąć ściśle określone wartości, o czym będzie mowa w dalszej części artykułu.

Osoby bliżej zainteresowane teorią odsyłam do pozycji (1, 2, 3).

Macierz A rozkłada się na macierz górnotrójkątną U i dolnotrójkątną L, zostaje odpowiednio przekształcony wektor B, a następnie za pomocą utworzonej macierzy U i przekształconego wektora B obliczony zostaje wektor niewiadomych X. Bity nr 4, 3, 2, 1, decydują o obliczaniu odpowiednio macierzy L, U, B, X. Bit 0 decyduje o zachowaniu macierzy A i wektora B, tzn, że można otrzymać rozwiązanie, zachowując te dwie macierze (są one dodatkowo zapamiętywane, co niestety zajmuje pewną część pamięci).

I ważne wyjaśnienie dotyczące bitów 7, 6, 5. Okazuje się, że największą dokładność w obliczeniach uzyskuje się, gdy tzw. element główny ma wartość bezwzględnie maksymalną. Dokładne wyjaśnienie tego problemu zawierają pozycje (2, 3). Element główny ustawiany jest przez odpowiednie przestawienie kolumn i wierszy. Wyszukiwanie elementu głównego zapewnia bit 7 + 1. O tym, czy mają być przestawiane kolumny, wiersze, czy kolumny i wiersze jednocześnie decydują odpowiednio bity 6 i 5. Wektory Y_w i X_w pamiętają te przestawienia i jeśli bit 7 ustawimy na wartość 0, to przestawienia będą się odbywać wg tych wektorów. Dzięki temu w czasie ciągu obliczeń iteracyjnych, kiedy wartości zmieniają się w niewielkim stopniu, przeszukiwanie macierzy A odbywa się tylko podczas pierwszego rozwiązania, co znakomicie przyspiesza obliczenia.

Proponuję przeprowadzić kilka testów. Utwórzmy tablicę 10 stopnia (n = 10). Lepiej wykonać podstawienie automatyczne macierzy — podstawiają się wtedy wartości generowane przez RND. Podajmy s + 11101110. W tym wypadku element główny będzie automatycznie ustawiany przez przestawianie kolumn i wierszy. Zostaną obliczone macierze X przy zachowaniu macierzy A i B. Naciśnięcie klawisza „3", spowoduje wykonanie procedury, a odczyt wyników nastąpi przez naciśnięcie klawisza „4". Warto zwrócić uwagę na czas i na błędy obliczeniowe. Po naciśnięciu dowolnego klawisza pojawi się ekran główny. Wektory X_w, Y_w pamiętają przestawienia. Podstawmy s = 01101110. Wynik będzie identyczny, ale czas krótszy. Przestawienia były narzucone i dzięki temu nie trzeba było wielokrotnie przeszukiwać macierzy A. Podstawmy s = 1000110. Czas będzie krótki, ale dokładność powinna być mniejsza. Jednak dobór elementu głównego ma znaczenie. Nie obliczaliśmy macierzy L ale i ona się przydaje, zwłaszcza wtedy, gdy rozwiązywanie przeprowadzamy wielokrotnie i zmieniamy tylko wektor wyrazów wolnych B. Na początku należy rozłożyć macierz A na macierz L i U (s = 11111001). Rozłożenie musi być jawne (bit 0 = 1). O zamianie kolumn i wierszy decydujemy sami (w tym wypadku są przestawiane kolumny i wiersze), ale bity 6 i 5 nie mogą być potem zmieniane. Podstawmy s = 01100111. Teraz możemy wielokrotnie wprowadzać wektor B i wywoływać proces obliczeniowy. Sam proces rozkładu wektora B i obliczania wektora X trwa krótko, co można stwierdzić oglądając wyniki.

Przykładowa tabela czasów wykonania:

1. s = BIN11111111 – obliczenie L,U,B,X z pełnym przestawieniem;
2. s = BIN10011111 – obliczenie L,U,B,X bez przestawiania;

czas w sekundach:

n=	1	3	5	7	10	15	20
1.	0,12	0,24	0,68	1,68	4,24	13,28	29,68
2.	0,12	0,2	0,52	1,08	2,72	8,4	18,6

W razie stwierdzenia błędu program uruchamia procedurę obsługi błędu Spectrum i emituje odpowiedni komunikat:

3 Subscript wrong — podczas zewnętrznego sterowania przestawieniami (bit 7 = 0) numerem wiersza lub kolumny jest liczba rzeczywista lub naturalna nie należąca do przedziału 1,...,n

4 Out of memory — gdy zabraknie miejsca w pamięci RAM

A Invalid argument — podczas zewnętrznego sterowania przestawieniami, gdy wektor Y_w lub X_w ma powtarzające się numery

Q Parametr error — wywołany przez interpreter Spectrum, gdy przesyłane parametry nie zgadzają się ilościowo, składniowo, lub nie istnieją w obszarze zmiennych wywoływany przez procedurę, gdy nie istnieje tablica numeryczna o podanej nazwie, gdy zabraknie nawiasu za literą bądź zostanie przesłana nazwa wieloliterowa, gdy rozmiar tablicy nie wyniesie (n,n + 4), gdy zmienna sterująca „s" nie zawiera się w przedziale (0—255)

Funkcja użytkownika FN po powrocie z procedury przyjmuje dwie wartości:

0 — gdy wszystkie narzucone przez zmienną sterującą rozkłady i przekształcenia zostały wykonane

1 — gdy element główny ma wartość zero. Oznacza to, że należy wprowadzić zamianę kolumn i wierszy, gdyż na głównej przekątnej znajdują się zera, a jeśli zamiana automatyczna kolumn i wierszy jest uruchomiona, świadczy to o niejednoznaczności układu równań.

Kod maszynowy jest trzymany w tablicy tekstowej nS. Takie ulokowanie procedury ma wiele zalet:

1) nie trzeba zmieniać rantropu i rezerwować miejsca dla kodu maszynowego
2) można w każdym momencie wykasować procedurę (np. DIM n$ (0)
3) można zapisać procedurę przez SAVE „nazwa" DATA n$ (), odczytać przez LOAD „nazwa" DATA n$ (), zweryfikować przez VERYFI „nazwa" DATA n$ (), czyli traktować ją jak zwykłą tablicę tekstową
4) jeśli wczytujemy program przez LOAD, to procedura wczytuje się jednocześnie z programem i pozostałymi zmiennymi.
5) w jednej tablicy tekstowej można przechowywać wiele procedur, a do FN przesyłać wycinek tablicy z daną procedurą. Proponuję zrobić eksperyment, ale najpierw lepiej utrwalić program na taśmie.

Utwórzmy tablicę DIM m$ (4774)
podstawmy LET m$ () = n$ () + n$ ()
teraz w linii 2 000 wymieńmy n$ na m$ (2388 TO). W ten sposób można przepisywać i porządkować procedury wg własnych potrzeb, tak jak zwykłe tablice tekstowe. Można nawet wpisać m$ (2388), co i tak uruchomi procedurę. Odnaleziony zostanie adres tego elementu tablicy, czyli adres początku procedury. Podczas tych operacji należy zachować ostrożność, aby nie doprowadzić do uruchomienia programu od przypadkowego bajtu. Najczęściej w takich wypadkach system się resetuje.

Krzysztof POŹNIAK

Bibliografia

1) K. Poźniak: „Rozwiązywanie układu równań liniowych" „Bajtek" 7/1987 str. 20—21
2) A. Kossur, P. Perkowski: „Obliczeniowe aspekty projektowania układów elektronicznych" WNT 1979 str. 44—49
3) J. Ogrodzki, E. Śliwa: „Analiza i projektowanie komputerowe układów elektronicznych" Wyd. Polit. Warsz. 1985 str. 47—53, 56—65


```
   1 REM ***********************
   2 REM
   3 REM    Krzysztof Pozniak
   4 REM  Klub Mikrokomputerowy
   5 REM     MOBBITE     © 1987
   6 REM
   7 REM ***********************
   8 DEF FN f(n$,a$,s)=USR (PEEK
 (PEEK 23563+256*PEEK 23564+4)+2
56*PEEK (PEEK 23563+256*PEEK 235
64+5))
  10 INK 7. PAPER 0. BORDER 0: C
LS : PRINT AT 10,8, FLASH 1;"pro
sze czekac": LET l=1. LET d=8000
: RESTORE d; DIM n$(2387)
  11 LET k1=0: LET k2=0: LET s=1
: FOR y=1 TO 20. READ x: LET n$(
l)=CHR$ x: LET k1=k1+x: LET k2=k
2+s*x: LET s=-s: IF l=2387 THEN
GO TO 13
  12 LET l=l+1: NEXT y
  13 READ k01: READ k02. IF k01<
>k1 OR k02<>k2 THEN BEEP 0.2,20.
 PRINT "blad w linii ";d: STOP
  14 LET d=d+10. IF l<2387 THEN
GO TO 11
  20 LET dane=0. LET n=1. LET s=
255. LET time=0. LET n=1: DIM n(
n,n+4): DIM s$(8): LET s$="11111
111"
 100 CLS : PRINT AT 0,0;"rozkazy
"
 110 PRINT AT 2,0,"1 - nowa tabl
ica"
 120 PRINT "2 - zmienna sterujac
a"
 130 PRINT "3 - wykonanie"
 140 PRINT "4 - wydruk"
 200 PRINT AT 10,0,"n=",n
 210 PRINT AT 12,0,"7-",s$(1),"=
>zamiana ","sterowana" AND NOT V
AL s$(1),"automatyczna" AND VAL
s$(1)
 211 PRINT TAB 0,"6-",s$(2);"=>k
olumny ","niezamieniane" AND NOT
 VAL s$(2);"zamieniane" AND VAL
s$(2)
 212 PRINT TAB 0;"5-",s$(3);"=>w
iersze ","niezamieniane" AND NOT
 VAL s$(3);"zamieniane" AND VAL
s$(3)
 213 PRINT TAB 0;"4-",s$(4);"=>"
"nie" AND NOT VAL s$(4);"" AND
VAL s$(4);"obliczanie macierzy L
"
 214 PRINT TAB 0,"3-",s$(5),"=>"
;"nie" AND NOT VAL s$(5);"" AND
VAL s$(5);"obliczanie macierzy U
"
 215 PRINT TAB 0,"2-",s$(6);"=>"
;"nie" AND NOT VAL s$(6);"" AND
VAL s$(6),"obliczanie macierzy E
"
 216 PRINT TAB 0,"1-",s$(7),"=>"
;"nie" AND NOT VAL s$(7);"" AND
VAL s$(7),"obliczanie macierzy X
"
 217 PRINT TAB 0;"0-";s$(8);"=>"
;"zachowanie" AND NOT VAL s$(8);
"niszczenie" AND VAL s$(8);" mac
ierzy A|B"
 300 PRINT AT 0,0; PRINT #1;AT 0
,0; INVERSE 1; BRIGHT 1;"podaj
rozkaz"; INVERSE 0, BRIGHT 0;"
"
 310 IF INKEY$="" THEN GO TO 310
 320 LET i$=INKEY$: IF i$<"1" OR
 i$>"4" THEN GO TO 310
 330 LET i$=INKEY$. IF i$="1" TH
EN GO SUB 1000: GO TO 100
 340 IF i$="2" THEN GO SUB 1100:
 GO TO 100
 350 IF i$="3" THEN PRINT #1;AT
0,0;"rozkaz 3 wykonywany": GO SU
B 2000: GO TO 300
 360 IF i$="4" THEN GO SUB 1200.
 PAUSE 0. GO TO 100
1000 INPUT "podaj wymiar tablicy
 n="; n: DIM n(n,n+4): PRINT #1;
AT 0,0,"dane : 0-wlasne"'"
 1-automatyczne"
1001 LET i$=INKEY$: IF i$<>"0" A
ND i$<>"1" THEN GO TO 1001
1005 LET dane=VAL i$
1010 IF dane THEN INPUT "": PRIN
T #1;AT 0,0;"dane podstawiane au
tomatycznie": FOR y=1 TO n: LET
m=0. FOR x=1 TO n. LET n(y,x)=10
0*RND: LET m=m+n(y,x)*x. NEXT x:
 LET n(y,x)=m. FOR k=1 TO 3: LET
 n(y,x+k)=100*RND. NEXT y: RETUR
N
1013 IF INKEY$<>"" THEN GO TO 10
13
1015 PRINT #1;AT 0,0;"0-pominiec
ie wektorow Xw,Yw"'"1-ustawianie
 wektorow Xw,Yw"
1016 LET i$=INKEY$ IF i$<>"0" A
ND i$<>"1" THEN GO TO 1016
1017 LET k=VAL i$
1020 FOR y=1 TO n  FOR x=1 TO n:
 INPUT "wspolczynnik n(";(y);","
;(x),")=",n(y,x) NEXT x: INPUT
"wspolczynnik b(";(y);")=";n(y,x
). IF k THEN INPUT "dana wektora
 Xw(",(y);")=";n(y,n+3)'"dana we
ktora Yw(",(y);")=";n(y,n+4)
1030 NEXT y. RETURN
1100 INPUT "76543210"' LINE s$
1110 FOR k=1 TO 8. IF s$(k)<>"1"
 AND s$(k)<>"0" THEN GO TO 1100
1120 NEXT k. LET s=VAL ("BIN "+s
$). RETURN
1200 IF i$="4" THEN CLS . PRINT
"n=";n'"czas=";time," s"
1210 FOR k=1 TO n. PRINT TAB 0,"
x",k,"=",n(k,n+2),: IF dane THEN
 PRINT "b=";n(k,n+2)-k
1211 NEXT k
1212 PRINT #1,AT 0,0,"czy dalej
listowac? tak-1"
1213 LET i$=INKEY$  IF i$="" THE
```

```
N GO TO 1213
1214 IF i$<>"1" THEN RETURN
1215 INPUT "" FOR y=1 TO n FOR
 x=1 TO n: PRINT TAB 0;"wspolczy
nnik n(";y;",";x;")=";n(y,x): NE
XT x: NEXT y
1216 PRINT : PRINT "wektor przes
tawien kolumn :": FOR y=1 TO n:
PRINT n(y,n+3);" "; NEXT y
1217 PRINT TAB 0,"wektor przesta
wien wierszy :": FOR y=1 TO n P
RINT n(y,n+1);" "; NEXT y
1220 PAUSE 0. RETURN
2000 PRINT AT 20,0,"
                     POKE 23672,0
 POKE 23673,0 POKE 23674,0 LET
 z=FN f(n$,"n()",1)
2100 LET x=256*PEEK 23673+PEEK 2
3672
2101 LET y=256*PEEK 23673+PEEK 2
3672
2110 LET time=(x+y-ABS (x-y))/50
2115 PRINT AT 20,0, FLASH 1;"ukl
ad rozwiazany" AND NOT z," STOP
-element glowny zerowy!" AND z
2120 RETURN
6000 DATA 221,42,11,92,221,229,4
2,93,92,221,126,20,221,182,21,22
1,182,23,221,182,2663,63
6010 DATA 24,32,58,229,221,126,1
3,50,93,92,221,126,14,50,94,92,2
6,178,40,193,2151,-196
6020 DATA 237,67,93,92,56,35,32,
33,221,225,221,70,22,229,221,226
,62,2,221,190,2554,218
6030 DATA 3,32,16,221,166,5,221,
182,7,32,10,221,126,4,198,4,221,
190,6,40,1967,45
6040 DATA 5,46,25,195,85,0,197,2
05,176,22,1,57,0,205,5,31,205,42
,161,92,1699,-91
6050 DATA 9,34,101,92,221,126,4,
1,6,0,221,9,221,229,221,225,221,
46,69,92,2190,472
6060 DATA 221,114,30,221,119,0,2
21,119,1,196,3,71,35,0,0,1,5,0
[illegible]
6070 DATA 253,221,117,16,221,116
,17,221,117,45,221,116,46,25,221
,117,14,221,116,15,2456,230
6080 DATA 221,117,43,221,116,44,
235,225,25,17,5,0,237,82,221,117
,10,221,116,11,2361,174
6090 DATA 221,117,26,221,116,27,
237,82,221,117,8,221,116,9,221,1
17,24,221,115,25,2463,146
6100 DATA 237,82,221,117,6,221,1
16,7,221,117,22,221,116,23,221,1
10,16,221,102,17,2414,142
6110 DATA 237,82,237,82,221,117,
31,221,116,32,237,82,221,117,33,
221,116,34,221,126,2784,556
6120 DATA 30,203,71,32,68,221,12
6,33,221,119,45,221,126,34,221,1
19,46,221,126,31,2314,-154
6130 DATA 221,119,43,221,126,32,
221,119,44,25,235,33,0,0,221,70,
0,25,16,253,2024,230
6140 DATA 229,193,205,5,31,237,9
1,101,92,225,213,221,126,0,221,7
0,31,221,70,32,2622,-4
6150 DATA 237,176,1,15,0,9,61,32
,241,237,83,101,92,225,221,117,2
,221,116,3,2190,-62
6160 DATA 221,117,18,221,116,19,
221,117,53,221,116,54,221,94,33,
221,86,34,25,221,2429,-209
6170 DATA 117,4,221,116,5,221,11
7,20,221,116,21,22,0,30,5,221,70
,6,33,0,1550,50
6180 DATA 0,25,16,253,221,117,55
,221,116,56,221,126,30,33,0,0,22
,0,221,94,1627,-23
6190 DATA 0,203,119,40,23,25,229
,42,101,92,221,117,49,221,116,50
,67,14,1,113,1943,9
6200 DATA 35,18,16,251,34,101,92
,225,203,111,40,23,25,229,42,101
,92,221,117,51,2031,-625
6210 DATA 221,116,52,67,14,1,113
,35,12,16,251,34,101,92,225,229,
42,101,92,221,2035,211
6220 DATA 117,12,221,116,13,17,5
,0,33,0,0,221,70,0,4,25,16,253,6
,2,1131,-161
6230 DATA 221,126,30,203,119,40,
17,203,111,40,13,62,1,221,190,0,
40 [illegible]
6240 DATA 0,193,[illegible],222,223,2
3,0,0,25,16,[illegible],229,193,2
[illegible]
[illegible] 221,126,30,220
[illegible],65,221 [illegible] 54,40,1,
[illegible]
[illegible],221,112,
[illegible],119,4 [illegible],12,39,221,1
[illegible],40 [illegible]
[illegible],0,0,17,5,0,25
,16 [illegible],221,116,56,22
[illegible]
6280 DATA [illegible],70,40,37,0,0,25,16
[illegible],116,42,221,1
[illegible]
6290 DATA 40,67,221,126,30,203,1
27,40,67,221,110,5,221,102,6,221
,126,0,229,220,2397,-37
6300 DATA 209,19,54,0,1,0,0,237,
176,221,94,14,221,86,15,225,25,6
1,32,234,1633,-279
6310 DATA 221,110,0,221,102,0,35
,35,62,1,221,70,0,17,5,0,116,25,
119,221,1601,165
6320 DATA 91,16,221,86,17,25,60,
16,240,24,116,46,2,195,65,0,221,
116,16,221,1605,327
6330 DATA 103,17,175,17,5,0,237,
62,229,221,110,8,221,102,9,221,7
0,0,1,6,185,2163,469
6340 DATA 35,182,35,35,182,35,1
[illegible],182,35,182
,45,46,16,[illegible],1900,-62
6350 DATA [illegible],205,6,2,221,126,0,2
21,113,[illegible],126,6,221,119,36,1
97,221,110,35,2270,-186
6360 DATA 221,102,36,35,35,221,7
0,0,7,12,221,94,12,221,86,13,12
6,15,16,213,1627,-31
6370 DATA 254,0,40,160,105,40,16
0,221,94,14,221,86,15,25,209,16,
245,24,9,24,2043,851
6380 DATA 211,24,66,46,9,195,85,
0,221,115,28,221,114,29,213,225,
19,54,0,221,2116,-144
6390 DATA 78,0,6,0,237,176,221,7
0,0,22,0,62,1,221,110,12,221,102
,13,94,1646,156
6400 DATA 35,229,221,110,28,221,
102,29,25,106,43,207,54,1,225,16
,238,221,110,26,2330,-174
6410 DATA 221,102,29,35,221,70,0
,1,126,190,40,168,35,16,250,193,22
1,126,10,221,116,2462,348
6420 DATA 35,221,126,11,221,119,
10,16,166,62,25,221,166,30,40,93
,221,126,6,254,4102,-114
6430 DATA 1,40,66,221,126,1,51,4
0,62,221,126,39,25,127,40,77,22
1,126,39,163,2065,-76
6440 DATA 40,67,221,94,12,221,86
,13,221,110,15,221,102,19,221,12
6,42,221,75,43,2176,-95
6450 DATA [illegible],44,225,197,221,
[illegible],193,225,
[illegible]
6460 DATA [illegible],221,94,41,22
[illegible],6,225,44
[illegible]
6470 DATA [illegible],30,0,24,0,24,1
[illegible],44,119,44,126,213,235,
[illegible],1,1489,543
6480 DATA 42,193,3,55,40,2,0,5,2
25,192,51,2,224,63,241,56,209,24
,206,42,1674,626
6490 DATA 101,92,35,35,221,70,40
,17,5,0,175,221,70,39,190,40,6,2
5,16,250,1658,64
6500 DATA 13,32,244,4,12,221,94,
14,221,86,15,221,126,30,203,119,
40,24,221,126,2066,312
6510 DATA 1,144,40,16,71,221,110
,24,221,102,26,35,35,126,220,25,
16,253,70,119,1665,-249
6520 DATA 225,112,221,126,30,203
,111,40,32,221,126,1,145,40,26,7
,221,110,66,221,2306,16
6530 DATA 102,27,35,35,126,229,2
4,5,24,132,24,105,24,105,25,16,2
53,46,119,235,1704,-195
6540 DATA 112,221,126,30,203,119
,40,93,230,221,40,65,221,110,24,2
21,102,25,35,35,2100,166
6550 DATA 126,6,0,221,110,49,221
,102,50,225,209,190,40,4,35,4,24
,246,79,175,2123,-335
6560 DATA 176,40,56,26,119,121,1
6,221,110,53,221,104,54,226,17,6
,0,25,16,253,1664,-266
6570 DATA 235,225,221,78,45,221,
70,46,221,126,0,245,167,6,6,78,2
6,119,121,18,2303,-21
6580 DATA 35,19,16,247,193,9,235
,9,235,241,61,32,234,24,6,24,147
,24,111,24,1026,620
6590 DATA 111,221,126,30,203,111
,40,104,221,110,26,221,162,27,35
,35,126,6,0,221,2076,-96
6600 DATA 110,51,221,102,52,229,
209,190,40,4,35,4,24,249,79,176,
176,40,73,26,2069,-51
6610 DATA 119,121,16,221,126,30,
230,24,40,64,197,221,110,18,221,
102,19,229,221,94,2425,177
6620 DATA 43,221,86,44,25,16,253
,235,225,221,78,66,221,70,65,12,
24,13,194,6,2101,313
6630 DATA 0,76,26,119,121,15,35,
19,16,247,193,13,32,240,175,176,
40,7,6,0,1561,-273
6640 DATA 12,197,71,24,232,193,2
4,6,24,141,24,56,24,36,221,126,3
0,203,67,40,1774,-676
6650 DATA 247,221,110,20,221,102
,21,229,221,94,43,221,86,44,25,1
6,253,245,225,6,2640,264
6660 DATA 6,78,26,119,121,15,35,
19,16,247,221,110,19,221,102,19,
175,6,[illegible],1743,-226
[illegible] 36,16,[illegible],32,0,1,1,0,0
01,24,193,24,70,221,126,1,221,11
0,[illegible],1626,444
6680 DATA 1,30,60,221,94,43,221,
86,44,221,110,15,221,102,19,25,2
[illegible],-65
6690 DATA 116,36,221,110,20,221,
102,21,[illegible],47,221,116,46,
221,94,[illegible],2276,54
6700 DATA 1,[illegible],70,30,221,1,203,
60,40,1,60,[illegible],66,40,10,221,131,
1,61,24,1566,-61
6710 DATA 4,44,122,24,84,221,119
,120,221,126,30,203,87,40,22,221,
110,47,[illegible],102,2116,44
6720 DATA 48,1,0,0,237,176,221,1
10,20,221,102,21,1,5,0,237,176,2
03,95,40,1023,-109
```

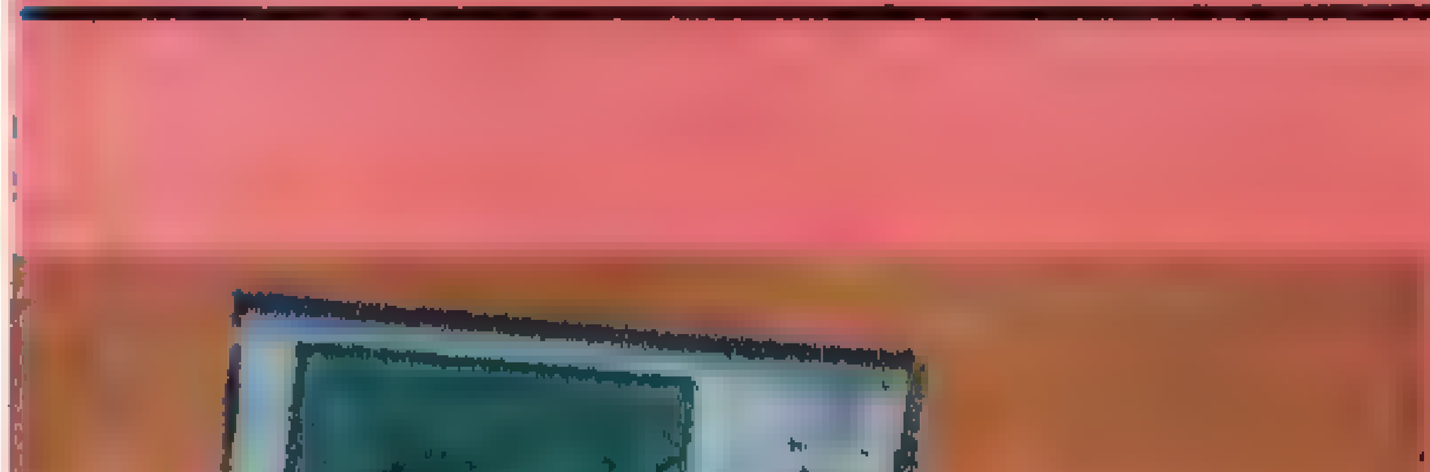
ZX-Spectrum

TEKST

```
   1 LET k=-1
   2 GO SUB 200  INPUT AT 0,0  B
RIGHT 1,"TEKST ", LINE t$  GO SU
B 300  LET w=20  LET k=k+1  IF k
=32 THEN LET k=0
   3 IF t$="" THEN LET t$="*"
   5 FOR n=1 TO LEN t$
   6 IF w=0 THEN LET w=20: LET k
=k+1  IF k=32 THEN LET k=0
   7 PRINT AT w,k; BRIGHT 1;t$(n
). BEEP .01,16
   8 LET w=w-1. NEXT n. IF w>0 T
HEN FOR p=w TO 1 STEP -1  PRINT
AT p,k, BRIGHT 1," "  NEXT p
   9 GO TO 2
 100 CLEAR 55815  BORDER 6  CLS
  PRINT AT 8,7,"CZEKAJ 54 SEKUND
Y !": GO SUB 500 . LET w=20. LET
k=0. LET t$=" * TEKST * ": GO SU
B 300. CLS . GO TO 5
 200 POKE 23606,0  POKE 23607,60
  RETURN   REM prosto
 300 POKE 23606,64  POKE 23607,2
16  RETURN : REM w lewo
 500 FOR q=32 TO 127  GO SUB 200
  PRINT AT 21,0, BRIGHT 1,CHR$ q
  FOR m=0 TO 7: POKE (55360+(8*q
)-m+7),(128 AND POINT (m,7))+(64
 AND POINT (m,6))+(32 AND POINT
(m,5))+(16 AND POINT (m,4))+(8 A
ND POINT (m,3))+(4 AND POINT (m,
2))+(2 AND POINT (m,1))+POINT (m
,0)  NEXT m  GO SUB 300  PRINT A
T 21,1, BRIGHT 1;CHR$ q  BEEP .0
1,16  NEXT q. RETURN
 930 SAVE "PIONOWO" LINE 100
```

J. Rubinowicz

Najmłodsze dziecko Olivetti



Jest to najnowszy komputer typu portable tej firmy. Zbudowany został na mikroprocesorze INTEL 80C88 (4.77 MHz). ROM M15 ma 16 Kb zaś RAM 512 Kb. Komputer pracuje pod nadzorem systemu operacyjnego MS-DOS (3.2). Na 10.5-calowym ciekłokrystalicznym ekranie, mieści się 25 wierszy tekstu w 80 lub 40 kolumnach (rozdzielczość 640x200 lub 320x200 punktów).

I jeszcze kilka szczegółów: rozmiary 350x70x280 mm; waga 5.2 kg.

```
8730 DATA 52,221,110,16,221,102
19,1,6,0,9,229,221,116,35,221,16
2,36,0,221,1542,-376
8740 DATA 126,1,61,40,26,237,176
,225,193,225,[illegible]
43,[illegible],176
8750 DATA 1,6,0,24,232,24,170,24
,142,24,106,51,61,221,110,35,221
,102,36,1,1581,555
8760 DATA 6,0,237,176,221,110,15
,221,102,19,1,6,0,237,176,213,22
6,176,6,6,2152,-173
8770 DATA 16,19,16,252,221,126,3
0,230,28,254,4,40,6,1,2,0,9,54,1
,237,1546,-876
8780 DATA 63,101,92,221,70,28,23
9,[illegible],51,2,6,192,63,3,61,9,4,3
,1213,81
8790 DATA 49,2,2,224,51,245,56,2
21,110,12,221,102,13,1,6,0,221,1
26,30,203,1694,-378
8800 DATA 103,40,16,221,94,35,22
1,86,36,237,176,24,6,24,146,[illegible],1
46,24,97,9,1765,31
8810 DATA 203,81,40,15,221,94,47
,221,86,46,1,6,0,237,176,1,6,0,9
,203,1659,-123
8820 DATA 95,40,26,221,126,1,235
,221,110,35,221,102,36,1,6,0,9,2
35,61,40,1620,26
8830 DATA 6,237,176,1,5,0,9,24,2
45,221,126,29,61,254,1,40,41,221
,119,26,1647,-265
8840 DATA 221,94,43,221,86,44,22
1,110,35,221,102,36,25,221,117,3
5,221,116,36,221,2426,-212
8850 DATA 110,47,221,102,48,25,2
21,117,47,221,116,48,24,157,24,1
53,24,114,221,126,2166,-64
8860 DATA 1,61,40,100,221,110,1,
221,94,43,221,86,44,221,116,26,2
21,102,21,25,1986,-32
8870 DATA 221,117,20,221,116,21,
221,110,19,221,102,19,25,17,5,0,
[illegible],221,117,[illegible],1635,-95
8880 DATA 221,116,19,221,110,53,
221,102,54,25,221,117,53,221,116
,54,221,94,14,[illegible],2471,26
8890 DATA 66,15,221,116,24,221,1
02,25,25,221,117,24,221,116,25,2
21,116,26,221,104,[illegible]
8900 DATA [illegible],221,114,[illegible],221,1
16,27,221,116,49,221,102,50,35,2
21,[illegible],221,116,[illegible]
8910 DATA [illegible],221,110,54,221,102,
[illegible],25,24,4,24,136,24,104,221,117
,51,221,116,52,1535,-152
8920 DATA 221,126,35,[illegible],40,73,2
21,126,30,[illegible],40,[illegible],221,53,3
,17,6,0,221,[illegible]
8930 DATA 116,37,[illegible]
[illegible]
8940 DATA [illegible]
[illegible]
8950 DATA 230,96,32,15,25,6,221,
94,[illegible],237,[illegible],52,221,117,41,2
21,116,42,221,2066,-190
8960 DATA 113,55,221,102,56,17,6
,0,[illegible],62,221,117,55,221,116,56,
24,146,221,126,2150,342
8970 DATA 40,[illegible],79,1,0,0,200,22
1,110,6,221,102,5,221,126,0,254,
1,40,11,[illegible],301
8980 DATA 71,[illegible],221,94,43,221,86,
44,[illegible],16,[illegible],51,35,221,86,34
,[illegible],43,
8990 DATA 20,221,116,21,17,5,0,2
[illegible],221,117,18,221,116,16,221,
117,35,221,116,[illegible],-251
9000 DATA [illegible],116,17,221,102,7
,[illegible],45,11,221,94,12
9010 DATA [illegible],16,253,221,117
,[illegible],11,47,221
,116,46,221,110,[illegible],405
9020 DATA 1,221,94,12,221,86,13,
1,6,0,221,116,35,221,102,36,237,
176,221,117,2133,170
9030 DATA 35,221,116,36,221,110,
29,221,102,21,1,5,0,237,176,221,
160,0,40,44,2617,-215
9040 DATA 66,1,5,0,221,110,35,22
1,102,36,237,176,221,117,35,221,
116,36,221,110,[illegible]
9050 DATA 47,221,102,48,1,5,0,23
7,176,221,116,16,221,70,17,5,221,
117,47,221,[illegible],-255
9060 DATA 116,48,24,237,237,83,1
61,92,221,126,0,[illegible],180,1,66,71,
[illegible]
9070 DATA [illegible],160,192,[illegible],53,3,61,
[illegible],224,16,192,2,61,246,224,3,1,
[illegible],1731,-111
9080 DATA 221,[illegible],221,85,23,22
1,110,[illegible],221,[illegible],13,1,5,0,237,17
[illegible],221,1,2040,-252
9090 DATA 40,[illegible],157,221,110,22,2
21,102,[illegible],221,117,47,221,116,46,
221,94,14,221,86,2405,151
9100 DATA 15,237,82,221,117,22,2
21,116,23,221,110,26,221,102,21,
221,94,43,221,86,2414,-164
9110 DATA 44,237,82,221,117,20,2
21,116,21,221,116,16,221,102,19,
237,82,1,5,6,2111,-267
9120 DATA 247,82,221,117,18,221
116,19,221,117,35,221,116,36,221
,126,1,24,135,221,2506,137
9130 DATA 126,30,254,6,1,0,0,200
,221,94,12,221,86,13,221,126,0,2
21,110,6,1648,114
9140 DATA 221,102,7,221,70,16,22
1,70,17,197,1,5,0,237,176,193,9,
61,34,245,2106,-565
9150 DATA 221,126,0,221,110,6,22
1,102,9,35,35,221,117,47,221,116
,48,221,110,14,2261,-17
9160 DATA 221,102,15,221,117,35,
221,116,36,221,110,47,221,102,46
,70,221,94,14,221,2451,-7
9170 DATA 85,15,25,221,117,47,22
1,116,48,221,110,6,221,102,7,16,
9,24,3,26,1633,47
9180 DATA 24,249,1,5,0,235,221,1
10,35,221,102,36,237,176,221,117
,35,221,116,36,2398,-416
9190 DATA 61,32,202,1,0,0,201,40
7,431
```



Labirynt

Jak odnaleźć drogę w gmatwaninie korytarzy? Czy komputer potrafi to zrobić szybko? Oto dwa przykłady. Program pierwszy to przykład jednego z wielu rozwiązań. Początkowe instrukcje (10÷530) wyświetlają listę funkcji programu, definiują labirynt, w którym kręte ścieżki przebija „świder" sterowany z klawiatury (opis klawiszy znajduje się w programie). Najistotniejszą dla działania programu sekwencją instrukcji jest zapis algorytmu (540÷780). Pomimo że program został napisany w amstradowskim dialekcie Basica łatwo go przeanalizować. Wyjaśnienia wymaga jedynie instrukcja **CHR$** (248) — (w wierszu 550), która jest kodem znaku graficznego występującego tylko w CPC 6128 oraz **TIME** — instrukcja pozwalająca korzystać z zegara mikroprocesora. Jej sens łatwo odgadnąć w wierszu 860 — to właśnie tu obliczany korzystając ze zmiennej „czas" jest czas (w sekundach), jaki potrzebuje komputer na wyznaczenie drogi wyjścia z labiryntu i jej graficzne przedstawienie.

A teraz uwaga! Czas odnajdywania wyjścia z plątaniny przejść nie zależy od jakości komputera, ale od sprawności szarych komórek! Oto przykład:

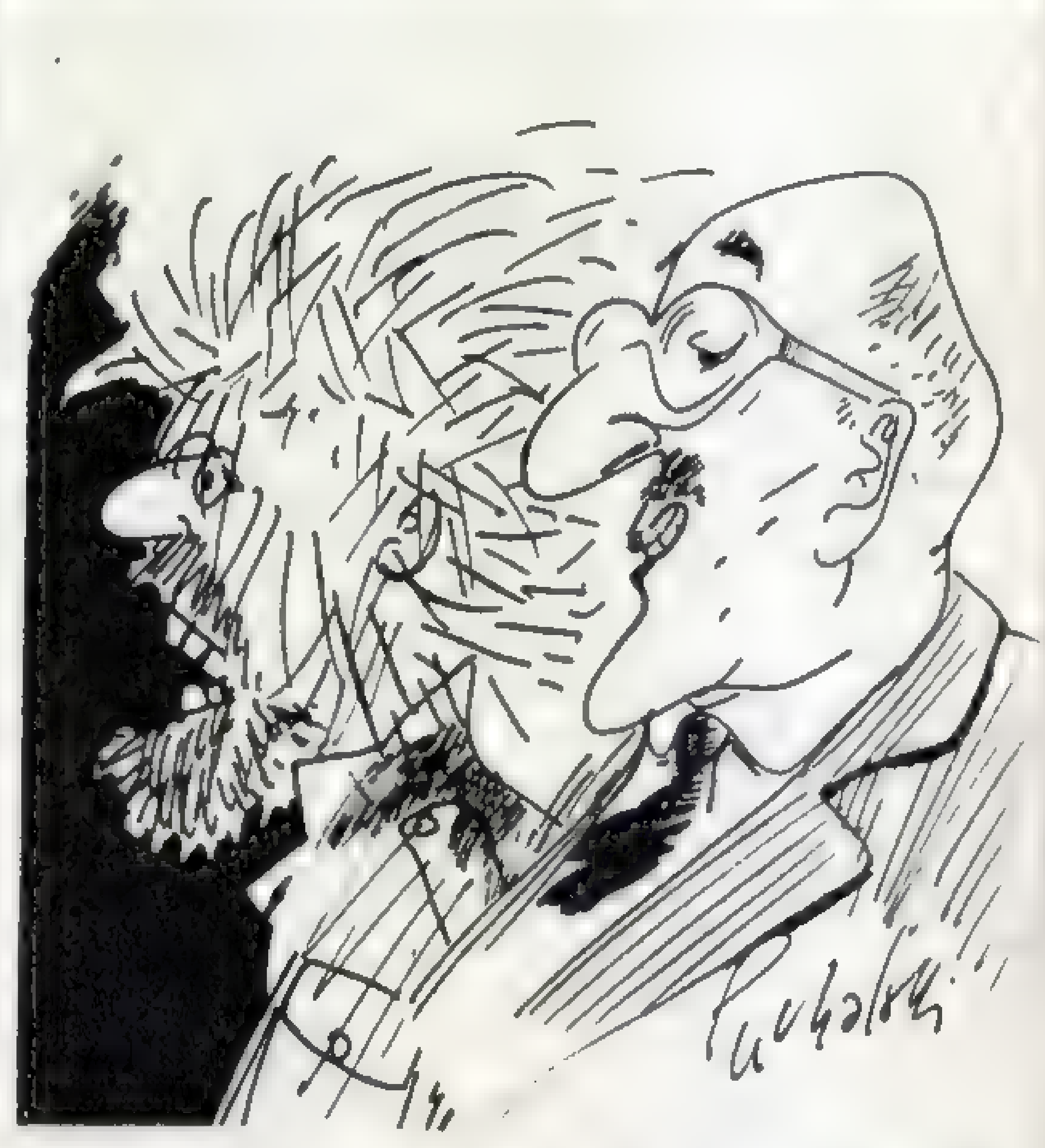
```
530 lab(x,y)=0
540 k=2
550 INPUT #1, "podaj wspolrzedne punktu startu  x,y";x,y
560 LOCATE x,y:PRINT CHR$(248)
570 lab(x,y)=k
580 i=2
590 j=2
600 IF lab(i,j)=k THEN GOTO 660
610 j=j+1
620 IF j<=n THEN GOTO 600
630 i=i+1
640 IF i<=n THEN GOTO 590
650 k=k+1:GOTO 580
660 IF lab(i,j+1)<>0 THEN GOTO 690
670 lab(i,j+1)=k+1
680 IF j+1=n THEN GOTO 780
690 IF lab(i+1,j)<>0 THEN 720
700 lab(i+1,j)=k+1
710 IF i+1=n THEN GOTO 780
720 IF lab(i,j-1)<>0 THEN GOTO 750
730 lab(i,j-1)=k+1
740 IF j-1=1 THEN GOTO 780
750 IF lab(i-1,j)<>0 THEN GOTO 610
760 lab(i-1,j)=k+1
770 IF i-1<>1 THEN GOTO 610
780 PRINT " koniec"
790 FOR n=1 TO 23
800 PRINT
810 FOR m=1 TO 23
820 IF lab(m,n)>1 THEN LOCATE m,n:PRINT CHR$(240):GOTO 840
830 LOCATE m,n:PRINT CHR$(143)
840 NEXT m:NEXT n
```

Powyższe instrukcje wstawmy do programu zamiast instrukcji 540—780. Efekt? Droga jest również wyznaczona, ale można się po niej poruszać jedynie od strony zewnętrznej labiryntu, a i czas jej wyznaczenia nie jest rewelacyjny. Szkoda jedynie, że opis takiej właśnie metody znaleźć można w podręczniku.

A. C.

Od redakcji. Program jest przykładem jednego z wielu rozwiązań. Czekamy zatem na propozycję efektywniejszych rozwiązań.

```
10 REM definiowanie labiryntu
20 DIM lab(24,24):DIM rob(24,24)
30 n=23
40 WINDOW #1,26,60,1,16:CLS
50 WINDOW #2,26,60,17,17
60 WINDOW #3,26,60,19,19
70 REM stan zerowy labiryntu
80 FOR i =1 TO 23
90 FOR k=1 TO 23
100 LOCATE i,k:PRINT CHR$(255)
110 lab(i,k)=1
120 NEXT k:NEXT i
130 x=1:y=1
140 PRINT #1,"wytyczenie tuneli"
150 PRINT #1,"   drazenie tunelu  - r"
160 PRINT #1,"   powrot swidra    - s"
170 PRINT #1,"           prawo    - p"
180 PRINT #1,"           lewo     - l"
190 PRINT #1,"           gora     - g"
200 PRINT #1,"           dol      - d"
210 PRINT
220 PRINT #1," zmiana trybu pracy - z"
230 PRINT #1,"  obliczenia        - k"
240 IF INKEY$="k" THEN PRINT #2,"trwaja obliczenia":GOTO 540
250 IF INKEY$="r" THEN r=1:PRINT #2,"drazysz tunel":GOTO 290
260 IF INKEY$="s" THEN PRINT#2,"nie drazysz": r=0:GOTO 290
270 GOTO 240
280 IF INKEY$="z" THEN PRINT #2,"zmiana trybu": GOTO 240
290 IF INKEY$<>"p" THEN GOTO 350
300 PRINT #2,"przesuwasz w prawo"
310 IF x+1 >23 THEN x=23:GOTO 330
320 x=x+1
330 LOCATE x,y:IF r<>0 THEN PRINT " ":PRINT #3,x,y:lab(x,y)=0
340 GOTO 290
350 IF INKEY$<>"l" THEN GOTO 410
360 PRINT #2,"przesuwasz w lewo"
370 IF x-1<1 THEN x=1:GOTO 390
380 x=x-1
390 LOCATE x,y:IF r<>0 THEN PRINT " ":PRINT #3,x,y:lab(x,y)=0
400 GOTO 350
410 IF INKEY$<>"g" THEN GOTO 470
420 PRINT #2,"przesuwasz w gore"
430 IF y-1<1 THEN y=1:GOTO 450
440 y=y-1
450 LOCATE x,y:IF r<>0 THEN PRINT " ":PRINT #3,x,y:lab(x,y)=0
460 GOTO 410
470 IF INKEY$<>"d" THEN GOTO 280
480 PRINT #2,"przesuwasz w dol"
490 IF y+1>23 THEN y=23:THEN GOTO 510
500 y=y+1
510 LOCATE x,y:IF r<>0 THEN PRINT " ":PRINT #3,x,y:lab(x,y)=0
520 GOTO 280
530 lab(x,y)=0
540 INPUT #1, "podaj wspolrzedne punktu startu  x,y";x,y
550 LOCATE x,y:PRINT CHR$(248)
560 FOR i=1 TO 23
570 FOR j=1 TO 23
580 rob(i,j)=lab(i,j)
590 NEXT j:NEXT i
600 czas=TIME
610 xp=x:yp=y
620 k=2
630 x=xp:y=yp
640 lab(x,y)=k
650 IF x=1 OR y=1 OR x=23 OR y=23 THEN GOTO 780
660 IF lab(x+1,y)<>0 THEN GOTO 690
670 k=k+1:x=x+1:lab(x,y)=k
680 GOTO 650
690 IF lab(x-1,y)<>0 THEN GOTO 720
700 k=k+1:x=x-1:lab(x,y)=k
710 GOTO 650
720 IF lab(x,y+1)<>0 THEN GOTO 750
730 k=k+1:y=y+1:lab(x,y)=k
740 GOTO 650
750 IF lab(x,y-1)<>0 THEN GOSUB 890: GOTO 620
760 k=k+1:y=y-1:lab(x,y)=k
770 GOTO 650
780 PRINT #2,"koniec obliczen"
790 PRINT
800 FOR n=1 TO 23
810 FOR m=1 TO 23
820 IF lab(m,n)>1 THEN LOCATE m,n:PRINT CHR$(240):GOTO 840
830 LOCATE m,n:PRINT CHR$(143)
840 NEXT m:NEXT n
850 PRINT "droga wyznaczona przez Twoj komputer"
860 czas=(TIME-czas)/300
870 PRINT #2,"czas obliczen = ";czas
880 STOP
890 rob(x,y)=1
900 FOR i=1 TO 23
910 FOR j=1 TO 23
920 lab(i,j)=rob(i,j)
930 NEXT j:NEXT i
940 RETURN
```



— Walka pokoleń?! Będziesz synu walczył całe życie!



1. Laser CO_2 daje impuls o bardzo dużej energii i krótkim czasie trwania. Za jego pomocą uzyskuje się plazmę i mini syntezy termojądrowe

2. Badanie zjawisk fizyki jądrowej przy użyciu komputerów znacznie przyspiesza także badania. Zajmują się tym w IFPiLM dr Józef FARNY i inż. Stanisław ... [illegible]



Według obliczeń naukowców za około 200 lat wyczerpią się kopalne źródła pozyskiwania energii ... [illegible] Nadzieję stanowią reakto[ry] jądrowe ... [illegible] ... wnętrza Ziemi. Ale bodaj niewyczerpalnym źródłem jest energia syntezy termojądrowej. Cechuje się ona tym, że nie pozostawia odpadów promieniotwórczych.

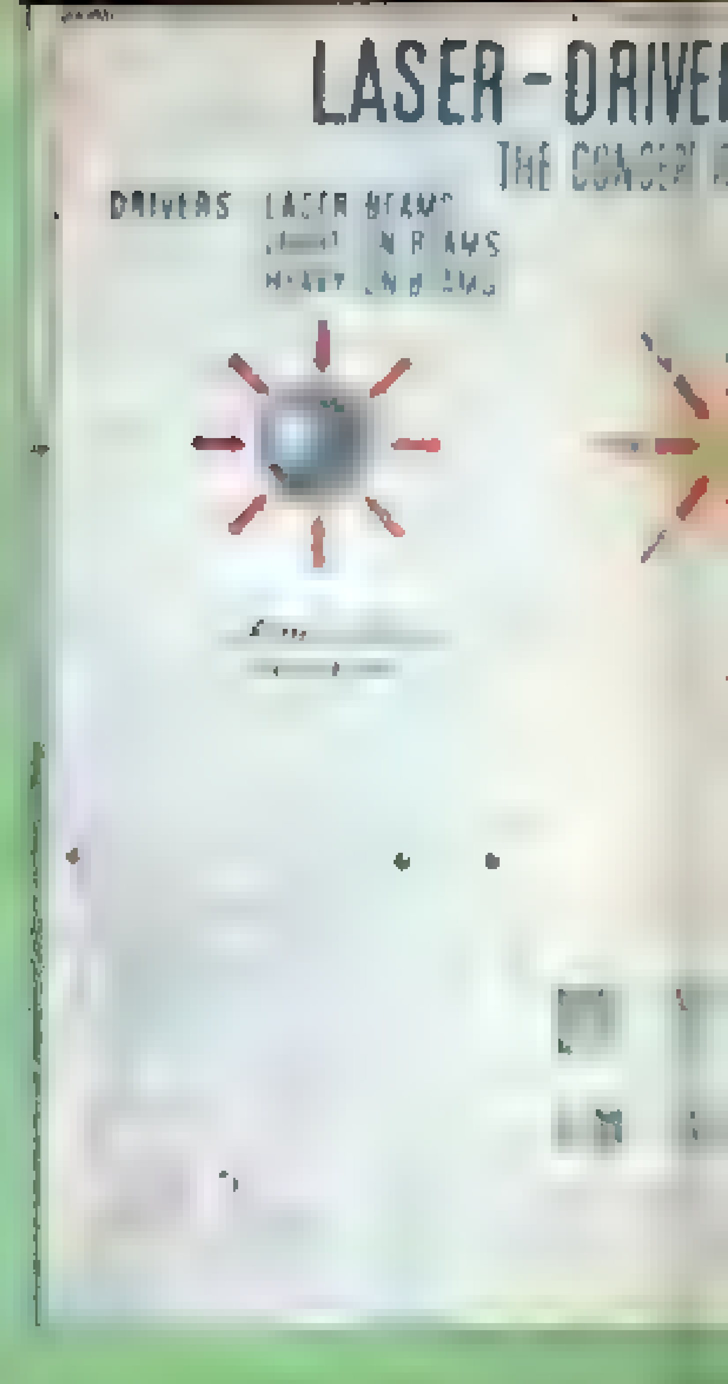
Termojądrowa reakcja (według Encyklopedii Powszechnej) zachodząca w bardzo wysokich temperaturach rzędu 10^7 K reakcja łączenia się lekkich jąder atomowych w jądro cięższe (np. jąder wodoru w jądro helu), z wydzieleniem ogromnych ilości energii; wykorzystywana w bombach wodorowych; źródło energii gwiazd).

Tak mówi teoria i praktyka sprowadzona do skrajnej postaci, czyli niszczenia. **W bombie wodorowej** do uzyskania wielkich temperatur paliwa syntezowego służy zapalnik w postaci bomby atomowej. Takie ładunki nie mogą być wykorzystywane do celów pokojowych, gdyż wyzwalają zbyt duże porcje energii i są źródłem skażenia promieniotwórczego. Do celów pokojowych potrzebny jest odbiór energii syntezy bądź w sposób ciągły, bądź przez częste wyzwalanie bardzo małych porcji energii w tzw. miniwybuchach. Stąd też wzięła się nazwa mikrosyntezy. Ponieważ konieczne jest zachowanie pełnej kontroli nad tym procesem, stąd określenie — kontrolowana synteza termojądrowa.

Pierwsze prace eksperymentalne w tej dziedzinie podjęto w latach pięćdziesiątych. Zasadniczym ... [illegible] ... było osiągnięcie odpowiednich gęstości i wielkich temperatur paliwa. **Materia** (w tym przypadku deuter i tryt — izotopy wodoru) nagrzana do gigantycznych temperatur rzędu milionów stopni musi być utrzymana w stanie skupienia tak, aby zdążyła przereagować jądrowo i oddać energię na zewnątrz. W takim stanie **staje się plazmą**, czyli czwartym stanem skupienia. A więc poza stanem ciekłym, stałym i gazowym.

Prof. SYLWESTER KALISKI: „Można chyba bez ryzyka powiedzieć, że gdyby udało się zrealizować pokojową syntezę termojądrową, wówczas oznaczałoby to dla ludzkości nie mniejszy skok rozwojowy niż ten, jaki przyniosło odkrycie elektryczności. Człowiek byłby wtedy uwolniony nie tylko od kryzysów energetycznych, lecz pojawiłyby się przed nim możliwości regulowania klimatu na Ziemi, sztucznego podgrzewania pól uprawnych, nowe perspektywy komunikacyjne, nieograniczone praktycznie możliwości podróży kosmicznych, nowe szanse przeobrażeń materii, badań naukowych i wiele, wiele innych. Słowem — nowa era w rozwoju ludzkości."

Należy stworzyć coś w rodzaju naczynia, w którym można było by utrzymać przez pewien czas nagrzaną do wielkich temperatur plazmę. Żadna



3. Schemat zjawiska synt[ezy] ...

WYŚCIG

materia nie mogła stanowić ... [illegible] ... gdyż najbardziej żaroodporn[e] ...

Najkorzystniejszy wpadł na pomysł ... [illegible] ... cząstek plazmy ... [illegible] ... tzw. **pułapkę magnetyczną**.

Tak pisał były komendant Wojsk[owej Akademii] Technicznej w książce „Lasery — ... [illegible] wa", która ukazała się w roku 1982 ... [illegible] ... z tym, który dążył do tego, tworząc **Instytut Fizyki Plazmy i Las[erowej Mikro]syntezy** ... [illegible]

Nie ... osiągnięć polskich ... [illegible] ... krajów ... [illegible] ... wają się one co dwa lata w innym ... Pradze. Następna, co stanowi ... Hiszpanii, co świadczy o znaczeniu prac i konieczności ... [illegible] ... przykład wiele badań obję... uwagi na realizowany ... gwiazdnych.

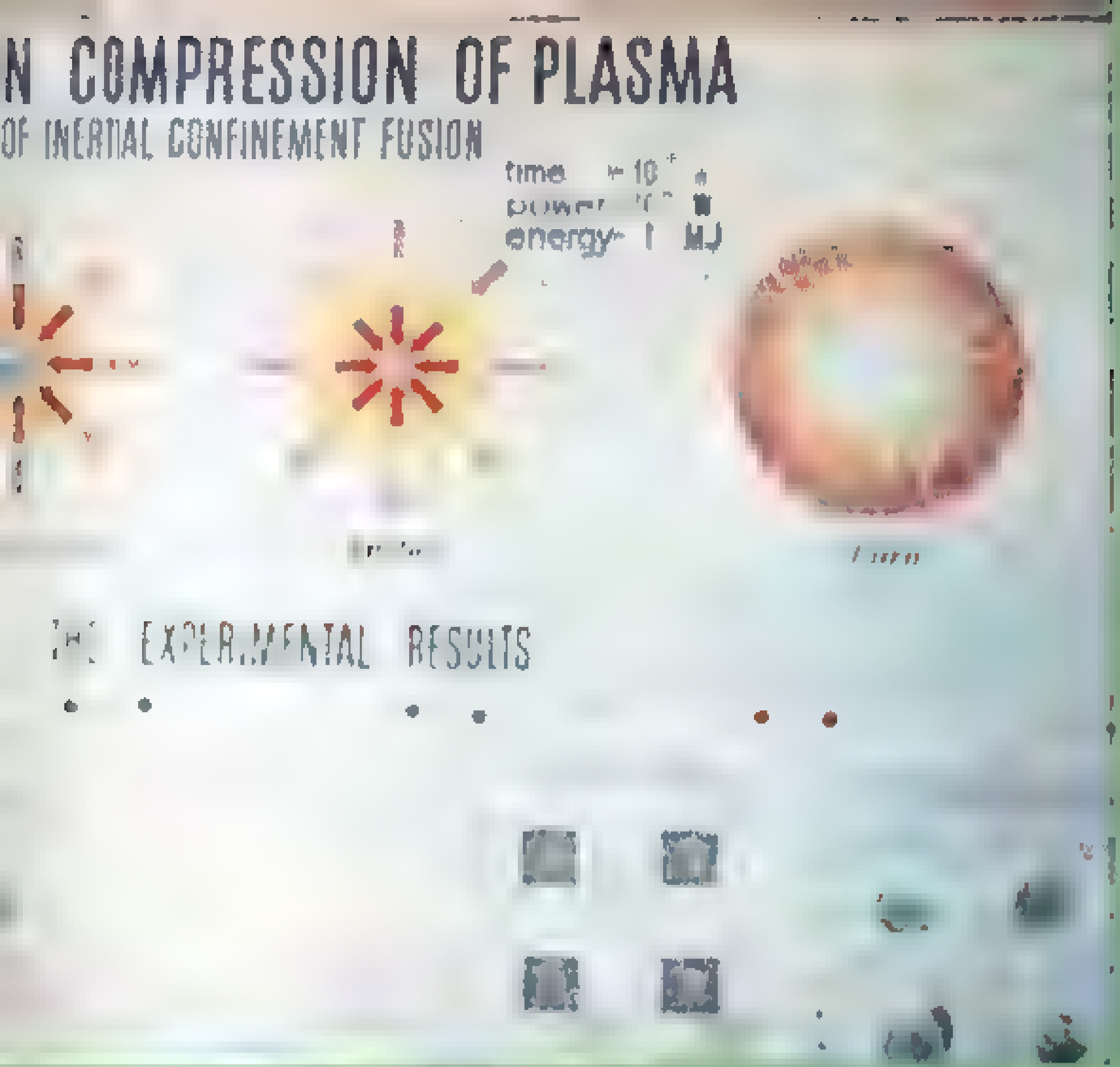
O ostatnich osiągnięciach z dziedziny laserowej mikrosyntezy mówi

*Prof. **Oleg Krochin** — Instytut Fizyki Akademii Nauk ZSRR*

— W warunkach ziemskich plazma jest wytwarzana sztucznie. I stąd trudności ... zjawiska. Naturalnie zachodzi ono np. na Słońcu, gdzie panuje temperatura 2 mln st... laboratorium musimy wytwarzać plazmę o 100 mln stopni. Jak badać takie wie... temperaturę plazmy w jej różnych warstwach? To niektóre z trudności, które prakty... wiązać, aby ruszyć dalej.

Podobnych detali typu technologicznego, czy uzyskania nowych bardzo odpor... rę i działanie silnego pola magnetycznego materiałów, jest więcej. Nie znamy do koń... się plazmy, która sprawia ciągle niespodzianki. Dlatego do badań i praktycznej ... zwanych u nas tokamakami podchodzimy bardzo ostrożnie. Maszyna taka kosztuje ... rubli i w końcu trzeba będzie ją rozmontować, aby uczynić kolejny krok. Dlatego za... cyzja o budowie, musi być ona poparta bardzo dokładnymi badaniami, przygotowa... ługi wyposażeniem w superdokładną aparaturę pomiarową. Polscy uczeni współ... tworzeniu m.in. aparatury do pomiaru rozkładu temperatur plazmy. Mają ... Cenimy je.

Obecnie tylko tokamaki są urządzeniami, z których uzyskuje się nieprzerw... jest to tylko 10 procent energii ... [illegible] ... dziesięciokrotnie. Ponieważ są to prace bardzo kosztowne, nasz kraj przedstaw... [Mię]dzynarodowej Agencji Atomowej w Wiedniu projekt współpracy INTOR, już osiem la[t] ... bowiem prowadzi się prace równolegle w trzech ośrodkach na świecie ... ZSRR, USA oraz Wielkiej Brytanii. Sądzę, że świat osiągnie taki postęp techniczny, ... rzystał z energii syntezy termojądrowej być może już u progu nowego stulecia.



...zy termojądrowej, zachodzącego w czterech etapach. Tak mówi teoria

...G PO ENERGIĘ

...y takim, naczy-
...działające na
...sposób tworzyć

...czyli cztery lata
...serowej Mikro-
...dra dobrze wy-
...na miarę świa-

...się w gronie
...plazmy. Odby-
...kraju. Ostatnia w
...przyspieszeniu
...nie wszystko
...m. Tzw. wojen

Właśnie dzięki zachodzącej na słońcu syntezie termojądrowej otrzymujemy ciepło i życiodajną energię. Podstawowym problemem w realizacji tzw. minisłońca na Ziemi, jest stworzenie warunków podobnych najpierw w laboratorium, a potem na skalę przemysłową.

Obecnie realizuje się ten cel trzema drogami. W urządzeniu o nazwie **„focus"** powstaje zjawisko pinch effect. Powoduje się skompresowanie plazmy szybko narastającym polem magnetycznym. Przy jednoczesnym podgrzewaniu plazmy przyjmuje ona postać tzw. sznura plazmowego. Gdyby zjawisko to trwało dostatecznie długo, mogłoby dojść do powstania syntezy termojądrowej. Jednak pojawia się niestabilność tego sznura, który się deformuje i rozlatuje. Wszystko odbywa się w czasie niezwykle małym, wynoszącym milionową część sekundy. Jednak nie rezygnuje się całkowicie z tej metody. Jak na razie przydatna jest ona do testowania i produkcji urządzeń pomiarowych i badania zachowania się plazmy, poznawania jej właściwości, nie do końca jeszcze odkrytych.

Podczas „spalania" 1 g mieszaniny deuteru i trytu wyzwoli się energia około 20 MWh. Inaczej, jeśli zamienić deuter zawarty w ciężkiej wodzie w energię syntezy, to jeden litr ciężkiej wody odpowiada pod względem energetycznym 250 litrom benzyny. Czyżby pojazdy na wodę? Czemu nie...

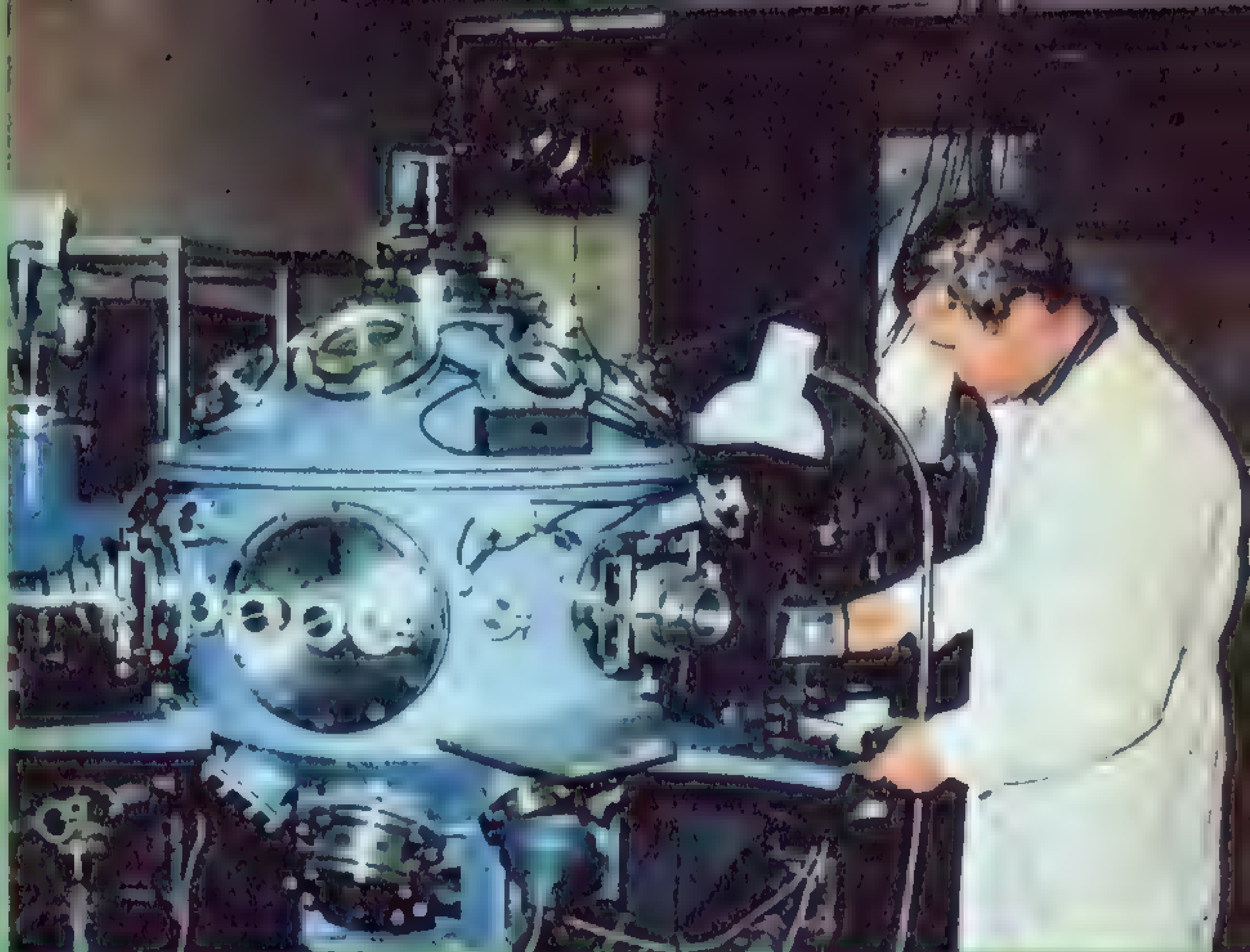
Druga metoda pułapki magnetycznej to tzw. **tokamaki**. Są to bardzo kosztowne urządzenia składające się (oczywiście upraszczając) z eliptycznej rury, na której znajduje się uzwojenie. Pole magnetyczne pozwala na utrzymanie plazmy stosunkowo długo, ale i tu nie udało się doprowadzić do sytuacji, w której ilość wydatkowanej energii byłaby równa uzyskanej. Wówczas to bowiem zachodziłby tzw. **eksperyment przełomowy**. Tokamag, z rosyjskiego akronimu nazwy Toroidalnaja KAmiera z MAGnitnym polem, jest jednak nadzieją wielu instytutów na świecie. Powstają urządzenia o coraz większych wymiarach i według zapowiedzi prawdopodobnie jeszcze pod koniec tego stulecia uda się doprowadzić do powstania eksperymentu przełomowego. Do wiodących należą ośrodki w Moskwie, Leningradzie (ZSRR), Osace (Japonia) Liwermore, Los Alamos (Stany Zjednoczone). Prekursorem metody tokamakowej był radziecki uczony prof. **Lew Arcymowicz**.

Z informacji podczas wystawy: „Osiągnięcia nauki i techniki radzieckiej" w Warszawie 1987 roku: „Czy można stworzyć termojądrowy reaktor dla elektrowni? Odpowiedzią na to pytanie może być urządzenie „Tokamak-10", które pokazujemy na wystawie. Jego parametry maksymalnie zbliżone są do przyszłego reaktora termojądrowego nowego typu siłowni. Zastosowano tu magnetyczny materiał nadprzewodzący".

...Natomiast w
...jak mierzyć

...koło 1 miliarda
...zapewnie de-
...osiągnięcia

...ergię. Na razie
...moc włożoną aż
...rezultatami, w
...że będzie ko-



4. W komorze próżniowej wykonuje się praktyczne eksperymenty laserowej mikrosyntezy. Nr 7. Henryk FEDEROWICZ

Jedną z licznych barier jest budowa potężnych urządzeń, ale też sposób odbioru olbrzymich ilości energii, chłodzenia itd. Potrzebne są nowe materiały, w tym nadprzewodzące, aby zmniejszyć zużycie dostarczanej energii i wielkość urządzeń.

Jest jeszcze trzecia droga osiągnięcia ostatecznego celu **„mikrosynteza laserowa"**, której prekursorami byli radziecki prof. **Basow** i amerykański **Dawson**. Zwrócili oni uwagę na możliwość zastosowania laserów do nagrzewania plazmy w bardzo krótkich czasach. W Polsce właśnie zespół kierowany przez prof. Kaliskiego doprowadził do uzyskania plazmy, a następnie mikrosyntezy.

— Zjawisko jest znane i możliwe do przeprowadzenia w małej skali — mówi dr inż. JÓZEF FARNY z IFPiLM w Warszawie. — Natomiast układ, który da energię musi mieć duże rozmiary. Po prostu w tej dziedzinie nie można efektu końcowego wykonać na małą skalę. Aby się posunąć naprzód, trzeba rozwijać technikę laserową, posiadać bardzo precyzyjną aparaturę pomiarową, aby na przykład badać zjawiska trwające dwie miliardowe części sekundy.

Przebieg mikrosyntezy ukazuje rysunek. Mamy do czynienia jakby z czterema etapami postępowania, podobnie jak w pracy silnika czterosuwowego. Najpierw formowanie plazmy poprzez jej oświetlenie laserem dużej mocy. Potem

Wystarczy połączyć z sobą dwa jądra deuteru, aby otrzymać tryt i wyzwolić sporą ilości energii. Co to jest deuter? To gaz-izotop wodoru. Uzyskiwany z tzw. ciężkiej wody. Gdyby zaś połączyć jądra deuteru i trytu, otrzyma się czterokrotnie więcej energii. Zapasy paliwa są niewyczerpalne. Starczyłyby ludzkości na około miliard lat.

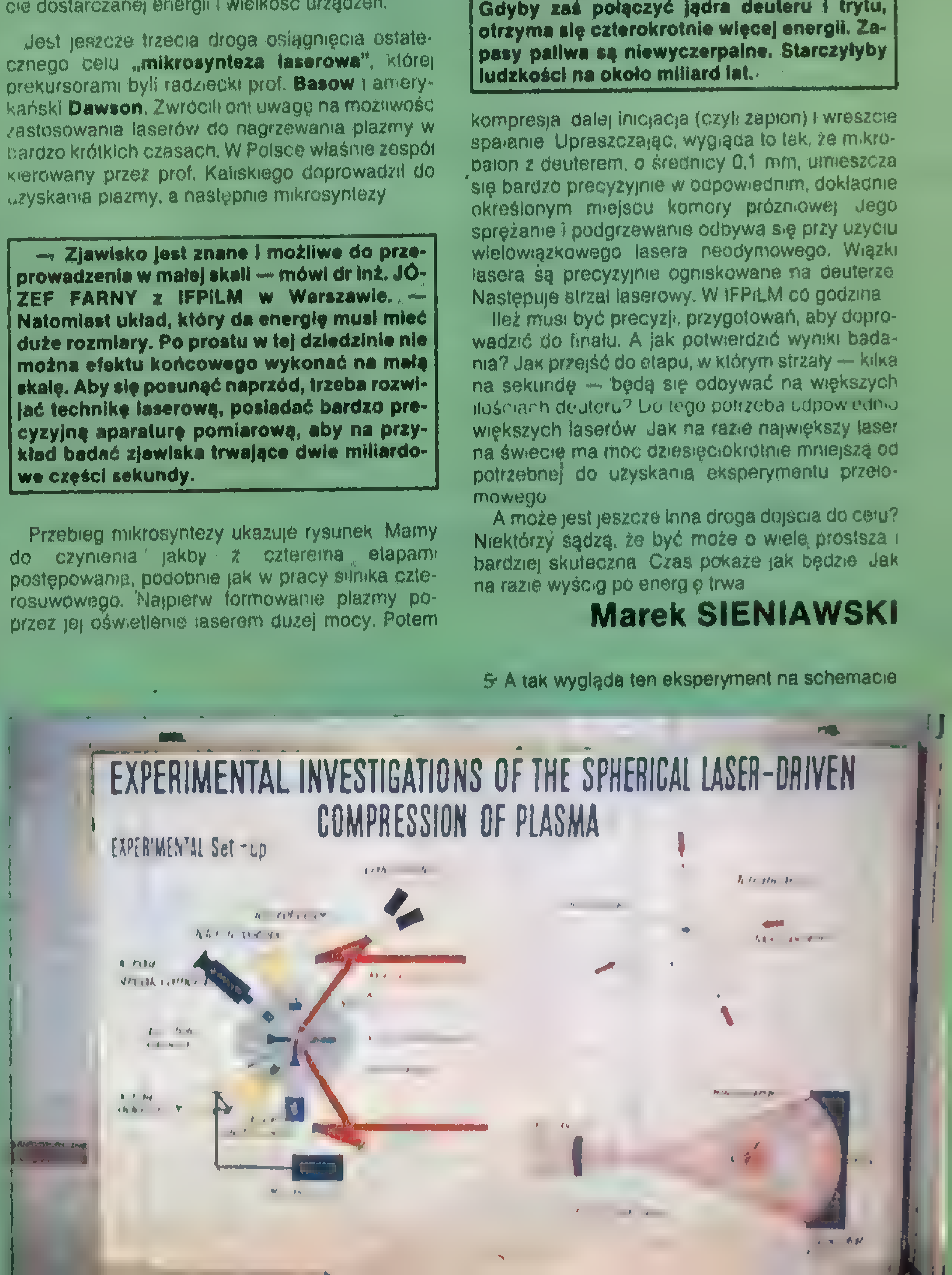
kompresja, dalej inicjacja (czyli zapłon) i wreszcie spalanie. Upraszczając, wygląda to tak, że mikrobalon z deuterem, o średnicy 0,1 mm, umieszcza się bardzo precyzyjnie w odpowiednim, dokładnie określonym miejscu komory próżniowej. Jego sprężanie i podgrzewanie odbywa się przy użyciu wielowiązkowego lasera neodymowego. Wiązki lasera są precyzyjnie ogniskowane na deuterze. Następuje strzał laserowy. W IFPiLM co godzina.

Ileż musi być precyzji, przygotowań, aby doprowadzić do finału. A jak potwierdzić wyniki badania? Jak przejść do etapu, w którym strzały — kilka na sekundę — będą się odbywać na większych ilościach deuteru? Do tego potrzeba odpowiednio większych laserów. Jak na razie największy laser na świecie ma moc dziesięciokrotnie mniejszą od potrzebnej do uzyskania eksperymentu przełomowego.

A może jest jeszcze inna droga dojścia do celu? Niektórzy sądzą, że być może o wiele prostsza i bardziej skuteczna. Czas pokaże jak będzie. Jak na razie wyścig po energię trwa.

Marek SIENIAWSKI

5. A tak wygląda ten eksperyment na schemacie



PROCEDURY ROM SPECTRUM (3)

Drugi zestaw procedur drukujących znaki zawiera bardziej wyrafinowane propozycje

16. PRINT UDG	2904	#0B58
CEL: wydruk znaku z puli grafik użytkownika		
WE: A — numer znaku (0...225) BC — fizyczna pozycja wydruku; HL — adres pozycji wydruku;		
WY: BC — nowa pozycja wydruku; HL — adres nowej pozycji; IX przechowane; używane AF';		

Procedura 16 pozwala stosować znacznie większy zbiór grafik użytkownika, niż w BASIC'u, mianowicie 256. Jeśli to za mało, to można przygotować więcej takich obszarów. Ważne, aby przed wywołaniem procedury 16 zmienna systemowa UDG zawierała adres początku właściwego obszaru. Przykład:

```
 10 ; ** UDG '87
 20 FETCH  EQU    #0B03
 30 STORE  EQU    #0ADC
 40 P_UDG  EQU    #0B58
 50        ORG    60000
 60        CALL   FETCH
 70        LD     A,200
 80        CALL   P_UDG
 90        CALL   STORE
100        RET
```

Chcąc zachować ciągłość wydruków, należy modyfikować pozycję.

17. PRINT ALL	2943	#0B7F
CEL: wydruk znaku z matrycy 8X8 przechowywanej jako ciąg ośmiu kolejnych bajtów w pamięci.		
WE: DE — adres matrycy; BC — fizyczna pozycja wydruku; HL — adres pozycji;		
WY: jak w proc. 16;		

Jak wiadomo definicja znaku polega na określeniu matrycy 8X8 bitów (punktów). Poszczególne linie matrycy zbierane są w bajty. Procedura 17 wydrukuje każdy znak, jeśli tylko podamy jej adres definicji. Przykładem niech będzie program drukujący grecką literę BETA.

```
 10 ; ** BETA '87
 20 PO_ALL EQU    #0B7F
 40 FETCH  EQU    #0B03
 50 STORE  EQU    #0ADC
 60        ORG    60000
 70        CALL   FETCH
 80        LD     DE,ZNAK
 90        CALL   PO_ALL
100        CALL   STORE
110        RET
120 ;
130 ZNAK   DEFB   0,56,68,68
140        DEFB   88,88,64,64
```

19. PRINT STRING	8252	#203C
CEL: wydruk łańcucha znaków		
WE: DE — adres łańcucha; BC — długość łańcucha,		
WY: DE — następny adres za łańcuchem; BC = 65535; flaga Z = 1; HL przechowane; używane rejestry pomocnicze (HL' przechowane);		

Łańcuch znaków ASCII może zawierać kody kontrolne, co pokazuje przykład:

```
 10 ; ** NAPIS '87
 20 PR_STR EQU    #203C
 30        ORG    60000
 40        LD     DE,NAPIS
 50        LD     BC,16
 60        CALL   PR_STR
 70        RET
 80 NAPIS  DEFB   18,1
 90        DEFB   20,1
100        DEFM   "COCA"
110        DEFB   20,0
120        DEFM   "COLA"
130        DEFB   18,0
```

Linia 80 zawiera kody dla FLASH 1, a następna dla INVERSE 1. Kody wyłączające znajdują się w liniach 110 i 130. Znacznie wygodniejsza do drukowania tekstów jest procedura 20.

20. PRINT MESSAGE	3082	#0C0A
CEL: wydruk łańcucha znaków;		
WE: DE — adres listy łańcuchów znakowych; A — numer kolejny łańcucha (pierwszy ma numer 0);		
WY: IX przechowane; używane rejestry pomocnicze (HL' przechowane);		

Jeśli program zawiera pewną ilość stałych tekstów, to wygodnie ułożyć je w listę. Pierwszy bajt traktujemy jako początek listy i zapalamy bit 7. Następnie umieszczamy kolejne teksty, zapalając 7

bit w ostatnim bajcie każdego łańcucha. Przykład takiej listy znajduje się w programie zamieszczonym w drugim zeszycie programów komputerowych „IKS-a".

Ciekawostką może być fakt, że przy innym wejściu do procedury 20 (#0C13) łańcuchy z listy o numerach od trzeciego w górę będą otrzymywać spację kończącą. Istnieje również możliwość wydruku firmowych komunikatów o błędach systemowych. Należy wtedy wejść do procedury z wartością w DE równą #1391.

21. PRINT SPACES 2768 #0AD0
CEL: wydruk łańcucha spacji
WE: D — liczba spacji do wydruku.
WY: E, IX przechowane: D=0, flaga Z=1;

Następne 4 procedury wiążą się z atrybutami.

22. BORDER 8859 #229B
CEL: ustawienie koloru ramki i jednocześnie atrybutów ekranu edycyjnego
WE: A — kolor tła (0..7);
WY: A — atrybut ekranu edycyjnego (dodany kontrastowy INK, wszystkie rejestry przechowane;

23 OUT FLASH 6337 #18C1
CEL: wydruk znaku z atrybutami: FLASH 1; INVERSE 0; OVER 0:
WE: A — kod znaku,
WY: BC, DE, HL, IX przechowane; używane rejestry pomocnicze (HL' nie przechowane);

Stosując procedurę 23 należy pamiętać, że przy przejściu do BASIC'a trzeba ustawić HL' na wartość 10072. Zawartości zmiennych systemowych ATTR—T i P—FLAG są ochraniane.

24. TEMPS 3405 #0D4D
CEL: przeniesienie wartości atrybutów głównych do zmiennych systemowych przechowujących atrybuty tymczasowe (dotyczy ATTR—P, MASK—P, P—FLAG):
WE: nieistotne;
WY: BC, DE, IX przechowane; A — aktualny atrybut tymczasowy, HL — wskazuje P—FLAG;

Znaki drukowane są na ekranie na podstawie atrybutów tymczasowych. Jeśli komenda tego wymaga, następuje wydruk według atrybutów głównych i wtedy stosuje się procedurę 24

25 SET ATTRIBUTE 3035 #0BDB
CEL: ustawienie wartości bajtu w obszarze atrybutów;
WE: HL — adres w obszarze ekranu; odpowiednie atrybuty w zmiennych ATTR—T, MASK—T i P—FLAG;
WY: HL — odpowiadający wejściowemu adres w obszarze atrybutów, A — atrybut w danym polu znakowym, D — wartość ATTR—T; E — wartość MASK—T; BC, IX przechowane;

Ostatni przykład dotyczy sposobu nadrukowywania znaku na znak tak, aby nie stracić żadnego bitu, jak to dzieje się w trybie OVER 1 (1 XOR 1=0).

```
 10 ; ** OR '86
 20 ATTR_T EQU        23695
 30 AT     EQU        #200A
 40 CLADDR EQU        #0E9E
 50        ORG        60000
 60 ; ustawienie atrybutów
 70        LD         A,57
 80        LD         (ATTR_T),A
 90 ; wydruk 1-ego znaku
100        LD         B,5
110        LD         C,5
120        CALL       AT
130        LD         A,"O"
140        RST        16
150 ; adres pola
160        LD         A,5
170        CALL       CLADDR
180        LD         BC,5
190        ADD        HL,BC
200 ; wydruk w trybie OR
210        LD         DE,ZNAK2
220        LD         B,8
230 PETLA  LD         A,(DE)
240        OR         (HL)
250        LD         (HL),A
260        INC        DE
270        INC        H
280        DJNZ       PETLA
290        RET
300 ; definicja znaku 2
310 ZNAK2  DEFB       1,2,4,8
320        DEFB       16,32,64,128
```

Ustawienie atrybutu nastąpiło w linii 140. Przedstawiony sposób nadruku jest pierwszym krokiem do opanowania grafiki tzw. duszków

Krzysztof MAMCARZ

Przyjdą do nas najzdolniejsi...



Rozmowa z dyr. naukowym Januszem CHODZEWICZEM

● Kto u was robi dobre interesy?

— Obie strony czyli klienci i my. Mamy satysfakcję jeśli nasz kontrahent wyjdzie od nas zadowolony

● Z tego trudno żyć?

— Ale zaufanie do firmy to gwarancje zysków

● Czyich?

— Jesteśmy jednostką gospodarki uspołecznionej, a głównym udziałowcem AlmaCompu jest Rada Naczelna Zrzeszenia Studentów Polskich. Do kasy ZSP wpływają więc zyski, zaś w kieszeniach pracowników powinna pozostawać godziwa zapłata, ale tylko za uczciwie wykonaną pracę

● Jest to jedyny wasz związek ze studentami?

— Nie. Wśród naszych współpracowników, wykonujących zresztą bardzo różnorodne prace, jest wielu studentów, do nich też kierujemy naszą dydaktyczną ofertę. Naturalnie pragniemy opierać naszą działalność na możliwościach i warunkach naukowo-technicznych studenckich kół naukowych i młodych pracowników nauki. Współpracujemy z ośrodkami naukowo-badawczymi PAN-u.

● Właśnie, oferujecie bardzo zróżnicowany wachlarz usług — czy w ten sposób nie rozpraszacie swych możliwości?

— Jesteśmy na początku działalności. Pracujemy dopiero od połowy tego roku. Dlatego staramy się zdobyć jak najwięcej klientów. W perspektywie skupimy się na kompleksowej obsłudze użytkowników sprzętu komputerowego. To znaczy rozpoznajemy potrzeby, dostarczamy i instalujemy sprzęt, udzielając na niego gwarancji rocznej, dostarczamy literaturę i oprogramowanie, wykonujemy prace projektowe i i programowe, a także prowadzić będziemy kursy przygotowujące nie tylko do obsługi sprzętu, ale także uczące oprogramowania.

● Będziecie mieli na nie chętnych? Przecież konkurencja jest duza

— Przyjmujemy zgłoszenia od zakładów pracy. Na przykład dla pracowników LOT-u prowadzić będziemy kurs programowania w Pascalu. Indywidualnych zgłoszeń mamy bardzo wiele. Mamy też w tym swój interes.

● Jaki?

— Najzdolniejszym zaoferujemy pracę

● Prawdę powiedziawszy jest to dość konwencjonalna oferta

— Ale niekompletna

● Macie jeszcze jakąś niespodziankę?

— Raczej konkretne plany. Otóż chcemy specjalizować się w systemach wielodostępnych. Jako rozwiązanie profesjonalne proponujemy Atari 520 ST z motorolą 68 020 z systemem wielodostępnym OS-9, co na naszym rynku jest raczej nowością

● Mamy zaufanie do IBM, nie mamy natomiast do sieci

— Bo ich nie ma.

● Co zatem proponujecie?

— Właśnie, sieci. Nasze modemy uzyskały już homologację: przeprowadziliśmy też pierwsze udane próby łączności

● To tylko pierwszy krok, a bazy danych?

— Tych niestety nie ma. Będziemy jednak tworzyć swoje.

● Przyznaję, że to bardzo pracochłonne zajęcie

— To inwestycja. Proszę sobie wyobrazić pracowników naukowych prowadzących swoje badania, którzy właśnie za pośrednictwem naszego komputera, z którym mogliby połączyć się nawet z domu mogliby otrzymać informacje bibliograficzne o najnowszych publikacjach.

● Dotychczas nikt tego nie proponował, a poza tym, czy to będzie się opłacało?

— Musi. Mamy nadzieję, że znajdziemy klientów. Wykupiony abonament umozliwi dostęp do naszego banku. Usprawni to dostęp do najnowszych informacji. Jesteśmy przecież Centrum Edukacji Informatycznej i właśnie na dostarczaniu wiedzy zalezy nam najbardziej

● Szczególnie w tej dziedzinie życzymy sukcesów

Dziękuję za rozmowę: W. Cetera

PROGRAM 36

C-64

MIKOŁAJ

MIKOŁAJ jest grą zręcznościową. Zadaniem gracza (Mikołaja) jest doręczenie dzieciom jak największej liczby prezentów. Szczegółowe zasady gry zawiera program. Do gry potrzebny jest joystick, który podłączamy do portu drugiego. Przed uruchomieniem programu należy stworzyć jego kopię.

S. Wasilewski

```
READY.

0 REM *** MIKOLAJ ***
10 PRINT"PROSZE CZEKAC":SK=0:FORI=0TO343:READA:POKE49152+I,A:SK=SK+A:NEXT
20 IFSK<>65195THENPRINT"BLAD W LINIACH: 1000 - 1033":END
30 SK=0:FORI=0TO327:READA:POKE14080+I,A:SK=SK+A:NEXT
40 IFSK<>20134THENPRINT"BLAD W LINIACH: 2000 - 1040":END
50 Z$=CHR$(147):Y$=CHR$(5):X$=CHR$(18):W$=CHR$(151):V$=CHR$(129):U$=CHR$(152)
51 T$=CHR$(155):S$=CHR$(111):R$=CHR$(183):N$=CHR$(160):M$=CHR$(17):L$=CHR$(29)
52 J$=CHR$(32):I$=CHR$(149):H$=CHR$(167):G$=CHR$(19):Q$=CHR$(112):P$=CHR$(109)
53 O$=CHR$(127):C$=X$+W$+S$+V$+R$+S$+R$+Q$+R$:B$=X$+W$+J$+U$+J$+T$+J$
54 D$=L$+X$+I$+S$+V$+R$+Q$+R$+Q$:A=53248
60 PRINTZ$Y$M$M$"JESTES MIKOLAJEM. TWOIM ZADANIEM JEST"
61 PRINT"ROZDAC JAK NAJWIECEJ PREZENTOW. DZIECI"
62 PRINT"OCZEKUJACE W MIESZKANIACH SA BARDZO"
63 PRINT"NIECIERPLIWE. KOLOR SWIATLA W OKNACH"
64 PRINT"OKRESLA CZAS OCZEKIWANIA NA PREZENTY."
65 PRINT"GDY JEDNO Z DZIECI STRACI NADZIEJE"
66 PRINT"NA OTRZYMANIE PREZENTU, TO W JEGO OKNIE"
67 PRINT"ROZBLYSNIE SWIATLO W KOLORZE ZOLTYM."
68 PRINT"NASTEPUJE KONIEC GRY. PREZENTY MIKOLAJ"
69 PRINT"WRZUCA PRZEZ KOMIN. PRZY ZETKNIECIU SIE"
70 PRINT"Z PLATKIEM SNIEGU.":PRINT:PRINT
80 INPUT"SZYBKOSC (1-4)",LE%:IFLE%<1ORLE%>4THEN80
90 POKE830,INT(12/LE%)
100 INPUT"POZIOM GRY (1-4)",LE%:IFLE%<1ORLE%>4THEN100
110 POKE49457,INT(8/LE%)
130 POKEA+33,0:POKEA+32,0:PRINTY$Z$:SC=0:POKE54296,15
140 PRINTG$M$M$M$M$M$M$M$M$M$M$M$:FORI=1TO2
150 PRINTX$L$L$B$CHR$(144)"    "B$L$L$L$B$L$L$L$B$L$L$L$B$:NEXT
160 PRINTD$D$D$D$D$:PRINTW$C$C$C$C$C$
170 A$="":FORI=1TO15:A$=A$+P$+J$:NEXT
180 PRINTW$X$J$A$O$:PRINTV$X$A$P$J$O$
190 A$=W$+X$:FORI=1TO33:A$=A$+N$:NEXT:PRINTW$X$A$
200 FORI=1TO5:PRINTTAB(31)I$H$:NEXT:PRINTTAB(31)I$H$;
210 POKEA+23,248:POKEA+27,255:POKEA,230:FORI=0TO16:POKEA+I,0:NEXT
220 POKEA,230:POKEA+1,100:POKE2040,220:POKEA+21,255:POKEA+28,255:POKEA+39,4
230 POKEA+38,11:POKEA+37,1:POKE832,0:POKE2042,223
240 POKEA+2,255:POKEA+3,100:POKE2041,221
250 PRINTG$M$M$M$M$M$M$TAB(31)W$"MAX":PRINTM$W$TAB(31)MX
260 POKEA+40,8:FORI=6TO14STEP2:POKEA+I,-103+(I*24)
270 POKEA+I+1,208:NEXT
280 FORI=3TO7:POKE2040+I,224:POKEA+39+I,0:NEXT
300 PRINTY$G$M$M$TAB(30)"WYNIK"
310 FORI=0TO10STEP2:B=4*256+(81*I):POKE834+I,B/256
320 POKE833+I,B-(PEEK(834+I)*256):NEXT
330 POKE831,1:POKE832,0
340 SYS49388:IFPEEK(829)=0THEN400
350 PRINTG$M$M$L$L$L$L$L$L$L$L$L$L$"KONIEC GRY"
360 POKE54276,0:POKE54277,255:POKE54273,10:POKE54276,129
370 GETF$:IFF$=""THEN370
380 IFSC>MXTHENMX=SC
390 GOTO130
400 SC=SC+100:PRINTG$M$M$M$M$V$TAB(30)SC
410 POKE54276,0:POKE54277,7:POKE54273,100:POKE54276,17:GOTO340

1000 DATA162,0,189,63,3,133,250,189
1001 DATA66,3,133,251,165,251,201,4
1002 DATA144,48,201,6,176,44,76,89
1003 DATA192,234,145,250,189,65,3,24
1004 DATA105,40,157,65,3,189,66,3
1005 DATA105,0,157,66,3,189,65,3
1006 DATA133,250,189,66,3,76,102,192
1007 DATA234,145,250,232,232,224,10,144
1008 DATA193,96,169,4,157,66,3,138
1009 DATA10,157,65,3,134,253,24,101
1010 DATA253,105,3,157,65,3,76,45
1011 DATA192,160,0,177,250,201,42,208
1012 DATA187,169,32,76,26,192,133,251
1013 DATA177,250,201,32,208,205,169,42
1014 DATA76,57,192,173,0,220,41,1
1015 DATA208,13,173,1,208,201,50,144
1016 DATA6,206,1,208,206,1,208,173
1017 DATA0,220,41,2,208,13,173,1
1018 DATA208,201,130,176,6,238,1,208
1019 DATA238,1,208,173,0,220,41,4
1020 DATA208,13,173,0,208,201,20,144
1021 DATA6,206,0,208,206,0,208,173
1022 DATA0,220,41,8,208,13,173,0
1023 DATA208,201,230,176,6,238,0,208
1024 DATA238,0,208,173,1,208,141,3
1025 DATA208,173,0,208,24,105,24,141
1026 DATA2,208,173,64,3,201,0,208
1027 DATA18,173,0,208,24,105,2,141
1028 DATA4,208,173,1,208,24,105,1
1029 DATA141,5,208,96,173,10,208,201
1030 DATA160,208,249,238,63,3,173,63
1031 DATA3,205,62,3,144,238,32,115
1032 DATA192,32,115,192,32,0,192,169
1033 DATA0,141,63,3,32,42,193,32
1034 DATA6,194,76,61,193,174,5,220
1035 DATA224,5,176,12,189,42,208,201
1036 DATA240,208,0,169,7,254,42,208
1037 DATA96,0,238,41,193,173,41,193
1038 DATA201,2,144,8,169,0,141,41
1039 DATA193,76,21,193,96,162,0,189
1040 DATA42,208,201,247,208,6,169,1
1041 DATA141,61,3,96,232,224,5,208
1042 DATA238,234,234,234,173,0,208,74
1043 DATA74,74,133,252,169,3,133,253
1044 DATA173,1,208,74,74,74,170,165
1045 DATA252,24,105,40,133,252,165,253
1046 DATA105,0,133,253,202,224,255,208
1047 DATA238,160,15,177,252,201,42,208
1048 DATA10,173,64,3,201,0,208,3
1049 DATA238,64,3,76,220,193,200,46
1050 DATA172,5,208,24,105,8,141,5
1051 DATA208,201,150,144,33,160,0,173
1052 DATA4,208,201,70,144,16,200,201
1053 DATA110,144,11,200,201,160,144,6
1054 DATA200,201,190,144,1,200,169,2
1055 DATA141,64,3,76,193,193,76,228
1056 DATA193,185,42,208,201,240,240,9
1057 DATA[illegible],42,208,56,233,1,[illegible],42
1058 DATA208,169,60,141,4,208,141,5
1059 DATA208,76,0,194,173,64,3,201
1060 DATA1,76,142,193,173,64,3,201
1061 DATA2,208,15,173,30,208,201,5
1062 DATA208,5,169,0,141,64,3,76
1063 DATA236,192,173,30,208,76,236,192
1064 DATA169,0,141,61,3,96,173,63
1065 DATA3,208,15,238,249,7,173,249
1066 DATA7,201,223,144,5,169,221,141
1067 DATA249,7,96,0,0,0,0,0
1999 REM SPRITE DATA
2000 DATA0,0,0,0,0,0,64,0
2001 DATA0,32,0,0,40,0,0,20
2002 DATA0,0,20,0,0,60,0,0
2003 DATA40,0,0,42,144,0,42,149
2004 DATA0,42,0,0,85,64,0,255
2005 DATA160,0,255,160,0,255,160,4
2006 DATA255,252,4,85,85,80,0,0
2007 DATA0,0,0,0,0,0,0,0
2008 DATA0,0,0,0,0,0,0,0
2009 DATA0,0,0,0,0,0,0,0
2010 DATA0,0,0,0,0,0,0,0
2011 DATA0,0,0,0,2,0,0,0
2012 DATA128,0,0,40,32,0,170,40
2013 DATA42,162,2,170,160,10,170,160
2014 DATA40,0,160,32,0,40,128,0
2015 DATA8,0,0,0,0,0,0,0
2016 DATA0,0,0,0,0,0,0,0
2017 DATA0,0,0,0,0,0,0,0
2018 DATA0,0,0,0,0,0,0,0
2019 DATA0,0,0,0,2,0,0,0
2020 DATA128,0,0,40,32,0,170,40
2021 DATA40,162,2,170,160,10,170,160
2022 DATA10,0,160,2,128,128,0,130
2023 DATA0,0,0,0,0,0,0,0
2024 DATA0,0,0,0,2,128,0,10
2025 DATA160,0,26,172,0,166,164,2
2026 DATA169,232,14,169,100,11,174,88
2027 DATA6,250,104,9,165,96,10,110
2028 DATA64,2,154,0,0,164,0,0
2029 DATA48,0,0,0,0,0,0,0
2030 DATA0,0,0,0,0,0,0,0
2031 DATA0,0,0,0,0,0,0,0
2032 DATA170,186,171,170,186,171,170,186
2033 DATA171,170,186,171,170,186,171,170
2034 DATA186,171,170,186,171,170,186,171
2035 DATA170,122,167,85,117,87,255,255
2036 DATA255,170,186,171,170,186,171,170
2037 DATA186,171,170,186,171,170,186,171
2038 DATA170,186,171,170,186,171,170,186
2039 DATA171,170,122,167,85,117,8[illegible],0
2040 DATA0,0,0,0,0,0,0,0
3000 RETURN

READY.
```

W RZĘDZIE

W tej grze dysponujesz tablicą o wymiarze 10 x 12. Aby wygrać, musisz mieć w wierszu dziesięć swoich znaków, które wstawiasz w dane miejsce tablicy, tzn. w najniżej położone wolne miejsce, w wybranej przez ciebie kolumnie, klawiszami od 0 do 9.

Komputerowi, aby wygrał z tobą, wystarczy ustawienie w kolejności pięciu znaków w dowolnym wierszu. W wypadku, gdy znaki zostały wstawione w ostatni wiersz tablicy, a nie nastąpiło rozstrzygnięcie gry, komputer sygnalizuje fakt oraz proponuje powtórną partię. Jako znaki przyporządkowano: tobie — C, komputerowi — K.

Janusz JANIEC

```
5 OPEN #1,4,0,"K:"
7 POKE 752,1
8 POKE 710,3
10 DIM S(160),H(11),W(4),Z$(16)
12 Z$="                "
15 ? CHR$(125)
17 POSITION 14,10:? CHR$(9);CHR$
(25);CHR$(32);CHR$(140);CHR$(143
);CHR$(15);CHR$(32);CHR$(32)
18 POSITION 22,10:? CHR$(153);CH
R$(32);CHR$(153);CHR$(149);CHR$(
149)

19 POSITION 14,11:? CHR$(12);CHR
$(25);CHR$(32);CHR$(25);CHR$(32)
;CHR$(25);CHR$(32);CHR$(32)
20 POSITION 22,11:? CHR$(25);CHR
$(32);CHR$(153);CHR$(149);CHR$(1
39)
21 POSITION 14,12:? CHR$(32);CHR
$(25);CHR$(32);CHR$(25);CHR$(32)
;CHR$(25);CHR$(32);CHR$(153)
22 POSITION 22,12:? CHR$(32);CHR
$(32);CHR$(15);CHR$(32);CHR$(153
)
23 POSITION 14,13:? CHR$(9);CHR$
(139);CHR$(32);CHR$(143);CHR$(14
0);CHR$(12);CHR$(32);CHR$(25)
24 POSITION 22,13:? CHR$(32);CHR
$(32);CHR$(143);CHR$(21);CHR$(13
7)

25 POSITION 8,22:? "Wcisnij dowo
lny klawisz"
27 GET #1,A
29 ? CHR$(125)
95 POSITION 8,2:? Z$:POSITION 12
,4:? Z$:POSITION 22,11:? Z$
98 POKE 764,255
100 FOR H=0 TO 9:H(H)=1:NEXT H
105 FOR Y=2 TO 13
110 FOR X=1 TO 10
115 S(Y*10+X)=46
120 NEXT X:NEXT Y
125 W(1)=2:W(2)=9:W(3)=10:W(4)=1
1
130 G=0:T=0
135 GOSUB 1100
140 POSITION 22,9:? "TWOJE POSUN
IECIE"
145 RESTORE
150 FOR AA=0 TO 9
153 READ A
154 SOUND 0,0,0,0
157 IF PEEK(764)=A THEN 188
159 SOUND 0,250,13,3
160 NEXT AA
163 POSITION 22,9:? Z$
167 FOR B=0 TO 50:NEXT B
170 GOTO 140
188 POKE 764,255
200 POSITION 22,9:? Z$
205 IF C>130 THEN 1900
210 R=67
220 C=H(AA)*10+AA+11
230 H(AA)=H(AA)+1
240 S(C)=R
300 R=75
310 AA=0
315 AA=AA+1
325 C=H(AA)*10+AA+11
327 IF S(C)<>46 THEN 450
360 R=75
370 C=C+10
380 IF S(C)<>46 THEN 450
450 IF AA<>9 THEN 315
500 AA=INT(RND(0)*10)
510 C=H(AA)*10+AA+11
530 S(C)=75
540 H(AA)=H(AA)+1
560 R=75
580 GOSUB 1100
590 FOR B=2 TO 12
592 BC=B*10
594 IF S(BC+1)=67 AND S(BC+2)=67
 AND S(BC+3)=67 AND S(BC+4)=67 A
ND S(BC+5)=67 AND S(BC+6)=67 THE
N 610
595 FOR BD=0 TO 5
596 BE=BD+BC
597 IF S(BE+1)=75 AND S(BE+2)=75
 AND S(BE+3)=75 AND S(BE+4)=75 A
ND S(BE+5)=75 THEN 1800
598 NEXT BD
599 NEXT B
600 GOTO 140
610 IF S(BC+7)=67 AND S(BC+8)=67
 AND S(BC+9)=67 AND S(BC+10)=67
THEN 1700
620 GOTO 140
1100 REM KRESLENIE TABLICY
1105 POSITION 9,7:? CHR$(160);CH
R$(160);CHR$(160);CHR$(160);CHR$
(160);CHR$(160);CHR$(160)

1106 POSITION 16,7:? CHR$(160);C
HR$(160);CHR$(160);CHR$(160);CHR
$(160)
1110 FOR Y=12 TO 2 STEP -1
1120 POSITION 9,20-Y:? CHR$(160)
;
1130 FOR X=1 TO 10
1140 ? CHR$(S(Y*10+X));
1150 NEXT X
1170 POSITION 20,20-Y:? CHR$(160
);
1180 NEXT Y

1190 POSITION 9,19:? CHR$(160);"
0123456789";CHR$(160)
1200 POSITION 9,20:? CHR$(160);C
HR$(160);CHR$(160);CHR$(160);CHR
$(160);CHR$(160);CHR$(160)
1201 POSITION 16,20:? CHR$(160);
CHR$(160);CHR$(160);CHR$(160);CH
R$(160)
1210 RETURN
1500 DATA 50,31,30,26,24,29,27,5
1,53,48
1600 POSITION 8,2:? "CZY GRASZ D
ALEJ?"
1610 POSITION 12,4:? CHR$(84);"a
k/";CHR$(78);"ie"

1620 IF PEEK(764)=45 THEN 95
1630 IF PEEK(764)=35 THEN 2000
1640 FOR BB=0 TO 40:NEXT BB
1650 RETURN
1700 FOR BB=0 TO 9
1710 POSITION 10+BB,20-B:? "c"
1715 NEXT BB
1720 GOSUB 1600
1722 POSITION 22,9:? "BRAWO!! WY
GRALES"
1730 FOR BB=0 TO 9
1740 POSITION 10+BB,20-B:? "C"
1750 NEXT BB
1755 FOR BB=0 TO 20:NEXT BB
1760 GOTO 1700
1800 FOR BB=BD TO 4+BD
1810 POSITION 10+BB,20-B:? "h"
1815 NEXT BB
1820 GOSUB 1600

1822 POSITION 22,9:? "WYGRAL KOM
PUTER!"
1830 FOR BB=BD TO 4+BD
1840 POSITION 10+BB,20-B:? "H"
1850 NEXT BB
1855 FOR BB=0 TO 20:NEXT BB
1860 GOTO 1800
1900 GOSUB 1600
1910 POSITION 22,9:? "KOMPUTER D
OSZEDL"
1920 POSITION 22,11:? "DO KONCA
TABLICY"
1930 GOTO 1900
2000 ? CHR$(125):POKE 752,0:NEW
```

EDYTOR KODU MASZYNOWEGO

A T A R I

Każdy użytkownik komputera wprowadzał zapewne niejednokrotnie kod maszynowy zapisany w liniach DATA programu. Wie zatem, jak żmudna jest to praca i jak łatwo przy niej o popełnienie błędów, których wykrycie jest również kłopotliwe. Narzędziem pomocnym do wprowadzenia kodu maszynowego z dowolnego listingu do pamięci mikrokomputera jest program MCEDYTOR — edytor kodu maszynowego. Sądzę, że po przeczytaniu opisu jego funkcji, każdy z czytelników podejmie trud włączenia go do swojej biblioteki programów. Program napisany jest w języku BASIC i zajmuje ok. 10KB pamięci.

Edytor po uruchomieniu żąda podania kilku parametrów sterujących jego pracą. Są to:

1) Liczba danych w wierszu (zmienna LW) — jest to liczba danych oddzielonych przecinkiem, zawartych w jednej linii DATA. Jedyne ograniczenie wartości tej liczby występuje wówczas, gdy z wprowadzonego kodu edytor będzie tworzył również linie DATA programu BASIC — długość takiej linii nie może przekroczyć 116 znaków. Odliczając pięć znaków na numer linii, cztery znaki „DATA" oraz odstępy — w tym wypadku liczba danych w wierszu nie może przekroczyć 26.
2) Nośnik, na którym należy utworzyć zbiór z wprowadzonym kodem maszynowym (zmienna NOS) — T-taśma, D — dysk.
3) Rodzaj zbioru, który należy utworzyć (zmienna RZB) — użytkownik edytora ma do wyboru następujące opcje

B — zbiór z programem binarnym: — typu „load and go"

D — zbiór typu BOOT TAPE lub BOOT DISC

L — zbiór zawierający linie DATA z wprowadzonym kodem maszynowym. Zbiór ten może być dołączany instrukcją ENTER „C:" lub ENTER „D: nazwa" do tworzonego programu w języku BASIC jako podprogram w kodzie maszynowym. Wybierając opcję L, użytkownik edytora, po zakończeniu wprowadzania kodu maszynowego, określa również numer pierwszej linii oraz krok numeracji.

4) Wskaźnik występowania sum kontrolnych podczas wprowadzania kodu maszynowego. Jeżeli sumy kontrolne będą wprowadzane — T, jeżeli nie będą — N. Każdy programista, przygotowując listing długiego programu napisanego w kodzie maszynowym, aby zmniejszyć prawdopodobieństwo popełnienia błędu przy wprowadzaniu kodu do komputera, oblicza sumy kontrolne na końcu każdego wiersza (linii DATA) listingu. Najczęściej sumą kontrolną jest suma wszystkich danych w jednej linii, lecz sumy kontrolne mogą być obliczane również w inny sposób. W razie wybrania opcji T, edytor „zapyta się" o sposób obliczania sum kontrolnych. W edytorze zakodowane są dwa sposoby: pierwszy — suma kontrolna jest sumą danych w wierszu (linii), drugi — suma kontrolna jest resztą z dzielenia sumy danych w linii przez 256. W sytuacji, gdy sumy kontrolne nie są obliczane żadnym z wymienionych sposobów, można zakodować sposób ich obliczania w liniach 502—506 edytora.

Edytor żąda ponadto podania trzech dodatkowych parametrów: adresu startowego dla wprowadzanego programu (zmienna STA) oraz adresu początkowego (zmienna AP) i końcowego (zmienna AK) obszaru pamięci, którą wprowadzany program zajmuje. Kilka słów wyjaśniających znaczenie tych parametrów.

W wypadku, gdy rodzaj zbioru, który edytor będzie tworzył z wprowadzanego kodu, został określony opcją B lub D parametry STA, AP i AK są zwykłymi parametrami uruchamiania, wymaganymi przez DOS lub system operacyjny komputera. Gdy została wybrana opcja L, znaczenie tych parametrów jest inne. Adres startowy jest w tej sytuacji niepotrzebny i edytor „nie pyta się" o niego. Parametry AP i AK określają wielkość obszaru pamięci (początek, koniec) w którym edytor będzie przechowywał wprowadzany kod maszynowy przed zapisaniem go na taśmę lub dysk. Na przykład: chcemy wprowadzić 30 linii kodu po 20 danych w każdej linii — razem 600 bajtów kodu. Jako adres startowy podajemy 1, a adres końcowy 600. Adresów AP i AK w żadnym wypadku nie należy utożsamiać z bezwzględnymi adresami obszaru RAM, w którym przechowywany jest wprowadzany kod — stąd nie należy obawiać się, że ich nieprzemyślane dobranie zniszczy aktualnie pracujący w pamięci edytor.

Po wprowadzeniu wszystkich wymienionych wyżej parametrów, edytor wyświetli podany adres początkowy i w tym momencie możemy przystąpić do wprowadzania kodu maszynowego. Edytor będzie kontrolował:

— czy każdy znak wprowadzanego kodu jest cyfrą?,

— czy każda wprowadzona liczba jest mniejsza od 256?,

```
50 DIM SEK$(128),SIOV$(4),CIOV$(
6)
52 SIOV$="hLS ":SIOV$(4,4)=CHR$(
228)
54 CZSEK=2000
56 CIOV$="h":CIOV$(2)=CHR$(162)
CIOV$(3)=CHR$(32):CIOV$(4,5)="LV
":CIOV$(6)=CHR$(228)
58 CZZB=2100
60 WPRD=2200
100 GRAPHICS 0:DL=PEEK(560)+256*
PEEK(561)+4:POKE DL-1,71:POKE DL
+2,6
110 POSITION 6,0:? "MCedytor":PO
KE 710,0
120 OPEN #1,4,0,"K:":GOSUB WPRD
130 DIM BUF$(AK-AP+127),WIER(LW)
,T$(20),F$(20)
140 BUF$=CHR$(0):BUF$(AK-AP+30)=
BUF$:BUF$(2)=BUF$:SEK$=BUF$:ADDR
=AP
142 DIM LINIA$(114),KOD$(3)
160 IF RZB=76 THEN 360
170 IF NOS<>84 THEN BUF$="":GOTO
 250
171 REM =========================
=====
172 REM
173 REM TWORZENIE LADOWACZA DLA
174 REM ZBIORU TYPU BOOT TAPE
175 REM
176 REM =========================
=====
190 AP=AP-24:BUF$=CHR$(0):BUF$(2
)=CHR$(INT((AK-AP+127)/128))
200 H=INT(AP/256):L=AP-H*256:BUF
$(3)=CHR$(L):BUF$(4)=CHR$(H)
210 AINIT=AP+8:H=INT(AINIT/256):
L=AINIT-H*256:BUF$(5)=CHR$(L):BU
F$(6)=CHR$(H)
220 FOR I=7 TO 24:READ A:BUF$(I)
=CHR$(A):NEXT I
225 DATA 24,96,169,60,141,2,211,
169,0,133,10,169,0,133,11,76,0,0
230 H=INT(STA/256):L=STA-H*256:B
UF$(15)=CHR$(L):BUF$(19)=CHR$(H)
240 BUF$(23)=CHR$(L):BUF$(24)=CH
R$(H)
250 IF NOS<>68 THEN 360
260 IF RZB=66 THEN 360
261 REM =========================
=====
262 REM
263 REM TWORZENIE LADOWACZA DLA
264 REM ZBIORU TYPU BOOT DISC
265 REM
266 REM =========================
=====
290 AP=AP-30:BUF$=CHR$(0):BUF$(2
)=CHR$(INT((AK-AP+127)/128))
300 H=INT(AP/256):L=AP-H*256:BUF
$(3)=CHR$(L):BUF$(4)=CHR$(H)
310 AINIT=STA:H=INT(AINIT/256):L
=AINIT-H*256:BUF$(5)=CHR$(L):BUF
$(6)=CHR$(H)
320 RESTORE 330:FOR I=7 TO 30:RE
AD A:BUF$(I)=CHR$(A):NEXT I
330 DATA 169,0,141,231,2,133,14,
169,0,141,232,2,133,15,169,0,133
,10,169,0,133,11,24,96
340 H=INT(AP/256):L=AP-H*256:BUF
$(8)=CHR$(L):BUF$(15)=CHR$(H)
350 H=INT(STA/256):L=STA-H*256:B
UF$(22)=CHR$(L):BUF$(26)=CHR$(H)
360 GRAPHICS 0:POKE 712,10:POKE
710,10:POKE 709,2
370 ? ADDR;":";:FOR J=1 TO LW
380 HK2=255:HK1=3:GOSUB 570:IF N
=-1 THEN J=J-1:GOTO 380
390 IF N=-19 THEN 720
400 IF N=-12 THEN CZYT=1:GOTO 72
0
410 TRAP 410:IF N=-14 THEN ? :?
"Podaj nowy adres poczatkowy ";
INPUT ADDR:? :GOTO 370
420 TRAP 40000:IF N<>-4 THEN 480
430 TRAP 430:? :? "Wyswietl wpro
wadzone dane":? :? "od adresu ";
:INPUT F:? " do adresu ";:INPUT
T:TRAP 40000
440 IF F<AP OR F>AK OR T<AP OR T
>AK OR T<F THEN ? CHR$(253);AP;"
 <= adres <= ";AK:GOTO 430
450 FOR I=F TO T STEP LW:? :? I;
":";:FOR K=0 TO LW-1:N=PEEK(ADR(
BUF$)+I+K-AP):T$="00U"
452 T$(4-LEN(STR$(N)))=STR$(N)
460 IF PEEK(764)<255 THEN GET #1
,A:POP :POP :? :GOTO 370
```

```
470 ? T$;",";:NEXT K:? CHR$(126)
;:NEXT I:? :? :GOTO 370
480 IF N<0 THEN ? :GOTO 370
490 WIER(J)=N:NEXT J
500 IF SUMK=0 THEN ? CHR$(126) G
OTO 530
502 SK=0
504 FOR I=1 TO LW:SK=SK+WIER(I):
IF RSK=1 THEN SK=SK-256*(SK>255)
506 NEXT I
510 RF=128:SOUND 0,200,12,8:HK2=
9999:HK1=4:GOSUB 570:SOUND 0,0,0
,0:RF=0:? CHR$(126)
520 IF N<>SK THEN ? :? "Bledna s
uma kontrolna",CHR$(253),   GOT
O 370

530 FOR W=15 TO 0 STEP -1:SOUND
0,50,10,W:NEXT W
540 FOR I=1 TO LW:POKE ADR(BUF$)
+ADDR-AP+I-1,WIER(I):NEXT I
550 ADDR=ADDR+LW:IF ADDR<=AK THE
N 370
560 GOTO 710
570 N=0:Z=0
580 GET #1,A:IF A=155 OR A=44 OR
 A=32 THEN 670
590 IF A<32 THEN N=-A:RETURN
600 IF A<>126 THEN 630
610 GOSUB 690:IF I=1 AND T=44 TH
EN N=-1:? CHR$(126);:GOTO 690
620 GOTO 570
630 IF A<48 OR A>57 THEN 580
640 ? CHR$(A+RF);:N=N*10+A-48
650 IF N>HK2 THEN ? CHR$(253);:A
=126:GOTO 600
660 Z=Z+1:IF Z<HK1 THEN 580
670 IF Z=0 THEN ? CHR$(253);:GOT
O 570
680 ? ",";:RETURN
690 POKE 752,1:FOR I=1 TO HK1:?
CHR$(30);:GET #6,T:IF T<>44 AND
T<>58 THEN ? CHR$(A);:NEXT I
700 POKE 752,0:? " ";CHR$(126);:
RETURN
710 GRAPHICS 0:POKE 710,26:POKE
712,26:POKE 709,2
720 IF RZB=76 THEN 1500
725 IF NOS=84 THEN 890:REM TASMA
740 IF CZYT THEN ? :? "Ladowanie
 zbioru dyskowego ":?
750 IF RZB=68 THEN 1050 REM BOOT
 DYSK
760 ? :? "Podaj nazwe zbioru ";
INPUT T$
770 F$=T$:IF LEN(T$)>2 THEN IF T
$(1,2)<>"D:" THEN F$="D:":F$(3)=
T$
780 TRAP 870:CLOSE #2:OPEN #2,8-
4*CZYT,0,F$
790 IF CZYT THEN FOR I=1 TO 6:GE
T #2,A:NEXT I:GOTO 820
800 PUT #2,255:PUT #2,255
810 H=INT(AP/256):L=AP-H*256:PUT
 #2,L:PUT #2,H:H=INT(AK/256):L=A
K-H*256:PUT #2,L:PUT #2,H
820 GOSUB CZZB
830 IF STA=0 OR CZYT THEN 850
840 PUT #2,224:PUT #2,2:PUT #2,2
25:PUT #2,2:H=INT(STA/256):L=STA-
H*256:PUT #2,L:PUT #2,H
850 TRAP 40000:CLOSE #2:IF CZYT
THEN CZYT=0:? :? :GOTO 360
860 END
870 ? "Blad nr ";PEEK(195);" pod
czas dostepu do zbioru":? F$:CLO
SE #2:? :GOTO 360

890 IF CZYT THEN ? :? "Czytanie
tasmy "
900 ? :? :? "Wloz i ustaw tasme"
 ? "nacisnij PLAY ";:IF  NOT CZY
T THEN ? " & RECORD"
920 TRAP 960:CLOSE #2:OPEN #2,8-
4*CZYT,128,"C:"
930 GOSUB CZZB:IF PEEK(195)>1 TH
EN 960
940 CLOSE #2:TRAP 40000:IF CZYT
THEN CZYT=0:GOTO 360
950 END
960 ? "Blad nr ";PEEK(195);" prz
y czytaniu/zapisie tasmy":CLOSE
#2:GOTO 890
1050 IF CZYT THEN 1100
1060 ? :? "Formatowac dyskietke
?  (T/N)"
1070 GET #1,A:IF A<>84 AND A<>78
 THEN 1070
1080 ? CHR$(A):IF A=78 THEN 1100
1090 ? :? "Formatowanie ...":XIO
 254,#2,0,0,"D:"
1100 NR=INT((AK-AP+127)/128):BUF
$(AK-AP+2)=CHR$(0):IF CZYT THEN
? "Czytanie dysku...":GOTO 1120
1110 ? "Zapis zbioru na dysk..."
1120 FOR I=1 TO NR:NS=I
1130 IF CZYT THEN GOSUB CZSEK:BU
F$(I*128-127)=SEK$:GOTO 1160
1140 SEK$=BUF$(I*128-127)
1150 GOSUB CZSEK
1160 IF PEEK(DCB+3)<>1 THEN 1200
1170 NEXT I
1180 IF  NOT CZYT THEN END
1190 ? :? :CZYT=0:GOTO 360
1200 ? "Blad przy dostepie do zb
ioru typu BOOT DISC":? "Moze nie
sformatowana dyskietka ? ":GOTO
1050
1490 REM =====================
=====
1491 REM
1492 REM MODUL TWORZY Z WPROWADZ
ONYCH
1493 REM DANYCH LINIE DATA PR. B
ASIC
1494 REM I ZAPISUJE JE NA TASME
LUB
1495 REM DYSK
1496 REM

1497 REM =====================
=====
1500 IF CZYT=1 THEN ? :? "Zbior
typu DATA wprowadz instr. ENTER"
:FOR J=1 TO 300:NEXT J:CZYT=0:GO
TO 370

1505 ? :? "Podaj poczatkowy nume
r lini DATA ":INPUT NRL
1510 ? "Podaj krok numeracji ";:
INPUT KROK
1520 IF NOS=68 THEN 1600:REM DYS
K
1525 REM TASMA  TRAP 1800
1530 CLOSE #2:OPEN #2,8,0,"C:":G
OTO 1650
1600 ? "Podaj nazwe zbioru ";:IN
PUT F$
1610 IF F$(1,2)<>"D:" THEN T$(1,
2)="D:":T$(3)=F$:GOTO 1630
1620 T$=F$:TRAP 1800
1630 CLOSE #2:OPEN #2,8,0,T$
1650 FOR I=AP TO AK STEP LW
1655 FOR J=1 TO 114:LINIA$(J)=CH
R$(32):NEXT J

1660 J=LEN(STR$(NRL))
1665 LINIA$(1,J)=STR$(NRL):J=J+2
:LINIA$(J)="DATA":J=J+6
1670 FOR K=0 TO LW-1
1675 N=PEEK(ADR(BUF$)+I+K-AP):KO
D$="000":KOD$(4-LEN(STR$(N)))=ST
R$(N)
1680 LINIA$(J)=KOD$:J=J+3
1685 IF J>112 THEN ? "Przekroczo
na dop.dl. linii DATA - 114zn":E
ND
1690 IF K<LW-1 THEN LINIA$(J)=C
HR$(44):J=J+1
1695 NEXT K
1700 LINIA$(J)=CHR$(155):PRINT #
2,LINIA$:NRL=NRL+KROK
1705 NEXT I
1710 END
1800 ? :? " Blad nr ";PEEK(195);
" podczas zapisywania linii  DA
TA":GOTO 1520
1993 REM =====================
=====
1994 REM MODUL ZAPISU/ODCZYTU SE
KT.
1995 REM NR SEKTORA W ZM. NS
1996 REM SKOK DO SIOV $E453
1997 REM DIM SIOV$(4)
1998 REM
1999 REM =====================
=====
2000 DCB=3*256:POKE DCB+1,1
2004 A=ADR(SEK$):H=INT(A/256):L=
A-H*256
2006 POKE DCB+5,H:POKE DCB+4,L:P
OKE DCB+2,87-5*CZYT
2008 H=INT(NS/256):L=NS-H*256
2010 POKE DCB+11,H:POKE DCB+10,L
2012 A=USR(ADR(SIOV$))
2014 RETURN
2090 REM =====================
=====

2091 REM
2092 REM MODUL ODCZYTUJE/ZAPISUJ
E
2093 REM ZBIOR Z TASMY LUB DYSKU
2094 REM SKOK DO CIOV = $E456 IO
CB #2
2095 REM
2096 REM =====================
=====
2100 IOCB=864
2102 H=INT(ADR(BUF$)/256):L=ADR(
BUF$)-H*256
2104 POKE IOCB+4,L:POKE IOCB+5,H
2106 L=AK-AP+1:H=INT(L/256):L=L-
H*256
2108 POKE IOCB+8,L:POKE IOCB+9,H
2110 POKE IOCB+2,11-4*CZYT
2112 A=USR(ADR(CIOV$))
2114 RETURN

2150 REM =====================
=====
2151 REM
2152 REM MODUL WPROWADZA DANE ST
ERUJ.
2153 REM PRZEBIEGIEM PROGRAMU
2154 REM LW liczba danych w wier
szu
2155 REM AP adres poczatkowy

2156 REM AK adres koncowy
2157 REM NOS nosnik,T-tasma,D-dy
sk
2158 REM RZB rodzaj zbioru,B-bin
arny,
2159 REM D-boot dysk,L-linie dat
a
2160 REM STA adres startowy

2161 REM =====================
=====
2200 ? "Liczba danych w wierszu:
";:INPUT LW
2201 DIM K1$(7),K2$(6),K3$(9),K4
$(11),K5$(17),K6$(5),K7$(29),K8$
(31),K9$(26)
2202 K1$="T-tasma":K2$="D-dysk"
2203 K3$="B-binarny":K4$="D-typu
 boot":K5$="L-zaw. linie DATA":K
6$="(T/N)"
2204 K7$="0 - sk.=suma danych w
wierszu":K8$="1 - sk.=reszta z d
zielenia sumy"
2205 K9$="danych w wierszu przez
 256"
2210 POSITION 2,2:? "Rodzaj nosn
ika (";K1$;",";K2$;")"
2220 GET #1,NOS:IF NOS<>84 AND N
OS<>68 THEN 2220
2222 IF NOS=84 THEN FOR I=1 TO L
EN(K1$):K1$(I,I)=CHR$(ASC(K1$(I,
I))+128):NEXT I

2224 IF NOS=68 THEN FOR I=1 TO L
EN(K2$):K2$(I,I)=CHR$(ASC(K2$(I,
I))+128):NEXT I
2226 POSITION 2,2:? "Rodzaj nosn
ika (";K1$;",";K2$;")"
2230 POSITION 2,4:? "Rodzaj zbio
ru ?":POSITION 10,5:? K3$:POSITI
ON 10,6:? K4$:POSITION 10,7:? K5
$
2240 GET #1,RZB:IF RZB<>66 AND R
```

```
Z8<>68 AND RZB<>76 THEN 2240
2245 POSITION 2,9
2250 IF RZB=66 AND NOS=84 THEN ?
 "Na tasmie mozna zapisac tylko
zbior":? "typu boot tape lub lin
ie data"
2255 IF RZB=66 AND NOS=84 THEN G
OSUB 2500
2260 IF RZB=66 AND NOS=84 THEN 2
230
2262 IF RZB=66 THEN FOR I=1 TO L
EN(K3$):K3$(I,I)=CHR$(ASC(K3$(I,
I))+128):NEXT I:POSITION 10,5:?
K3$
2264 IF RZB=68 THEN FOR I=1 TO L
EN(K4$):K4$(I,I)=CHR$(ASC(K4$(I,
I))+128):NEXT I:POSITION 10,6:?
K4$
2266 IF RZB=76 THEN FOR I=1 TO L
EN(K5$):K5$(I,I)=CHR$(ASC(K5$(I,
I))+128):NEXT I:POSITION 10,7:?
K5$
2270 IF RZB=76 THEN 2310
2280 POSITION 2,9:? "Adres start
owy ";:INPUT STA
2310 ? "Adres poczatkowy ";:INPU
T AP
2320 ? "Adres koncowy    ";:INPU
T AK
2330 POSITION 2,15:? "Czy beda s
umy kontrolne ? ":POSITION 30,15
:? K6$:SUMK=0
2340 GET #1,A:IF A<>84 AND A<>78
 THEN 2340
2350 IF A=84 THEN SUMK=1:K6$(2)=
CHR$(212):POSITION 30,15:? K6$
2352 IF A=78 THEN K6$(4)=CHR$(20
6):POSITION 30,15:? K6$:GOTO 242
0
2360 ? "Sposob obliczania sum ko
ntrolnych":RSK=0
2370 POSITION 2,17:? K7$
2380 POSITION 2,18:? K8$
2390 POSITION 6,19:? K9$
2400 GET #1,A:IF A<>48 AND A<>49
 THEN 2400
2410 IF A=49 THEN FOR I=1 TO LEN
(K8$):K8$(I,I)=CHR$(ASC(K8$(I,I)
)+128):NEXT I:POSITION 2,18:? K8
$
2412 IF A=49 THEN FOR I=1 TO LEN
(K9$):K9$(I,I)=CHR$(ASC(K9$(I,I)
)+128):NEXT I:POSITION 6,19:? K9
$
2414 IF A=48 THEN FOR I=1 TO LEN
(K7$):K7$(I,I)=CHR$(ASC(K7$(I,I)
)+128):NEXT I:POSITION 2,17:? K7
$
2416 IF A=49 THEN RSK=1
2418 FOR I=1 TO 300:NEXT I
2420 RETURN
2500 FOR I=1 TO 350:NEXT I:POSIT
ION 2,9:? "
                         "
2502 ? "
                         "
2504 RETURN
```

Gdy wprowadzany kod składa się z liczb trzycyfrowych (można używać zer prowadzących tzn. zamiast 1 wprowadzać 001) edytor sam drukuje oddzielające je przecinki. Gdy wprowadzane są liczby jedno- lub dwucyfrowe, to po wciśnięciu jednego z trzech klawiszy: spacji, przecinka lub RETURN, edytor wyprowadzi przecinek i jest gotowy do przyjęcia następnej liczby. Po wprowadzeniu wszystkich danych z jednej linii edytor emituje dźwięk (buczenie) sygnalizujące wprowadzanie sumy kontrolnej. Suma kontrolna może się składać maksymalnie z czterech cyfr. Po jej wprowadzeniu edytor wyświetla ją automatycznie w inwersie wideo i porównuje ją z sumą kontrolną obliczoną na podstawie wprowadzanych w linii danych. Jeżeli obie są równe, edytor emituje krótki dźwięk i jest gotowy do wprowadzenia następnej linii kodu. Gdy sumy kontrolne, obliczona przez edytor i wprowadzona, nie są równe, edytor sygnalizuje konieczność ponownego wprowadzenia linii. Korektę wprowadzonego kodu, przed podaniem sumy kontrolnej, można dokonać, kasując określone liczby kodu klawiszem DELETE/BACK/ SPACE i wpisując je ponownie.

Edytor wykonuje cztery komendy, przeznaczone do zapisu i odczytu wprowadzonego kodu na taśmę lub dysk oraz przydatne przy wprowadzaniu kodu z długiego listingu. Komendy te można wydać w dowolnym momencie pracy z edytorem, wciskając jednocześnie następujące klawisze:

CONTROL — S — zapisuje wprowadzony kod maszynowy na taśmę lub dysk, tworząc zbiór określony opcją B, D lub L,

CONTROL — L — wprowadza do pamięci mikrokomputera kod maszynowy z taśmy lub dysku,

CONTROL — D — wyświetla na ekranie monitora kod maszynowy, aktualnie wprowadzany lub wczytany z urządzenia zewnętrznego,

CONTROL — N — komenda ta jest przydatna do wprowadzania kodu maszynowego w kilku etapach. Na przykład, chcemy wprowadzić program w kodzie maszynowym, o adresie początkowym równym 20000 i adres końcowy równy 30000. Rozpoczynamy pracę z edytorem i po wprowadzeniu 1000 bajtów kodu chcemy ją przerwać. W tej sytuacji komendą CONTROL—S zapisujemy wprowadzony kod na taśmę lub dysk i zapamiętujemy adres, na którym przerwaliśmy wprowadzanie. Po pewnym czasie, gdy będziemy chcieli kontynuować wprowadzanie, komendą CONTROL—L wczytujemy wcześniej wprowadzony, zapisany na taśmie lub dysku, kod do pamięci i komendą CONTROL—N podajemy edytorowi adres, od którego chcemy kontynuować wprowadzanie — w tym przykładzie jest to adres 21000.

Program MCEDYTOR znacznie usprawnia wprowadzanie kodu maszynowego z dowolnych listingów do pamięci mikrokomputera. Kontrolując wprowadzanie liczby kodu, odciąża użytkownika, skracając przez to czas wprowadzania. Ponadto gwarantuje, że wprowadzony kod będzie pozbawiony błędów.

H. KRASUSKI

* SUMA KONTROLNA / ETYKIETA *

YE	50	SZ	52	WZ	54	VY	56
RP	58	RG	60	XK	100	YZ	110
CE	120	RK	130	PG	140	HZ	142
ZG	160	JQ	170	FW	171	BC	172
OB	173	DK	174	BL	175	GL	176
UE	190	OT	200	EG	210	DJ	220
EJ	225	PZ	230	NU	240	BR	250
YV	260	FV	261	BB	262	OA	263
WR	264	BK	265	GK	266	SU	290
OU	300	DY	310	HG	320	AD	330
LQ	340	IY	350	AQ	360	ON	370
RM	380	GX	390	AH	400	JU	410
QA	420	LH	430	MI	440	FN	450
RI	452	EM	460	CL	470	ZN	480
PT	490	KN	500	CX	502	GA	504
GN	506	MI	510	LO	520	DH	530
FA	540	AO	550	OP	560	VN	570
VA	580	RD	590	AI	600	XD	610
QC	620	KC	630	BB	640	OD	650
QG	660	JD	670	SH	680	RV	690
RG	700	SM	710	NH	720	NC	725
JF	740	TK	750	NC	760	GY	770
UE	780	SX	790	EI	800	GK	810
WL	820	EI	830	WE	840	PI	850
YF	860	KP	870	NU	890	ZE	900
UY	920	GH	930	IS	940	YE	950
PX	960	EH	1050	OZ	1060	RH	1070
PR	1080	YD	1090	IJ	1100	WE	1110
TC	1120	BC	1130	OS	1140	ZL	1150
AE	1160	FM	1170	RB	1180	XM	1190
YD	1200	EB	1490	UL	1491	IN	1492
TS	1493	QK	1494	ER	1495	VF	1496
FD	1497	WP	1500	WP	1505	BD	1510
ZG	1520	YD	1525	RR	1530	XN	1600
IR	1610	RG	1620	XB	1630	RC	1650
JW	1655	FI	1660	MG	1665	PM	1670
AR	1675	SW	1680	YK	1685	YT	1690
HS	1695	SK	1700	FX	1705	QG	1710
CP	1800	EX	1993	LA	1994	NS	1995
YV	1996	HG	1997	VX	1998	FV	1999
WX	2000	WR	2004	LX	2006	JE	2008
FT	2010	EZ	2012	PN	2014	DU	2090
UE	2091	XH	2092	RW	2093	YY	2094
UU	2095	ES	2096	WL	2100	RB	2102
MU	2104	YK	2106	UM	2108	GL	2110
VF	2112	PP	2114	DK	2150	TU	2151
CV	2152	EV	2153	CV	2154	CS	2155
OG	2156	MS	2157	BJ	2158	JO	2159
FW	2160	DR	2161	KE	2200	JS	2201
CY	2202	QR	2203	PR	2204	SO	2205
KF	2210	SS	2220	NH	2222	UC	2224
LG	2226	JR	2230	SE	2240	BS	2245
XL	2250	AB	2255	EB	2260	SP	2262
GR	2264	TC	2266	OR	2270	EF	2280
HK	2310	UW	2320	FH	2330	SP	2340
SC	2350	XN	2352	FR	2360	KG	2370
LY	2380	PY	2390	LG	2400	BJ	2410
BN	2412	PF	2414	NO	2416	XX	2418
PI	2420	GN	2500	SA	2502	PU	2504

Liga Myślących

Zadanie 1

W koszyku znajduje się 30 jajek w trzech gatunkach. Jajek drugiego gatunku jest dwa razy więcej niż pierwszego, a jajek trzeciego gatunku jest trzy razy mniej niż pierwszego gatunku. Jakie jest prawdopodobieństwo, że losowo wyciągnięte z koszyka jajko będzie drugiego gatunku?

Zadanie 2

W gospodarstwie rolnym w 1983 roku zebrano pszenicę z 36 ha. Na rok 1984 zaplanowano wzrost wydajności plonów z 1 ha o 8 proc., wzrost całego zbioru pszenicy o 20 proc. w stosunku do roku 1983. O ile hektarów trzeba zwiększyć obszar uprawy pszenicy, aby wykonać ten plan?

Zadanie 3

Należy znaleźć taką liczbę dwucyfrową, żeby jej suma cyfr wynosiła 9 i żeby po przestawieniu jej cyfr otrzymać:
1) liczbę mniejszą od połowy szukanej liczby,
2) liczbę większą od połowy szukanej liczby.
Rozwiązując zadanie znajdźcie wszystkie liczby w każdym z przypadków.

Zadanie 4

Dwie maszyny pracujące z dwóch stron tunelu powinny zakończyć przekop w ciągu 60 dni. Jeżeli pierwsza maszyna wykona 30 proc. swej pracy, a druga $26\frac{2}{3}$ proc. swej pracy, to obie wykopią 60 m tunelu. Gdyby zaś pierwsza maszyna wykonała $\frac{2}{3}$ całej pracy drugiej maszyny, a druga 0,3 całej pracy pierwszej maszyny, to pierwsza maszyna pracowałaby o 6 dni dłużej niż druga. Wyznaczyć o ile metrów dziennie przesuwa się każda maszyna?

Zadanie 5

Przy badaniu ekonomiczności dwóch silników spalinowych o jednakowej mocy stwierdzono, że jeden z nich zużył 600 g benzyny, a drugi pracując o 2 godziny krócej, zużył 384 g benzyny. Gdyby pierwszy silnik zużywał na godzinę tyle benzyny, ile drugi, a drugi tyle ile pierwszy, to przy tym samym czasie pracy każdego z nich, co poprzednio, zużycie benzyny w obu silnikach byłoby jednakowe.
Ile benzyny na godzinę zużywa każdy silnik?

Rozwiązania zadań prosimy przesyłać do redakcji do końca stycznia 1988 r. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe cenne nagrody-niespodzianki.

Mikrokomputery IBM PC (cz. 6)

Charakterystyka oprogramowania

Szybkość mikroprocesorów INTEL 8088 i 80286 oraz współpracujących z nimi koprocesorów arytmetycznych, pojemność i czas dostępu do pamięci zapisywalnej oraz pamięci zewnętrznych (dyskietki, dyski stałe), a także rozdzielczość kart graficznych (CGA, EGA, HERCULES, MA) pozwalają wykorzystywać mikrokomputery IBM PC i kompatybilne do różnych celów. Wśród zastosowań można wymienić:
— małe i średnie systemy wspomagania projektowania (CAD/-CAE);
— stacje graficzne systemów komputerowych;
— systemy pomiarowe, sterowania i uruchamiania;
— systemy biurowe, a wśród nich: programy gromadzenia danych, programy analizy statystycznej, programy finansowe, edytory (procesory) tekstów;
— systemy medyczne np. intensywnej terapii;
— systemy edukacyjne.

Mikrokomputery wykorzystywane są przez dyrektorów i sekretarki, urzędników i nauczycieli, artystów i lekarzy, inżynierów mechaników, elektroników i architektów. Niedoskonałości rozwiązań konstrukcyjnych mikrokomputerów PC rekompensowane są przez oprogramowanie, którego wybór zadowolić może większość użytkowników.

Przypomnijmy, wśród **systemów operacyjnych** podstawowym jest DOS, popularnie wykorzystywany. Użytkownik mający dysk stały może założyć na nim kilka partycji, każda z innym systemem operacyjnym. Mogą to być:
— CONCURRENT DOS, wielozadaniowy okienkowy system operacyjny, umożliwiający równoczesne wykonywanie do czterech zadań, każde w oddzielnym oknie:
— CP/M-86, system, który powstał w wyniku rozwinięcia procesorów 8-bitowych;
— CONCURRENT CP/M-86, wielozadaniowy i wielostanowiskowy system, umożliwiający wprowadzenie do 16 terminali logicznych i obsługę do 16 stacji dysków;
— XENIX-286, system umożliwiający pracę wieloprogramową i wielodostępną, wzorowany na systemie UNIX, a przeznaczony dla IMB PC/AT.

Olbrzymia grupa oprogramowania to **języki programowania**, im kompilatory, interpretery, biblioteki z różnymi procedurami, np. graficznymi, translatory itp.

Język **ADA** reprezentowany jest przez Artek ADA Compiter firmy Artek Co. i Janusz ADA D-Pack, pakiet m.in. z kompilatorem asemblerem, programem łączącym, optymalizującym firmy RR Software. Ta ostatnia firma oferuje również translator z języka Pascal na język ADA.

Dla użytkowników piszących programy w **kodzie maszynowym** przygotowano wiele propozycji, wśród których najciekawszą jest chyba MACRO ASSEMBLER-86 firmy MicroSoft w.4.0, umożliwiający translację programów w kodzie maszynowym procesorów INTEL 8088 lub 8086 na relokowalny kod maszynowy oraz dopuszczający używanie makroinstrukcji i pseudoinstrukcji.

Język **BASIC**, bardzo popularny, zwłaszcza wśród początkujących, chyba najlepiej reprezentowany jest przez:
— BASIC Compiler firmy MicroSoft, odznaczający się podwójną precyzją, wyposażony w łańcuchy i bloki „common", tworzący po dołączeniu biblioteki programy przeznaczone do bezpośredniego wykonywania (typu. EXE);
— Turbo BASIC firmy Borland, tworzący programy typu EXE, z własnym edytorem, programem łączącym i bibliotekę, umożliwiający obsługę karty EGA i koprocesorów 8087/80287;
— Better BASIC, kompilator firmy Summit Software.

Język **C** na mikrokomputerach PC, obecnie najpopularniejszy, ze względu na swoje cechy, język programowania, dostępny jest:
— kompilatorem MS C w 4.00 firmy MicroSoft, który ma pięć modeli pamięci, możliwość łączenia różnych modeli w pojedynczym programie, bogatą bibliotekę (200 funkcji), kompatybilny jest z XENIX 286 oraz możliwość łączenia programów napisanych w MSC z napisanymi w MS FORTRAN-ie, MS Pascalu, MS MACRO ASSEMBLER-ze;
— kompilatorem Aztec C86-c firmy Manx, będącym standardem w systemie operacyjnym UNIX i uważanym za najlepszy kompilator języka C;
— najnowszą propozycją firmy Borland Turbo C, będącym najszybszym chyba kompilatorem;
— innymi popularnymi kompilatorami: Lattice C, Mark Williams C, Wizard C;
— interpreterem Run/C, realizującym całkowicie standard języka i będącym wprowadzeniem do programowania.

Wielu projektantów systemów mikroprocesorowych wykorzystuje **cross — assemblery**, które na mikrokomputerach PC dostępne są dla mikroprocesorów:
— ZILOG: Z8, Z80, Z8000;
— MOTOROLA: 6809, 6804, 68000;
— INTEL: 8080, 8085, 8089, 8049/41/51, 8096;

— MOS TECHNOLOGY: 6501, 6502.

Dla użytkowników języka **COBOL** dostępne są kompilatory: MBP Cobol firmy MBP Software, MS COBOL-86 firmy MicroSoft, RM/COBOL-86 firmy Ryan McFarland pracujący pod CM/M-86.

Język **FORTH** ma dwóch ciekawych reprezentantów, są to: HS/-FORTH firmy Harvard Softworks i PC/FORTH + firmy Lab.Micro.

Do dziś bardzo popularnym językiem programowania jest **FORTRAN**, wśród jego kompilatorów na uwagę zasługują: MS FORTRAN w.4.00, RM/FORTRAN oraz FORTRAN-77 firmy Softech Microsystems.

Język **LIST** najlepiej reprezentowany jest przez pakiet GC LIST firmy Gold Hill Computers, a język **LOGO** — przez PC LOGO, firmowany przez Harvard Assoc.

MODULA-2 język do programowania strukturalnego, z możliwością tworzenia niezależnych modułów programowych i własnych typów danych, dostępny pakietami: LOGITECH MODULA-2/86 PLUS, MODULA II BY VOLITION SYSTEMS i M2SDS firmy Interface Tech. Corp.

Język **PASCAL** reprezentowany jest przez: kompilator firmy MicroSoft, kompilujący moduły do 64 kB i łączący programy o długości do 1 MB; pakiet Pascal MT+, zawierający m.in.: kompilator, asembler, konsolidator, program uruchamiania (debugger) i bibliotekę procedur pomocniczych oraz kompilator firmy Borland Turbo Pascal, łącznie z biblioteką procedur graficznych Turbo Graphix, najpopularniejszy i powszechnie wykorzystywany. Kompilator Turbo Pascal ma możliwość swobodnego dostępu do pamięci operacyjnej i portów we/wy i może współpracować z koprocesorem arytmetycznym. Zbiór procedur graficznych Turbo Graphix może być wykorzystywany przez użytkowników do tworzenia własnych programów graficznych i obsługuje zarówno kolorową kartę graficzną, jak i monochromatyczną. Początkujący użytkownicy mogą korzystać z programu ALICE.

Najlepszym reprezentantem języka **PROLOG** jest również pakiet firmy Borland o nazwie Turbo Prolog.

Wśród języków symulacyjnych dotychczas jedynym reprezentantem jest GPSS.

Kolejną grupą programów są **pakiety zintegrowane i programy finansowe**, wśród których najpopularniejszymi są FRAMEWORK, LOTUS 1-2-3, SYMPHONY, OPENACCESS, MULTIPLAN, SUPERCALC, VISICALC.

FRAMEWORK jest pakietem zawierającym arkusz elektroniczny, bazę danych, edytor tekstowy, graficzne zobrazowanie danych oraz program komunikacyjny.

LOTUS 1-2-3 zawiera arkusz elektroniczny o wymiarach 2048 wierszy i 256 kolumn, do 240 znaków w komórce, umożliwiający graficzne zobrazowanie danych, automatyczną rekalkulację, makroinstrukcje i różnorodne operacje na tabelach.

SYMPHONY jest pakietem zintegrowanym zawierającym arkusz elektroniczny (8192 wiersze i 256 kolumn), bazę danych, graficzne zobrazowanie danych, edytor tekstów i moduł służący do komunikacji.

OPENACCESS to pakiet oparty na relacyjnej bazie danych, zawierającej do 30 000 rekordów, arkusz elektroniczny, edytor tekstów, trójwymiarowe zobrazowanie danych oraz moduł komunikacyjny.

MULTIPLAN to program zawierający arkusz elektroniczny o wymiarze 255 wierszy i 63 kolumny, umożliwiający przechowywanie danych oraz automatyczną rekalkulację i prognozowanie.

SUPERCALC zawiera arkusz elektroniczny o wymiarze 254 wiersze i 63 kolumny. Ekran monitora może być dzielony na 2 okienka do przeglądania 2 tabel. Program umożliwia wykonywanie operacji na tabelach i automatyczną rekalkulację. Podobne możliwości ma program VISICALC.

W grupie programów **bazy danych** można wymienić: DBASE III, i DBASE III Plus, RBASE 4000 i 5000, dB VISTA.

DBASE III jest systemem zarządzania relacyjną bazą danych, umożliwiającym pisanie programów do zautomatyzowanej obsługi danych. Baza może zawierać 1 miliard rekordów, a rekord do 128 pól pięciu typów. System umożliwia pracę z 15 zbiorami jednocześnie. Do kompilacji komend wykonywanych w trybie interpretacyjnym może być wykorzystany CLIPPER dB III Compiler.

RBASE 4000 jest to system obsługi relacyjnych baz danych zawierających do 100 miliardów rekordów, przy 2,5 miliardzie rekordów w zbiorze. Umożliwia wykonywanie wszelkich operacji na zbiorach i rekordach.

Wśród edytorów dużą popularnością cieszą się SPF/PC i IBM Professional EDIT. A wśród edytorów (procesorów) tekstów WORDSTAR, WORDSTAR 2000, MS WORD, CHI WRITER.

WORDSTAR 2000 umożliwia redakcję i korektę tekstów, zapamiętywanie zdań, definiowanie formatu drukowanego tekstu, sprawdzanie poprawności pisowni, automatyczną obsługę korespondencji i przesyłanie zbiorów.

CHI WRITER jest edytorem pracującym na stronie graficznej, ma 18 typów klawiatur (wersja 2.04), m.in. z literami polskimi. Umożliwia definiowanie własnych znaków.

Kolejna grupa programów to programy komunikacyjne, transmitujące i konwersji. Wśród nich można wymienić CROSSTALK XVI, pakiet komunikacyjny do emulowania terminali; programy konwersji dBASE II/III to C, PASCAL — to — MODULA-2.

Następna grupa programów to programy narzędziowe systemowe, umożliwiające wszelkie operacje na zbiorach, np.: kopiowanie, usuwanie, porządkowanie, zmiana atrybutów; odzyskiwanie utraconych zbiorów itp. Można tu wymienić: XTREE, PC TOOLS, NORTON UTILITIES, NORTON COMMANDER, NORTON INTEGRATOR, EXPLORER.

Dodatkiem do kompilatorów są biblioteki z procedurami matematycznymi i graficznymi. Wśród bibliotek matematycznych do języka FORTRAN popularne są: SSP, SANDIA i IMSL. Inną powszechnie wykorzystywaną, wymienioną wcześniej biblioteką jest Turbo GRAPHIX, zawierająca procedury graficzne do kompilatora firmy Borland Turbo PASCAL. Umożliwiają one rysowanie podstawowych elementów geometrycznych (linie, punkty, okręgi, elipsy), umieszczenie ich w przesuwalnych oknach, rysowanie obiektów złożonych, np. histogramów. Procedury umożliwiają łatwą manipulację obiektami, oknami. Innym przykładem jest pakiet STATGRAPHICS (Statistical Graphics System), uzupełniający działanie programów dBASE III, LOTUS 1-2-3, SYMPHONY. Pakiet zawiera 255 procedur z zakresu analizy wariancji, analizy regresji, analizy danych eksperymentalnych oraz analizy szeregów czasowych. Istotną cechą pakietu jest zobrazowanie wyników analizy, w formie atrakcyjnych dwu- lub trójwymiarowych kolorowych wykresów. W układzie współrzędnych (X, Y) można przedstawić punkty, linie, histogramy, krzywe. W układzie współrzędnych (X, Y, Z) można przedstawić punkty, linie, krzywe, powierzchnie. Pakiet współrzędnych z 5 typami ploterów, umożliwia wykonanie rysunku w 8 kolorach.

Projektowanie wspomagane komputerem, czyli zastosowanie komputera do prac projektowych, wykorzystywane przez mechaników, elektroników, architektów i artystów obejmuje także przetwarzanie danych geometrycznych (graficznych).

Możliwości oprogramowania z grupy CAD (ang. Computer Aided Desing — projektowanie wspomagane komputerem) i CAE (ang. Computer Aided Enginnering — inżynieria wspomagana komputerem) zależą od budowy programu oraz struktury danych.

Projektant używając np. edytora rysunków systemu dwuwymiarowego, wykorzystując podstawowe elementy geometryczne (punkt, prosta, łuk, okrąg) lub figury (obiekty) zdefiniowane i wcześniej umieszczone w bibliotece graficznej, edytuje i kopiuje rysunki, zmienia skale i wymiaruje, nasuwa fragmenty rysunku, łączy rysunki wywoływane z biblioteki. Efektywność działania z programem zależy od wielkości zbioru poleceń edytora rysunku.

Kolejnym ważnym składnikiem systemu jest sprzęt. Korzystanie z systemu wspomagania projektowania na minikomputerze jest bardzo kosztowne, a wykorzystywanie całego ośrodka obliczeniowego wiąże się z ograniczeniem dostępu projektantów do systemu. Natomiast system mikrokomputerowy w następującej konfiguracji sprzętowej: mikroprocesor 80 286, koprocesor arytmetyczny 80 287, dysk stały 40 MB, stacje dysków elastycznych 1,2 MB, karta graficzna EGA, pamięć operacyjna 4 MB, monitor jednostki centralnej oraz monitor graficzny wysokiej rozdzielczości, digitizer z myszką, ploter o formacie A3 (B) niejednokrotnie wystarcza. Plotery o większych formatach zapewniają lepszą rozdzielczość, ale są drogie.

Najbardziej popularnym systemem grafiki dwuwymiarowej jest program AutoCAD, umożliwiający narysowanie praktycznie każdego obiektu i opisanie go.

Rozdzielczość systemu zależy od wykorzystywanego sprzętu, tzn. od digitizerów wprowadzających dane i ploterów, na które wyprowadzane są rysunki, nie jest wewnętrzną rozdzielczością AutoCAD-a. Wpływ na rozdzielczość mają również karta graficzna komputera oraz zastosowanie monitora wysokiej rozdzielczości. Program nie narzuca formatu rysunku, może być to A0.

Rysunek wykonywany za pomocą AutoCAD-u to zbiór informacji, które opisują obraz graficzny. Może mieć dowolny wymiar, może być narysowany w dowolnych jednostkach miary: calach, stopach, milimetrach, angstremach.

Praca systemu może być prowadzona w trybie konwersacyjnym lub w trybie wyboru komendy z zestawu komend (menu).

Ekran monitora, w razie wykorzystywania go również do edycji rysunków, dzieli się na trzy obszary: komunikatów, zestawu komend, edycji rysunku.

Edytowany rysunek tworzony jest z podstawowych elementów graficznych: punktów, linii, łuków, okręgów, tekstów. Z elementów tych można budować obiekty złożone i manipulować nimi, powiększyć fragment lub szczegół edytowanego rysunku. Każdemu punktowi na ekranie odpowiada punkt w układzie współrzędnych (X, Y).

Program umożliwia definiowanie warstw rysunku, przypisanie do warstwy koloru i typu linii, powiększenie lub zmniejszenie rysunku (fragmentu). Możliwe jest tworzenie własnego zestawu komend z przygotowanych rysunków.

Do pakietu podstawowego ACAD dołączone są moduły zapewniające: półautomatyczne wymiarowanie, szkicowanie odręczne, trójwymiarowy układ współrzędnych.

Podczas edytowania rysunków można wykorzystać bibliotekę z elementami (np. elektronicznymi, mechanicznymi).

RoboCAD jest systemem, który może pracować na wielu typach mikrokomputerów, między innymi IBM PC, APPLE.

System ma w porównaniu z AutoCAD-em wiele zalet. Oto niektóre z nich: edytor rysunków zaopatrzony jest w dużo większy zbiór poleceń, jest efektywniejszy w użyciu. Biblioteką graficzną łatwo się posługiwać, do wywołania lub zapisania do niej elementu czy rysunku wystarczy jeden ruch. Zmiana jednostek miary obiektu zapisanego wcześniej w bibliotece i wstawienie go w inny obiekt, usuwanie fragmentów rysunku, czyli czynności decydujące o szybkości systemu graficznego — są w programie RoboCAD rozwiązane bardzo efektywnie. Program ma automatyczne wymiarowanie oraz podręczny edytor tekstów i bazę danych.

Grafika komputerowa jest integralną częścią programów z grupy CAE. Programy te umożliwiają np. konstruowanie, analizę i symulację działania układów elektronicznych oraz wykonanie płytek drukowanych układów. Powyższe czynności można wykonać za pomocą specjalizowanych pakietów lub współpracujących ze sobą programów.

PCAD jest systemem komputerowego wspomagania projektowania płytek drukowanych wielowarstwowych i tworzenia dokumentacji technicznej.

System PCAD przeznaczony jest do: wprowadzania schematów ideowych układów elektronicznych, analizy i symulacji układów elektronicznych, automatycznego lub półautomatycznego projektowania płytek drukowanych oraz sporządzania dokumentacji technicznej, obejmującej kliszę ścieżek, rysunku montażowego, schematu ideowego układu, maski lutowniczej, wykazu użytych elementów.

W skład biblioteki programu wchodzi około 400 układów TTL, około 100 elementów dyskretnych, około 200 układów liniowych, około 160 układów CMOS, około 160 układów dużej skali integracji firmy INTEL.

Istnieje wiele mniejszych programów przeznaczonych do projektowania analizy i symulacji działania układów analogowych i cyfrowych, do projektowania płytek drukowanych i sporządzania schematów ideowych itp.

Oto niektóre z nich:

MICRO CAP — przeznaczony jest do projektowania, analizy i symulacji działania układów analogowych. Zawiera bibliotekę, w której znajdują się następujące elementy: rozmieszczenie operacyjne, tranzystory bipolarne i unipolarne, źródła prądowe i napięciowe, przełączniki, diody i inne elementy dyskretne. Po zaprojektowaniu układu — prowadzona jest symulacja, której wynikiem są charakterystyki częstotliwościowe i amplitudowe. Program umożliwia prowadzenie analizy Fouriera. Duże znaczenie ma zobrazowanie wyników analizy, rozdzielczość optymalnych przebiegów.

MICRO LOGIC — jest programem przeznaczonym do projektowania, analizy i symulacji działania układów cyfrowych. Program uzupełnia działanie MICRO CAP-a.

ANALOP — analizuje i optymalizuje dwukrotne układy liniowe w dziedzinie częstotliwości. Pozwala analizować szeregowe, równoległe, kaskadowe oraz rozgałęzienia i ich kombinacje, uwzględniając nawet układy z zespolonym sprzężeniem zwrotnym. Maksymalnie można wprowadzić 200 elementów. Optymalizacja może być prowadzona dla 15 zmiennych na 15 częstotliwościach. Macierzą rozproszenia można opisać do 10 układów jednowrotnych lub dwuwrotnych. Specyfikacja optymalizacji może być prowadzona w sposób elastyczny, zarówno dla fazy, jak i magnitudy. Program ANALOP analizuje niesymetryczne linie paskowe.

NODAL — przeznaczony jest do analizy symetrycznych i niesymetrycznych linii paskowych. Jako wynik analizy otrzymywany jest wykres Smithca.

Jacek WOJTALA

Giełda pomysłów

ZEGAR — ZX Spectrum

Po uruchomieniu programu w prawym górnym rogu ukazuje się zegar (GG:MM:SS). Ponieważ jego obsługa jest wykonana na przerwaniach, sprawia to wrażenie nieabsorbowania komputera, który może w tym samym czasie wykonywać inne zadania (np. wykonywanie programu interpretera języka BASIC).

URUCHOMIENIE

1) wpisać program nr 2 i nagrać go na magnetofon

SAVE „ZEGAR" LINE 1Ø

2) wykonać komendę NEW

3) wpisać i wykonać program nr 1. Program ten będzie zgrywał dwie części kodu maszynowego, który należy nagrać za nagranym wcześniej programem nr 2.

4) cofnąć taśmę przed programem numer 2 i wykonać:

LOAD „ZEGAR"

5) wykonać polecenia zawarte w programie

UWAGA: Zegar zatrzymuje się przy pracy komputera z magnetofonem. Obszar pamięci od adresu 647ØØ nie może być zmieniany.

R. FOLTYNIEWICZ

```
  10 REM KOD MASZYNOWY cz.1
  20 CLEAR 64699  LET a=64700
  25 LET s=0
  30 FOR f=a TO a+20
  40 READ l  POKE f,l  LET s=s+l
  50 NEXT f
  60 READ l: IF s<>l THEN PRINT
AT 5,7,"ZLE DANE W LINII 210  !!
" STOP
  70 RANDOMIZE USR 64700
 100 REM KOD MASZYNOWY cz.2
 110 LET a=65278  LET linia=310
 120 LET s=0: FOR f=1 TO 10  REA
D l: LET s=s+l: POKE a,l. IF a=6
5475 THEN GO TO 600
 130 LET a=a+1: NEXT f
 140 READ l: IF s<>l THEN PRINT
AT 5,7,"POPRAWIC LINIE ",linia
STOP
 150 LET linia=linia+10  IF a<65
475 THEN GO TO 120
 200 REM dane dla cz.1
 210 DATA 33,219,255,54,1,43,54,
50,33,0,253,6,0,54,254,55,16,251
,54,254,201,2120
 300 REM dane dla cz.2
 310 DATA 245,197,213 229,253,22
9,221,229,33,219,2088
 320 DATA 255 55,175,190 32,82,5
4,1,43,50,935
 330 DATA 175,190,32,64,54 50,33
220,255,205,1298
 340 DATA 98,255,32,63,54,48,35,
205,102,255,1147
 350 DATA 32 55,54,48,35,35,205,
98,255,32,849
 360 DATA 46,54,48,35,205,102,25
5,32,38,54,869
 370 DATA 48,35,35,35,62,50,190,
32,10,43,540
 380 DATA 62,51 190,32,22,54,48,
24,5,43,534
 390 DATA 205,98,255 32,12,54,48
,35,62,50,851
 400 DATA 190,32,4,54,43,24,30,5
8,24,77,555
 410 DATA 62 57,190,201,62,53,19
0,201,14,0,1030
 420 DATA 253,33,31 64,6 0,33,22
0,255,9,904
 430 DATA 126,197,205,134,255,19
3,253,43,12,62,1480
 440 DATA 8 166,32 236 24,41 221
,42,54,92,935
 450 DATA 33,0 111 41,41,41 229
221,25,17,770
 460 DATA 0,1,253,229,225,6,8,22
1,126,0,1069
 470 DATA 119,221,35,25,16,247,3
3,24,88,6,814
 480 DATA 8,54,71,35,16,251,201,
221,225,253,1335
 490 DATA 225,225,209,193,241,18
5,56,0,62,253,1659
 500 DATA 237,71,237,94,201,237,
85,201,0,0,1364
 600 REM ZGRANIE NA MAGNETOFON
 610 PRINT AT 0,5,"Przygotuj mag
netofon"
 620 SAVE "dane.1"CODE 64768,257
.630 SAVE "dane.2"CODE 65278 198
 710 PRINT AT 10,10. KONIEC
   1 REM PROGRAM LADUJACY
  10 CLS . PRINT AT 10,0;"Czy pr
zewidujesz uzycie zlacza krawed
ziowego podczas dzialania progra
mu (t/n)"
  20 INPUT a$
  30 IF a$="n" OR a$="N" THEN GO
 TO 50
  40 CLEAR 64767  LOAD ""CODE 64
768. LOAD ""CODE 65278: GO TO 10
0
  50 CLEAR 65022  LOAD ""CODE 65
023. LOAD ""CODE 65278
 100 REM WPROWADZENIE PARAMETROW
 105 REM atrybuty
 110 CLS  PRINT AT 10,0;"PODAJ
"
 120 PRINT AT 11,0;"KOLOR TLA CY
FR 0-7"  INPUT a
 130 PRINT AT 11,0,"KOLOR ATRAME
NTU CYFR": INPUT b
 140 PRINT AT 11,0;"PODWYZSZONA
JASKRAWOSC t(1)/n(0)"  INPUT c
 150 POKE 65450,a*8+b+c*64
 200 REM godzina
 210 CLS : PRINT AT 10,0; FLASH
1,"Podaj czas: gg-mm:ss"
 220 INPUT c$
 230 IF LEN c$<>8 THEN GO TO 220
 240 LET ad=65503  FOR n=1 TO 8
 250 POKE ad-n,CODE c$(n)
 260 NEXT n
 270 CLS : PRINT AT 0,24;c$
 280 PRINT AT 10,0;"ABY URUCHOMI
C ZEGAR SKASUJ TERAZTEN PROGRAM
KOMENDA NEW.         NASTEPNIE WP
ROWADZ:              RAND. USR 65
168."
 290 PRINT AT 15,0;"ZATRZYMANIE
ZEGARA               RAND. USR 65
170"
```

BIQUINARY NOTATION – zapis dwupiątkowy, notacja dwupiątkowa,
BISECTOR – dwusieczna.
BISTABLE – bistabilny, dwustabilny, dwustanowy,
BISTABLE CIRCUIT – przerzutnik bistabilny,
BISTABLE ELEMENT – element bistabilny,
BISTABLE LATCH – zatrzask (typ przerzutnika),
BISTABLE TRIGGER ELEMENT – przerzutnik bistabilny,
BIT – bit, jednostka miary określająca zawartość informacyjną wiadomości,
BIT CONFIGURATION – wzorzec dwuwartościowy, wzorzec binarny, szablon binarny,
BIT DENSITY – gęstość zapisu binarnego,
BIT ERROR PROBABILITY – stopa błędów bitowa,
BIT ERROR RATE – elementarna (bitowa) stopa błędu, stosunek bitów błędnych do całkowitej liczby bit w transmitowanych,
BIT MAPPING DISPLAY – grafika punktowa,
BIT PACKING DENSITY – gęstość zapisu binarnego,
8-BIT PARALLEL PERIPHERIAL INTERFACE ADAPTER – równoległy 8 bitowy adapter,
BIT PATTERN – wzorzec dwuwartościowy, wzorzec binarny, szablon binarny,
BIT PER SECOND – bit na sekundę (szybkość zapisu),
BIT POSITION – pozycja bitu,
BIT RATE – szybkość transmisji bitów,
BIT SIGNIFICANT – pozycja znacząca bitowa,
BIT-BY-BIT ORGANIZATION – organizacja pamięci z dostępem bitowym,
BIT-SLICE MICROPROCESSOR – mikroprocesor segmentowy,
BIT-SLICED ARCHITECTURE – architektura segmentowa,
BITS PER INCH – liczba bitów na cal,
BIU – patrz: BUS INTERFACE UNIT,
BLACK BOX – "czarna skrzynka", urządzenie lub proces o nie znanej zasadzie, także: układ [illegible] są jedynie wejścia i wyjścia,
BLANK – znak pusty, także: przerwa, odstęp międzyimpulsowy,
BLANK BIT – bit pusty,
BLANK CHARACTER – znak odstępu, spacja,
BLANK COIL – [illegible]
BLANK CYCLE – cykl pusty,
BLANK FORM – formularz, druk,
BLANK INSTRUCTION – [illegible] rób",
BLANK POSITION – pozycja pusta,
BLANK TAPE – taśma czysta,
BLANKED ELEMENT – element niewidzialny [illegible] wia się na urządzeniu graficznym, [illegible] życie elementu [illegible] obraz),
BLANKING – wygaszanie, wymazywanie wybiórcze,
BLEED – także: wyciek kapilarny farby, [illegible] poza kontury znaku,
BLINKING – migotanie (służy do wyróżniania fragmentu obrazu),
BLOCK – blok fizyczny, blok (część programu),
BLOCK ADDRESS – adres bloku,
BLOCK CANCEL CHARACTER – znak pominięcia bloku,
BLOCK CHECK – sprawdzanie bloku,
BLOCK COUNT – liczenie bloków,
BLOCK COUNTER – licznik bloków,
BLOCK END PROTECTION CHARACTER – znak zabezpieczania [illegible]
BLOCK FACTOR – współczynnik blokowania,
BLOCK FORMAT – format bloku,
BLOCK GAP – przerwa międzyblokowa, przerwa blokowa,
BLOCK HANDWRITING READER – czytnik pisma ręcznego blokowego,
BLOCK IGNORE CHARACTER – znak pominięcia bloku,
BLOCK LENGHT – długość bloku,
BLOCK OF WORDS – blok słów (komputerowych),
BLOCK PROTECTION – zabezpieczanie blokowe,
BLOCK SELECTION TIME – czas znalezienia bloku (fizycznego),
BLOCK SIZE – długość bloku,
BLOCK TRANSFER – przesyłanie (informacji) blokowe,
BLOCK WRITING – pisanie bloku,
BLOCK-ORIENTED RAM – pamięć RAM o strukturze blokowej,
BLOCKED RECORD – zapis zblokowany,
BLOCKING – tworzenie bloków, łączenie zapisów (rekordów) w bloki,
BLOCKING FACTOR – współczynnik zblokowania,
BLOWER – dmuchawa, wentylator (np. w przełączniku taśmy),
BLUEPRINT – światłokopia, także: odbitka na światłokopii,
BLUNDER – poważna pomyłka, znaczna pomyłka,
BLUR – zacierać, zamazywać,
BOARD – płyta, tablica, tablica połączeń, tablica (deska) rozdzielcza,
BODY-PUNCHED ASPECT CARD – karta przeziernа,
BOLDFACE – tłusty druk,
BOND – złącze, łącznik,
BONDING PAD – pole kontaktowe (w strukturze układu scalonego),
BOOKKEEPING MACHINE – maszyna do księgowania,
BOOKKEEPING OPERATION – operacje pomocnicze (przy porządkowaniu danych),
BOOLEAN ALGEBRA – algebra BOOLE'a,
BOOLEAN VALUE – wartość typu BOOLEAN,
BOOT – inicjować, ładowanie systemu operacyjnego [illegible],
BOOT RECORD – rekord wprowadzający (ścieżka 0, sektor 1, strona [illegible]),
BOOTING – wciąganie, wprowadzanie,
BOOTSTRAP – program ładujący, ładowanie,
BOOTSTRAP INPUT PROGRAM – program wprowadzania [illegible] wstępnego,
BOOTSTRAP LOADER – program ładujący, procedura ładująca,
BOOTSTRAP PROGRAM – program wprowadzający,
BOOTSTRAPING – wciąganie, wprowadzanie, ładowanie początkowe,
BORAM – patrz: BLOCK-ORIENTED RAM,
BORROW – pożyczka, także: zanegowane przeniesienie,
BOT MARKER – znacznik początku taśmy,
BOTTOM DISC – dolny dysk (w pakiecie),
BOTTOM OF THE PAGE – dolna część strony (arkusza),
BOTTOM-UP – w [illegible] (w systemach ekspertowych),
BOUND – granica obszaru,
BOUNDARY – brzeg, granica (obszaru),
BOUNDARY SET – zbiór brzegowy,
BOUNDARY VALUE – wartość brzegowa,
BOUNDED – odgraniczony, ograniczony,

BOUNDED FROM ABOVE – ograniczony z góry,
BOUNDEDNESS – ograniczoność,
BOX – kratka do wypełnienia na formularzu, kratka na wykresie, także: figura geometryczna na schemacie blokowym, skrzynka,
BPI – patrz: BITS PER INCH,
BPS – patrz: BITS PER SECOND,
BRACE – nawias klamra, nawias klamrowy (lub wężykowaty),
BRACKET – nawias kwadratowy,
BRAIN – mózg, także: w żargonie oznacza komputer, mózg elektronowy,
BRANCH – skok krótki, rozgałęzienie programowe, odgałęzienie,
BRANCH CIRCUIT – obwód odgałęziony,
BRANCH INSTRUCTION – rozkaz rozgałęzienia,
BRANCH OPERATION – operacja rozgałęzienia,
BRANCH POINT – miejsce, punkt odgałęzienia (przewodu), także: zwrotnica, przełącznik),
BRANCH PROGRAM – program uboczny,
BRANCHING BOX – mufa rozdzielcza,
BRANCHING POINT – miejsce odgałęzienia,
BRANCHING PROCESS – proces gałązkowy,
BREADBOARDING OF IC – makietowanie układu scalonego,
BREAK – przerwanie, rozerwanie, urwanie się,
BREAK DOWN – ulegać awarii,
BREAK KEY – klucz przerwania, klawisz zatrzymania,
BREAK SIGNAL DETECTION – detekcja sygnału wstrzymania,
BREAK THE PROGRAM INTO ... – dzielić program na ...,
BREAK THE SEQUENCE – [illegible],
BREAKDOWN-TIME – czas awarii maszyny,
BREAKPOINT – punkt przerwania, miejsce [illegible] (zatrzymania),
BREAKTHROUGH IN TECHNOLOGY – [illegible] postęp w dziedzinie techniki,
B-REGISTER – rejestr modyfikacji, B rejestr,
BRIGHTNESS – jaskrawość, luminancja,
BRIGHTNESS CONTROL – regulacja jaskrawości,
BRIGHTNESS OF TONE – jasność tonu, jasność brzmienia,
BRING INTO STEP – [illegible],
BRING UP TO DATE – aktualizować,
BRITISH STANDARD INTERFACE (BSI) – złącze standardowe BSI [illegible] urządzenie [illegible] do komputerów,
BROAD-BAND LINE – [illegible],
BROADCASTING SYSTEM – [illegible],
BROKEN NUMBER – ułamek,
BRUSH BLOCK – blok ze szczotkami odczytującymi,
BRUSH SCANNER – szczotka odczytu,
BRUSH SHIFT – przesunięcie szczotek,
BRUSH-ROCKER (RING) – trzymadło szczotkowe,
BRUST MODE – praca uderzeniowa (dot. kanału multipleksorowego),
BPS (BITS PER SECOND) – bitów na sekundę,
B STORE – rejestr modyfikacji, rejestr B, rejestr indeksowy,
BUBBLE DOMAINS STORE – pamięć magnetyczna pęcherzyków,
BUBBLE MEMORY – pamięć domenowa,
BUCKET – [illegible],
BUCKET HEADER – etykieta porcji (bloku),
BUCKET PACKING DENSITY – gęstość bloku programowego.

BUCKET-BRIGADE DEVICE - element łańcuchowy, analogowy rejestr przesuwający,
BUFFER - bufor,
BUFFER CAPACITY - pojemność pamięci buforowej,
BUFFER MEMORY - pamięć buforowa, bufor,
BUFFER REGISTER - rejestr pośredniczący, rejestr buforowy,
BUFFER STORE - pamięć buforowa,
BUFFERED - zbuforowany,
BUFFERING - buforowanie,
BUG - defekt w programie, błąd w programie,
BUGGED PROGRAM - program z błędami,
BUILD-IN COMMANDS - polecenia (rozkazy) wbudowane (rezydentne),
BUILDING BLOCK - układ, element realizacyjny,
BUILT-IN CONTROL - sterowanie układowe,
BUILT-IN FUNCTION - funkcja wewnętrzna, funkcja wbudowana,
BUILT-IN LOGIC - wbudowany układ logiczny,
BUILT-IN MICROCOMPUTER - mikrokomputer wbudowany,
BUILT-IN-CHECK - kontrola automatyczna istniejąca w sprzęcie,
BULK STORAGE - pamięć o wielkiej pojemności,
BULK STORE - pamięć masowa,
BULK-EFFECT INTEGRATED CIRCUIT - układ scalony działający z wykorzystaniem zjawisk objętościowych,
BURN IN - w teorii niezawodności oznacza wstępne starzenie obiektu,
BURN-IN PERIOD - w teorii niezawodności oznacza okres wstępnego starzenia obiektu,
BURST - sekwencja sygnałów,
BURSTER - przyrząd do cięcia tabulogramu, także: przerywacz,
BUS - magistrala,
BUS AVAILABLE - dostęp do szyny systemu,
BUS CONTROL REQUEST - wejścia sygnałów procedury dostępu mikroprocesora do magistrali,
BUS ERROR - błąd magistrali,
BUS INTERFACE UNIT (BIU) - jednostka sprzęgająca z magistralą,
BUS MASTER - moduł rozstrzygający o dostępie do szyny
BUS REQUEST/GRANT LOGIC - układ żądania i przyznania magistrali,
BUS REQUESTER - sterownik zgłoszeń,
BUSINESS DATA PROCESSING - przetwarzanie danych gospodarczych,
BUSY/READY SIGNAL - sygnał zajętości/gotowości,
B/W - monochromatyczny (dot. monitora),
BYTE - bajt, także: grupa 8-miu bitów,
BYTE-SERIAL TRANSMISSION - przesyłanie szeregowe bajtów, przesyłanie kolejnych bajtów,

C

CABLE - kabel,
CACHE MEMORY - pamięć podręczna,
CAI - patrz: COMPUTER-AIDED INSTRUCTION,
CALCULATE - obliczać,
CALCULATING MACHINE - maszyna licząca, kalkulator,
CALCULATING OPERATION - operacja arytmetyczna,
CALCULATION - obliczanie,
CALCULATION PROCEDURE - tok obliczania, procedura obliczania,
CALCULATOR - maszyna licząca, kalkulator,
CALCULUS - rachunek,
CALCULUS OF FINITE DIFFERENCES - rachunek różnicowy,
CALCULUS OF PROBABILITY - rachunek prawdopodobieństwa, probabilistyka,
CALCULUS OF VARIATIONS - rachunek wariacyjny,
CALCULUS OF VECTORS - rachunek wektorowy,
CALL - wywołanie, przekazanie kontroli do podprogramu,
CALL SLIP - rewers biblioteczny,
CALLIGRAPHIC DISPLAY DEVICE - konturowe urządzenie graficzne,
CALLING INDICATOR - wskaźnik zgłoszeniowy,
CALLING SEQUENCE - sekwencja wywołująca,
CAM - patrz: COMPUTER AIDED MANUFACTURE,
CAMAC - patrz: COMPUTER AIDED MEASUREMENT AND CONTROL,
CAN - patrz: CANCEL CHARACTER,
CANCEL - anulować, kasować, przekreślać, także: skracać,
CANCEL CHARACTER - znak kasowania,
CANCEL MODE - tryb anulowania,
CANCELATION - anulowanie, kasowanie, także: skracanie,
CANCELLING - kasowanie,
CAPABILITY - zdolność,
CAPACITANCE - pojemność,
CAPACITIVE - pojemnościowy,
CAPACITY - pojemność, zdolność przepustowa, wydajność,
CAPACITY COUPLING - sprzężenie pojemnościowe,
CAPITAL - duża litera alfabetu,
CAPSTAN - oś w przewijaczu taśmy magnetycznej,
CAPTION - nagłówek (np. formularza),
CAPTURE - utrwalać (np. dane na taśmie), zapisywać,
CARBON - kopia spod kalki, także: kalka maszynowa,
CARBON PAPIER - kalka maszynowa,
CARD - karta,
CARD BED - łoże karty (w maszynie),
CARD COUNTER - licznik kart,
CARD FACE - przednia strona karty (dziurkowanej),
CARD FEED - podajnik kart,
CARD FILE - kartoteka,
CARD FILING CABINET - skrzynka kartotekowa,
CARD HOPPER - zasobnik kart (dziurkowanych),
CARD INDEX - kartoteka,
CARD INPUT - wejście na kartach (dziurkowanych),
CARD LAYOUT - wzór karty dziurkowanej,
CARD LOADER - program wprowadzający do pamięci dane zawarte w pliku kart,
CARD PUNCH - dziurkarka kart,
CARD RANDOM ACCESS MEMORY - pamięć na kartach magnetycznych z bezpośrednim dostępem do danych,
CARD READER - czytnik kart,
CARD READER/PUNCH - czytnik dziurkarka kart,
CARD REPRODUCER - reproducer kart (dziurkowanych),
CARD STACKER - zasobnik kart (dziurkowanych),
CARD VERIFIER - sprawdzarka kart (dziurkowanych),
CARD VERIFYING - sprawdzanie kart,
CARD-TO-CARD TRANSCEIVING - teledacyjne powielanie kart dziurkowanych,
CARD-TO-TAPE CONVERSION - przenoszenie zapisu z kart dziurkowanych na taśmę magnetyczną,
CARDCAGE - kaseta (do wkładania modułów sprzętowych),
CARRIAGE - karetka, wózek (np. maszyny do pisania),
CARRIAGE LENGTH - długość przesuwu (np. papieru),
CARRIAGE RETURN CHARACTER (CR) - znak powodujący powrót karetki,
CARRIAGE TAPE CONTROL - sterowanie przesuwu papieru,
CARRIER - nośnik danych,
CARRIER-SENSE MULTIPLE-ACCESS WITH COLISION-DETECTION (CSMA-CD) - sieć lokalna o dostępie wspólnym z wykrywaniem kolizji,
CARRY - przeniesienie,
CARRY BIT - bit przeniesienia,
CARRY CIRCUIT - układ przenoszenia,
CARRY COUNT - liczenie przeniesień,
CARRY DELAY - opóźnienie przeniesienia,
CARRY DIGIT - cyfra przeniesienia,
CARRY GATE - bramka przeniesienia,
CARRY LOOKAHEAD - układ przeniesienia na bardziej znaczące pozycje,
CARRY OVER - przenosić na drugą stronę równości,
CARRY PROPAGATION - propagacja przeniesienia,
CARRY PROPAGATION DELAY - czas propagacji (impulsu wejściowego),
CARRY SIGNAL - sygnał nadmiaru,
CARRY STORE - pamięć przeniesień,
CARRY-OVER - przeniesienie,
CARRYING CAPACITY - nośność, ładowność,
CARTESIAN COORDINATES (SYSTEM) - układ współrzędnych prostokątnych,
CARTRIDGE - ładunek, także: kaseta z taśmą o obiegu zamkniętym,
CARTRIDGE TAPE - taśma kasetowa,
CASCADE SORTING - sortowanie kaskadowe,
CASCADED CARRY - przeniesienie pozycyjne, przeniesienie kaskadowe,
CASE - przykład, przypadek,
CASE-SENSITIVE - posiadający możliwość rozróżniania liter małych od dużych,
CASSETTE - kaseta,
CASSETTE RECORDER - magnetofon kasetowy,
CASSETTE TAPE - taśma kasetowa,
CASTING OUT NINES - próba dziewiątkowa,
CATALOG - katalog,
CATCH UP - także: nadążać,
CATCHWORD - słowo kluczowe,
CATEGORY - dział, grupa,
CATENATE - łączyć,
CATHODE-RAY TUBE (CRT) - lampa elektronopromieniowa, kineskop,
CATHODE-RAY TUBE DISPLAY - zobrazowanie oscyloskopowe,
CAUSAL - przypadkowy, obojętny,
CAUSE - powód, przyczyna,
CCD-TECHNOLOGY - technologia CCD (Charge Coupled Device Technology),

Tajemnice informatyki (2)

Twierdzenie Fermata

Program „Twierdzenie Fermata" ilustruje dobrze znaną w teorii liczb następującą zasadę: dowolnie wybrana liczba pierwsza p wraz z dowolnie wybraną liczbą całkowitą x spełniają następującą równość:

$(x^{p-1} = 1) \bmod p$,

przy czym liczba x nie może być podzielna przez p.

Przykładowo:

$(2^2 = 1) \bmod 3$

$(3^4 = 1) \bmod 5$

$(9^4 = 1) \bmod 5$ itd

Program nasz sprawdza, czy spełnione są założenia twierdzenia Fermata, tj. p — liczba pierwsza, x nie jest podzielny przez p oraz — w wypadku gdy $x^{p-1} > 999999999$ podaje odpowiedni komunikat, zapraszając do wyboru innych liczb

A.T.

```
10 PRINT"podaj liczbe pierwsza"
15 INPUT p
20 PRINT"podaj liczbe naturalna"
25 INPUT x
30 r=x-INT(x/p)*p
35 IF r<>9 THEN 45
40 PRINT"liczba";p,"dzieli";x:GOTO 20
45 xp=x:pl=p-1:xp$="dobrze"
50 FOR i=2 TO pl
55 xp=xp*x
60 IF xp<999999999 THEN GOTO 72
65 PRINT "liczba ";x;" ^ ";pl;" przekracza
   dopuszczalny zakres (1-999999999)"
67 xp$="nadmiar"
70 i=pl
72 NEXT i
73 IF xp$= " nadmiar" THEN 10
75 k=INT(xp/p)
80 r=xp-k*p
85 IF r<>1 THEN PRINT"nie podales
   liczby pierwszej":GOTO 10
90 PRINT "twierdzenie Fermata"
95 PRINT x;" ^ ";pl;" mod ";p;" = ";r
97 PRINT x;" ^ ";pl;" = ";xp
100 PRINT "koniec (t/n) "
105 INPUT x$
110 IF x$="t" THEN GOTO 1000
115 IF x$="n" THEN GOTO 10
120 GOTO 100
1000 END
```

Na skróty

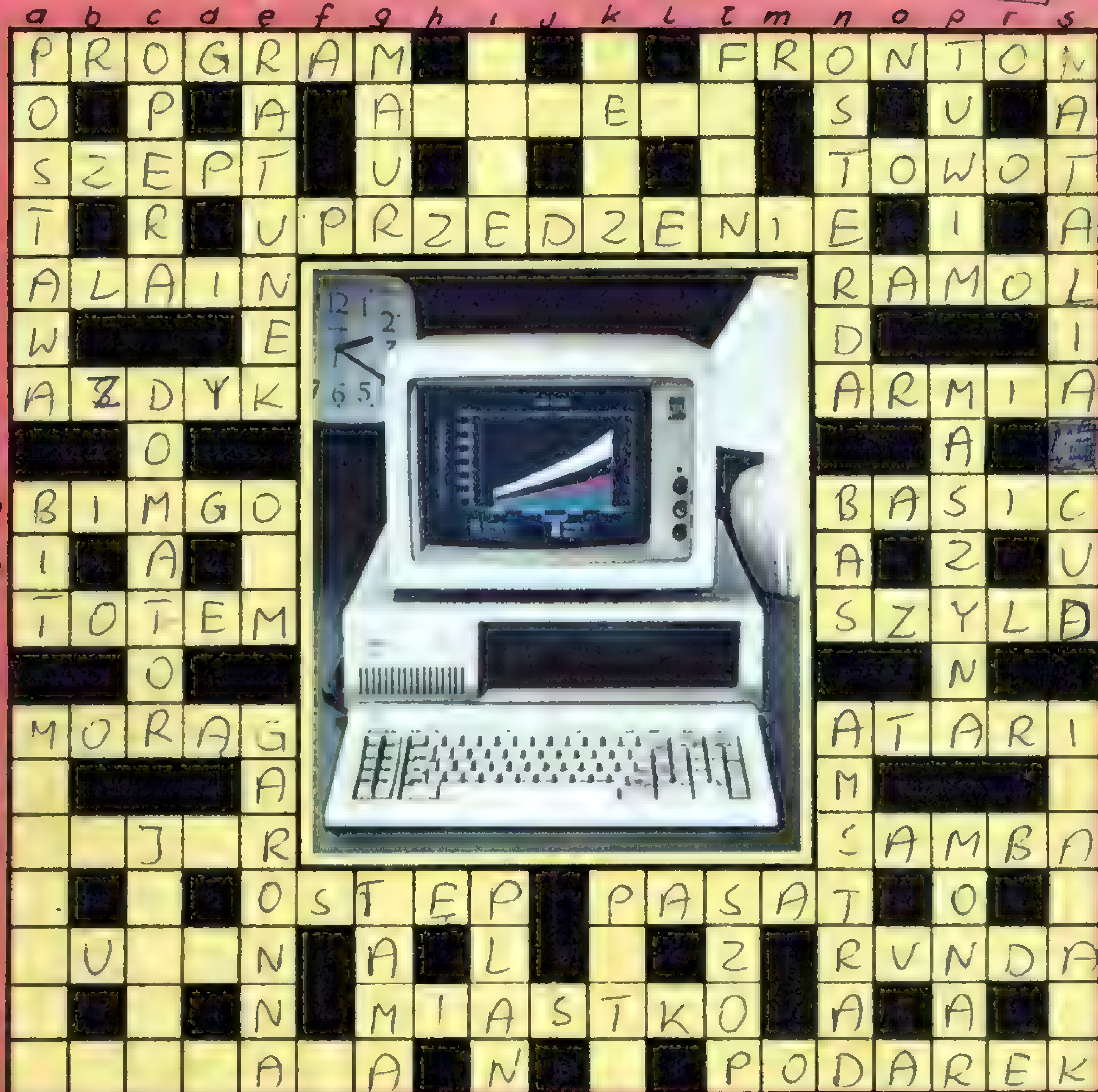
Przedstawiamy zaktualizowaną listę (por. X nr 1/87) skrótów, które związane są z mikrokomputerem, łącznie z ich oryginalną interpretacją. Nadal czekamy na inne, waszym zdaniem, ważne skróty, może z ich polską interpretacją?

AAAI — American Association of Artificial Intelligence
ADA — All — points — addressable
ALU — Arithmetic and Logic Unit
ASCII — American Standard Code for Information Interchange
AT — for Advanced Technology
BASIC — Beginner's All-purpose Symbolic Instruction Code
BDOS — Basic DOS
BIOS — Basic Input-Output System
bps — bits per second
CAD — Computer-Aided Design
CAM — Computer-Aided Manufacturing
CAMAC — Computer Application of Measurments
CCP — Console Command Processor
CD ROM — Compact Disc ROM
CGA — Color Graphics Adapter
CIA — Complex Interface Adapter
CMOS — Complementary Metal-Oxide Semiconductor
CPI — Characters Per Inch
CPM — Control Program for Microcomputers (CP/Monitor)
CPS — Characters per Second
CPU — Central Processing Unit
CRT — Cathode-ray tube
CT — Computer Tomograph
DCE — Data Communication Equipment
DIP — Dual In-line Pacage
DMA — Direct Memory Access
DOS — Disc Operating System
DRAW — Digital Read and Write
DTE — Data Terminal Equipment
EARS — Electronic Audio Recognition System
ECD — Enhanced Console Driver
EDRAW — Erasable Optical Disc Technology
EGA — Enhanced Graphics Adapter
ELD — Electroluminescent Display
EPROM — Erasable PROM
FF — Form Feed
FPU — Floating Point Processing Unit
GEM — Graphics Environment Manager
GSX — Graphics System Extension
IBM — International Business Machines
IC — Integrated Circuit
ICL — International Computers LTD
IFF — Interchange File Format
ILBM — Interleave Bit Map
I/O — Input/Output
LAN — Local Area Network
LCD — Liquid-Crystal Display
LF — Line Feed
LUT — Lookup Table
MDA — Monochrome Display Adapter
MNU — Memory Management Unit
MS DOS — Microsoft DOS
PGS — Professional Graphics System
PIF — Program Information File
PIPS — Pan Information Processing System
POST — Power-On Self Test
PROM — Programmable Sound Generator
RAM — Random Access Read/Write Memory
ROM — Read Only Memory
RGB — Red — Green — Blue
RISC — Reduced Instruction Set Computer
SCCS — Source Code Control System
SID — Sound Interface Device
SIMPP — Simple Image — Processing Package
SLAM — Scan — Line Access Memory
SPRS — Signal Processing Router/Scheduler
TUNIS — Toronto University System
VDU — Visual Display Unit
VIC — Video Interface Chip
WIMPS — Windows, Icons, Mice and Pull — Down Menus
WORM — Write Once, Read Many
YHP — Yokogawa Hewlett — Packard

Opracował:

W. GOGOŁEK

KRZYŻÓWKA NR 11 KUPON



POZIOMO: 1a. niejeden w „IKS-ie" 1ł. przednia część budynku 2g. kościelna klątwa 3a. cichy głos 3n. gęsty smar 4e. nieuzasadniona, podświadoma niechęć 5a. imię słynnego francuskiego aktora filmowego 5n. zniedołężniały staruszek 7a. rodzaj hydrolokatora 7n. siły zbrojne 9a. jelonek — bohater kreskówki Disneya 9n. popularny język komputerowy 11a. czczona w religiach pierwotnych roślina lub zwierzę 11n. tablica informacyjna 13a. miasto w woj. olsztyńskim 13n. typ mikrokomputera 15a. czaple pióro przy kapeluszu 15n. brazylijski taniec 16e. niedostępne miejsce w lesie 16k. stały wiatr w strefie międzyzwrotnikowej 17a. miasto we Włoszech 17n. część walki w ringu 18g. miasto w woj. słupskim 19a. ozdobna plecionka ze sznurka 19ł. w worku Mikołaja

PIONOWO: a1. może być zasadnicza a9. jednostka ilości informacji a13. typ mikrokomputera c1. dramat muzyczny c7. popularny poranny program telewizyjny c15. jego czaszkę nosi Hamlet e1. pomoc w nagłym wypadku e9. firma produkująca sprzęt komputerowy e13. rzeka we Francji g1. dawny muzułmański mieszkaniec Płw. Iberyjskiego g16. zapora i1. paragwajska herbata i16. projekt k1. rosyjski fizyk, autor reguły dotyczącej kierunku przebiegu prądu elektrycznego k16. angielski mąż stanu z przełomu XVIII i XIX w. ł1. mitologiczny leśny bożek ł16. zwierzę myjące pożywienie przed spożyciem n1. słynny aktor i reżyser polski (1885—1947) n9. niski głos męski n13. typ komputera p1. autor „Kwiatów polskich" p7. elektryczna lub parowa p15. organizacja walcząca z narkomanią s1. imię żeńskie s9. niewytłumaczalne zjawisko s13. duża rzeka azjatycka

Rozwiązanie stanowi hasło utworzone z liter przeniesionych z następujących pól: 16i 18ł 7b 18e 1f 15c (7e 11b 9c 1a 17b 4a 2k 1m) — (3b 11p 19j 10l 5o 15n 11o) 16k 7o 4h 7d 15c 5a 9s 7r 4i 5s 9o)

Rozwiązania (tylko hasło) należy przesyłać pod adresem redakcji, **na kartkach pocztowych naklejając kupon „IKS-a".**

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8884. Nr. ind. 361682.

POZNAJ KOMPUTER ZYSKASZ PRZYJACIELA

INFORMATYKA KOMPUTERY SYSTEMY



CENA 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" Nr 1/1988 ISSN 0860–2794







V Krajowa Konferencja Informatyków

V Krajowa Konferencja Informatyków odbyła się 8—10 grudnia 1987 roku w Poznaniu (hotel Poznań). Zaplanowanym tematem spotkania (262 uczestników) był Rozwój Metod i Zastosowań Informatyki. Na konferencję nadesłano 108 referatów. Zostały one opublikowane w 534 stronicowym tomie. Organizatorów Konferencji wspomagali: Komitet Wspomagający oraz Rada Programowa, w sumie 48-osobowy zespół fachowców.

Tyle spraw formalnych, jeśli idzie o merytoryczne to warto podkreślić, iż posiedzenia odbywały się z udziałem wszystkich uczestników pierwszego dnia (około 30—50% drugiego, trzeciego bywało różnie...)

Każdemu z posiedzeń przewodniczyło kilku Panów Profesorów lub Panów Docentów. Wygłaszali oni słowo wstępne, a następnie zachęcali do dyskusji (referaty nie były wygłaszane).

Pierwszego dnia zwrócono między innymi uwagę na rangę sieci komputerowych, które są obecnie powszechną formą wykorzystania informatyki niemal we wszystkich dziedzinach życia wysokorozwiniętych krajów. Wiąże się z tym konieczność dysponowania dużymi komputerami — superkomputerami. Tworzą one tak zwane węzły sieci. Krajowe wysiłki w tej kwestii reprezentowały wypowiedzi na temat doświadczeń i perspektyw sieci komputerowych w Polsce. Najbardziej zaawansowane są prace nad Międzyuczelnianą Siecią Komputerową (MSK) realizowaną od 1985 roku oraz nad Krajową Akademicką Siecią Komputerową (KASK), która budowana jest w ramach CPBR nr 8.13. Warto w tym miejscu podkreślić, że już od kilku lat pracuje z powodzeniem u nas w kraju komputerowa sieć GABRIEL LOT-u. Obsługujący ją komputer jest w Nowym Jorku.

Poza kwestiami technicznymi budowy sieci, zwrócono uwagę na bardzo istotny problem zachowania standaryzacji sprzętu i oprogramowania. Podkreślono także, że w naszych warunkach, posiadanie nawet doskonałej sieci, to dopiero początek długiej, bardzo długiej drogi do jej efektywnego wykorzystania — bowiem sieć tę trzeba „wypełnić odpowiednimi informacjami" i chcieć (umieć) z nich korzystać. Jest to problem ogromny, tym bardziej że nasze doświadczenia w tym względzie są ciągle pomijalne.

Drugiego dnia konferencji zwrócono między innymi uwagę na centralne komputery — stanowiące serce (mózg?) sieci. Jeden z prowadzących posiedzenie przedstawił bardzo ciekawe zestawienie czasu liczenia problemu XYZ

Typ komputera	Czas liczenia problemu XYZ w latach
VAX II/780	30 000
Cosmic cube	3 000
Cray I	100
Columbia	10
GF 11	1

Ostatni z wymienionych — GF 11 — w chwilach „dobrych" (tak zwanych pikach) liczy około 11 gigaoperacji w czasie jednej sekundy (11 miliardów operacji/sek.) Innymi słowy w czasie, gdy komputer wykonuje jedną operację (złożoną) np.: mnożenie dwóch wielocyfrowych liczb — światło przebędzie zaledwie nieco ponad 2 cm (dwa)! Jak to możliwe? — zadanie to (operację) wykonuje jednocześnie setki procesorów.

Cena superkomputera wynosi aż 5—20 mln dolarów. Reprezentantami tego sprzętu są np.: Cyber 205, Cray I, Cray II, a w Japonii urządzenia tego typu produkuje NEC i parę innych firm. Ciekawym osiągnięciem naukowców Kraju Kwitnącej Wiśni jest mikroprocesor (liczy szeregowo), który rachuje z szybkością przekraczającą 1 gigaoperację na sekundę (światło, w czasie wykonywania przez ten procesor jednej operacji, przebiega 30 cm).

Dostęp do superkomputerów np.: w Holandii (są to Cyber 205 i Cray) realizowany jest poprzez sieci połączone komputerami VAX.

Ciekawymi spostrzeżeniami jednego z uczestników konferencji jest to, że Fortran jest obecnie podstawowym językiem programowania superkomputerów.

Aby nie popadać w kompleksy zauważono, że odpowiednie „uzbrojenie" IBMa PC (których mamy w kraju sporo) w „przyspieszacze" tworzy nam komputery o tempie liczenia tylko 50—100 razy mniejszym od Cray'a.

Odmienny podział komputerów zaproponował inny z uczestników konferencji

Mikrokomputery	— IBM, PS/2
Superminikomputery	— VAX, SCS-40
Superkomputery	— Cray, CD6, NEC
Hiperkomputery	— ETA-10

Architektura superkomputerów została zaprezentowana przez najmłodszych uczestników konferencji — członków studenckiego Koła Zainteresowań Cybernetycznych z WATu.

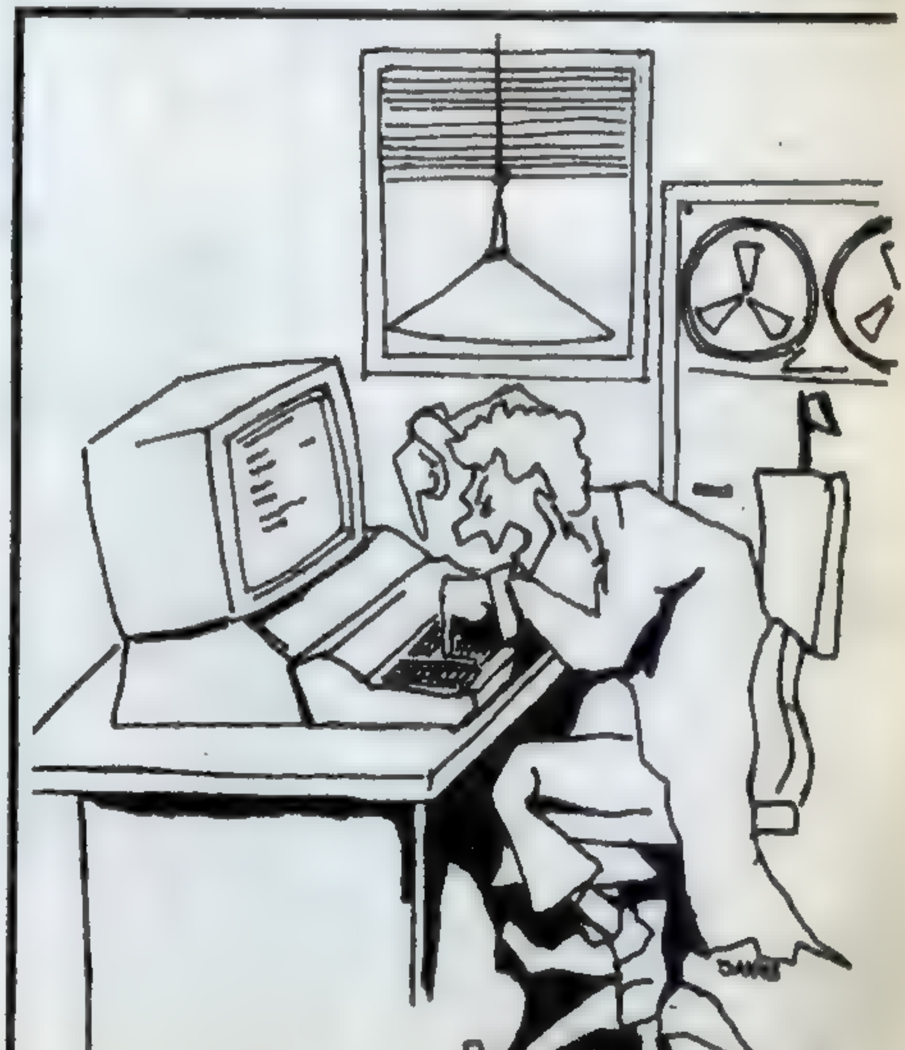
Warto w tym miejscu podkreślić wątpliwość, którą wywołała wypowiedź jednego z Panów Profesorów — *„Potrzebujemy w kraju superkomputera, a jest to jedyny nakład techniczny — sprzęt ten nie wymaga konserwacji"*... czyzby?

Na świecie pracuje obecnie około 260 superkomputerów, w tym, ponad 50 eksploatowanych jest w Japonii.

Powstało zatem pytanie — co robić w naszych warunkach? Pytanie to stało się retorycznym, a nas zaproszono do Ośrodka Krajowej Państwowej Komunikacji Samochodowej w Poznaniu. Tyle sprzętu informatycznego w jednym niewielkim budynku jeszcze nie widziałem: kilkadziesiąt klonów IBMa PC, w tym ELWRO 801At (dwie sztuki) oraz RIAD 34 we wspaniałym zestawie: dwie drukarki wierszowe, pięć napędów dyskowych ze sterownikami (2X100 Mb plus 3X217 Mb z możliwością dołączenia innych napędów, także 600 Mb), cztery przewijaki („samoładujące się") i adapter do wielu, bardzo wielu terminali (także JSG). Najwięcej zainteresowania budziła jednostka centralna — zbudowana u nas w kraju — wygląda niepozornie — solidny „słup" trochę wyższy od piszącego te słowa. Jednostka mieści w sobie 8 Mb RAM. Wykonuje około 0,5 mln operacji na sekundę. Sprzęt ten jest w trakcie uruchamiania.

Kończąc spisane na gorąco niektóre wrażenia, chciałbym podkreślić nieprzypadkowe wyliczanie we wstępie: 262 uczestników, 48 organizatorów! Wydaje się, że gdyby uzupełniono tytuł konferencji o magiczne słowo „naukowa", mniej było by osób zawiedzionych, które oczekiwały informacji bliższych naszej rzeczywistości i przyszłości — tej najbliższej.

Uczestnik



— Kocha, lubi, szanuje, nie dba, nie chce, żartuje...

Laser za złotówki



Rozmowa z Zygmuntem Klukiem — kierownikiem sklepu Centralnej Składnicy Harcerskiej Oddział Warszawa na ulicy Mokotowskiej 26. Jednym z 7 sklepów CSH specjalizujących się w sprzedaży sprzętu komputerowego i wideo.

Czego mogą spodziewać się klienci CSH w tym roku?

— Utrzymania dotychczasowych zasad. Będziemy nadal sprowadzać sprzęt komputerowy firm Spectra Video, Timex i Bondwell. W sprzedaży będą modele Spectra video SV 738, Timex, klony IBM z Dalekiego Wschodu — Bondwell 38-2-XT, Bondwell Portable (przenośny). Zamierzamy sprowadzić Bondwell 63 AT, uzupełnieniem do modeli XT i AT będą karty (pakiety) programowe. Nowością na polskim rynku będzie Spectra video X-PRESS 16 kompatybilny z IBM PC. Oczywiście nadal będziemy oferować urządzenia zewnętrzne i różne drobiazgi komputerowe: interfejsy, joystiki, dyskietki.

Od pewnego czasu poszerzyliśmy ofertę o sprzęt wideo — magnetowidy VHS, kamwidy, kasety, telewizory. Melomanów z pewnością ucieszy wiadomość, że w tym roku zamierzamy sprowadzić kompact dyski i odtwarzacze laserowe.

Jednym słowem klient będzie mógł kupić w CSH nowoczesny sprzęt. Czy nie boi się Pan konkurencji? W tym roku z pewnością powstaną liczne firmy ze zbliżonej branży.

— Jak do tej pory, to konkurencja boi się CSH. Bondwell XT sprzedawany był po cenach znacznie niższych niż w innych firmach, w związku z czym musiały je obniżyć. Jesteśmy na tyle solidną firmą, że „dorobiliśmy" się własnych klientów przyjeżdżających z odległych miast, nawet takich, gdzie są podobne sklepy. Z pewnością w jakimś stopniu przyciągają ich dwie literki w nazwie „PP" — przedsiębiorstwo państwowe.

Kto najczęściej korzysta z pańskiego sklepu?

— Blisko 50 procent sprzętu zakupił resort oświaty. Na drugim miejscu są kluby osiedlowe, miejskie, zakładowe, wiejskie. Z pewnością proporcje wyglądałyby inaczej gdybyśmy mieli więcej towaru, klienci prywatni kupują tylko „nadwyżki".

Jakie wymagania stawia Pan pracownikom?

— Muszą być uczciwi, znać się na komputerach, lubić handel. Tu przerzuca się tony sprzętu.

Ile zarabiają?

— W naszym oddziale jesteśmy w czołówce, powyżej średniej krajowej. Zarobki są ważne, bo mobilizują pracowników i przywiązują do firmy.

Kto wybiera dostawców sprzętu?

— Zarząd przedsiębiorstwa. Oceniamy ofertę potencjalnych kooperantów, ich wyroby, programy. Coraz więcej polskich firm z nami współpracuje — rzemieślnicy, spółdzielnie, zakłady państwowe. Firmy zagraniczne nie dostarczą nam polskich programów użytkowych, edukacyjnych czy gier.

Pańskie życzenia?

— Od początku uruchomienia sklepu to samo. Chciałbym sprzedawać polski komputer.

Czego Panu w imieniu redakcji życzę.

Dziękuję za rozmowę.

Dariusz OTTO

NAUKA. TECHNIKA ...

Naukowcom amerykańskim z laboratorium fizycznego w Waltham (stan Massachusetts) udało się opracować technologię produkcji tranzystorów, które niemal w całości , wytwarza ... natura. Proces budowy tranzystorów rozpoczyna się od stopienia krzemu i tantalu powodując ich całkowite wymieszanie. Gdy tak przygotowaną mieszaninę powoli ochłodzić, to w kryształkach krzemu zaczynają powstawać wtrącenia tantalu. Występują one jednak tylko w tych punktach, w których uprzednio zanurzone były cienkie krzemowe pałeczki pełniące rolę zarodników. W trakcie krzepnięcia pałeczki te są stopniowo usuwane, a na ich miejscu rosną nitki tantalu o średnicy około 1 µm. Każda z tantalowych nici zamieniana zostaje w tranzystor w wyniku nałożenia wokół niej dwóch koncentrycznych warstw przewodnika prądu. Ten nowy sposób budowy tranzystorów pozwala na uniknięcie podstawowych trudności wiążących się z produkcją układów scalonych rozmieszczanych na płytkach krzemowych, jak również umożliwia obciążanie układów prądem o wyższym natężeniu bez możliwości uszkodzenia elementu elektronicznego.

■ ■ ■

W ZSRR prowadzone są ostatnio próby aparatury, która w lipcu br., w ramach międzynarodowej ekspedycji "PHOBOS" poleci w kierunku Marsa. W odstępie kilku dni wystartują dwa aparaty kosmiczne zawierające ponad 20 różnych urządzeń naukowych przeznaczonych do badania "Czerwonej Planety", jego satelity Phobosa, a także Słońca i przestrzeni międzyplanetarnej. W ekspedycji tej po raz pierwszy w praktyce kosmonautyki zostanie przeprowadzone badanie składu gruntu ciała niebieskiego przy pomocy promienia laserowego.

■ ■ ■

W roku 1988 planuje się w Wielkiej Brytanii założenie banku informacji o zwierzętach znajdujących się w ogrodach zoologicznych. Centralny bank zlokalizowany ma być w Londynie. Przewiduje się zainstalowanie terminali komputerowych w 18 ogrodach umożliwiających dostęp do danych o 40000 różnych zwierząt należących do ponad 1500 gatunków. Pozwoli to na opracowywanie nowych programów hodowli rzadkich i ginących zwierząt jak również wydawanie zaleceń dotyczących hodowli zwierząt obecnie żyjących w sztucznych warunkach. Należy oczekiwać, że do 1994 roku powstanie międzynarodowy bank informacji, oparty na systemie opracowanym w Wielkiej Brytanii, z którego będą mogły korzystać ogrody zoologiczne w Ameryce, Australii i Europie.

■ ■ ■

Znana amerykańska firma elektroniczna TEXAS INSTRUMENTS opracowała nowy specjalizowany moduł mikroprocesorowy VHSIC oznaczony symbolem 1750A.

Będzie on, od 1988 r., podstawowym układem sterowania i kontroli w wojskowych samolotach firmy LOCKHEED.

Systemy doradcze

Janusz MORBITZER

Jednym z praktycznych rezultatów badań w dziedzinie sztucznej inteligencji jest powstanie tzw. systemów doradczych, zwanych również systemami eksperckimi (ang. expert systems).

System doradczy to program symulujący pracę grupy ekspertów rozwiązującej problem z zakresu jednoznacznie sprecyzowanej dziedziny. System taki jest bardzo wąsko specjalizowany.

Jaki jest cel stosowania systemów doradczych? Otóż praca eksperta-człowieka jest zwykle bardzo droga. Nie zawsze możliwe jest szybkie przybycie eksperta na miejsce, gdzie trzeba postawić diagnozę. Ponadto zdania poszczególnych ekspertów rozwiązujących ten sam problem mogą być różne. Opłacalne jest więc kilkakrotne zorganizowanie spotkań ekspertów w celu „wydobycia" od nich wiedzy z zakresu wybranej dziedziny. Wiedza ta jest podstawą budowy systemu doradczego. Zakłada się, że suma wiedzy uzyskanej od ekspertów jest większa od wiedzy każdego z nich z osobna. Raz zbudowany system można powielać w dowolnej liczbie egzemplarzy i dostarczać tam, gdzie jego stosowanie jest celowe i opłacalne.

Dla jakich problemów tworzy się systemy doradcze? Ogólnie istnieją dwie grupy takich problemów. Pierwsza, to problemy, dla których nie jest znany algorytm rozwiązania. Drugą grupą są problemy, dla których algorytm rozwiązania jest wprawdzie znany, ale czas jego realizacji znacznie przekracza rozsądne, dopuszczalne granice. Przykładem takiego problemu jest gra w szachy.

Algorytm rozwiązania jest tutaj znany. Wszystkie możliwe warianty gry można przedstawić w postaci drzewa (rys. 1), przypisując końcowym węzłom atrybuty: partia wygrana, przegrana i remis. Wystarczy teraz tylko wybierać te ruchy, które prowadzą do wygranej lub w najgorszym wypadku (gdy przeciwnik również realizuje tę samą strategię) do remisu.

sytuacja startowa

20 możliwych ruchów białych

20 możliwych odpowiedzi czarnych na każdy ruch białych

Wyczerpująca analiza takiego drzewa nie jest jednak możliwa. Liczba wszystkich dróg do przebadania wynosi około 10^{120}. Zakładając, że analiza każdego ruchu trwa 1 ns i wiedząc, że 1 rok = 3.15×10^{13} (ns), łatwo można obliczyć, że analiza całego drzewa trwałaby około 10^{99} lat. Warto podkreślić, że istotne ograniczenia, dyskwalifikujące przedstawiony algorytm, nie tkwią w sprzęcie, gdyż nawet przyjęcie większej o kilka rzędów prędkości obliczeń nie prowadzi do zadowalających rezultatów. W jaki sposób można więc rozwiązać problem gry w szachy? Problem ten można rozwiązać stosując tzw. programowanie heurystyczne. Heurystyka to ogólna zasada, wskazówka lub taktyka oparta na doświadczeniu, drastycznie ograniczająca poszukiwanie rozwiązań w dużych obszarach problemowych.

Heurystyka prawie zawsze daje wystarczająco dobre rozwiązanie. Oznacza to, że czasami na skutek drastycznego ograniczania poszukiwań, najlepsze rozwiązanie (a nawet wszystkie rozwiązania) mogą zostać pominięte. Programowanie heurystyczne odgrywa dominującą rolę w systemach doradczych. Systemy te nie działają bowiem na podstawie algorytmów, lecz wiedzy empirycznej, a więc heurystyki.

System doradczy składa się z następujących modułów funkcjonalnych:
- bazy danych,
- bazy reguł,
- układu kontroli.

Baza danych zawiera fakty z danej dziedziny. Relacje pomiędzy faktami przechowywane są w bazie reguł. Są to najczęściej reguły typu

IF warunek THEN akcja — 1
ELSE akcja — 2

przy czym warunek nie musi mieć określonej wartości logicznej lub wartość ta może być określona z pewnym prawdopodobieństwem.

Układ kontroli określa sposób korzystania z relacji. Służy on do:
- selekcjonowania reguł.
- dopasowywania reguł
- kolejkowania reguł (gdy kilka z nich rokuje sukces)
- wykonywania reguł.

Jakość systemu doradczego zależy głównie od jakości wiedzy. Wiedza obejmuje fakty, związki między faktami oraz metody korzystania z tych związków. Istotnym zagadnieniem jest również zupełność wiedzy, tj. objęcie wszystkich możliwych problemów z danej dziedziny. Jeżeli wiedza jest niezupełna, to część problemów nie może być rozwiązana, gdyż brak jest powiązań z wiedzą zawartą w systemie.

W jaki sposób buduje się system doradczy? Proces ten przebiega w kilku po sobie następujących etapach:

1 — pozyskanie wiedzy (ang. knowledge acquisition process) — np. 2
2 — identyfikacja (określenie problemów i zakresu działania systemu)
3 — konceptualizacja (określenie sposobu reprezentowania wiedzy i przepływu informacji)
4 — formalizacja (wyborów języka, opis)
5 — implementacja systemu (programowanie)
6 — testowanie.



Rys. 2 Etap pozyskiwania wiedzy

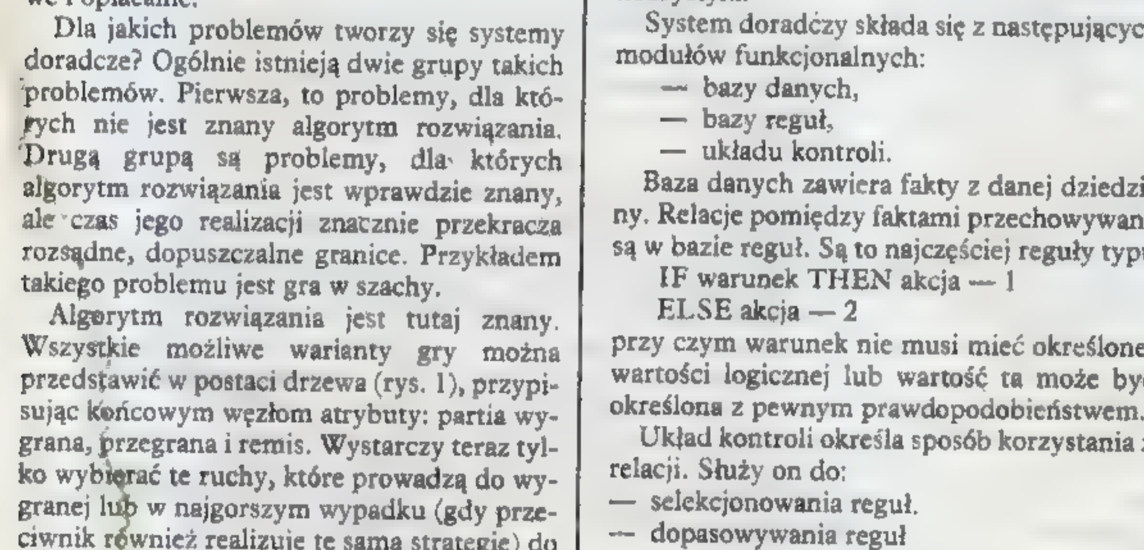
Systemy doradcze tworzone są zwykle w języku LISP, PROLOG, C, PASCAL lub innym zorientowanym na przetwarzanie logiczne. Systemy te nie tylko powinny dostarczyć rozwiązania problemu, ale również odtworzyć na żądanie sposób dojścia do tego rozwiązania. Jest to tzw. odpowiedź na pytanie WHY? (dlaczego system zrobił to, co zrobił).

Systemy doradcze są systemami otwartymi — mogą przyjmować nowe reguły wzbogacając tym samym bazę wiedzy. System automatycznie sprawdza, czy nowa reguła nie jest sprzeczna z istniejącymi lub czy z nich w oczywisty sposób nie wynika.

Systemy doradcze znajdują obecnie zastosowanie w wielu dziedzinach:
- w medycynie (np. MYCYN, ARAMIS, INTERNIST)
- w rozpoznawaniu struktur chemicznych (np. DENDRAL)
- w diagnozowaniu chorób roślin
- w klasyfikowaniu roślin i zwierząt
- w diagnozowaniu uszkodzeń elektrycznych i mechanicznych (np. silników)
- w ocenie strat wynikłych z trzęsienia ziemi (system SPERILL)
- w systemach biurowych.

Jednym z najbardziej znanych systemów jest MYCYN. Służy on do diagnozowania chorób zakaźnych krwi. System ten składa się z 450 reguł.

Zbudowane dotychczas systemy są stosunkowo prymitywne. Zawarta w nich wiedza jest dużym uproszczeniem rzeczywistości. Podstawowymi problemami są sposoby reprezentowania wiedzy oraz sekwencyjność pracy komputerów (człowiek myśli równolegle). O ile drugi problem może być rozwiązany na drodze sprzętowej (wieloprocesowość, transputery), o tyle rozwiązanie pierwszego wymaga opracowania nowych metod reprezentacji wiedzy — jej rozumienia i organizacji, tj. powiązań między poszczególnymi informacjami. Jest to podstawowy kierunek badań sztucznej inteligencji. Duże nadzieje na postęp w tej dziedzinie związane są z komputerami piątej generacji.



Rys. 1. Fragment drzewa gry w szachy

W szponach Atari (11)

„Przewijanie" obrazu

Tomasz MROWIEC
Ludwik PIELA

Często zdarza się, że chcemy wyświetlić taką ilość informacji, która przekracza pojemność ekranu. Jednym ze sposobów rozwiązania tego problemu jest przesuwanie informacji po ekranie. Przykładem takiego rozwiązania jest przeglądanie programu po podaniu polecenia **LIST**. Poszczególne linie programu przesuwają się po ekranie od dołu do góry. Ten rodzaj przesuwania stosowany jest we wszystkich mikrokomputerach. Komputery Atari dysponują dwoma dodatkowymi sposobami przesuwania, oferującym niezwykłe możliwości: zgrubnym i płynnym.

W większości komputerów używane jest przesuwanie zgrubne: pola przechowujące znaki mają ustalone położenie na ekranie i przesuwanie tekstu polega na przemieszczaniu bajtów w obszarze pamięci ekranu. Takie przesuwanie odbywa się skokowo, o jeden wiersz lub kolumnę. Rozdzielczością przesuwania jest wysokość lub szerokość znaku. Ruch jest szarpany i dosyć nieprzyjemny. Poza tym uzyskiwany jest przez przemieszczanie w pamięci tysięcy bajtów, co odbywa się stosunkowo wolno.

W pewnych mikrokomputerach można uzyskać nieco płynniejszy ruch przez kreślenie obrazów w trybie graficznym o wyższej rozdzielczości. Podczas przesuwania uzyskuje się większą rozdzielczość, jednak kosztem znacznego spowolnienia działania programu. Cała trudność polega na tym, że przesuwanie informacji na ekranie realizowane jest przez przemieszczanie danych w obszarze pamięci ekranu.

W komputerach Atari przesuwanie zgrubne można uzyskać w nieco inny sposób: przesuwając ekran nad danymi, które mają być wyświetlone. Umożliwia to rozkaz LMS, który występuje w programie wyświetlania i informuje ANTIC skąd mają być brane dane do wyświetlania. Normalnie występuje jeden rozkaz LMS na początku programu wyświetlania. Wskazuje on początek obszaru RAM zawierającego dane, które mają być wyświetlone na ekranie. A zatem zgrubne przesuwanie możemy uzyskać przez manipulowanie bajtami argumentu rozkazu LMS. Przez zmianę zawartości dwóch bajtów adresu możemy osiągnąć efekt podobny do przemieszczania setek bajtów w obszarze pamięci ekranu. **Program nr 1** demonstruje wykorzystanie tego sposobu. Przesuwa on

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Prosram nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
ST 10 DL=PEEK(560)+256*PEEK(561)
HE 20 LMSL=DL+4
GP 30 LMSH=DL+5
KS 35 ? CHR$(125):LIST
JU 40 FOR I=PEEK(89) TO 255
IG 50 POKE LMSH,I
JO 60 FOR J=PEEK(88) TO 255
KS 70 POKE LMSL,J
ES 80 FOR W=1 TO 160:NEXT W
JG 90 NEXT J
FR 100 NEXT I
NS 110 END
```

ekran nad obszarem pamięci zawierającym wydruk programu oraz procedury systemu operacyjnego. W rezultacie otrzymujemy przesuwanie poziome i pionowe. Czyste przesuwanie pionowe możemy uzyskać przez dodawanie lub odejmowanie długości linii w bajtach do argumentu rozkazu LMS. Wykonuje to **program nr 2.**

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Prosram nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
HG 10 LIST 1,140
SU 20 DL=PEEK(560)+256*PEEK(561)
HF 30 LMSL=DL+4
GQ 40 LMSH=DL+5
ZR 50 EL=24:EH=PEEK(561)
PY 60 EL=EL+40
HO 70 IF EL<256 THEN 110
QZ 80 EL=EL-256
SP 90 EH=EH+1
FF 100 IF EH=256 THEN END
MQ 110 POKE LMSL,EL
IG 120 POKE LMSH,EH
LF 125 FOR W=1 TO 250:NEXT W
RG 130 GOTO 60
NY 140 END
```

Czyste przesuwanie poziome nie jest tak łatwe do uzyskania jak pionowe. Powodem jest szeregowa organizacja pamięci ekranu. Bajty danych umieszczone są kolejno: bajty jednej linii występują bezpośrednio za bajtami poprzedniej linii. Możemy przesuwać linie poziomo w lewo przez zwiększanie argumentu LMS. W tym wypadku jednak, skrajne lewe bajty każdej linii będą przesuwane do skrajnej prawej pozycji następnej wyższej linii. Było to widoczne po uruchomieniu programu nr 1.

Rozwiązaniem może być rozszerzenie obszaru danych ekranu i rozbicie go na ciąg niezależnych obszarów poziomych linii danych.

Musimy pamiętać, że RAM jest jednowymiarowa i bajty umieszczone są kolejno, lecz teraz używana jest inaczej. Dla każdej linii poziomej rozszerza się obszar RAM dużo dalej niż może być pokazane na ekranie. Nie jest to przypadek, wszak chcemy wyświetlić większą ilość danych niż możemy zmieścić na ekranie. Aby pokazać tę dodatkową informację, musimy ją wpierw przechować w pamięci. Przy takim uporządkowaniu możemy stosować prawdziwe przesuwanie poziome. Możemy przesuwać ekran nad danymi bez niepożądanego przesuwania pionowego.

Pierwszym krokiem w uzyskaniu czystego przesuwania poziomego jest określenie całkowitej długości linii poziomej i przydzielenie odpowiedniego obszaru RAM. Następnie musimy napisać całkowicie nowy program wyświetlania z rozkazami LMS w każdej linii obrazu. Program wyświetlania będzie oczywiście dłuższy niż zwykle. Jako argumentu LMS najwygodniej jest użyć adresu pierwszego bajtu każdej poziomej linii danych ekranu. Dla każdej linii obrazu będzie jeden taki adres.

W celu uzyskania przesuwania każdy argument rozkazu LMS w programie wyświetlania musi być zwiększony, aby uzyskać ruch w lewo, albo zmniejszony dla ruchu w prawo. Program musi być tak skonstruowany, aby obraz nie przesunął się poza granice przydzielonych obszarów RAM; w przeciwnym przypadku mogą być wyświetlone „śmieci". Podczas tworzenia programu należy pamiętać, że argument LMS wskazuje pierwszy bajt danych ekranu w wyświetlanej linii. Maksymalna wartość argumentu równa jest adresowi ostatniego bajtu w długiej linii poziomej zmniejszonemu o liczbę bajtów w wyświetlanej linii. Pamiętajmy także, że wartość argumentu LMS nie powinna, wewnątrz jednej długości linii ekranu (w bajtach), przekraczać granicy adresu 4KB.

Opisany proces jest dosyć skomplikowany, dlatego przedstawimy go na przykładzie. Najpierw wybieramy długość linii poziomej. Najwygodniej będzie użyć linii o długości 256B, uprości to nam obliczanie adresu. Każda linia będzie zatem wymagać jednej strony pamięci. Używając trybu 2 BASIC-a zmieścimy na ekranie 12 linii obrazu, zatem potrzebne będzie 12 stron (3KB) RAM. Dla uproszczenia i zagwarantowania że pamięć ekranu nie będzie zawierać samych zer, użyjemy dolnych 3KB RAM. Obszar ten jest używany przez OS i DOS, dlatego powinien zawierać interesujące dane. Dla uczynienia eksperymentu bardziej interesującym, umieścimy program wyświetlania na stronie 6, dzięki temu możemy wyświetlić go na ekranie podczas przesuwania. Początkowe wartośc argumentów LMS będą zatem łatwe do obliczenia. wszystkie mniej znaczące bajty będą zerami, a bardziej znaczące będą miały kolejno wartośc 0,1,2,...itd. **Program nr 3** wykonuje opisane operacje i przesuwa program poziomo. Przegląda on dane z

```
FK 1 REM ****************
FJ 2 REM *              *
YO 3 REM * Prosram nr 3 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
QY 10 REM ZAPIS PROGRAMU WYSWIETL
   ANIA
WO 20 FOR I=1536 TO 1538
LD 30 POKE I,112
IS 40 NEXT I
HL 50 FOR I=1 TO 12
XI 60 POKE 1536+3*I,71
ZV 70 POKE 1536+3*I+1,0
TU 80 POKE 1536+3*I+2,I
IX 90 NEXT I
WU 100 POKE 1575,65
NH 110 POKE 1576,0
RI 120 POKE 1577,6
IT 130 REM PODANIE ADRESU PROGRAM
   U WYSWIETLANIA
HS 140 POKE 560,0
LM 150 POKE 561,6
ZB 160 REM PRZESUWANIE POZIOME
DV 170 FOR I=0 TO 235
SS 180 FOR J=1 TO 12
LW 190 POKE 1536+3*J+1,I
GC 200 NEXT J
JJ 205 FOR W=1 TO 50:NEXT W
FU 210 NEXT I
OK 220 GOTO 170
NX 230 END
```

prawa na lewo. Po osiągnięciu końca strony zaczyna przeglądanie od początku. Program wyświetlania można znaleźć w szóstej linii od dołu (tj. na stronie 6).

Następnym krokiem jest połączenie przesuwania pionowego i poziomego dla uzy-

skania ruchu ukośnego. Przesuwanie poziome otrzymujemy przez dodanie lub odjęcie 1 od argumentu LMS. Przesuwanie pionowe — przez dodanie lub odjęcie długości linii od argumentu. Wykonanie obu operacji daje w efekcie ruch ukośny.

Poprzez różnorodne manipulowanie bajtami argumentu LMS możemy uzyskać różne rodzaje uporządkowań. Linie mogą przesuwać się względem siebie lub wzajemnie przeskakiwać. Część z tych efektów można uzyskać za pomocą konwencjonalnego manipulowania obrazem, lecz dla ich osiągnięcia musi być przemieszczonych więcej danych. Zaletą przesuwania za pomocą argumentów LMS jest jego szybkość. Zamiast manipulowania całym ekranem danych, o wielkości wielu tysięcy bajtów, wystarcza manipulowanie dwoma lub kilkoma tuzinami bajtów.

Drugą interesującą zaletą komputerów Atari jest możliwość płynnego przesuwania, tj. przesuwania w odstępach mniejszych niż wielkość znaku. Odbywa się ono w krokach jednej linii ekranu pionowo oraz jednego taktu koloru poziomo. Ten typ przesuwania nie może iść zbyt daleko; dla otrzymania w pełni płynnego przesuwania na dużych odległościach, musimy połączyć płynne przesuwanie ze zgrubnym.

Istnieją dwa kroki dla implementacji płynnego przesuwania. Pierwszy, ustawiamy bity zezwolenia na płynne przesuwanie w bajtach rozkazów programu wyświetlania dla linii obrazu, w których chcemy mieć taki ruch. W większości przypadków interesuje nas cały ekran, zatem ustawiamy odpowiednie bity we wszystkich rozkazach programu wyświetlania. Bit D5 rozkazu zezwala na przesuwanie pionowe, a bit D4 — na poziome. Następnie zapamiętujemy żądaną odległość przesuwania w odpowiednim rejestrze. Istnieją dwa, jeden dla przesuwania poziomego, drugi dla pionowego. Rejestr przesuwania poziomego (HSCROL) jest pod adresem $D404(54276), rejestr przesuwania pionowego (VSCROL) — pod adresem $D405(54277). Dla ruchu poziomego musimy zapamiętać w HSCROL liczbę taktów koloru, o którą chcemy przesunąć linie obrazu. Dla ruchu pionowego zapamiętujemy w VSCROL liczbę linii ekranu, o którą chcemy przesunąć linie obrazu. Zapamiętane wartości przesuwania będą używane dla każdej linii, dla której dozwolone jest płynne przesuwanie.

Podczas używania płynnego przesuwania możemy napotkać dwa problemy. Oba wynikają z faktu, że częściowo przesunięty obraz pokaże więcej informacji niż normalny. Rozważmy, na przykład, co się zdarzy, kiedy przesuniemy poziomo linię o pół znaku w lewo. W linii jest 40 znaków. Połowa pierwszego znaku zniknie za lewym brzegiem ekranu. Czterdziesty znak przesunie się w lewo. Co zajmie jego miejsce? Powinna przesunąć się połowa nowego znaku i zająć miejsce przesuniętego teraz czterdziestego znaku. Mógłby to być 41 znak. Lecz co się zdarzy, jeśli w normalnej linii jest tylko 40 znaków?

Jeśli zastosujemy przesuwanie zgrubne, wtedy 41 znak nieoczekiwanie pojawi się na ekranie po zniknięciu pierwszego znaku za lewym brzegiem. Rozwiązanie tego problemu zostało wbudowane w sprzęt. Istnieją trzy możliwe szerokości ekranu: wąski (o szerokości 128 taktów koloru), normalny (o szerokości 160 taktów koloru) i szeroki (o szerokości 192 taktów koloru). Opcje te wybierane są przez ustawienie odpowiednich bitów w rejestrze DMACTL. Podczas używania płynnego przesuwania poziomego ANTIC automatycznie pobiera więcej danych z RAM niż wyświetla. Jeśli, na przykład, DMACTL ustawiony jest dla normalnej szerokości ekranu, który w trybie 0 BASIC-u ma 40 bajtów na linię, to ANTIC będzie pobierał ilość danych odpowiednią dla szerokiego ekranu — 48 bajtów na linię. Jeśli nie będzie to brane pod uwagę, to spowoduje przesunięcie linii w poziomie. Problem ten nie wystąpi, jeśli zorganizowaliśmy już RAM w długie linie poziome.

Podobny problem dla przesuwania pionowego może być obsługiwany jednym z dwóch sposobów. Pierwszym, niezbyt starannym jest zignorowanie go. Wtedy nie otrzymamy półobrazów na obu końcach obrazu. Zamiast tego obrazy u dołu ekranu nie będą przesuwać się właściwie; będą one pojawiać się nieoczekiwanie. Dla uzyskania właściwego płynnego przesuwania do I z rejonu ekranu musimy przeznaczyć jedną linię obrazu do działania jako bufor. Wykonamy to przez nieustawianie bitu przesuwania pionowego w rozkazie programu wyświetlania ostatniej linii strefy obrazu przesuwanej pionowo. Okno przesunie się teraz bez nieprzyjemnego szarpnięcia. Obraz ekranu będzie skrócony o jedną linię. Zaleta przesuwania obrazów staje się teraz widoczna. Możliwe jest tworzenie obrazów, które mają więcej niż 192 linie ekranu.

Płynne przesuwanie ma ograniczony zasięg. Pionowym ograniczeniem jest 15 linii ekranu, poziomym — 16 taktów koloru. Jeżeli spróbujemy przesunąć poza te granice, ANTIC zignoruje starsze bity rejestrów przesuwania. Dla uzyskania pełnego płynnego przesuwania, w którym cały ekran płynnie przesuwa się tak daleko, jak chcemy, musimy połączyć płynne przesuwanie ze zgrubnym. W tym celu najpierw płynnie przesuwamy obraz, śledząc jak daleko się przemieścił. Kiedy długość przesunięcia równa się wielkości znaku, zerujemy rejestr płynnego przesuwania i wykonujemy przesuwanie zgrubne

Program nr 4 ilustruje proste płynne przesuwanie realizowane z małą szybkoś-

```
FK 1 REM ****************
FJ 2 REM *              *
ZI 3 REM * Prosram nr 4 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
TR 8 HSCR=54276
VU 9 VSCR=54277
BS 10 GRAPHICS 0:LIST 1,200
SU 20 DL=PEEK(560)+256*PEEK(561)
XZ 30 POKE DL+10,50
YN 40 POKE DL+11,50
KR 50 FOR Y=0 TO 7
UN 60 POKE VSCR,Y
OT 70 GOSUB 200
OK 80 NEXT Y
WO 110 FOR X=15 TO 0 STEP -1
ZN 120 POKE HSCR,X
RD 130 GOSUB 200
LT 140 NEXT X
QG 150 GOTO 30
VG 200 FOR J=1 TO 200:NEXT J:RETU
   RN
JJ 205 FOR W=1 TO 50:NEXT W
NT 210 END
OK 220 GOTO 170
NX 230 END
```

cią. Demonstruje on kilka problemów, które wystąpią podczas przesuwania. Pierwszy, linie obrazu poniżej przesuwanego okna są przesunięte w prawo. Jest to zgodne z wcześniejszym opisem, że ANTIC automatycznie pobiera 48 bajtów zamiast 40. Problem ten występuje tylko w nierealistycznych programach demonstracyjnych, takich jak ten. W rzeczywistych zastosowaniach uporządkowanie danych ekranu wyklucza ten problem. Drugi, bardziej poważny, występuje, kiedy rejestry przesuwania są modyfikowane w czasie, gdy ANTIC jest w środku procesu wyświetlania. Wprowadza to pewne zamieszanie i powoduje, że obraz na ekranie drży. Rozwiązaniem jest zmiana rejestrów przesuwania tylko w czasie okresów przerwy pionowej. Może to być wykonywane tylko za pomocą procedur języka asemblera.

Zastosowania pełnego płynnego przesuwania są niezliczone. Oczywistym jest, na przykład, wykorzystanie go do przeglądania dużych map, utworzonych za pomocą grafiki znakowej. Używając trybu 2 BASIC-u można utworzyć dużą mapę, która zawiera np.: 10 pełnoekranowych obrazów. Ekran staje się oknem dla mapy. Użytkownik może przesuwać całą mapę za pomocą manipulatora. Taki system jest bardzo efektywny pod względem zajętości pamięci.

W ten sposób mogą być również prezentowane duże schematy elektroniczne. Manipulator może być używany zarówno do poruszania się po schemacie, jak i wskazywania określonych składników, które użytkownik chce wybrać. Za pomocą tej techniki może być prezentowany dowolny duży obraz, który nie musi być oglądany w całości.

W podobny sposób mogą być przeglądane duże bloki tekstu. Wprawdzie czytanie ciągłych bloków tekstu przez przesuwanie obrazu nie jest zbyt wygodne, jednak może być używane, na przykład, do sprawdzenia, czy tekst został właściwie zredagowany. Wykorzystano to w programie ATARI WRITER. Na ekranie dysponującym 40 znakami w linii możemy obejrzeć zredagowaną stronę (taką, jak otrzymamy po wydrukowaniu) zawierającą np.: 80 znaków w linii. Technika ta bardziej przydaje się do prezentowania oddzielnych bloków tekstu. Szczególnie interesujące może być zastosowanie jej do tworzenia rozwijających się menu.

Program nr 5 prezentuje jeden z możliwych do uzyskania efektów. Ma on zmodyfi-

```
FK 1 REM ****************
FJ 2 REM *              *
AC 3 REM * Prosram nr 5 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
BO 8 DIM SC$(17):FOR I=1 TO 17:RE
   AD A:SC$(I,I)=CHR$(A):NEXT I
MW 9 GRAPHICS 0:POKE 752,1:POKE 6
   22,255
NU 10 POKE 1536,112
BF 20 POKE 1537,66:POKE 1538,64:P
   OKE 1539,156
KM 25 FOR I=0 TO 23:POKE 1540+I,2
   :NEXT I:POKE 1564,112
BD 30 POKE 1565,82:POKE 1566,0:PO
   KE 1567,100:POKE 1568,65:POKE
   1569,0:POKE 1570,6
KD 40 POKE 560,0:POKE 561,6:POKE
   54276,7:LIST 0,70
HR 50 FOR I=0 TO 920:POKE 25640+I
   ,PEEK(40000+I):NEXT I
MV 55 X=0:Y=100:POKE 1567,Y:POKE
   1566,X
OK 56 FOR J=0 TO 230
VG 60 FOR I=7 TO 0 STEP -1:POKE 5
   4276,I:GOSUB 100:NEXT I
```

```
PL 62 X=X+2:IF X>255 THEN X=0:Y=Y
   +1
JU 63 A=USR(ADR(SC$),7,X,Y)
YL 64 REM POKE 54276,7:POKE 1566,
   X:POKE 1567,Y
JP 65 NEXT J
UW 70 GOTO 55
VU 100 FOR T=0 TO 5:NEXT T:RETURN
QU 200 DATA 104,104,104,141,4,212
   ,104,104,141,30,6,104,104,141,
   31,6,96
JJ 205 FOR W=1 TO 50:NEXT W
OK 220 GOTO 170
HX 230 END
```

kowany program wyświetlania, dzięki czemu została wyraźnie wydzielona jedna linia u dołu ekranu. Otrzymujemy w niej LIST programu z przesuwaniem w poziomie, a nie w pionie. Podprogram w języku maszynowym (umieszczony w zmiennej SC$) umożliwia zrealizowanie płynnego przesuwania poziomego na dużej odległości.

Program nr 6 zamienia ekran naszego telewizora w przesuwający się papier, na którym dwa pisaki kreślą linie. Jeden pisak stoi w miejscu w celu oznaczenia linii środkowej,

```
FK 1 REM ****************
FJ 2 REM *              *
RM 3 REM * Prosram nr 6 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
QZ 50 GOSUB 800
RK 60 GOSUB 900
SX 70 POKE X2,0
OS 80 POKE 1675,0:REM USTAWIENIE
   SZYBKOSCI
SW 90 START=100
VU 100 ? :? :? :? :? "NACISNIJ <S
   TART> DLA NOWEGO PRZEBIEGU"
RM 110 IF PEEK(53279)<>6 THEN 110
TE 120 GOSUB 700
HP 130 W=INT(PEEK(53770)/32)
AG 140 WYNIK=W:PHI=T(W)
UK 150 Q=-16:PROBA=100
IK 160 FOR PR=1 TO PROBA-1:Q=-Q:S
   =Q+16
ON 170 R=PEEK(53770)
VT 180 IF PEEK(53279)=3 THEN R=25
   6-R
FU 190 IF R>127 THEN 220
ZH 200 W=W+1:IF W=8 THEN W=7
NA 210 GOTO 230
OM 220 W=W-1:IF W=-1 THEN W=0
TT 230 WYNIK=WYNIK+W:PHI=T(W)
BC 240 FOR Z=32+S TO 63+S
TG 250 POKE X1,PEEK(SB+Z-PHI)+PEE
   K(SB+Z+PHI-1)
UJ 260 NEXT Z:NEXT PR
FN 270 GOSUB 790:? :? :? :?
ES 280 ? "   WARTOSC JEST DODATNI
   A LUB UJEMNA"
UA 290 ? "   JESLI SREDNIA AMPLIT
   UDA BYLA"
NG 300 ? "   POWYZEJ LUB PONIZEJ
   OCZEKIWANEJ."
GZ 310 ? :? :? "     WARTOSC = ";W
   YNIK-PROBA*7/2
DJ 320 ? :? :? :GOTO START
XL 700 GRAPHICS 7+16:POKE 559,0
DI 710 SETCOLOR 4,0,0:SETCOLOR 2,
   1,8:SETCOLOR 0,0,4
KP 720 PG=PEEK(106)
TT 730 POKE 203,56:POKE 204,PG-1
CJ 740 POKE 205,PEEK(560):POKE 20
   6,PEEK(561)
LF 750 KEEPL=PEEK(546):KEEPH=PEEK
   (547)
HZ 760 POKE 546,135:POKE 547,6
KP 770 POKE 559,34:RETURN
CO 780 POKE 559,0:GRAPHICS 7+16:P
   OKE 546,KEEPL:POKE 547,KEEPH
ED 790 GRAPHICS 0:RETURN
VZ 800 X1=1664:X2=1365
VX 810 DATA 200,200,48,192,48,12,
   3,216,165,20,41,0,24,200,65,16
   5
YO 811 DATA 203,109,130,6,133,203
   ,144,26,165,204,105,0,41,15,72
   ,13
AC 812 DATA 175,6,133,204,104,200
   ,11,174,130,6,169,0,202,157,0,
   112
OR 813 DATA 200,250,172,130,6,169
   ,0,136,145,203,208,251,160,4,1
   77,205
RP 814 DATA 109,130,6,145,205,200
   ,177,205,105,0,41,15,109,175,6
   ,145
WV 815 DATA 205,169,255,141,250,6
   ,173,128,6,32,234,6,169,65,141
   ,250
GW 816 DATA 6,173,129,6,32,234,6,
   76,95,228,201,100,176,17,72,74
WM 817 DATA 74,168,104,41,3,170,1
   89,131,6,41,255,17,203,145,203
   ,96
HF 820 DATA 559,0
UW 830 FOR N=1664 TO 1791
FM 840 READ DAT:POKE N,DAT:NEXT N
DS 850 POKE 1711,PEEK(106)-16
XO 860 POKE 559,34:RETURN
HY 900 DIM T(8)
LN 910 POKE 559,0
MG 920 DEG :SB=1536
FR 930 FOR N=0 TO 127
QV 940 POKE SB+N,40*SIN(N*45/8)+4
   0:NEXT N
AP 950 T(0)=16:T(1)=15:T(2)=14:T(
   3)=13
VG 960 T(4)=11:T(5)=9:T(6)=6:T(7)
   =0
RP 970 POKE 559,34
ZV 980 RETURN
```

a drugi wykonuje ruch sinusoidalny ze zmieniającą się losowo amplitudą. Przy każdym przekraczaniu linii środkowej podejmowana jest losowo decyzja o zwiększeniu lub zmniejszeniu amplitudy dla następnego półokresu (z uwzględnieniem dolnego i górnego ograniczenia amplitudy)

Każda binarna decyzja (zwiększenie lub zmniejszenie) wynika z kombinacji decyzji człowieka i komputera. W komórce o adresie 53770(RANDOM) występuje szybko zmieniający się ciąg liczb pseudolosowych z przedziału 0—255. Podczas każdego przecinania linii środkowej komputer decyduje o zwiększeniu lub zmniejszeniu amplitudy pisaka w zależności od tego, czy aktualna liczba losowa jest większa, czy mniejsza od 127. Określa to przebieg zdarzeń do czasu naciśnięcia przez użytkownika klawisza OPTION. Jeśli jest on naciśnięty w czasie przecinania linii środkowej, wtedy decyzja komputera, o zwiększeniu lub zmniejszeniu, jest odwracana.

Po pewnej liczbie okresów program kończy działanie i wyświetlany jest wynik. Wartość dodatnia lub ujemna wskazuje, że średnia amplituda ruchomego pisaka była większa lub mniejsza od oczekiwanej w tym przebiegu.

Opisując manipulowanie obrazem za pomocą rozkazów LMS programu wyświetlania nie można pominąć przesuwania „bardzo zgrubnego", czyli wymiany całych obrazów na ekranie.

Konwencjonalny sposób, stosowany w większości komputerów, polega na kopiowaniu zawartości wybranego obszaru RAM do obszaru pamięci wyświetlanego na ekranie. Czynność taka może być czasochłonna, dlatego musi być realizowana przez procedurę w języku maszynowym. Pomimo tego, podczas przesyłania kilku tysięcy bajtów, może być zauważalne opóźnienie zmiany obrazu

Zamiast przesyłania setek i tysięcy bajtów, w Atari wystarczy zmienić zawartość dwóch (lub czterech) bajtów, aby uzyskać taki sam efekt w znacznie krótszym czasie Możliwe jest również przygotowanie kilku obrazów i szybkie wyświetlanie ich na ekranach w określonej kolejności

Opisany sposób demonstruje **program nr 7**. Po uruchomieniu następuje krótka przerwa, w czasie której komputer zapisuje ekran nr 2. Następnie widoczne jest zapisy-

```
FK 1 REM ****************
FJ 2 REM *              *
BQ 3 REM * Prosram nr 7 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
FL 100 DL=PEEK(560)+256*PEEK(561)
ZW 110 MEMTOP=PEEK(742)
TO 120 EKRAN1=PEEK(89):EKRAN2=MEM
   TOP-5
PT 130 POKE 89,EKRAN2:POKE 106,EK
   RAN2+4:? CHR$(125)
WB 140 FOR X=1 TO 22:? "*********
      EKRAN 2       *********":NE
   XT X
NX 150 POKE 89,EKRAN1:POKE 106,EK
   RAN1+4:? CHR$(125)
XL 160 FOR X=1 TO 22:? "---------
      ekran 1       ---------":NE
   XT X
WG 180 OPEN #1,4,0,"K:":GET #1,X:
   CLOSE #1:X=X-48:IF X<1 THEN 18
   0
TO 190 ON X GOTO 210,220
OR 200 GOTO 180
UA 210 GOSUB 250:POKE DL+5,EKRAN1
   :GOTO 180
YG 220 GOSUB 250:POKE DL+5,EKRAN2
   :GOTO 180
OK 230 END
M 250 SOUND 0,100,10,15:FOR X=1
   TO 20:NEXT X
K 260 SOUND 0,0,0,0:RETURN
```

wanie informacji na ekranie nr 1. Naciskając klawisze 1 lub 2 otrzymujemy na ekranie jeden z dwóch obrazów

Proszę zwrócić uwagę na linie 130 i 150 Pierwsza z nich umożliwia zapisanie obrazu w obszarze pamięci, który nie jest wyświetlany, tak jakby to był obszar pamięci ekranu. W taki sposób możemy oszukać system operacyjny i przygotować jeden lub więcej obrazów, które później wyświetlimy na ekranie.

*Przygotowując ten artykuł korzystaliśmy z opisów i programów zamieszczonych w: **De Re Atari, Second book of Atari** oraz **Second book of Atari graphics.***

Tomasz MROWIEC
Ludwik PIELA

Uwaga!

Dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną linii dla Edytora BASIC-u („IKS" nr 4 z 1987 r.). Nie wprowadzamy ich do komputera.

Występujące w programach podkreślone znaki należy wprowadzić w negatywie.

Czy komputery lubią kolejki?

Kolejki już od dawna wrosły w naszą rzeczywistość. Nie można wyobrazić sobie pracy jakiejkolwiek kasy bez kolejek.

Postawmy nasz komputer np. w budce, w której sprzedają lody. Przychodząc wrzucasz do automatu pieniądze i wtedy możesz wpisać się do kolejki. Ponieważ w kolejce nie ma wielu klientów wystarczą tylko inicjały i np. wiek. Komputer musi wiedzieć, ile można danej osobie sprzedać kulek lodów.

Gdy przyjdzie pora, automat zawoła cię i wyda lody. Oczywiście natychmiast zapomina o tym, że kupowałeś lody. Nawet komputer *„nie ma głowy"* do zapamiętania wszystkich klientów.

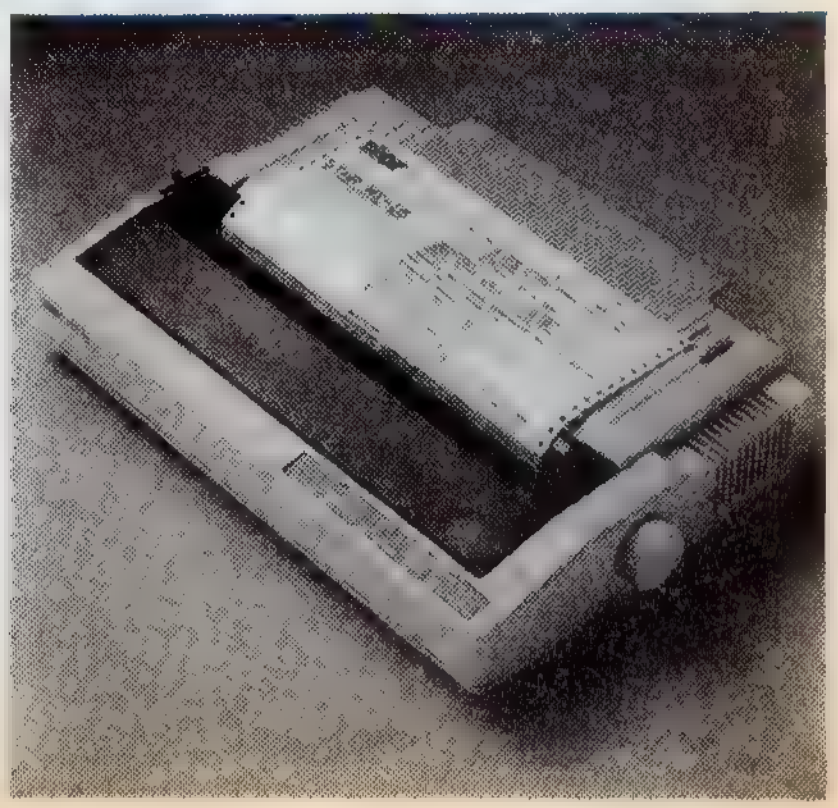
Dla zainteresowanych opowiem, jak tworzymy taką kolejkę. Stojąc w kolejce, musimy pamiętać, za kim stoimy. Warto również zapamiętać, kto stoi z tyłu. W zasadzie inne wiadomości nie są ważne. Komputer również tak czyni. Pozostaje jeszcze problem przed kim stoi pierwszy klient i za kim ostatni. W tym celu tworzymy dodatkowego sztucznego klienta (tzw. głowę kolejki), który stoi przed pierwszym, a za ostatnim klientem. Przy tak zorganizowanej kolejce można pozwolić sobie np. na wejście w środek kolejki, choć nie staliśmy w niej od początku. Wystarczy wmówić panu, za którym chcemy stanąć, że staliśmy za nim, a panu, przed którym stajemy, że staliśmy przed nim. Oczywiście nasza kolejka nie przewiduje takich możliwości, lecz możecie w ramach treningu dopisać taką procedurę.

Prezentowany program napisany w języku Pascal operuje na strukturze nie występującej w Basicu. Jest nią REKORD. Rekord jest podobny do wektora (tablicy o jednym wierszu), lecz o ile w wektorze każdy jego element jest jednego typu, np. liczby rzeczywiste, o tyle w rekordzie kolejne jego elementy mogą być różnych typów. W naszym programie inicjały są typu znakowego (char), zaś wiek i liczba kulek typu całkowitego.

```
 { *****************************
   *      PROGRAM KOLEJKA       *
   *          autor:            *
   *    Cezary Dobrowolski      *
   ***************************** }

Program kolejka;
label 10,40;
type
  t1=record nazw,imie: char;
            lat,il:integer;
            end;
  t2=array[1..100] of t1;
var
  i,ik:integer;
  kol:t2;
  a:char;
  procedure wejscie(var ik:integer;var kol:t2);
  label 15,20;
  var
    a:char;
    cena,wartosc,ilosc,reszta:integer;
 begin
15:
  if ik=100 then
  begin
   writeln('brak miejsc w kolejce');
   writeln('nacisnij ENTER');readln(a);
   goto 20;
  end;
  cena:=15;
  write('podaj wplate? ');readln(wartosc);
  write('ile chcesz kupic? ');readln(ilosc);
  reszta:=wartosc-cena*ilosc;
  writeln('wyplata reszty',reszta);
  if reszta<0 then begin writeln ('za malo wplaciles');readln(a); goto 20;end;
  ik:=ik+1;
  kol[ik].il:=ilosc;
  write('podaj pierwsza litere nazwiska klienta:   ');readln(kol[ik].nazw);
  write('podaj pierwsza litere imienia klienta:   ');readln(kol[ik].imie);
  write('ilosc lat:   ');readln(kol[ik].lat);
  write('czy jeszcze ktos chce kupic lody (T/N)');readln(a);
  if a='t' then goto 15;
20:
  end;
 procedure obsluga (var ik:integer; var kol:t2);
 label 30;
  var i:integer;
  begin
   if ik=0 then
   begin
    writeln('nie ma klientow do obslugi');
    writeln('nacisnij ENTER');
    goto 30;
   end;
  writeln('obsluga klienta: ',kol[1].nazw,'  ',kol[1].imie,'  ',kol[1].lat);
  for i:=2 to ik do
  begin
   kol[i-1].nazw:=kol[i].nazw;
   kol[i-1].imie:=kol[i].imie;
   kol[i-1].lat:=kol[i].lat;
  end;
  ik:=ik-1;
  writeln('nacienij enter');readln(a);
 30:
  end;
 procedure wydruk (ik:integer;kol:t2);
 var
  i:integer;
  begin
   for i:=1 to ik do
   begin
    writeln(i,'  ',kol[i].nazw,'  ',kol[i].imie,'  lat ',kol[i].lat);
   end;
   write('nacisnij ENTER');readln(a);
  end;
  begin (programu glownego)
   ik:=0;
 10:
   for i:=1 to 25 do begin writeln; end;
   writeln('wybiez opcje: ');
   writeln('1-wejscie klienta');
   writeln('2-obsluga klienta');
   writeln('3-lista klientow w kolejce');
   writeln('4-koniec pracy');
   readln(i);
   if i=1 then wejscie(ik,kol);
   if i=2 then obsluga(ik,kol);
   if i=3 then wydruk(ik,kol);
   if i=4 then begin writeln('Koniec pracy przepraszamy'); goto 40; end;
   goto 10;
 40:
  end.
```

Star NX-15



„Lilavati" — Okopy (1)

Mieczysław SKONIECZNY

Większości Czytelników nie trzeba tłumaczyć czym jest „Lilavati". Są to interesujące rozrywki matematyczne zebrane w ciekawą całość przez Szczepana Jeleńskiego. Jak zaznacza autor w przedmowie, celem tej publikacji jest „...*służyć przez dłuższy czas i wielokrotnie za podręcznik dostarczający Czytelnikowi okazji do pożytecznej rozrywki w chwilach wolnych, spędzanych czy to samotnie, czy też w gronie przyjaciół i znajomych*". Chcę zaproponować, aby w tej zabawie wziął udział także mikrokomputer. Artykuł ten rozpocznie cykl poświęcony rozwiązywaniu anegdot matematycznych zaczerpniętych w „Lilavati" z wykorzystaniem mikrokomputera.

Ze względu na charakter czasopisma na początku proponuję anegdotę prezentowaną przez S. Jeleńskiego pod numerem 55 o nazwie „W okopach".

Do okopu na froncie weszło 8 żołnierzy i oficer, który zajął stanowisko na lewym skrzydle. Z pewnych względów oficer musi znaleźć się na prawym skrzydle, co nie jest proste, gdyż okop jest ciasny, co uniemożliwia przesuwanie się poza żołnierzami. Na zewnątrz nie można nawet wysunąć głowy, bo nieprzyjaciel prowadzi ciągły ogień na nasze pozycje. Ale w okopie są trzy wgłębienia, w których może chwilowo stanąć jeden żołnierz. Problem do rozwiązania jest następujący: jak powinni poruszać się żołnierze, aby oficer przeszedł wzdłuż całego okopu i stanął na prawym skrzydle?

Charakterystyka programu

Wersja źródłowa programu, opracowanego w BASIC-u, jest następująca.

W instrukcjach 12—16 są wypełniane wartościami początkowymi 2 pomocnicze tablice 13-elementowe

a) td — zajętość określonej pozycji
 td(i)=0 — i-ta pozycja wolna
 td(i)=j — żołnierz nr j stoi na pozycji i.

b) w — współrzędne znakowe x dla rysowania głowy żołnierza w i-tej pozycji.

Program pracuje w 2 wariantach:

1) Samodzielne poszukiwanie rozwiązania problemu
2) Prezentacja rozwiązania modelowego (75 przesunięć).

Program w instrukcjach 20—180 zapoznaje użytkownika z problemem i wybiera żądany wariant pracy.

Instrukcje 300—499 to realizacja wariantu 1. Po zapoznaniu z zasadami gry, użytkownik samodzielnie rozwiązuje anegdotę, wprowadzając przed każdym posunięciem numer przesuwanego żołnierza (1—9) oraz numer pozycji, na której ma stanąć żołnierz (1—3).

Instrukcje 460—480 badają czy ruch jest możliwy. Informacje o każdym przesunięciu są wyświetlane w górnym okienku ekranu.

Instrukcja 500—550 to realizacja wariantu 2. Parametry przesunięcia są pobierane z DATA i automatycznie wykonuje się ruch. W ten sposób odbywa się 75 przesunięć.

```
5 DIM td(13):DIM w(13)
10 kod=0:MODE 1:INK 0,24:INK 1,0:INK 2,9:INK 3,6:PAPER 2:BORDER 9:PEN 1
11 WINDOW £1,1,40,1,6:PAPER £1,0:CLS £1
12 CLS:RESTORE 12:FOR i=1 TO 13:READ w(i):NEXT i
13 DATA 3,7,11,15,19,23,27,31,35,39,15,23,31
15 td(1)=0:td(10)=1:FOR i=2 TO 9:td(i)=i:NEXT i
16 FOR i=11 TO 13:td(i)=0:NEXT i
20 GOSUB 9000:PRINT"Do okopu na froncie, w styczno?ci z nie-przyjacielem, weszlo 8 ?olnierzy (kolor
zielony, numery 2-9) i oficer (k
olor    czerwony, numer 1)."sGOSUB 1000
80 PAPER 2 :LOCATE 1,5:PRINT"Oficer musi z pewnych wzgl[d5w znale;"  si[ na prawym skrzydle. Okop je
st ciasnynie mo?e wi[c przesun;"
 si[ poza ?olnie-rzami.Na zewn!trz nie mo?na nawet wysu-"
90 PRINT"n;" glowy, bo nieprzyjaciel prowadzi    ci!gly ogie3. Ale w okopie s! trzy wgl[-bienia, w k
t5rych mo?e chwilowo stan[" ?ol
nierz."
100 PEN 0:LOCATE 1,25:PRINT"NACISNIJ DOWOLNY KLAWISZ"
110 IF INKEY$="" GOTO 110
120 LOCATE 10,12:FOR I=1 TO 541:PRINT" ";:NEXT I
130 PEN 1:LOCATE 1,13:PRINT"Jak powinni porusza" si[ ?olnierze, aby oficer przeszedl wzdlu? calego o
kopu i  stan!l na prawym skrzydl
e?"
140 LOCATE 1,18:PRINT"Wybierz jeden z wariant5w:              1.Sam szukasz rozwi!zania problemu.
    2.Chcesz obejrze" rozwi!zani
e modelowe    (75 przesuni["})."
150 LOCATE 1,23:INPUT"Podaj kod wariantu (1 lub 2):",kod
160 IF kod=1 GOTO 300
170 IF kod=2 GOTO 500
180 LOCATE 1,23:PRINT"Bl!d - zly kod wariantu          ":FOR i=1 TO 1000:NEXT i:GOTO 150
300 CLS:PRINT"Przesuni[cia dokonujesz podaj!c:        -nr przesuwanego ?olnierza;           -nr p
ozycji, na kt5rej ma stan!"."
310 PRINT"Po podaniu ka?dego numeru nale?y wcisn!"klawisz ENTER. W bialym oknie na g5rze  ekranu s!
wy?wietlane informacja doty-  cz
!ce ka?dego przesuni[cia."
320 PRINT:PRINT"Gra ko3czy si[ sukcesem po przesuni[ciu dow5dcy (kolor czerwony - nr1) na pozy- cj[
nr 1(lewy skraj okopu) i ustawie
nie ?olnierzy w pocz!tkowym szyku tzn. ?ol- nierz nr2 na pozycji nr2,...,?olnierz   nr9 na pozycji n
r9."
330 PRINT:PRINT"?olnierze maj! swoje numery wypisane na piersi, natomiast numery pozycji s! wi- docz
ne na zewn!trz okopu.":PRINT
340 PRINT"?yczymy powodzenia w ilo?ci ruch5w      mniejszej ni? 75 (rozwi!zanie modelowe).
350 PEN 0:LOCATE 1,25:PRINT"NACISNIJ DOWOLNY KLAWISZ"
360 IF INKEY$="" GOTO 360
370 CLS:GOSUB 1000:ruch=1
380 PAPER 2:LOCATE 1,8:INPUT "Podaj nr przesuwanego ?olnierza: ",nr2
390 IF nr2>0 AND nr2<10 GOTO 420
400 LOCATE 1,8:PRINT"Bl!d - podano zly numer ?olnierza    "
410 FOR t=1 TO 1000:NEXT t:GOTO 380
420 LOCATE 1,9:INPUT "Podaj nr pozycji na,kt5rej ma stan!":",nr1
430 IF nr1>0 AND nr1<14 GOTO 455
440 LOCATE 1,9:PRINT"Bl!d - podano zly numer pozycji.
450 FOR t=1 TO 1000:NEXT t:GOTO 420
455 CLS £1:IF td(nr1)=0 GOTO 460
456 LOCATE £1,13,4:PEN £1,3:PRINT £1,"POZYCJA ZAJOTA"
457 FOR t=1 TO 1500:NEXT t:CLS £1
459 LOCATE 1,8:FOR i=1 TO 80:PRINT" ";:NEXT i:GOTO 380
460 FOR i=1 TO 13:IF td(i)=nr2 GOTO 462
461 NEXT i
462 IF nr1=1 AND i=2 GOTO 490
463 FOR j=2 TO 9
464 IF nr1=j AND i=j-1 GOTO 490
465 IF nr1=j AND i=j+1 GOTO 490
466 NEXT j
467 IF nr1=10 AND i=9 GOTO 490
468 IF nr1=4 AND i=11 GOTO 490
469 IF nr1=11 AND i=4 GOTO 490
470 IF nr1=12 AND i=6 GOTO 490
471 IF nr1=6 AND i=12 GOTO 490
472 IF nr1=8 AND i=13 GOTO 490
473 IF nr1=13 AND i=8 GOTO 490
480 LOCATE £1,11,4:PEN £1,3:PRINT £1,"RUCH NIEDOZWOLONY":GOTO 457
490 GOSUB 5000:PAPER 2:ruch=ruch+1
495 IF td(1)<>1 GOTO 459
496 FOR p=2 TO 9:IF td(p)<>p GOTO 459
497 NEXT p
499 LOCATE 1,8:FOR p=1 TO 80:PRINT" ";:NEXT p:GOTO 600
500 CLS:GOSUB 1000:FOR ruch=1 TO 75
505 PEN 0:PAPER 2:LOCATE 14,8:PRINT"GRA POKAZOWA"
510 READ nr1,nr2
520 DATA 1,2,2,3,3,4,4,5,11,5,3,6,4,6,6,7,5,7,7,8,6,8,8,9,7,9,9,1,8,1,12,1,8,9,9,9,10,9,7,8,8,8,9,8,
8,1,7,1,6,1,12,1,6,7,7,7,8,7,13,
7,5,6,6,6,7,6,8,6,4,5,5,5,6,5,7,5
530 DATA 6,1,5,1,4,1,11,1,4,4,5,4,6,4,12,4,3,3,4,3,5,3,6,3,2,2,3,2,4,2,5,2,4,1,3,1,2,1,1,1,4,2,3,2,2
,2,5,3,4,3,3,3,6,4,5,4,4,4,6,5,5
,5,7,6,6,6,8,7,7,7,8,8,9,9
540 GOSUB 5000
550 NEXT ruch
600 PEN 0:LOCATE 12,8:PRINT"OGI)5NIZE0 SUKCES"
610 FOR i=0 TO 26:FOR t=1 TO 100:NEXT t:BORDER i:NEXT i
620 GOTO 10
```

Charakterystyka podprogramów

Poszczególne czynności wykonywane w trakcie realizacji programu są zorganizowane w postaci następujących podprogramów

1) rysowanie okopu na ekranie i początkowego rozmieszczenia żołnierzy (instrukcje 1000—1170).

```
1000 PEN 1:PAPER 2:FOR i=1 TO 40:LOCATE i,24:PRINT CHR$(131):NEXT i
1020 PAPER 0: FOR i=1 TO 37 STEP 4
1030 LOCATE i,18:PRINT CHR$(135);CHR$(131);CHR$(131);CHR$(131);
1040 FOR j=19 TO 23
1050 LOCATE i,j:PRINT CHR$(133);"   ":NEXT j:NEXT i
1060 FOR i=13 TO 29 STEP 8
1070 PAPER 0:LOCATE i,12:PRINT CHR$(135);CHR$(131);CHR$(131);CHR$(131);:PAPER 2:PRINT CHR$(133)
1080 FOR j=13 TO 17
1090 PAPER 0:LOCATE i,j:PRINT CHR$(133);"   ";:PAPER 2:PRINT CHR$(133):NEXT j:NEXT i
1100 a=2:y=19:FOR x=39 TO 7 STEP -4
1105 FOR t=1 TO 1000:NEXT t
1110 GOSUB 2000:NEXT x
1120 a=3:x=39:GOSUB 2000
1130 PEN 1: FOR i=2 TO 9:PAPER 2:LOCATE 4*i-1,20:PRINT USING "£";i:NEXT i
1140 PAPER 3:LOCATE 39,20:PRINT"1"
1145 IF kod=0 GOTO 1170
1150 PAPER 2:PEN 1:FOR i=1 TO 10:LOCATE 4*i-2,25:PRINT USING "££";i:NEXT i
1160 FOR i=0 TO 2:LOCATE 14+i*8,11:PRINT USING "££";11+i:NEXT i
1170 RETURN
```

2) Rysowanie jednego żołnierza w wybranej pozycji okopu (instrukcje 2000—2100).

Przed wywołaniem podprogramu należy nadać wartość następującym parametrom:

a — kolor żołnierza (a=2 kolor zielony, a=3 kolor czerwony — dowódca)

x, y — współrzędne znakowe głowy żołnierza.

```
2000 PAPER 0:PEN a:LOCATE x,y:PRINT CHR$(190)
2010 LOCATE x-1,y+1:PRINT CHR$(191);CHR$(143);CHR$(192)
2020 LOCATE x-1,y+2:PRINT CHR$(193);CHR$(143);CHR$(194)
2030 LOCATE x-1,y+3:PRINT CHR$(195);CHR$(196);CHR$(197)
2040 LOCATE x-1,y+4:PRINT CHR$(198);CHR$(199);CHR$(200)
2050 PLOT (x-1)*16+3,(26-y)*16-10:DRAWR 0,-4:DRAWR 2,-2:DRAWR 6,0:DRAWR 2,2:DRAWR 0,4
2060 PLOT (x-1)*16+1,(26-y)*16-40:DRAWR 14,0:DRAWR 0,-2:DRAWR -14,0
2070 PLOT (x-2)*16+1,(26-y)*16-34:DRAWR 0,-36:DRAWR 2,0:DRAWR 0,10:DRAWR 0,-20:DRAWR 2,0:DRAWR 0,12:
PLOTR 2,-4:DRAWR 0,-8
2100 RETURN
```

3) Wpisanie w okno informacji o przesunięciu (instrukcje 3000—3030)

Parametry:

ruch — numer kolejnego przesunięcia,

nr 1 — numer pozycji, na której ma stanąć żołnierz,

nr 2 — numer przesuwanego żołnierza,

```
3000 CLS £1:PEN £1,3:LOCATE £1,2,2:PRINT £1,"Przesuniecie nr:";ruch
3020 LOCATE £1,2,4:PEN £1,1:PRINT £1,"Przesuwam zolnierza nr:";nr2:PRINT £1," na pozycje nr:
:";nr1
3030 RETURN
```

4) Opróżnienie pozycji zajmowanej do tej pory przez żołnierza (instrukcje 4000—4020).

Parametry:

par, py — współrzędne znakowe pola, w którym była głowa żołnierza

```
4000 PAPER 0:LOCATE par,py:PRINT" "
4005 FOR m=1 TO 4
4010 LOCATE par-1,py+m:PRINT"   ":NEXT m
4020 RETURN
```

5) Realizacja i przesunięcie na ekranie (instrukcje 5000—5110)

```
5000 FOR i=1 TO 13
5010 IF td(i)=nr2 GOTO 5030
5020 NEXT i
5030 GOSUB 3000
5040 par=w(i):py=19:IF i>10 THEN py=13
5050 GOSUB 4000
5060 a=2:IF nr2=1 THEN a=3
5070 y=19:IF nr1>10 THEN y=13
5080 x=w(nr1):GOSUB 2000
5085 PAPER 2:IF nr2=1 THEN PAPER 3
5086 PEN 1:LOCATE x,y+1:PRINT USING "£";nr2
5090 IF kod=2 THEN FOR t=1 TO 1500:NEXT t
5100 td(nr1)=nr2:td(i)=0
5110 RETURN
```

6) Definicje własnych symboli graficznych (instrukcje 9000—9980)

```
9000 SYMBOL AFTER 170
9010 SYMBOL 171,0,0,120,12,124,204,118,3
9020 SYMBOL 172,24,60,102,102,126,102,102,3
9030 SYMBOL 173,24,0,60,102,96,102,60,0
9040 SYMBOL 174,24,60,102,192,192,102,60,0
9050 SYMBOL 175,0,0,60,102,126,96,60,6
9060 SYMBOL 176,254,98,104,120,104,98,254,3
9070 SYMBOL 177,56,24,28,24,56,24,60,0
9080 SYMBOL 178,240,96,112,96,226,102,254,0
9090 SYMBOL 179,24,0,220,102,102,102,102,0
9100 SYMBOL 180,24,198,230,246,222,206,198,0
9110 SYMBOL 181,24,0,60,102,102,102,60,0
9120 SYMBOL 182,12,56,108,198,198,108,56,0
9130 SYMBOL 183,24,0,60,96,60,6,124,0
9140 SYMBOL 184,24,60,96,60,6,102,60,0
9150 SYMBOL 185,24,0,126,76,24,48,126,0
9160 SYMBOL 186,254,198,140,126,48,102,254,0
9170 SYMBOL 187,12,24,126,76,24,48,126,0
9180 SYMBOL 188,48,254,172,24,50,102,254,0
9190 KEY DEF 10,1,55,171,172
9200 KEY DEF 11,1,56,173,174
9210 KEY DEF 3,1,57,175,176
9220 KEY DEF 20,1,52,177,178
9230 KEY DEF 12,1,53,179,180
9240 KEY DEF 4,1,54,181,182
9250 KEY DEF 13,1,49,183,184
9260 KEY DEF 14,1,50,185,186
9270 KEY DEF 5,1,51,187,188
9280 SYMBOL 190,60,126,126,255
9290 SYMBOL 191,0,3,7,15,15,14,14,14
9300 SYMBOL 192,0,192,224,240,240,112,112,112
9310 SYMBOL 193,14,14,14,30,62,60,120,112
9320 SYMBOL 194,112,112,112,112,112,112,112,112
9330 SYMBOL 195,112,96:SYMBOL 196,255,255,231,231,231,231,231,231
9340 SYMBOL 197,112,112,112:SYMBOL 198,0,0,0,0,0,1,7,7
9350 SYMBOL 199,231,231,231,231,231,231,231,231:SYMBOL 200,0,0,0,0,0,128,224,224
9980 RETURN
```

Symbole o numerach 171—188 to definicje znaków charakterystycznych dla języka polskiego (ą, ę, ć, ś).

Litery małe otrzymujemy przytrzymując klawisz SHIFT i wciskając jeden z klawiszy funkcyjnych (f0—f9)

Litery duże otrzymujemy analogicznie przytrzymując klawisz CONTROL.

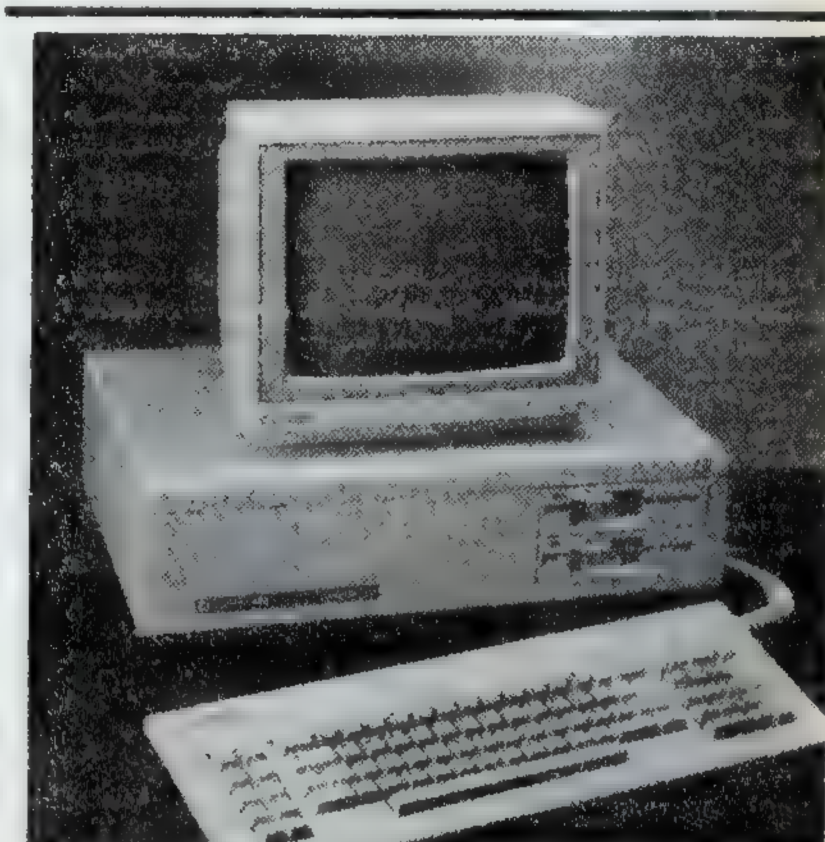
Symbole 190—200 to części składowe rysunku postaci żołnierza.

W trakcie samodzielnego szukania rozwiązania (wariant 1) należy unikać 2 sytuacji sygnalizowanych przez program:

POZYCJA ZAJĘTA — nie należy próbować przesuwać żołnierza na pozycję, na której stoi inny żołnierz.

RUCH NIEDOZWOLONY — dopuszczalne są tylko ruchy na pozycje sąsiadujące z pozycją zajmowaną przez żołnierza.

Na zakończenie życzę Czytelnikom udanych poszukiwań w liczbie ruchów mniejszych niż 75 (rozwiązanie modelowe proponowane przez autora „LILAVATI")





Komputer kontra miś

— Mamusiu, dzisiaj jest wtorek — pyta po przebudzeniu się pięcioletni Karol

— Tak syneczku. Dlaczego się ubierasz, przecież jesteś chory i nie idziesz do przedszkola.

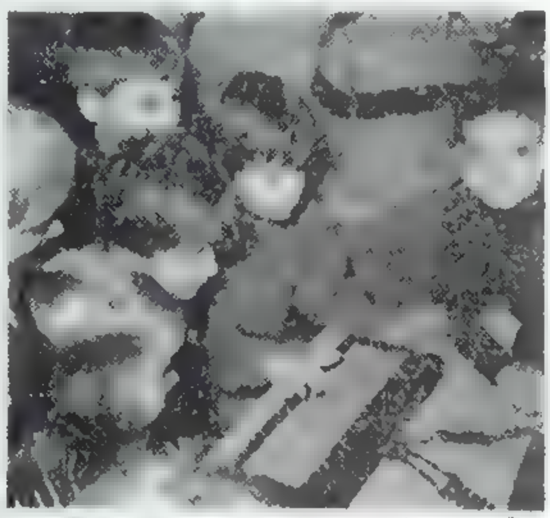
— Ale ja muszę? Dzisiaj są zajęcia z komputerami.

Gdy „zaczarowana skrzynka" pojawia się w przedszkolu, takie sceny mają miejsce w niejednym domu. Dzieci momentalnie przestają się ociągać, ba nawet ponaglają rodziców, byleby szybciej znaleźć się wśród rówieśników

Wanda Kostrzyńska, dyrektorka Przedszkola nr 407 na osiedlu Gocław Lotnisko nie kryje swojego zadowolenia

— Dzięki pomocy i uprzejmości rodziców mogliśmy szybciej uruchomić tak bardzo na





nowym osiedlu potrzebne przedszkole. Rodzice sami wnosili meble, zbijali szafeczki, zawieszali firanki. Również nawiązałam współpracę z kołem ZSMP działającym przy Spółdzielni Mieszkaniowej Gocław-Lotnisko. Zetesempowcy wraz z pracownikami administracji, w ramach czynu społecznego wykonali m.in. uchwyty przy balustradzie, skrzynki do kwiatów. Pojawiają się w każdej wolnej chwili, na każde moje wezwanie, bo niestety drobnych, ale uciążliwych awarii, tzw. niedoróbek nie brakuje

Dobrze, że chociaż dzieci tymi przyziemnymi sprawami nie muszą się interesować Od czasu, gdy jeden z rodziców raz w tygodniu właśnie we wtorek przynosi komputer, w grupach pięcio- i sześciolatków nikogo nie brakuje

— W programie nauczania starszych grup przedszkolnych jeszcze nie uwzględniono zajęć z komputerem. Program jest stary sugerowały wychowawczynie, nie nastawiony na żadne nowinki A przecież dzieci są tak chłonne i podatne na nową wiedzę W mig chwytają wszelkie zawiłości związane z obsługą komputera. Wystarczy im po-



kazać, wytłumaczyć, by bezbłędnie powtórzyły podpatrzone czynności Co tydzień włączają inną dyskietkę, z inną grą.

— Co prawda, wolałybyśmy na ekranie monitora obserwować wraz z dziećmi programy ułatwiające m.in. poznanie cyferek czy liter niż gry komputerowe — dodała pani dyrektor. *— Właściwie, to nie mam co narzekać. Przynajmniej dzieci od podstaw zagłębiają się w abecadło i tajniki komputera..*

— Czy komputer zastąpi misia Puchatka — głośno zastanawia się wychowawczyni pani **Jolanta Szewczyk.** *— Uważam, że raczej nie Przecież nie można go ani przytulić, ani z nim się utożsamiać. Na pewno znajdzie swe miejsce obok misia, a nie zamiast niego Zaobserwowałam, że niektóre gry wywołują u dzieci agresję. Tak nie powinno być Może jestem w błędzie, może się nie znam, ale takie mam odczucie. Gdy nadejdzie taki dzień, że komputery zadomowią się na stałe w naszych mieszkaniach, to nie powinny być traktowane, jako wypełniacz czasu*

Dla mnie przerażający jest fakt, iż przez telewizory emitowane programy tak bardzo absorbują naszych podopiecznych. Przedszkolaki w poniedziałki nic nie robią tylko komentują poszczególne pozycje telewizyjne, nawet filmy wyświetlane późnym wieczorem...

Jednak co by się powiedziało, to wtorek w przedszkolu jest wielkim świętem Postępu nie da się zatrzymać Dobrze się dzieje, że dzieci nie zostają w tyle za nowinkami techniki, że zaczarowana skrzynka traci magię, a staje się wspaniałym towarzyszem intelektualnej zabawy.

(a)



MERITUM

„Nietoperze"

Program „Nietoperze" to prosta gra zręcznościowa. Zabawa polega na zestrzeleniu jak największej ilości latających obiektów przypominających nietoperze. Czas gry jest ograniczony, a najlepsze wyniki przedstawia lista rekordów. Na wydruku kody programu maszynowego podzielone są na bloki, co pozwala na stopniowe wpisywanie programu. Każdy blok posiada swoją sumę kontrolną, która wyklucza możliwość uruchomienia programu z błędnie wprowadzonymi wartościami. Linie 6 i 7 należy wpisać dopiero po uruchomieniu programu, gdyż blokują one klawisze BREAK i NMI.

Piotr BIEDRZYCKI

```
3       CLEAR 200  : GOSUB 3000
4       POKE 16396,175: POKE 16397,201
5       DIM R(5), R×(5): A=30000
6       OUT 250,0 'ZABLOKOWANIE NMI
7       POKE 16526,0: POKE 16527,100
9     '-----------------------------------
10      GOSUB  1000: GOSUB 2000
20      U=USR(0)
25      NW=PEEK(A)+100*PEEK(A+1)
30      GOSUB 620  : GOTO    10
35    '-----------------------------------
619      'SPRAWDZANIE WYNIKU
620         FOR H=1 TO 5
625          IF NW ) R(H) THEN 655
630          NEXT H
635      RETURN
654     'WYNIK JEDEN Z PIECIU NAJLEPSZYCH
655        FOR J=5 TO H STEP -1
660        R(J)=R(J-1): R×(J)=R×(J-1)
665        NEXT J
668        CLS: PRINT CHR×(23)
670        PRINT@ 512,"   TWOJE IMIE";
672       INPUT I×
673        I×=I×+"      "
675        R(H)=NW       : R×(H)=LEFT×(I×,6)
680      RETURN
999     'PIERWSZY EKRAN
1000      CLS: PRINT CHR×(23)
1005      PRINT: PRINT"           NIETOPERZE"
1010      PRINT: PRINT
1015      PRINT"     '1'-LEWO","PRAWO-':'"
1020      PRINT"        'SHIFT'-STRZAL"
1025      PRINT: PRINT"          LISTA REKORDOW"
1026      PRINT: FOR H=1 TO 5
1030      PRINT "     ";H;" - ";R×(H);R(H)
1035      NEXT H
1036      A×=INKEY× 'KASOWANIE KODU OSTATNIEGO KL.
1037      IF INKEY×="" THEN 1037
1040    RETURN
1999    'PLANSZA GRY
2000      CLS: PRINT CHR×(23)
2005      PRINT@ 960,CHR×(143);" ";
2010      PRINT CHR×(143);" ";CHR×(143);
2015      PRINT "  CZAS: 0000   WYNIK: 0000";
2020      FOR H=0 TO 127: SET(H,43):NEXT
2025     RETURN
2888  'KOD WEWNETRZNY ZAJMUJE OBSZAR
2889  'OD 6400H (25600)   DO 677FH (26495)
2890  'ADRES STARTOWY 6400H (25600)
2999       'LADOWANIE KODU MASZYNOWEGO
3000          S=0:A=25600:P=7:GOSUB 4000  'BLOK 1
3005 DATA E5,D5,C5,21,00,00,22,30,75,21,44,75,22,3A,75,21
3010 DATA 30,05,22,32,75,21,03,00,22,36,75,21,04,04,22,38
3015 DATA 75,21,60,3F,22,34,75,CD,2E,64,C1,D1,E1,C9,ED,73
3020 DATA 3E,75,3E,02,32,AF,40,3A,38,75,F5,21,7F,02,CD,9A
3025 DATA 0A,CD,C9,14,CD,37,0B,2A,21,41,11,00,3C,CB,85,A7
3030 DATA 19,EB,2A,3A,75,73,23,72,23,22,3A,75,F1,3D,20,DA
3035 DATA 21,44,75,22,3A,75,CD,DD,66,CD,04,65,3A,40,3B,CB
3040        IF S()10139  THEN PRINT "BLAD W BLOKU NR 1":END
3050          S=0: A=Z   :P=8:GOSUB 4000  'BLOK 2
3055 DATA 57,20,F9,CD,FD,66,21,38,75,3A,39,75,96,3D,06,A0
3060 DATA 10,FE,3D,20,F9,CD,04,65,CD,DE,65,06,4A,10,FE,10
3065 DATA FE,3D,20,F7,CD,46,67,CD,65,65,18,CD,3A,38,75,F5
3070 DATA 3E,02,32,AF,40,21,0C,00,CD,9A,0A,CD,C9,14,CD,37
3075 DATA 0B,11,3A,00,2A,21,41,A7,19,CB,85,EB,2A,34,75,4E
3080 DATA 23,46,C5,E1,3E,20,02,03,03,02,19,11,7C,3F,3E,02
3085 DATA 32,AF,40,CD,39,0A,FE,01,28,22,EB,2A,3A,75,73,23
3090 DATA 72,23,22,3A,75,3E,AC,D3,FE,12,13,13,3E,9C,12,D3
3095        IF S()12355  THEN PRINT "BLAD W BLOKJ NR 2":END
3100          S=0: A=Z   :P=8:GOSUB 4000  'BLOK 3
3105 DATA FE,F1,3D,20,AA,21,44,75,22,3A,75,C9,11,7C,03,A7
3110 DATA ED,52,18,D6,2A,34,75,3A,10,38,CB,4F,28,0E,2B,2B
3115 DATA 11,42,3F,CD,39,0A,FE,FF,28,17,18,13,3A,20,38,CB
3120 DATA 57,28,0E,23,23,11,78,3F,CD,39,0A,FE,01,28,02,18
3125 DATA 03,2A,34,75,11,40,00,22,34,75,36,B9,A7,ED,52,36
3130 DATA B4,23,23,36,BC,23,23,36,B8,19,36,B6,28,2B,36,3F
3135 DATA 2A,34,75,01,08,00,2B,2B,36,20,ED,52,36,20,09,36
3140 DATA 20,19,36,20,C9,2A,34,75,01,04,00,11,40,00,3E,B9
3145        IF S()9777   THEN PRINT "BLAD W BLOKU NR 3":END
3150          S=0: A=Z   :P=8:GOSUB 4000  'BLOK 4
3155 DATA BE,20,14,ED,52,3E,B4,BE,20,0D,09,3E,B8,BE,20,07
3160 DATA 19,3E,B6,BE,20,01,C9,E5,CD,33,66,E1,CD,04,65,21
3165 DATA 00,00,01,FE,C0,C5,ED,B3,C1,10,FA,3A,36,75,3D,32
3170 DATA 36,75,FE,02,20,09,3E,20,32,C0,3F,D1,C3,69,64,FE
3175 DATA 01,20,09,3E,20,32,C4,3F,D1,C3,69,64,FE,00,20,09
3180 DATA 3E,20,32,C8,3F,D1,C3,69,64,16,C8,CD,ED,66,CD,ED
3185 DATA 66,16,14,CD,ED,66,CD,ED,66,EB,7B,3E,75,C9,3A,80
3190 DATA 38,CB,47,20,03,3E,32,C9,3E,0C,2A,34,75,11,40,00
3195        IF S()13734  THEN PRINT "BLAD W BLOKU NR 4":END
3200          S=0: A=Z   :P=8:GOSUB 4000  'BLOK 5
3205 DATA ED,52,22,3C,75,F5,2A,3C,75,11,40,00,ED,52,11,04
3210 DATA 00,22,3C,75,3E,20,BE,C4,33,66,36,95,19,BE,C4,33
3215 DATA 66,36,AA,F1,F5,07,0E,5A,D3,FE,47,10,FE,3D,D3,FE
3220 DATA 3C,0D,20,F4,2A,3C,75,36,20,19,36,20,F1,3D,20,C5
3225 DATA 3E,03,C9,3E,9C,E5,D5,BE,20,02,2B,2B,EB,3A,38,75
3230 DATA F5,2A,3A,75,4E,23,46,23,22,3A,75,C5,E1,CD,39,0A
3235 DATA A7,28,0D,F1,3D,20,E9,21,44,75,22,3A,75,D1,E1,C9
3240 DATA 36,20,23,23,36,20,2A,3A,75,E5,D1,1B,1B,01,00,03
3245        IF S()12564  THEN PRINT "BLAD W BLOKU NR 5":END
3250          S=0: A=Z   :P=8:GOSUB 4000  'BLOK 6
3255 DATA ED,B0,CD,BB,66,2A,3C,75,36,20,11,40,00,19,36,20
```

```
3260 DATA 3A,38,75,3D,20,1D,32,38,75,3E,48,21,00,11,0E,FE
3265 DATA 57,5F,ED,69,43,10,FE,ED,61,15,20,F5,D6,09,20,F0
3270 DATA F1,18,B4,3A,39,75,3C,3C,32,38,75,21,44,75,22,3A
3275 DATA 75,32,39,75,ED,7B,3E,75,C3,2E,64,3A,30,75,C6,05
3280 DATA 32,30,75,FE,64,C0,AF,32,30,75,3A,31,75,3C,32,31
3285 DATA 75,FE,14,C0,3D,32,31,75,ED,7B,3E,75,C9,16,14,CD
3290 DATA ED,66,16,B4,CD,ED,66,16,14,CD,ED,66,C9,0E,C8,D3
3295      IF S()13083  THEN PRINT "BLAD W BLOKU NR 6":END
3300          S=0: A=Z   :P=9:GOSUB 4000  'BLOK 7
3305 DATA FE,42,10,FE,10,FE,3C,D3,FE,0D,20,F3,C9,CD,33,67
3310 DATA 11,32,75,21,E0,3F,CD,0F,67,11,30,75,21,FC,3F,06
3315 DATA 02,CD,1A,67,13,2B,2B,10,F8,C9,1A,0E,FF,CD,2A,67
3320 DATA F6,30,77,2B,2B,79,F6,30,77,C9,0C,D6,0A,F2,2A,67
3325 DATA C6,0A,C9,21,32,75,35,C0,36,63,23,35,3E,FF,BE,C0
3330 DATA CD,DD,66,C3,C9,65,C3,76,67,3A,40,75,FE,01,28,13
3335 DATA 3E,01,32,40,75,3E,AD,32,E6,64,3E,9E,32,ED,64,32
3340 DATA 34,66,C9,3E,02,32,40,75,3E,AC,32,E6,64,3E,9C,32
3345 DATA ED,64,32,34,66,C9,CD,49,67,CD,9C,64,C9,00,00,00
3350      IF S()15675  THEN PRINT "BLAD W BLOKU NR 7":END
3800    RETURN   'KONIEC DANYCH
3999       'PODPROGRAM LADUJACY
4000        FOR Z=A  TO A+P*16-1
4005        READ A$:PRINTA$;"  ";
4010        N=ASC( LEFT$(A$,1))-48
4015        M=ASC(RIGHT$(A$,1))-48
4020        N=N-INT(N/12)*7
4025        M=M-INT(M/12)*7
4030        B=N*16+M: S=S+B
4035        POKE Z,B : NEXT Z
4040       RETURN
```



AMSTRAD „Generator"

Ten krótki program umożliwia wyświetlenie na ekranie kolejnych bajtów zadanego obszaru pamięci (np. kodu maszynowego własnego podprogramu) w postaci kolejnych (o ustalonej przez siebie numeracji i odstępie) wierszy DATA oraz zeskładowanie go na dyskietce (kasecie) w określonym, również przez siebie, zbiorze.

Otrzymany przez siebie zbiór można załadować do pamięci komendą LOAD i dokonywać poprawek używając edytora. Można go również dołączać do własnego programu w BASIC-u komendą MERGE.

Bajty przedstawione są w postaci heksadecymalnej (linia 230), a chcąc przedstawić je np. dziesiętnie, należy linię tę zmienić na: a$ = PEEK (x).

Ilość liczb w jednym wierszu DATA ustalona jest na 10 (zmienna c) i chcąc ją zmienić należy poprawić linię 260.

Cezary SOBCZAK

```
10 REM ***************
20 REM Generator
30 REM wierszy 'DATA'
40 REM ***************
50 MEMORY &2FFF
60 MODE 2
70 LOCATE 20,1:WINDOW #1,40,80,25,25
80 PRINT CHR$(24)"G E N E R A T O R  WIERSZY 'DATA'"CHR$(24)
90 PRINT:PRINT:PRINT
100 CLEAR INPUT
110 LOCATE 1,5: INPUT "Podaj poczatkowy adres pamieci ( > &2fff ) ",x
120 IF x<&2FFF THEN PRINT #1,"Musi byc WIEKSZY niz &2fff":FOR i=1 TO 4000:NEXT:CLS #1:LOCATE 1,5:PRINT SPACE$(79):GOTO 110
130 LOCATE 1,6:INPUT "Podaj koncowy adres pamieci                ",y
140 IF y<x THEN  PRINT #1, "Musi byc WIEKSZY niz poczatkowy":FOR i=1 TO 4000:NEXT:CLS #1:LOCATE 1,6:PRINT SPACE$(79):GOTO 130
150 LOCATE 1;7:INPUT "Numerowac linie 'DATA' od                ",z
160 IF z<1 OR z>=65536 THEN PRINT#1,"Numery moga byc od 1 do 65535":FOR i=1 TO 4000:NEXT:CLS #1:LOCATE 1,7:PRINT SPACE$(79):GOTO 150
170 LOCATE 1,8:INPUT "Odstep miedzy liniami                ",dz
180 LOCATE 1,9:INPUT "Nazwa zbioru na kopie linii 'DATA'         ",n$
190 CLS
200 WINDOW #1,2,77,4,22:PRINT "      G E N E R A T O R  WIERSZY  'DATA'"
210 OPENOUT n$
220 FOR x=x TO y
230 a$=HEX$(PEEK(x),2)
240 c=c+1
250 IF LEN(z$)=0 THEN z$=MID$(STR$(z),2)+" DATA "+a$  ELSE z$=z$+","+a$
260 IF c=10 OR x=y THEN PRINT #1, z$:PRINT #9,z$:z$="":c=0:z=z+dz
270 NEXT
280 CLOSEOUT
290 LOCATE 10,25:PRINT"K O N I E C          Kontynuacja  -  S P A C J A"
300 WHILE INKEY(47)=-1:WEND:GOTO 60
```

Sprzęg, łącznia, interfejs

Układ elektroniczny łączący urządzenia systemu cyfrowego i umożliwiający współpracę wg określonych zasad nazywany jest interfejsem. Jest to pewien standard techniczny rozwiązania problemu połączenia komunikacyjnego (do przesyłania informacji jedno- lub dwukierunkowo) między np. procesorem a monitorem ekranowym, lub monitorem ekranowym a drukarką.

W celu uniknięcia sytuacji wieży Babel w połączeniach dokonano pewnych światowych ustaleń w zakresie połączeń zwanych popularnie interfejsami. Najczęściej interfejs to wiązka przewodów o określonych parametrach elektrycznych, zakończona dwustronnie łączówkami, umożliwiającymi dołączenie go do dwu urządzeń.

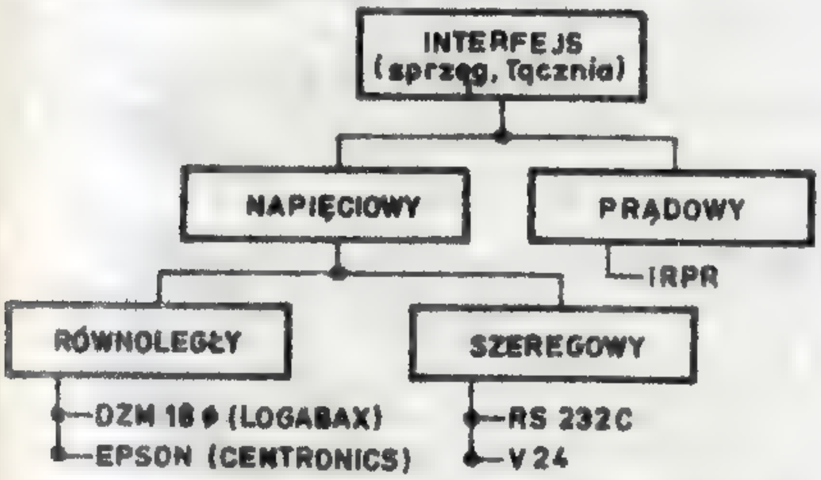
Kilka słów dygresji na temat nazwy „interfejs". Jest to spolszczenie angielskiego słowa **interface**, co oznacza dosłownie połączenie **między twarzami** (urządzeń), a w potocznym żartobliwym przekładzie — interfejs to **„międzymordzie"**. W lingwistycznych zapędach posunęliśmy się tak daleko, że w zamian mamy: sprzęg, łącznia, złącze, łącze, układ pośredniczący, układ sprzęgający, sprzężenie, styk.

Niezależnie od nadmiaru polskich „ersatzów" fachowcy uparcie używają obcego słowa interfejs, które się tak utrwaliło w informatyce, że nawet końmi go nie wyrwie.

Rodzaje interfejsów

W zależności od sygnału przesyłanego liniami między urządzeniami interfejsy dzieli się na prądowe (przesyłane są impulsy prądu elektrycznego) i napięciowe (przesyłane są impulsy napięcia elektrycznego). Interfejsy prądowe są mniej popularne, najbardziej rozpowszechnione są **interfejsy napięciowe**, wśród których z kolei rozróżnia się **szeregowe** i **równoległe** przesyłanie informacji. Interfejsem szeregowym informacje są przesyłane bit po bicie w jedną i w drugą stronę, a równoległym przesyłanych jest 8 bitów naraz, tworzących jeden znak.

Rysunek przedstawia podział interfejsów i nazwy najpopularniejszych ich przedstawicieli.



Najczęściej stosowane są standardy interfejsów napięciowych równoległych typu DZM-180, Centronics i szeregowych typu V 24 i RS 232C.

W tabelach połączeń tych interfejsów używane będą zapisy, do których odnoszą się następujące uwagi:

a) używane jest słownictwo i skróty pochodzące od określeń z języka angielskiego

b) kreseczka nad nazwą sygnału oznacza jego negację, to znaczy, że działanie tego sygnału zaczyna się w chwili, gdy jego wartość spada do poziomu zera logicznego (Ø) — tak pisane, w odróżnieniu od literki „O" — duże.

c) pin oznacza element łączówki o określonym numerze, do którego dolutowujemy przewód.

Tabelka przedstawia standard interfejsów szeregowych V 24 i Rs 232C, które zasadniczo nie różnią się między sobą, jednakże zostały powołane do życia przez różne organizacje międzynarodowe. Interfejs V 24 przez CCITT (fr. Comit'e Consultatif International Telegrafique et Telephonique / Międzynarodowy Doradczy Komitet Telegraficzny i Telefoniczny) natomiast RS 232C przez EIA (Electronic Industries Association)

Sygnały i wyprowadzenia interfejsów RS 232C i V 24

RS 232C	V 24	Pin = wyprowadzenie	Oznaczenie mnemoniczne	Nazwa linii
AA	101	1	—	ekran
AB	102	7	GND	masa sygnałowa
BA	103	2	TxD	dane nadawane
BB	104	3	RxD	dane odbierane
CA	105	4	RTS	żądanie nadawania
CB	106	5	CTS	gotowość nadawania
CC	107	6	DSR	gotowość modemu
CD	108.2	20	DTR	żądanie dołącz. modemu do linii
CE	125	22	RI	dzwonek
CF	109	8	DCD	nośna odbierana
CG	110	21	SQD	poziom sygnału odb
CH	111	23	—	wybór prędk. trans.
DA	113	24	—	zegar transm. nad.
DB	114	15	—	zegar transm. nad
DD	115	17	—	zegar transm. odb
SBA	118	15	—	dane nad. — kanał 2
SBB	119	14	—	dane odb. — kanał. 2
SCA	120	19	—	żąd. nad. — kanał 2
SCB	121	13	—	got. nad. — kanał 2
SCF	122	12	—	nośna odb. — kan. 2

Standard dopuszcza możliwość zredukowania niektórych linii kontrolnych i linii synchronizacji. W minimalnej postaci wystarczą trzy przewody: 1 — linia masy, 2 — dane nadawane, 3 — dane odbierane. Linie sterowane są sygnałami napięciowymi zmieniającymi się w zakresie od —12V do +12V.

Pozostały nam do omówienia dwa interfejsy równoległe. Są to DZM-180 (używa się również nazwy LOGABAX) i CENTRICS (używa się nazwy EPSON). Warto je znać, ponieważ większość mikrokomputerów ma wyjścia umożliwiające podłączenie drukarki przez jeden z tych standardów.

STANDARD DZM-180

Nr pinu	Nr pinu masy	Nazwa sygnału	Opis sygnałów
1	19	$\overline{SE}$	Impuls powodujący wpisanie danych do bufora drukarki
2	20	$\overline{ENT1}$	osiem linii danych, sygnały te reprezentują informację wejściową w postaci 8-bitowej danej równoległej
3	21	$\overline{ENT2}$	
[illegible]	22	$\overline{ENT3}$	
5	23	$\overline{ENT4}$	
6	24	$\overline{ENT5}$	
7	25	$\overline{ENT6}$	
[illegible]	26	$\overline{ENT7}$	
9	27	$\overline{ENT8}$	
10	28	$\overline{ACK}$	Potwierdzenie przyjęcia informacji wejściowej
11	29	Nie używany	
12	30	FIN PAP	Brak lub koniec papieru
13	—	RELSEL	Gotowość drukarki do przyjmowania danych
14	—	Nie używany	
15	—	Nie używany	
16	—	Ø V	Ø V logiczne
17	—	GND	Masa obudowy drukarki
18	—	Nie używany	
31	—	Nie używany	
32	—	Nie używany	
33	—	Nie używany	
34	—	Nie używany	
35	—	FIN	Sygnalizacja pustego bufora drukarki
36	—	Nie używany	
37	—	CPLX	Zegar 2,5 µs

Standard CENTRONICS

Nr pinu	Nr pinu masy	Nazwa sygnału	Opis sygnałów
1	19	STROBE	Impuls powodujący wpisanie danych do bufora drukarki
2	20	DATA 1	Osiem linii danych
3	21	DATA 2	Reprezentują 8-bitową
4	22	DATA 3	równoległą informację
5	23	DATA 4	wejściową
6	24	DATA 5	
7	25	DATA 6	
8	26	DATA 7	
9	27	DATA 8	
10	28	ACK	Potwierdza przyjęcia inform.
11	29	BUSY	Drukarka zajęta
12	30	PAPER END	Brak lub koniec papieru
13		SELECTED	Gotowość drukarki do pracy
14		Nie używany	
15		Nie używany	
16		SIGNAL GND	Masa sygnałowa
17		CHASSIS GND	Uziemienie
18		+5V Dc	Nap. zasil. +5V (ok. 50 mA)
31		INPUT PRIMPE	Zerowanie układów drukarki
32		ERROR	Sygnalizacja nieprawidłowości w przesyłanej informacji
33		EXT GND	Masa zewnętrzna
34		Nie używany	
35		Nie używany	
36		—	Poziom niski

Uwaga praktyczna: w celu zmniejszenia podatności na zakłócenia proponuje się, aby każdy przewód sygnałowy został skręcony z przewodem masy sygnałowej oznaczonej numerem pinu masy, który jest połączony elektrycznie z pozostałymi masami.

opr. inż. Edward WRZEŚNIEWSKI

Program stosuje procedurę przeniesienia blokowego dla zapełnienia rysunkiem całego ekranu (24 wiersze!).

W linii 999 wykorzystano możliwości CHR$ Ø — CHR$ 23 dla uzyskania nietypowego wydruku nazwy programu.

```
   1 REM   Program ██████

  10 BORDER 5: BEEP .02,14: CLEA
R 63999: FOR n=64000 TO 64011: R
EAD bt: POKE n,bt: NEXT n
  15 DIM a$(64)
  20 FOR n=7 TO 127 STEP 15: PLO
T 127,144: DRAW -n,-97: PLOT 128
,144: DRAW n,-97: NEXT N
  25 PLOT 0,143: DRAW 0,-95: DRA
W 255,0: DRAW 0,95
  30 PRINT #0;AT 0,0; PAPER 7;a$
  40 RANDOMIZE USR 64000
  50 FOR n=1 TO 8: PRINT a$: NEX
T n
  55 PLOT 0,48: DRAW 0,127: DRAW
 255,0: DRAW 0,-127
  60 FOR n=7 TO 127 STEP 15: PLO
T 127,80: DRAW -n,95: PLOT 128,8
0: DRAW n,95: NEXT N
  70 FOR n=7 TO 127 STEP 15: PLO
T 127,80: DRAW -n/3,-32: PLOT 12
8,80: DRAW n/3,-32: NEXT n
  80 FOR n=0 TO 175 STEP 16: PLO
T 127,80: DRAW -127,(n-80): PLOT
 128,80: DRAW 127,(n-80): NEXT n
  90 PAUSE 100: RUN
 100 DATA 33,0,72,17,0,80,1,0,8,
237,176,201
 999 CLEAR : LET b$=CHR$ 22+CHR$
 11+CHR$ 14+CHR$ 19+CHR$ 1+"24x3
2": PRINT b$: SAVE b$ LINE 1
```

PROGRAM 38

Spadająca piłka

(Gra zręcznościowa)

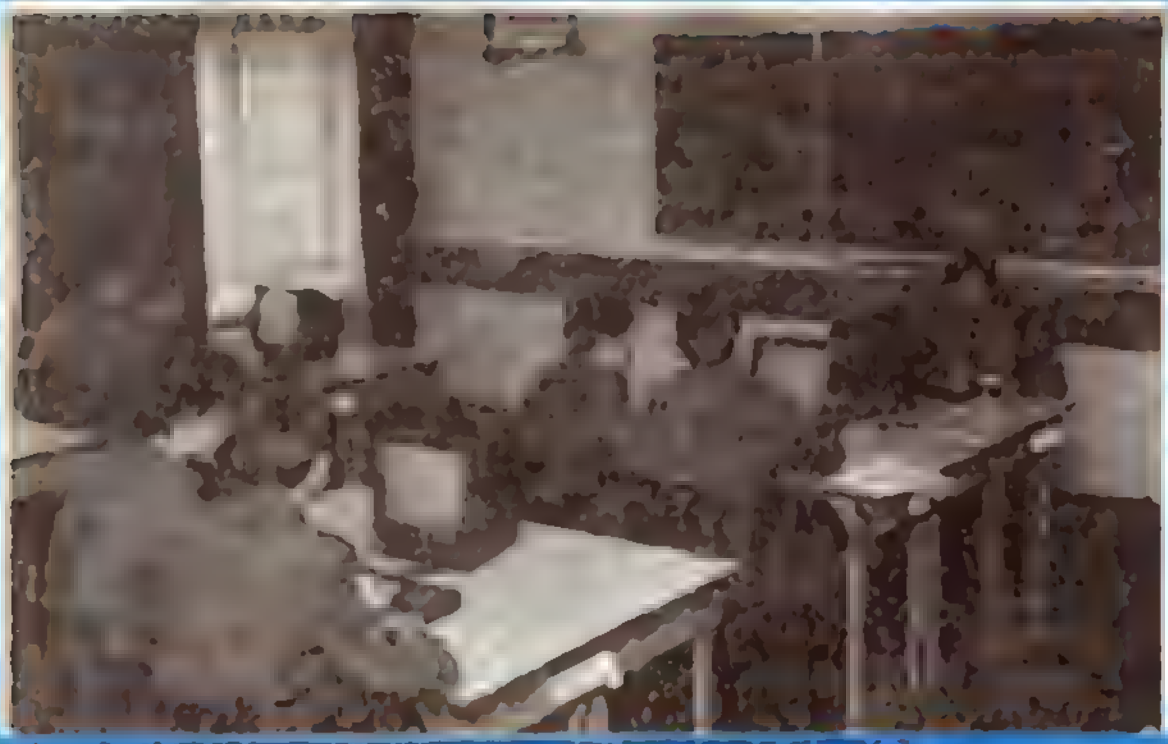
Gra polega na złapaniu maksymalnej liczby piłek do pojemnika (litera U). Gracz może poruszać wózkiem za pomocą kursorów w lewo bądź w prawo. Piłka zmienia kierunek losowo, gdy natrafi na przeszkodę (litera O). Jest to możliwe dzięki kontroli ekranu przez SCREEN$.

Nie zalecam zwalniać piłki za pomocą PAUSE, gdyż ta instrukcja nie jest wykonywana, jeżeli jest wciśnięty dowolny klawisz. Można natomiast wprowadzić jałową pętlę FOR-NEXT.

Krzysztof POŹNIAK
„Hobbyte"

```
   1 REM
********************************
*                              *
*      Krzysztof  Pozniak      *
*                              *
*    Klub Mikrokomputerowy     *
*       HOBBYTE   © 1987       *
*                              *
*      ████████████████        *
*                              *
********************************

  10 PAPER 0: INK 7: BORDER 0: C
LS : FOR k=1 TO 8 STEP 2: FOR l=
0 TO 1: PRINT AT 2*(k+l),l;: FOR
 a=1 TO 15: PRINT "O ";: NEXT a:
 NEXT l: NEXT k
  15 LET p=1: LET zp=0
  20 PRINT #1;AT 0,0;"pilka :";p
,"zlapanych :";zp: LET k=8+INT (
10*RND): LET w=8+INT (10*RND)
  30 FOR l=0 TO 20: LET z=SCREEN
$ (l+1,k)<>" ": LET k1=k+z*SGN (
.5-RND)*(l<20)
  35 LET d=CODE INKEY$: LET w=w+
((d=9) AND (23-w))-((d=8) AND (w
-2)): PRINT AT 21,w;" _U_ "
  40 PRINT AT l+1,k1;"O";AT l,k;
" ": LET k=k1: NEXT l
  45 LET p=p+1: IF k1=w+2 THEN L
ET zp=zp+1
  50 BEEP 0.05,20: BEEP .05,30:
PRINT AT 21,w;"     ";AT 21,k1;"
 ": GO TO 20
```



W tym roku szkolnym chęć pobierania nauki w Liceum Lotniczym im. Żwirki i Wigury w Dęblinie wyraziło prawie dwa tysiące kandydatów. Po badaniach lekarskich do egzaminu wstępnego przystąpiło tylko 30 procent chętnych.

Dyrektor istniejącej już piętnaście lat szkoły, [illegible] **Janusz Ziółkowski** [illegible] ceum Lotnicze, to zwykłe liceum zawodowe, a wyróżnia je się jedynie tym, że mieści się przy Wyższej Oficerskiej Szkole Lotniczej i od momentu powstania doskonale wywiązuje się z zadania zapewnienia stałego dopływu maturzystów do WOSL.

Na czym więc ta odrębność polega? Liceum kształci w specjalności „mechanik osprzętu lotniczego". W rzeczywistości jednak dęblińskie „orlęta" nie poprzestają na zdobywaniu tylko tych kwalifikacji. Są pilotami szybowców, skoczkami spadochrono-

Skomputery



[illegible]
[illegible] modelarnia. A w ubiegłym roku uruchomiono nowoczesne, zautomatyzowane gabinety: elektroniki, osprzętu lotniczego i informatyki. W tym ostatnim gabinecie zatrzymajmy się na chwilę. Oddajmy głos uczniom. Oto kilka charakterystycznych wypowiedzi:

— Od kilku lat interesuję się komputerami. Mam w domu nawet Atari i chyba sporo już umiem. Nie na tyle jednak, by nie nauczyć się więcej. Tutaj mam fachową opiekę, poznałem nowy typ komputera...

— Dla mnie komputery to nowość. Na początku nie byłem do nich przekonany. Myślałem, że będzie to przemijająca moda. [illegible] mnie gry komputerowe, teraz uczę się sam układać programy...

Chcę być pilotem i temu podporządkowuję wszystko. Czy możliwe jest, żeby bez



...yzowane „orlęta"

J. RAJCH

[illegible] tak skomplikowane maszyny nafaszerowane elektroniką? Na pewno nie. Dlatego cieszę się, że tak wcześnie mamy okazję poznać komputery i pracować na nich. Zresztą jest to bardzo zajmujące i wciąga...

Obecnie Liceum Lotnicze dysponuje 10 mikrokomputerami typu Amstrad.

— Są one dla nas najbardziej przydatne — mówi zastępca dyrektora szkoły, mjr **Zenon Ziemkiewicz**. — *Wykorzystywane są przede wszystkim na lekcjach informatyki. [illegible] w nich uczestniczyć każdy uczeń liceum. Wystarczą dobre wyniki w nauce i także chęci. Zajęcia prowadzą fachowcy, oficerowie z tutejszego Ośrodka Obliczeniowego.*

Plany [illegible] Kadra liceum stara się o zdobycie nowych mikrokomputerów. [illegible] Obecnie do pracy i nauki z komputerami przygotowywani są na kursach nauczyciele różnych zawodów. Może będzie można skomputeryzować i uatrakcyjnić na przykład lekcje języka polskiego?

A na razie uczą się języków, tych komputerowych, i programowania. Programy stosują coraz trudniejsze, o większym stopniu skomplikowania. Przystosowują je do swoich potrzeb. Zbierają fachową literaturę i wiadomości z komputerowego mikroświatka.

Na lekcjach informatyki i zajęciach koła zainteresowań uczniów dęblińskiego Liceum Lotniczego korzystających z komputerów podglądał **Jan Zetman**



Komputer w dydaktyce • ZX — Spectrum

Komputer jest cennym narzędziem ułatwiającym uczniom poznawanie rzeczywistości. W dydaktyce szczególnie dużą wartość mają takie metody przekazu wiedzy, które wymagają od ucznia jego aktywnej, a nawet twórczej postawy. Dysponując odpowiednim programem komputerowym, uczeń może przeprowadzać różnorodne eksperymenty. Są one tanie, w pełni bezpieczne, niedestrukcyjne, umożliwiają dowolną zmianę skali czasu (przyspieszenie lub zwalnianie procesu) i zmianę nawet tych parametrów, którymi w realnym świecie nie można swobodnie manipulować (np. pogoda, gęstość zaludnienia, popyt na określone towary itp.). Zmieniając w dużym zakresie parametry symulowanego procesu i obserwując uzyskiwane rezultaty, uczeń może sam formułować pewne wnioski dotyczące badanego fragmentu rzeczywistości. Uczeń może też formułować hipotezy i sprawdzać je korzystając z odpowiedniego programu komputerowego. Uzyskana w ten sposób wiedza jest dużo głębsza i trwalsza niż wiedza przekazana metodami podawczymi. Jest ona wynikiem własnych dociekań ucznia i jego aktywnej postawy, stąd też zorientowana jest bardziej problemowo. Nie sposób także pominąć radości i satysfakcji ucznia z samodzielnego odkrywania nowych dla niego reguł, praw czy związków.

Prostym przykładem programu umożliwiającego samodzielne formułowanie wniosków o wybranym fragmencie rzeczywistości jest prezentowany RZUTUKOS (dla ZX Spectrum).

Program ten symuluje rzut ukośny. Zastosowany algorytm, oparty na równaniach rzutu ukośnego, jest bardzo prosty i nie wymaga komentarza. Uczeń podaje dwa parametry:

— kąt wyrzutu,

— prędkość początkową rzutu,

a program kreśli trajektorię lotu pocisku. Na ekranie można uzyskać całą rodzinę trajektorii dla różnych kątów i prędkości początkowych. Trajektorie kreślone są w dyskretnych momentach czasu, a stopień zagęszczenia punktów trajektorii jest odwrotnie proporcjonalny do prędkości chwilowej pocisku (najlepiej jest to widoczne przy rzucie pionowym — kąt = 90°).

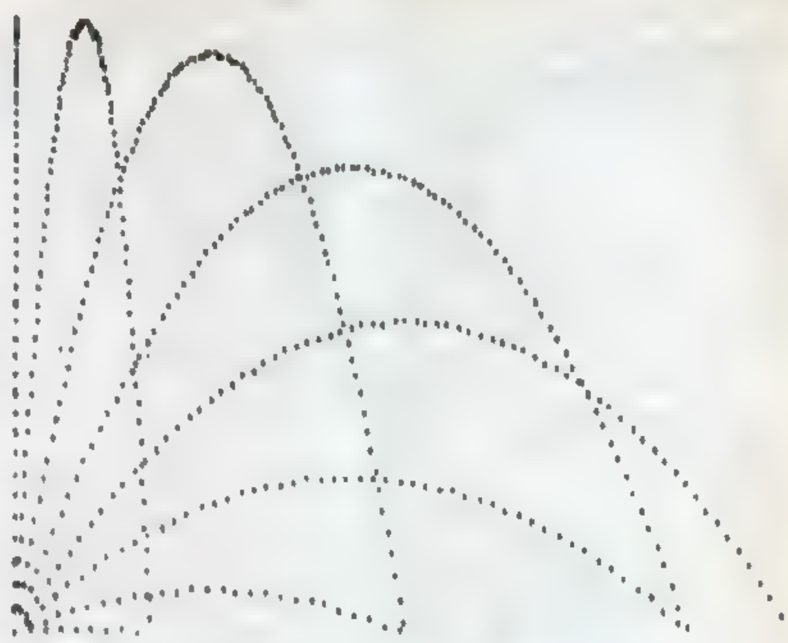
Na podstawie uzyskiwanych trajektorii uczeń może samodzielnie odpowiedzieć na pytania:

— od jakich parametrów i w jaki sposób zależy zasięg w rzucie ukośnym,

— dla jakiego kąta wyrzutu zasięg jest maksymalny,

— jaki jest związek między dwoma różnymi kątami, dla których zasięg jest taki sam.

Program może być również wykorzystany (po prostej modyfikacji) do prowadzenia gry, polegającej na trafieniu w określony punkt, przy jak najmniejszej liczbie rzutów. Przedstawiony program ma na celu zainspirowanie zainteresowanych Czytelników do własnych poszukiwań metod unowocześnienia i uatrakcyjnienia procesu dydaktycznego przez sensowne wdrażanie nowego technicznego środka nauczania, jakim jest mikrokomputer. Szczególnie duże pole do popisu istnieje w naukach matematyczno-przyrodniczych, ale ciekawe programy można napisać dla każdego przedmiotu.



```
  10 REM [illegible]
  20 INPUT "kat rzutu (0-90): ";
k: IF k<0 OR k>90 THEN GO TO 20
  30 LET k=2*PI*k/360:
     REM [illegible]
  40 INPUT "predkosc poczatk. (0
-200): ";v0: IF v0<0 OR v0>200 T
HEN GO TO 40
  50 LET wx=.0625: LET wy=.085:
     REM [illegible]
  60 LET vx=v0*COS k: LET vy=v0*
SIN k: REM [illegible]
  70 LET dt=.4: LET g=9.81: LET
g=g/2:
     REM [illegible]
  80 FOR t=0 TO 50 STEP dt
  90 LET y=vy*t-g*t*t:
     REM [illegible]
 100 IF y<0 THEN GO TO 20:
     REM [illegible]
 110 LET x=vx*t
 120 PLOT x*wx,y*wy
 130 NEXT t
```

J.M.

Symulacja EDIT

Prezentowany program pozwala wprowadzać dowolne zmiany w zbiorze tekstowym, symulując swoim działaniem edycją linii. Mniej wtajemniczonym zwracam uwagę na kontrolę klawiszy **DELETE**, **EDIT**, **ENTER** i kursorów za pomocą INKEY$ oraz na uzyskanie wydruku na dole ekranu.

Można obejrzeć kody klawiszy wprowadzane przez INKEY$ dzięki linii 1000:

Krzysztof POŹNIAK
„Hobbyte"

```
  10 REM symulacja EDIT
  20 LET t$="ten tekst nalezy wy
edytowac i w dowolny sposob popr
awic"
  30 PRINT t$
  40 IF INKEY$<>CHR$ 7 THEN GO T
O 40
  50 GO SUB 100: CLS : GO TO 30
 100 LET k=0: LET d=0
 110 BEEP .05,20: PRINT #1;AT 0,
0;t$( TO k); FLASH 1;"L"; FLASH
0;t$(k+1 TO );" " AND d: LET d=0
 120 LET inkey=CODE INKEY$
 130 IF inkey=13 THEN RETURN
 140 IF inkey=12 AND k AND LEN t
$ THEN LET d=1: LET t$=t$( TO k-
1)+t$(k+1 TO ): LET k=k-1: GO TO
 110
 160 IF inkey=8 AND k THEN LET k
=k-1: GO TO 110
 170 IF inkey=9 AND k<>LEN t$ TH
EN LET k=k+1: GO TO 110
 180 IF inkey<=31 THEN GO TO 120
 190 LET t$=t$( TO k)+CHR$ inkey
+t$(k+1 TO ): LET k=k+1: GO TO 1
10

1000 PRINT AT 0,0;CODE INKEY$;"
": GO TO 1000
```

Długość życia

Program oparty na badaniach naukowych pozwala na obliczenie przewidywanej długości życia osób dorosłych w wieku 20—65 lat. To jak długo będziemy żyć, zależy jak się okazuje od... wielu czynników, o które będziemy pytani w programie. Sprawdźmy, o ile dłużej możemy żyć, jeżeli w Nowym Roku 1988 rzucimy na przykład palenie, zaczniemy się gimnastykować i stracimy nieco na wadze. Różnica może być zaskakująca. Może zatem warto spróbować?

Tadeusz CISEK

```
1 REM DLUGOSC ZYCIA
2 PRINT CHR$(147)
3 PRINT'DLUGOSC ZYCIA'
5 PRINT
40 PRINT'OKRESLENIE PRZEWIDYWANEJ DLUGOS
CI ZYCIA DLA OSOB W WIEKU 20-65 LAT'
45 PRINT
50 PRINT'1=TAK,2=NIE':PRINT
60 A=72
70 INPUT'CZY JESTES MEZCZYZNA ?';B:PRINT

80 IF B=1 THEN A=A-3:GOTO 90
85 A=A+4
90 PRINT'CZY MIESZKASZ W MIESCIE O LICZB
IE'
95 INPUT'MIESZKANCOW > 1 MILIONA ?';B:PR
INT
100 IF B=1 THEN A=A-2:GOTO 110
105 PRINT'CZY MIESZKASZ W MIESCIE O LICZ
BIE '
107 INPUT'MIESZKANCOW (10000) ?';B:IF B=
1 THEN A=A+2:PRINT

110 PRINT'JEZELI PRACUJESZ ZA BIURKIEM W
PISZ '1'':PRINT
120 PRINT'JEZELI TWOJA PRACA WYMAGA REGU
LARNEGO, CIEZKIEGO'
125 INPUT'FIZYCZNEGO WYSILKU WPISZ '2'';
B :PRINT
130 IF B=1 THEN A=A-3:GOTO 140
135 IF B=2 THEN A=A+3
140 PRINT'JEZELI CWICZYSZ INTENSYWNIE WI
ECEJ NIZ'
145 PRINT'5 RAZY PO POL GODZ W TYGODNIU'

150 PRINT'WPISZ  '1'. JEZELI CWICZYSZ 2-
3 RAZY'
155 PRINT'W TYGODNIU, WPISZ '2', JEZELI
 NIE'
157 INPUT' WPISZ  '3'';B:PRINT
160 IF B=1 THEN A=A+4:GOTO 170
165 IF B=2 THEN A=A+2
170 INPUT 'CZY JESTES ZONATY (ZAMEZNA) ?
';B:PRINT

180 IF B=1 THEN A=A+5:GOTO 190
185 PRINT'ILE DZIESIECIOLECI MIESZKALES
SAMOTNIE '
187 INPUT '(LICZAC OD 25 ROKU ZYCIA)?';B
:A=A-B:PRINT
190 PRINT' CZY SPISZ WIECEJ NIZ 10 GODZI
N '
195 INPUT'NA DOBE ?';B:PRINT
200 IF B=1 THEN A=A-4
210 PRINT'JEZELI JESTES OSOBA  AGRESYWNA
, ZYJACA INTENSYWNIE '
215 PRINT'WPISZ '1' , JEZELI JESTES OSOB
A'
220 INPUT 'SPOKOJNA, POGODNA WPISZ '2''
;B:PRINT
230 IF B=1 THEN A=A-3:GOTO 240
235 IF B=2 THEN A=A+3
240 PRINT'JEZELI JESTES SZCZESLIWY WPISZ
 '1';NIESZCZESLIWY,'
245 INPUT 'WPISZ  '2'';B:PRINT
250 IF B=1 THEN A=A+1:GOTO 260
255 IF B=2 THEN A=A-2

260 PRINT'CZY DOSTALES MANDAT ZA PRZEKRO
CZENIE PREDKOSCI'
265 INPUT 'W MINIONYM ROKU ?';B :PRINT
270 IF B=1 THEN A=A-1
280 PRINT 'CZY TWOJ DOCHOD W CIAGU ROKU'
:INPUT' PRZEKRACZA MILION ?';B :PRINT
290 IF B=1 THEN A=A-2
300 PRINT'JEZELI SKONCZYLES SZKOLE SREDN
IA WPISZ  '1';'
305 INPUT 'SZKOLE WYZSZA, WPISZ  '2':';B
:PRINT
310 IF B=1 THEN A=A+1:GOTO 320
315 IF B=2 THEN A=A+2
320 PRINT'JEZELI MASZ WIECEJ NIZ 65 LAT
I NADAL PRACUJESZ,'
325 INPUT' WPISZ  '1'';B :PRINT
330 IF B=1 THEN A=A+3
340 PRINT'JEZELI KTORES Z TWOICH DZIADKO
W ZYLO'
345 INPUT' 85 LAT WPISZ '1' ';B:PRINT
350 IF B=1 THEN A=A+2:INPUT 'CZY WSZYSCY
 4 ZYLI 80 LAT';B:IF B=1 THEN A=A+6

360 PRINT:PRINT'CZY KTORES Z TWOICH RODZ
ICOW ZMARLO'
365 PRINT 'NA ZAWAL LUB ATAK SERCA PRZED
':INPUT 'UKONCZENIEM 50 LAT';B :PRINT
370 IF B=1 THEN A=A-4
380 PRINT'CZY KTORES Z TWOICH RODZICOW L
UB RODZENSTWA PRZED 50'
385 PRINT'MIALO CHOROBE SERCA LUB RAKA,
LUB'
390 INPUT 'CUKRZYCE';B:PRINT
400 IF B=1 THEN A=A-3
410 PRINT'JEZELI PALISZ: >2 PACZKI/DZIEN
 WPISZ '1';'
415 PRINT '1-2 PACZKI/DZIEN, WPISZ '2';'

420 PRINT '1/2-1 PACZKI/DZIEN WPISZ '3''
:INPUT'JEZELI MNIEJ WPISZ '4'';B:PRINT
430 IF B=1 THEN A=A-8:GOTO 440
432 IF B=2 THEN A=A-6:GOTO 440
434 IF B=3 THEN A=A-3
440 PRINT'CZY PIJESZ CODZIENNIE CO NAJMN
IEJ'
445 INPUT '100 G ALKOHOLU ';B:PRINT
450 IF B=1 THEN A=A-1
460 PRINT'JEZELI MASZ NADWAGE POWYZEJ 25
 KG, WPISZ '1', 15-25 KG WPISZ '2''
470 PRINT ',5-15 KG , WPISZ '3'':INPUT '
W INNYM PRZYPADKU WPISZ '4' ';B:PRINT
480 IF B=1 THEN A=A-8:GOTO 490
482 IF B=2 THEN A=A-4:GOTO 490
484 IF B=3 THEN A=A-2
490 PRINT'CZY PRZECHODZISZ RAZ W ROKU OG
OLNE BADANIA'
495 INPUT'LEKARSKIE ?';B:PRINT
510 IF B=1 THEN A=A+2
520 PRINT'JEZELI MASZ 30-40 LAT WPISZ '1
';40-50, WPISZ'2'; 50-60, WPISZ'
530 INPUT ''3'; >70 WPISZ'4'';B:PRINT
540 A=A+B+1
550 PRINT ''
560 PRINT:PRINT' PRZEWIDYWANA DLUGOSC TW
OJEGO ZYCIA       WYNOSI ';A;' LAT !!!'
```

Sami piszemy gry

Cezary SOBCZAK

Zanim przystąpimy do pisania kolejnej gry, przedstawię Wam prosty program pozwalający uzyskać na ekranie efekt ruchu piłki bilardowej podstawę do dalszej zabawy

```
10 REM ruch pileczki
20 MODE 1:INK 0,9:INK 1,26:INK 2,7
30 PAPER 2:BORDER 9:WINDOW #1,2,39,2,24
40 PAPER #1,0:CLS:CLS #1
50 x=20:y=12:dx=1:dy=-1
60 WHILE INKEY(47)=-1
70 LOCATE #1, x,y:PRINT #1, CHR$(231);
80 LOCATE #1,x,y:PRINT #1, " ";
90 x=x+dx:y=y+dy
100 IF x<=1 OR x>=38 THEN dx=-dx
110 IF y<=1 OR y>=23 THEN dy=-dy
120 LOCATE #1,x,y:PRINT #1, CHR$(231);
130 WEND
140 WHILE INKEY(60)=-1:WEND:GOTO 60
```

Po uruchomieniu programu komendą **RUN**, naciśnięcie spacji spowoduje zatrzymanie piłeczki, a klawisza 'S' wprowadzenie jej znowu w ruch. Możemy teraz przystąpić do budowy gry. Zaczynamy, jak zwykle, od wymyślnego scenariusza. Na początek proponuję coś takiego: po ekranie poruszają się dwie piłeczki. Naszym zadaniem będzie sterowanie ruchem ludzika, tak aby w ustalonym czasie (np. 100 sekund) złapać jak najwięcej piłeczek. Za każde złapanie piłeczki otrzymujemy jeden punkt

Wprowadźmy oznaczenia na zmienne

n — liczba piłeczek (n=2)
z, v — współrzędne ludzika (wart. początkowe z=20, v=12)
x(i) y(i) — współrzędne i-tej piłeczki, i=1,n
dx(i) dy(i) — przyrosty współrzędnych i-tej piłeczki
cg — łączny czas gry (cg=100)
t — bieżący czas
ta — czas początkowy
pt — liczba uzyskanych punktów
lewo, prawo, góra, dół } przechowują numery klawiszy sterujących ruchem ludzika. Przyjęto następujące klawisze: lewo=71(z), prawo=63(x), góra=19(l), dół=22(/)

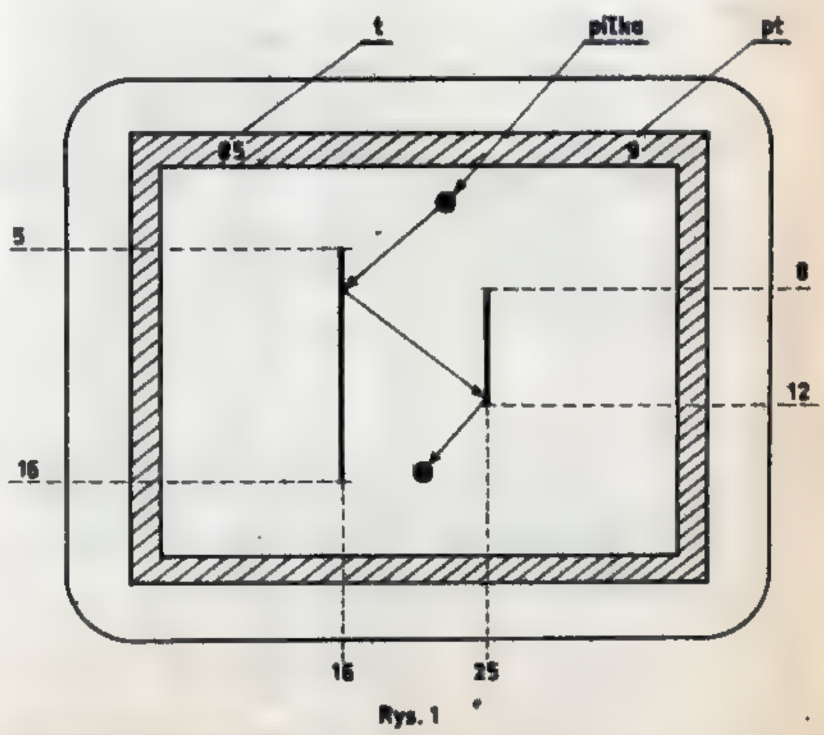
Wykorzystując przedstawioną wcześniej zasadę, piszemy.

```
10 REM zlap pilke
20 REM -----------------------------------
30 MODE 1:INK 0,0:INK 1,26:INK 2,7:INK 3
,11
40 BORDER 0:CLS
50 REM okreslenie tablic
60 n=2
70 DIM x(n):DIM y(n):DIM dx(n):DIM dy(n)
80 REM ramka
90 MOVE 12,12:DRAWR 0,360:DRAWR 615,0
100 DRAWR 0,-360:DRAWR -615,0
101 FOR y=370 TO 383:MOVE 12,y: DRAWR 61
5,0:NEXT
110 MOVE 12,399:FILL 2
140 REM nadanie wart. pocz. zmiennym
142 ENV 1,15,-1,10,15,1,10
150 ta=INT(TIME/300):pt=0:t=0:cg=100
160 z=20:v=12
170 lewo=71:prawo=63:gora=19:dol=22
180 x(1)=4:x(2)=30:y(1)=4:y(2)=20
190 FOR i=1 TO n
200 dx(i)=1:dy(i)=-1:NEXT
210 REM -----------------------------------
220 WHILE t<cg
230 t=INT(TIME/300)-ta:LOCATE 3,1
240 PEN 1:PRINT t:LOCATE 36,1:PRINT pt;
250 REM druk ludzika
260 LOCATE z,v:PEN 3:PRINT CHR$(248)
270 REM ruch pileczek
280 FOR i=1 TO n
290 IF x(i)<=2 OR x(i)>=39 THEN dx(i)=-d
x(i)
300 IF y(i)<=3 OR y(i)>=24 THEN dy(i)=-d
y(i)
310 IF x(i)=z AND y(i)=v THEN 460
320 LOCATE x(i),y(i):PRINT " ":x(i)=x(i)
+dx(i)
330 y(i)=y(i)+dy(i):LOCATE x(i),y(i):PEN
 1:PRINT CHR$(231)
340 NEXT
350 REM ruch ludzika
360 IF INKEY(lewo)=0 AND z>2 THEN GOSUB
1000:z=z-1
370 IF INKEY(prawo)=0 AND z<39 THEN GOSU
B 1000:z=z+1
380 IF INKEY(gora)=0 AND v>3 THEN GOSUB
1000:v=v-1
390 IF INKEY(dol)=0 AND v<24 THEN GOSUB
1000:v=v+1
400 WEND
410 REM koniec gry
420 INK 3,26,18:LOCATE 12,1:PEN 3:PRINT"
GRASZ JESZCZE (T/N)";
430 IF INKEY(51)=0 THEN RUN
440 IF INKEY(46)=0 THEN PEN 1:CLEAR INPU
T: CLS: END ELSE 430
460 REM zlapanie pilki
470 pt=pt+1:SOUND 3,340,50,15,1
472 FOR i=1 TO 150:BORDER 7,26:NEXT
473 BORDER 0
480 z=20:v=12:GOTO 230
1000 LOCATE z,v:PRINT " ":RETURN
```

Uruchom program i zagraj. Po pewnym czasie zapewne zauważysz, że piłeczki poruszają się po stałych torach i wystarczy ustawić ludzika w jednym miejscu, by piłeczka sama w niego trafiała. Zmodyfikuj program, generując początkowe współrzędne piłeczek w sposób losowy. Napisz:

```
145 randomize time
180 gosub 2000
475 gosub 2000
2000 for i=1 to n
2010 if x(i)<>Ø then locate x(i), y(i): print " "
2100 x(i)=int (RND*23+7)
2200 y(i)=int (RND*12+4): NEXT
2300 return
```

i uruchom program. Aby grę nieco uatrakcyjnić, wprowadzimy jeszcze niewidoczne dla nas przeszkody, od których piłeczki będą się odbijały, zmieniając niespodziewanie kierunek swojego ruchu. Proponuję ustawienie dwóch przeszkód (rys.1).



Rys. 1

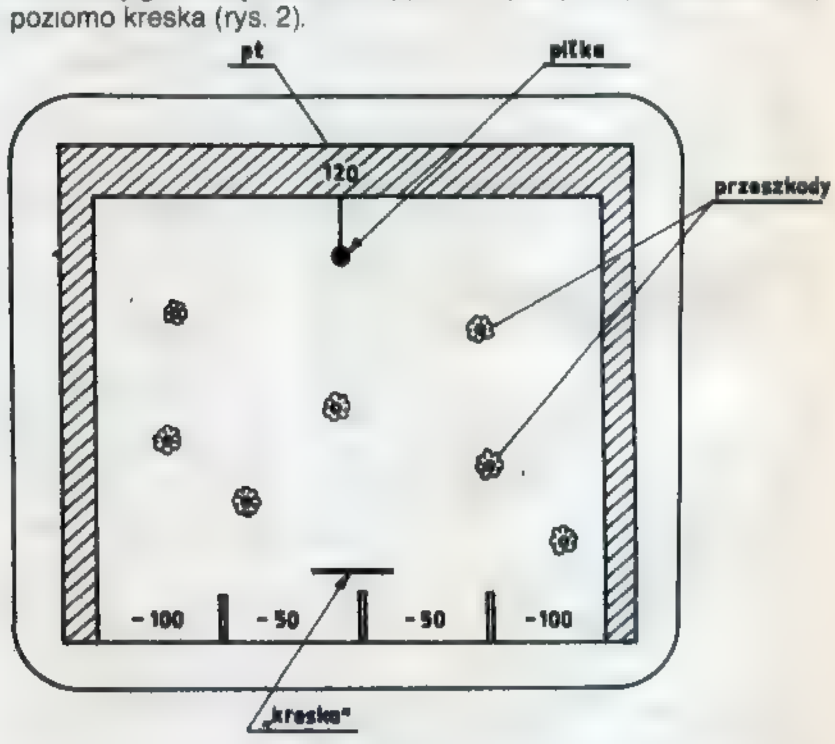
Uzupełnij program pisząc

```
285 if x(i)=16 and (y(i)<=16 and y(i)>=5) then dx(i)=—dx(i)
286 if x(i)=25 and (y(i)<=12 and y(i)>=8) then dx(i)=—dx(i)
```

Będzie nam na pewno przyjemniej grać słuchając muzyki. Dźwięki (ich okresy drgań) wprowadzone będą w wierszach DATA (od linii 5000) i chcąc zmienić melodię (czyli wprowadzić nowe liczby i nową ich ilość) trzeba pamiętać o uaktualnieniu zmiennej „mel" na bieżącą liczbę dźwięków, (wiersz 155).
Jako melodyjkę proponuję fragment Partity h-moll na skrzypce solo J. S Bacha.
Ostatecznie program ma postać

```
10 REM zlap pilke
20 REM --------------------------------
30 MODE 1:INK 0,0:INK 1,26:INK 2,7:INK 3
,16
40 BORDER 0:CLS
50 REM okreslenie tablic
60 n=2
70 DIM x(n):DIM y(n):DIM dx(n):DIM dy(n)
80 REM ramka
90 MOVE 12,12:DRAWR 0,360:DRAWR 615,0
100 DRAWR 0,-360:DRAWR -615,0
101 FOR y=370 TO 383:MOVE 12,y: DRAWR 61
5,0:NEXT
110 MOVE 12,399:FILL 2
140 REM nadanie wart. pocz. zmiennym
142 ENV 1,15,-1,10,15,1,10
145 RANDOMIZE TIME
150 ta=INT(TIME/300):pt=0:t=0:cg=100
155 RESTORE 5000:nt=1:mel=72
160 z=20:v=12
170 lewo=71:prawo=63:gora=19:dol=22
180 GOSUB 2000
190 FOR i=1 TO n
200 dx(i)=1:dy(i)=-1:NEXT
210 REM     gra
220 WHILE t<cg
230 t=INT(TIME/300)-ta:LOCATE 3,1
240 PEN 1:PRINT t:LOCATE 36,1:PRINT pt;
250 REM druk ludzika
260 LOCATE z,v:PEN 3:PRINT CHR$(248)
270 REM ruch pileczek
280 FOR i=1 TO n
285 IF x(i)=16 AND y(i)<16 AND y(i)>5 TH
EN dx(i)=-dx(i)
286 IF x(i)=25 AND y(i)<12 AND y(i)>8 TH
EN dx(i)=-dx(i)
290 IF x(i)<=2 OR x(i)>=39 THEN dx(i)=-d
x(i)
300 IF y(i)<=3 OR y(i)>=24 THEN dy(i)=-d
y(i)
310 IF x(i)=z AND y(i)=v THEN 460
320 LOCATE x(i),y(i):PRINT " ":x(i)=x(i)
+dx(i)
330 y(i)=y(i)+dy(i):LOCATE x(i),y(i):PEN
 1:PRINT CHR$(231)
340 NEXT
350 REM ruch ludzika
360 IF INKEY(lewo)=0 AND z>2 THEN GOSUB
1000:z=z-1
370 IF INKEY(prawo)=0 AND z<39 THEN GOSU
B 1000:z=z+1
380 IF INKEY(gora)=0 AND v>3 THEN GOSUB
1000:v=v-1
390 IF INKEY(dol)=0 AND v<24 THEN GOSUB
1000:v=v+1
391 READ nuta
392 SOUND 135,nuta,18,12,1
393 nt=nt+1
394 IF nt=mel THEN RESTORE 5000:nt=1
400 WEND
410 REM koniec gry
420 INK 3,26,18:LOCATE 12,1:PEN 3:PRINT"
GRASZ JESZCZE (T/N)";
430 IF INKEY(51)=0 THEN RUN
440 IF INKEY(46)=0 THEN PEN 1:CLEAR INPU
T: CLS: END ELSE 430
460 REM zlapanie pilki
470 pt=pt+1:SOUND 3,340,50,15,1
472 FOR i=1 TO 150:BORDER 7,26:NEXT
473 BORDER 0
475 GOSUB 2000
480 z=20:v=12:GOTO 230
1000 LOCATE z,v:PRINT " ":RETURN
2000 FOR i=1 TO n
2010 IF x(i)<>0 THEN LOCATE x(i),y(i):PR
INT" "
2100 x(i)=INT(RND*23+7)
2200 y(i)=INT(RND*12+4):NEXT
2300 RETURN
2350 REM  nuty
5000 DATA 127,106,84,80,95,106,113,71,80
5100 DATA 84,106,113,106,142,169,213,106
,113
5200 DATA 127,159,169,190,95,106,179,113
,127
5300 DATA 134,84,113,134,159,169,190,213
,225
5400 DATA 213,169,127,225,127,142,253,14
2,159
5500 DATA 284,225,190,159,113,95,134,84,
95
5600 DATA 106,127,142,159,95,106,113,134
,127
5700 DATA 169,134,113,84,80,84,95,106,11
3
```

Innym niemal już klasycznym przykładem wykorzystania efektu ruchu piłki bilardowej, jest „flipper". W grze tej wystrzelona piłeczka, trafiając w różnorodne przeszkody, zdobywa dla nas punkty do czasu, aż wypadnie poza pole gry. Sterując odpowiednio „flipperami", od których piłeczka się odbija, staramy się jak najdłużej utrzymać ją w ruchu na polu gry.
W naszej grze rolę dwóch „flipperów" przejmie poruszająca się poziomo kreska (rys. 2).



Rys. 2

Na początku piłeczka, wisząc na cienkiej nitce, mruga do nas zachęcająco. Po wystartowaniu (naciskając spację) piłka spada krążąc po polu gry, odbija się od przeszkód. Za każdorazowe trafienie w przeszkodę otrzymujemy 10 punktów. Jeżeli nie uda nam się, poruszając „kreską" odbić piłeczki, wówczas wpada ona do jednej z czterech komór, „zdobywając" dla nas punkty ujemne.

```
9 SYMBOL AFTER 256
10 MEMORY 20000:SYMBOL AFTER 38:CLS
15 ENT 1,89,37,238:ENV 1,84,7,5
16 ENT -2,13,-9,1,1,121,1:ENV 2,1,14,1,1
4,-1,11
20 MODE 1:INK 0,0:INK 1,26:INK 2,24:INK
3,17:BORDER 0
30 FOR i=16000 TO 16028:READ a$:v=VAL("&
"+a$):POKE i,v:NEXT
```

```
40 n=15:pt=0
45 SYMBOL 38,60,66,153,189,189,153,66,60
50 LOCATE 1,1:PEN 2:PRINT STRING$(39,143
);:LOCATE 1,2:PEN 2:PRINT STRING$(39,143
);:LOCATE 19,1:PEN 1:PRINT pt;
60 FOR y=1 TO 25:LOCATE 1,y:PEN 2:PRINT
CHR$(143):LOCATE 39,y:PEN 2:PRINT CHR$(1
43):NEXT
70 LOCATE 3,25:PEN 3:PRINT"-100";
80 LOCATE 14,25:PEN 3:PRINT"-50";:LOCATE
 24,25:PEN 3:PRINT"-50";:LOCATE 32,25:PE
N 3:PRINT"-100";
85 FOR i=10 TO 30 STEP 10:LOCATE i,25:PE
N 2:PRINT CHR$(149);:LOCATE i,24:PEN 2:P
RINT CHR$(149);:NEXT
87 MOVE 0,0:DRAWR 620,0,2
90 MOVE 313,370:DRAWR 0,-45,2:LOCATE 20,
5:PEN 1:PRINT CHR$(231)
91 IF x<>0 THEN LOCATE x,24:PRINT" ":LOC
ATE 2,23:PRINT SPACE$(37);
100 z=20:v=23:x=20:y=5:dx=SGN(INT(RND*5-
3)):dy=1
105 LOCATE z,v:PEN 1:PRINT STRING$(3,154
);
110 WHILE INKEY(47)=-1:LOCATE 20,5:PEN 1
:PRINT CHR$(231):LOCATE 20,5:PRINT" ":WE
ND:MOVE 313,365:DRAWR 0,-45,2,1:LOCATE 2
0,5:PRINT" "
120 RESTORE 2000:FOR i=0 TO n-1:READ x,y
:LOCATE x,y:PEN 2:PRINT "&":POKE 15000+i
,x:POKE 15100+i,y:NEXT
125 x=20:y=5
130 WHILE y<24
140 LOCATE x,y:PRINT" ":x=x+dx:y=y+dy:LO
CATE x,y:PRINT CHR$(231)
160 IF INKEY(1)=0 AND z<36 THEN LOCATE z
,23:PRINT SPACE$(2);:z=z+2
170 IF INKEY(8)=0 AND z>=3 THEN LOCATE z
,23:PRINT SPACE$(3);:z=z-2
175 IF y=22 AND (x>=z-1 AND x<=z+3) THEN
 dy=-dy
180 LOCATE z,v:PEN 1:PRINT STRING$(3,154
);
190 IF x<=3 OR x>=38 THEN dx=-dx
200 IF y<=3 OR y>=25 THEN dy=-dy
210 POKE 14990,x:POKE 14991,y:POKE 14992
,n:POKE 14995,0:CALL 16000
220 IF PEEK(14995)=1 THEN 250 ELSE 310
250 SOUND 129,142,50,1,1
300 IF ABS(dx)= ABS(dy) THEN 301 ELSE 30
4
301 dx=SGN(INT(RND*7-3)):IF dy=1 THEN GO
SUB 3000:GOTO 309
302 IF dy=-1 THEN GOSUB 4000:GOTO 309
304 IF ABS(dx)=1 THEN dx=-dx:dy=SGN(INT(
RND*7-3))
305 IF ABS(dy)=1 THEN dy=-dy:dx=SGN(INT(
RND*7-3))
309 LOCATE x,y:PEN 2:PRINT "&":x=x+dx:y=
y+dy:pt=pt+10:LOCATE 19,1:PEN 1:PRINT pt
310 WEND
312 SOUND 129,250,0,0,2,2
315 FOR a=1 TO 1000:NEXT
320 IF (x>=1 AND x<=10) OR (x>=31 AND x<
=40) THEN pt=pt-100:LOCATE 19,1:PRINT pt
;:GOTO 90
330  pt=pt-50:LOCATE 19,1:PRINT pt;:GOTO
 90
340 GOTO 90
1000 DATA 3a,8e,3a,21,98,3a
1100 DATA ed,4b,90,3a,ed,b1,c0,1,63,0,9
1200 DATA 3a,8f,3a,be,c0,1,93,3a,3e,1,2,
c9
2000 DATA 6,13,11,14,12,7,14,21,20,15,26
,9,31,16,34,19,35,12,25,6,18,17,21,8,13,
10,20,15,36,5
3000 IF RND>0.5 THEN dy=-1:RETURN
3100 dy=0:RETURN
4000 IF RND>0.5 THEN dy=1:RETURN
4100 dy=0:RETURN
```

Podprogram umieszczony w liniach 1000—1200 wczytywany jest w linii 30, pary współrzędnych (xp, yp) rozmieszczenia przeszkód podane w linii 2000 wczytujemy w linii 120.

Znak, który ma być drukowany jako przeszkoda jest zdefiniowany w wierszu 45.

Po wywołaniu podprogramu (linia 210) następuje badanie wskaźnika (linia 230). W razie trafienia piłeczki w przeszkodę następuje (w liniach 300—305) losowa zmiana kierunku ruchu piłeczki.

Efekty dźwiękowe występują tylko w wypadku trafienia w przeszkodę lub wyjścia z pola gry. Spróbuj rozszerzyć program, tak aby np. w czasie całej gry przygrywała muzyczka.

Podprogram wygląda następująco:

Kod mnemoniczny	Opis	Liczba bajtów rozkazu	Odpowiednik rozkazu w kodzie maszynowym (ósemkowo)
LD A, (14990)	Podstaw do akumulatora współrzędną x piłeczki	3	3A, 8E, 3A
LD HL, 15000	Podstaw do pary rejestrów HL adres początku tablicy współrzędnych xp przeszkód	3	21, 98, 3A
LD BC, (14992)	Podstaw do pary rejestrów BC ilość przeszkód (wartość n).	4	ED, 4B, 90, 3A
CPIR	Porównaj wartość współrzędnej x piłeczki (w akumulatorze) z kolejnymi elementami tablicy xp przeszkód	4	ED, B1
RET NZ	Powrót do programu BASICA, jeżeli nie znaleziono odpowiednika w tablicy	1	C0
LD BC, 99	Znaleziono. Współrzędna x=xp Podstaw do pary rejestrów BC wartość 99	3	1, 63, 0
ADD HL, BC	Zmodyfikuj adres tablicy wsp. przeszkód w celu porównania wsp. y piłeczki z odpowiadającą jej współrzędną yp przeszkody. Do pary rejestrów HL dodaj zawartość pary rejestrów BC. Wynik w HL	1	9
LD A, (14991)	Podstaw do akumulatora współrzędną y piłeczki	3	3A, 8F, 3A
CP (HL)	Porównaj wartość współrzędnej y piłeczki (w akumulatorze) z zawartością komórki o adresie umieszczonym w HL-element tablicy yp odpowiadający znalezionej wartości	1	BE
RET NZ	Powrót programu BASICA, jeżeli nie znaleziono odpowiednika	1	C0
LD BC, 14995 LD A, 1 LD, BC, A	Znaleziono, czyli x=xp i y=yp Ustawienie wskaźnika (komórki 14995) na jeden	3 2 1	1, 93, 3A 3E, 1 2
RET	Powrót do programu BASICA	1	C9

Oznaczenia zmiennych

n — liczba przeszkód
x, y — aktualne współrzędne piłki
xp, yp — współrzędne przeszkód
z, v — współrzędne „kreski" (v=23)
pt — punkty
lewo, prawo — numery klawiszy sterujących ruchem „kreski. Przyjęto lewo = 8 (←), prawo = 1 (→)

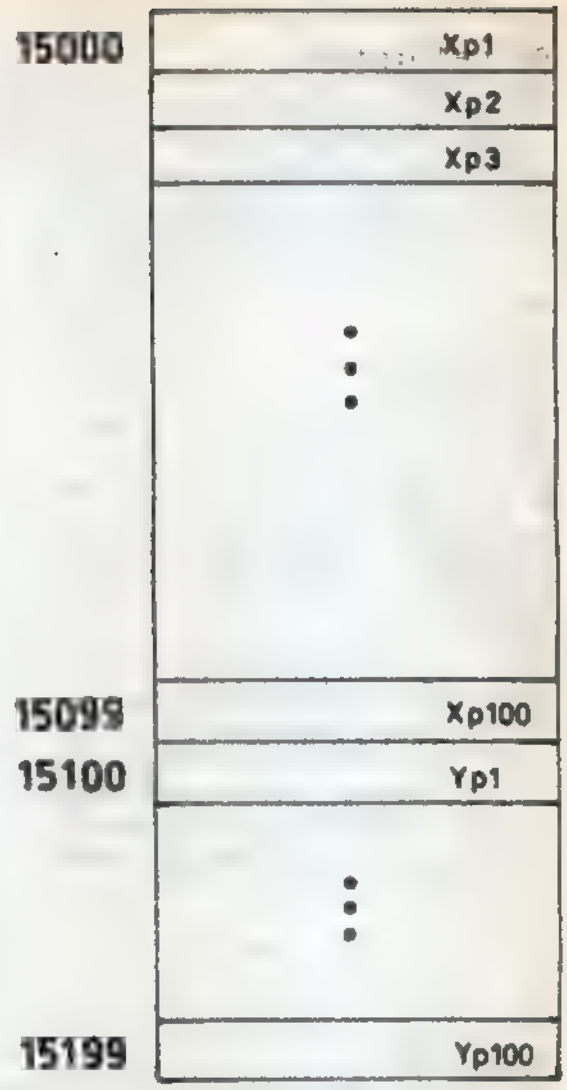
Współrzędne kolumn (xp) i linii (yp) przeszkód umieszczone zostaną w tablicach, które będziemy przeglądać po każdorazowej zmianie współrzędnych piłki. W tym celu napiszemy prosty podprogram w kodzie maszynowym, który będzie wywoływany w BASIC-u.

Przyjmijmy, że tablica współrzędnych xp zajmować będzie obszar pamięci od adresu 15000 (dziesiętnie) do 15099, a tablica odpowiadających im współrzędnych yp obszar od adresu 15100 do 15199 (rys. 3).

Ustalmy ponadto, że:
- komórka o adresie 14990 zawierać będzie bieżącą wartość x piłeczki
- komórka o adresie 14991 zawierać będzie bieżącą wartość y piłeczki
- komórka o adresie 14992 zawierać będzie ilość przeszkód (wartość n)
- komórka o adresie 14995 będzie wskaźnikiem przyjmującym wartość 0 (zero), gdy współrzędne (x, y) piłeczki nie pokrywają się ze współrzędnymi (xp, yp) przeszkody, i wartość 1 (jeden) w przeciwnym wypadku.

Przed wywołaniem podprogramu wskaźnik ustawimy na zero.

Rys. 3



Pascal (8)

Kontynuując rozważania na temat grafiki komputerowej, w kolejnej części przedstawiamy zasady wykorzystania pióra graficznego.

Z piórem graficznym związanych jest znacznie więcej procedur, niż z piórem tekstowym. Przed użyciem pióra graficznego niezbędne jest wywołanie bezparametrowej procedury

initgrafik

ustalającej początkowe warunki pracy. Po zakończeniu wykorzystywania pióra graficznego należy wywołać bezparametrową procedurę

leavegrafik

Odpowiednikiem procedury pen dla pióra graficznego jest procedura

grafpen (nrk)

a odpowiednikiem procedury **paper** — procedura

grafpaper (nrk)

gdzie **nrk** oznacza numer kałamarza.

Zasady wykorzystania kałamarzy przez pióro graficzne są identyczne, jak dla pióra tekstowego.

Z punktu widzenia pióra graficznego kartka podzielona jest na 400x600 punktów, (niezależnie od trybu wykorzystania ekranu) tj. 400 wierszy (numerowanych od 0 do 399) po 640 punktów (numerowanych od 0 do 639) w każdym z nich. Punkt można zakreślić kolorem tuszu za pomocą procedury

plot (x,y)

gdzie:
x — nr wiersza,
y — nr kolumny

Od ostatnio ustalonego położenia pióra można wykreślić odcinek do punktu wskazanego w procedurze

draw (x,y)

gdzie:
x,y — współrzędne punktu, do którego będzie kreślony odcinek.

Dodajmy, że punkt wskazany w **draw** staje się ostatnio ustalonym położeniem pióra graficznego.

Przedstawione procedury wykorzystano w programach kreślenia prostokątów:
— wyznaczającego ramy kartki (prog. 26),
— o współrzędnych wierzchołków (100, 100), (100,250), (550—250), (550—100) wraz z przekątnymi (prog. 27).

```
PROGRAM prog26;
 {$i grafik3.inc}
 BEGIN
  ink(0,0,0);
  ink(1,19,19);
  initgrafik;
  grafpaper(0);
  grafpen(1);
  plot(0,0);
  draw(0,399);
  draw(639,399);
  draw(639,0);
  draw(0,0);
  REPEAT UNTIL keypressed;
  leavegrafik;
  ink(0,1,1);
  ink(1,26,26);
 END.

PROGRAM prog27;
 {$i grafik3.inc}
 BEGIN
  initgrafik;
  mode(0);
  ink(2,0,0);
  ink(3,12,26);
  ink(4,16,16);
  ink(5,11,11);
  grafpaper(2);
  clg(2);
  grafpen(5);
  plot(100,100);
  draw(100,250);
  draw(550,250);
  draw(550,100);
  draw(100,100);
  grafpen(3);
  draw(550,250);
  grafpen(4);
  plot(100,250);
  draw(550,100);
  REPEAT UNTIL keypressed;
  leavegrafik;
 END.
```

W programie prog 27 wykorzystano ponadto procedurę

clg (nrk)

„czyszczącą" kartkę w kolorze ustalonym w ostatnio wykonanej procedurze grafpaper i ustanawiającą nowy numer kałamarza (nrk) dla tła pióra graficznego.

W kolejnych przykładach zilustrowaliśmy sposoby kreślenia figur graficznych oraz niektóre problemy związane z projektowaniem bardziej złożonych obrazów, ograniczając się do płaszczyzny.

W przykładzie 29 znajdziecie, program kreślenia okręgu o zadanym promieniu i współrzędnych środka. Wykorzystano w nim

```
PROGRAM prog28;
{$i grafik3.inc}
 PROCEDURE okrag(xs,ys,r:integer);
  VAR
   x,y,i:integer;
   wsp:real;
```

```
 BEGIN
  wsp:=pi/180;
  FOR i:=0 TO 359 DO
   BEGIN
    x:=round(r*sin(i*wsp));
    y:=round(r*cos(i*wsp));
    plot(xs+x,ys+y);
   END;
 END;
BEGIN
 initgrafik;
 ink(0,0,0);
 grafpaper(0);
 clg(0);
 grafpen(1);
 okrag(320,200,100);
 REPEAT UNTIL keypressed;
 leavegrafik;
 ink(0,1,1);
END.
```

znaną z geometrii metodę wyznaczania punktów leżących w jednakowej odległości od zadanego punktu, przy czym uzyskiwane wartości współrzędnych są zaokrąglane do części całkowitych (funkcja **round**), a przyjęta odległość kątowa między kolejnymi kolorowanymi punktami wynosi 1 stopień.

```
PROGRAM prog29;
{$i grafik3.inc}
 VAR
  i,j:integer;
  ch:char;
  wsp:real;
 PROCEDURE okrag(xs,ys,r:integer);
  VAR
   x,y,i:integer;
   wsp:real;
  BEGIN
   wsp:=pi/180;
   FOR i:=0 TO 359 DO
    BEGIN
     x:=round(r*sin(i*wsp));
     y:=round(r*cos(i*wsp));
     plot(xs+x,ys+y);
    END;
  END;
 BEGIN
  initgrafik;
  mode(1);
  ink(2,0,0);
  grafpaper(2);
  clg(2);
  ink(0,12,12);
  ink(3,10,10);
  grafpen(0);
  okrag(200,200,150);
  wsp:=pi/180;
  j:=0;
  FOR i:=0 TO 29 DO
   BEGIN
    grafpen(3);
    plot(200,200);
    draw(200+round(149*sin(6*j*wsp)),
       200+round(149*cos(6*j*wsp)));
    j:=j+1;
    grafpen(1);
    plot(200,200);
    draw(200+round(149*sin(6*j*wsp)),
       200+round(149*cos(6*j*wsp)));
    j:=j+1;
   END;
  WHILE NOT keypressed DO
   BEGIN
    ink(1,10,10);
    ink(3,26,26);
    FOR i:=1 TO 4000 DO;
    ink(3,10,10);
    ink(1,26,26);
```

Wrażenie obracania się koła uzyskano w programie z przykładu 30 przez zmianę kolorów szprych koła. Efekt ten jest namiastką bardzo dynamicznie rozwijającej się animacji obrazów.

```
PROGRAM prog30;
 {$i grafik3.inc}
 VAR
  i,nr:integer;
  ch:char;
 BEGIN
  initgrafik;
  mode(0);
  ink(0,0,0);
  grafpaper(0);
  clg(0);
  grafpen(1);
  plot(100,100);
  draw(100,250);
  draw(550,250);
  draw(550,100);
  draw(100,100);
  REPEAT
   gotoxy(0,23);
   WRITE('podaj nr kalamarza  ');
   gotoxy(19,23);
   READ(nr);
   grafpen(nr MOD 16);
   FOR i:=100 TO 250 DO
    BEGIN
     plot(100,i);
     draw(550,i);
    END;
   REPEAT UNTIL keypressed;
   READ(kbd,ch);
  UNTIL ch=' ';
  ink(0,1,1);
  mode(2);
  leavegrafik;
 END.
```

Inny problem to kolorowanie figur. W programie prog 30 zilustrowano jeden ze sposobów kolorowania pola prostokąta.

```
PROGRAM prog31;
 {$i grafik3.inc}
 VAR
  x11,y11,x12,y12,
  x21,y21,x22,y22:integer;
 BEGIN
  WRITELN('podaj wspolrzedne dolnego ',
          'prostokata');
  READLN(x11,y11,x12,y12);
  WRITELN('podaj wspolrzedne gornego ',
          'prostokata');
  READLN(x21,y21,x22,y22);
  initgrafik;
  mode(0);
  ink(0,0,0);
  clg(0);
  grafwindow(x11,y11,x12,y12);
  grafpaper(4);
  clg(4);
  grafwindow(x21,y21,x22,y22);
  grafpaper(8);
  clg(8);
  REPEAT UNTIL keypressed;
  mode(2);
  ink(0,1,1);
  leavegrafik;
 END.
```

Występująca w programie instrukcja **gotoxy** (x,y) umożliwia ustawienie położenia pióra tekstowego w określonym miejscu ekranu (ekran jest tu widziany jako tablica znaków o wymiarach 25 wierszy i liczbie kolumn zależnej od trybu wykorzystania ekranu.

Wszędzie tam, gdzie zamieszcza się obrazy obiektów rozlokowanych w przestrzeni trójwymiarowej, pojawia się kwestia likwidowania rysunków niewidocznych części obiektów, ścian i krawędzi. Najprostszym przykładem może tu być kreślenie nakładających się częściowo prostokątów, zilustrowane w przykładzie 32.

W programie wykorzystano właściwości tzw. okien graficznych wyznaczonych za pomocą procedury

grafwindow (x_L, y_L, x_p, y_p)

gdzie:

x_L, y_L — współrzędne lewego dolnego wierzchołka okna

x_p, y_p — współrzędne prawego górnego wierzchołka okna.

Okno graficzne jest prostokątem. Przyjmuje się, że dla pióra graficznego kartka jest początkowym, standardowym oknem graficznym.

W podanym przykładzie każdy prostokąt reprezentowany jest przez okno graficzne, a do ich kolorowania wykorzystano znaną już procedurę **clg** (dotyczy ona faktycznie ostatnio określonego okna). Efekt przesłaniania uzyskano sterując kolejnością kolorowania prostokątów. W praktycznie spotykanych zadaniach uzyskanie tego efektu wymaga skomplikowanych obliczeń i należy do najtrudniejszych operacji w grafice komputerowej.

Okna graficzne wykorzystane są w programie wyświetlania kolorów tuszów, którymi standardowo napełniane są poszczególne kałamarze.

```
PROGRAM prog32;
 {$i grafik3.inc}
VAR
 x1,x2,y1,y2,
 i,j,k:integer;
BEGIN
 x1:=10;
 x2:=60;
 y1:=50;
 y2:=150;
 initgrafik;
 mode(0);
 ink(0,0,0);
 grafpaper(0);
 clg(0);
 k:=1;
 FOR i:=0 TO 1 DO
  FOR j:=0 TO 7 DO
   BEGIN
    grafwindow(x1+j*80,y1+i*125,
               x2+j*80,y2+i*125);
    clg(k);
    k:=k+1;
   END;
 REPEAT UNTIL keypressed;
 ink(0,1,1);
 ink(1,26,26);
 mode(2);
 leavegrafik;_
END.
```

Podane przykłady na pewno nie oddają w pełni stopnia trudności grafiki komputerowej. Ilustrują jedynie wybrane, często spotykane problemy.

Stefan ROZMUS

Grafika na ekranie (3)

Przesunięcie oraz współrzędne jednorodne

Dotychczas w prezentowanym cyklu nie omawialiśmy przesunięcia punktów i linii na płaszczyźnie, a także figur składających się z punktów i linii. Spowodowane to było niemożliwością wprowadzenia stałych przesunięcia wewnątrz struktury ogólnej macierzy transformacji 2X2. Trudność tę można ominąć przez wprowadzenie trzeciego składnika do wektorów $[x\ y]$ i $[x\ y]$, przedstawiając je w postaci $[x\ y\ 1]$ i $[x^*\ y^*\ 1]$. Macierz transformacji po tej operacji staje się macierzą o rozmiarach 3X2:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix}$$

Jest to konieczne, gdyż w celu wykonania mnożenia macierzy liczba kolumn w macierzy definiującej punkt winna równać się liczbie wierszy w macierzy transformacji. W ten sposób

$$[x\ y\ 1]\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix} = [\,x + m\ \ y + m\,] = [\,x^* y^*\,]$$

Wynika z tego, że stałe m i n powodują przesunięcie x^* i y^* w stosunku do x i y. Aby istniała możliwość wyznaczenia macierzy odwrotnej (patrz część 1) do macierzy transformacji uzupełniamy tę macierz do kwadratowej o rozmiarach 3X3.

Na przykład:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

Trzeci składnik wektorów położenia punktów nie zmienia się po dodaniu trzeciej kolumny do macierzy transformacji. Wykorzystując tę macierz oraz uwzględniając (1) otrzymujemy przekształcony wektor $[\,x^*\ y^*\ 1]$. Dołożenie trzeciego elementu do wektora położenia i trzeciej kolumny do macierzy transformacji pozwala wykonać przesunięcie wektora położenia. Tak więc, wektor położenia $[x\ y\ 1]$ po pomnożeniu przez macierz transformacji 3X3 staje się wektorem położenia w ogólnym przypadku postaci $[X\ Y\ H]$. Przedstawione przekształcenie było wykonane tak, że $[X\ Y\ H] = [x\ y\ 1]$.

Przekształcenie zachodzące w przestrzeni trójwymiarowej w rozpatrywanym wypadku jest ograniczone do płaszczyzny gdyż $H = 1$. Jeżeli jednak trzecia kolumna

$$\begin{bmatrix} p \\ q \\ s \end{bmatrix}$$

macierzy transformacji T rozmiaru 3X3 nie jest równa

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

to w rezultacie przekształcenia otrzymamy $[x\ y\ 1] = [X\ Y\ H]$, gdzie $H \neq 1$. Płaszczyzna, na której obecnie znajduje się przekształcony wektor położenia, znajduje się w przestrzeni trójwymiarowej.

Zastąpienie dwuwymiarowego wektora trójwymiarowym lub w ogólnym przypadku n-wymiarowego $(n + 1)$ wymiarowym nazywa się wprowadzeniem tzw. współrzędnych jednorodnych. Opisane zostanie teraz dość obszernie zagadnienie współrzędnych jednorodnych. Współrzędne te mają duże zastosowanie, zwłaszcza w przestrzeni trójwymiarowej. Należy więc potraktować poniższy opis jako wprowadzenie do grafiki trójwymiarowej.

Idea tworzenia współrzędnych jednorodnych jest następująca. Punkt p przestrzeni n-wymiarowej charakteryzowany jest układem n liczb $p = (p_1, p_2, ..., p_n)$ będących jego współrzędnymi. Przedstawia się ten punkt za pomocą układu liczb $n+1$ w postaci:

$$\acute{p} = (p_1, p_2, ..., p_n, 1)$$

lub w ogólnym przypadku

$$p = (\lambda p_1, \lambda p_2, ..., \lambda p_n, \lambda) \text{ gdzie } \lambda \neq 0.$$

Liczby $\lambda p_1, \lambda p_2, ..., \lambda p_n, \lambda$ nazywa się współrzędnymi jednorodnymi punktu p określonego w przestrzeni n-wymiarowej.

Współrzędne jednorodne wykorzystuje się do wykonywania przekształceń w przestrzeni o jednym wymiarze większym niż ta, w której działamy. W ten sposób dwuwymiarowy wektor $[x\ y]$ przedstawia się trójskładnikowym wektorem $[hx\ hy\ h]$. Związek pomiędzy współrzędnymi punktu na płaszczyźnie a jego współrzędnymi jednorodnymi wyraża zależność:

$$x = \frac{hx}{h} \qquad y = \frac{hy}{h}$$

Nie ma jednorodnego przekształcenia współrzędnych punktu w przestrzeni dwuwymiarowej. Na przykład jednorodne współrzędne (12, 3, 3), (8, 2, 2), (4, 1, 1) przedstawiają wyjściowy punkt (4, 1). Dla uproszczenia wyliczeń ustalamy, że $[x\ y\ 1]$ przedstawia nieprzekształcony punkt w dwuwymiarowych współrzędnych jednorodnych.

Przekształcenie

$$[x^*\ y^*] = [x\ y]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

we współrzędnych jednorodnych ma postać

$$[X\ Y\ H] = [x\ y\ 1]\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Wykonanie przedstawionego powyżej przekształcenia współrzędnych wykazuje, że $X = x^*$, $Y = y^*$, $H = 1$. Jeżeli dodatkowa współrzędna jest równa jedynce, to przekształcone współrzędne jednorazowe są równe przekształconym współrzędnym (zwykłym). W ogólnym przypadku $H \neq 1$ i przekształcone zwykłe współrzędne otrzymuje się poprzez normalizację przekształconych współrzędnych jednorodnych, to jest

$$x^* = \frac{X}{H} \quad i \quad y^* = \frac{Y}{H}$$

Geometrycznie wszystkie przekształcenia x i y realizowane są — po normalizacji przekształconych współrzędnych jednorodnych — na płaszczyźnie $h = 1$.

Wykorzystanie współrzędnych jednorodnych do operacji wykonywanych na płaszczyźnie dwuwymiarowej pozwala rozszerzyć zakres wykonywanych przekształceń, poprzez wprowadzenie macierzy transformacji o wymiarach 3X3.

Ogólna postać macierzy transformacji jest następująca:

$$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

Wykorzystując tę macierz można dołączyć do już przedstawionych przekształceń inne np. przesunięcie, zmianę skali i przesunięcie. Pokażemy teraz wpływ trzeciej kolumny macierzy transformacji 3X3 na wynik przekształceń na płaszczyźnie.

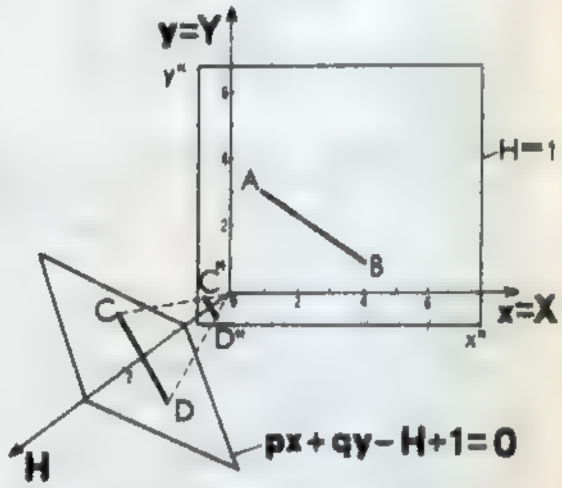
Prześledźmy wyniki następującej operacji:

$$[X\ Y\ H] = [x\ y\ 1]\begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= [x\ y\ (px + qx + 1)]$$

Teraz $X = x$, $Y = y$, $H = px + qx + 1$. Zmienna H określa płaszczyznę przekształcenia punktu (przedstawionego we współrzędnych jednorodnych) w przestrzeni trójwymiarowej.

Przekształcenie to pokazano na rys. 1.



Rys. 1 Przekształcenie we współrzędnych jednorodnych. Linia AB leżąca na płaszczyźnie xy jest rzutowana jako linia CD na płaszczyznę $pX+qY-H+1=0$

Na rys. 1 $p=q=1$. Wykonamy teraz normalizację w celu uzyskania zwykłych współrzędnych.

$$x^* = \frac{X}{H} = \frac{X}{pX+qY+1}$$

$$y^* = \frac{Y}{H} = \frac{Y}{pX+qY+1}$$

Przyjmując $p=q=1$ oraz współrzędne punktów A i B odpowiednio (1,3) i (4,1) otrzymujemy

$$x^* = \frac{1}{1+3+1} = \frac{1}{5},\quad y^* = \frac{3}{5}$$

po przekształceniu A w C* oraz

$$x^* = \frac{4}{1+4+1} = \frac{2}{3},\quad y^* = \frac{1}{6}$$

po przekształceniu B w D^*

Jednorodne współrzędne dla punktów C^* i D^* pokazanych na rys. 1 odpowiednio wynoszą $(\frac{1}{5}, \frac{3}{5}, 1)$ i $(\frac{2}{3}, \frac{1}{6}, 1)$

W rezultacie normalizacji uzyskano przejście trójwymiarowej linii CD w jej rzut $C^* D^*$

```
10 REM ****************************************
20 REM
30 REM Przesuniecie
40 REM
50 REM ****************************************
60 DIM wsp(101,3),mat(3,3),nwsp(101,3)
70 MODE 1:INK 0,13:INK 1,3:INK 2,12:INK 3,0
80 BORDER 0:PEN 1:PAPER 2
90 CLS
100 PRINT "Przesuwanie obrazu w ukladzie";
110 PRINT "wspolrzednych:-319<x<319,-191<y<191"
120 PRINT
130 PRINT "Siatka wspolrzednych (t/n)"
140 z$=INKEY$
150 IF z$=" " THEN 140
160 IF z$="t" THEN fl=1:GOTO 190
170 IF z$="n" THEN fl=0:GOTO 190
180 GOTO 140
190 GRAPHICS PAPER 1:REM Tlo grafiki
200 ORIGIN 320,208,0,640,16,400
210 CLEAR INPUT
220 WINDOW 1,40,25,25
230 GOSUB 1230:REM Rys. ukladu wspol.
240 i=0 :REM Licznik punktow
250 INPUT "Podaj wsp. punktu x,y";x,y
260 IF x>998 THEN 330
270 i=i+1
280 wsp(i,1)=x
290 wsp(i,2)=y
300 wsp(i,3)=1
310 GOSUB 780:REM Wykresl. punkt lub odcinek
320 GOTO 250
330 INPUT "Podaj przesuniecie:x,y";p,q
340 GOSUB 850 :REM Ustaw macierz przeksz.
350 GOSUB 1030 :REM Mnozenie
360 GOSUB 1140 :REM Zobrazowanie
370 INPUT "Przesunieto:ponownie-r,koniec-k";nk$
380 IF nk$="r" THEN 410
390 IF nk$="k" THEN 1730
400 GOTO 370
410 INPUT "kasowanie obrazu-k,rysowanie-r";nk$
420 IF nk$="k" THEN 450
430 IF nk$="r" THEN 670
440 GOTO 410
450 INPUT"nowego obrazu-n,starego-s";nk$
460 IF nk$="n" THEN 620
470 IF nk$="s" THEN 490
480 GOTO 450
490 REM Kasowanie starego obrazu
500 GOSUB 1230:REM Rys.ukladu wspolrz.
510 FOR k=1 TO i
520 wsp(k,1)=nwsp(k,1)
530 wsp(k,2)=nwsp(k,2)
540 wsp(k,3)=nwsp(k,3)
550 NEXT k
560 GRAPHICS PEN 3
570 GOSUB 1630
580 INPUT "przesuniecie-p,dorysowanie-d";nk$
590 IF nk$="p" THEN 330
600 IF nk$="d" THEN 250
610 GOTO 580
620 REM Kasowanie nowego obrazu
630 GOSUB 1230
640 GRAPHICS PEN 3
650 GOSUB 1630
660 GOTO 580
670 FOR n=1 TO i:REM Rysowanie od poczatku
680 FOR k=1 TO 3
690 wsp(n,k)=0
700 nwsp(n,k)=0
710 NEXT k
720 NEXT n
730 GOTO 230
740 REM ****************************************
750 REM       PODPROGRAMY
760 REM ****************************************
770 REM
780 REM Rysowanie punktu lub odcinka
790 REM
800 GRAPHICS PEN 3:REM stare wspolrzedne
810 IF i=1 THEN PLOT wsp(i,1),wsp(i,2)
820 IF i>1 THEN DRAW wsp(i,1),wsp(i,2)
830 RETURN
840 REM
850 REM Ustawianie macierzy
860 REM
870 FOR k=1 TO 3
880 mat(k,k)=1
890 NEXT k
900 mat(1,2)=0
910 mat(2,1)=0
920 mat(1,3)=0
930 mat(2,3)=0
940 mat(3,1)=p
950 mat(3,2)=q
960 FOR k=1  TO i:REM Zerowanie nwsp
970 FOR l=1 TO 3
980 nwsp(k,l)=0
990 NEXT l
1000 NEXT k
1010 RETURN
1020 REM
1030 REM Mnozenie macierzy
1040 REM
1050 FOR n=1 TO i
1060 FOR k=1 TO 3
1070 FOR m=1 TO 3
1080 nwsp(n,k)=nwsp(n,k)+wsp(n,m)*mat(m,k)
1090 NEXT m
1100 NEXT k
1110 NEXT n
1120 RETURN
1130 REM
1140 REM Rysowanie nowej figury
1150 REM
1160 GRAPHICS PEN 0:REM PIORO nowych wspol.
1170 PLOT nwsp(1,1),nwsp(1,2)
1180 FOR n=2 TO i
1190 DRAW nwsp(n,1),nwsp(n,2)
1200 NEXT
1210 RETURN
1220 REM
1230 REM Rysowanie ukladu wspolrzednych
1240 REM
1250 CLG 1
1260 GRAPHICS PEN 2:REM Pioro ukladu wspol.
1270 FOR a=-200 TO 200 STEP 20
1280 PLOT -2,a
1290 DRAWR 4,0
1300 NEXT a
1310 PLOT 0,-200
1320 DRAW 0,200
1330 TAG
1340 MOVER -16,-20
1350 PRINT "Y";
1360 MOVER -310,0
1370 PRINT "20";
1380 TAGOFF
1390 PLOT -307,170 : REM Kreslenie podzialki
1400 DRAWR 0,-4
1410 DRAWR 0,2
1420 DRAWR 20,0
1430 DRAWR 0,2
1440 DRAWR 0,-4
1450 IF fl=1 THEN 1460 ELSE 1510
1460 FOR b=-200 TO 200 STEP 20
1470 FOR a=-320 TO 320 STEP 20
1480 PLOT a,b
1490 NEXT a
1500 NEXT b
1510 FOR a=-320 TO 320 STEP 20
1520 PLOT a,-2
1530 DRAWR 0,4
1540 NEXT a
1550 PLOT -320,0
1560 DRAW 320,0
1570 TAG
1580 MOVER -16,-4
1590 PRINT "X";
1600 TAGOFF
1610 RETURN
1620 REM
1630 REM:Rysowanie starej figury
1640 REM
1650 PLOT wsp(1,1),wsp(1,2)
1660 FOR n=2 TO i
1670 DRAW wsp(n,1),wsp(n,2)
1680 NEXT n
1690 RETURN
1700 REM
1710 REM Koniec
1720 REM
1730 WINDOW 1,40,1,25
1740 CLS
1750 CLG 1
1760 END
```

na płaszczyznę H=1. Jak pokazano na rysunku punktem centralnym rzutowania jest środek układu współrzędnych.

Podstawowa macierz transformacji rozmiaru 3X3 dla dwuwymiarowych jednorodnych współrzędnych może być podzielona na cztery części

$$T = \left[\begin{array}{cc|c} a & b & p \\ c & d & q \\ \hline m & n & s \end{array}\right]$$

Jak uprzednio wykazano elementy a, b, c, d wpływają na szalowanie, obrót i skręcenie. Elementy m.in. umożliwiają przesunięcie, zaś p i q wpływają na rzutowanie. Pozostała część macierzy, element s powoduje ogólne skalowanie. Aby pokazać efekt ogólnego skalowania wykonamy następujące przekształcenie.

$$[X\ Y\ H] = [x\ y\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = [x\ y\ s]$$

Teraz $X=x$, $Y=y$, $H=s$ Współrzędne punktu po przekształceniu wynoszą: $x^* = \frac{x}{s}$, $y^* = \frac{y}{s}$. W wyniku przekształcenia $[x\ y\ 1] \rightarrow [\frac{x}{s}\ \frac{y}{s}\ 1]$ ma miejsce ogólna zmiana skali wektora położenia. Gdy $s<1$ zachodzi powiększenie, a przy $s>1$ pomniejszenie skali.

Przykład programu realizującego przesunięcie figury na płaszczyźnie

Po uruchomieniu programu „Przesunięcie" następuje przedstawienie zakresu umownych współrzędnych figur obrazowanych na ekranie monitora oraz pytanie o siatkę współrzędnych.

Na układzie współrzędnych przyjęto podziałkę wynoszącą 20 jednostek. W celu ułatwienia identyfikacji współrzędnych kreślonych figur wprowadza się siatkę współrzędnych, która pokrywa cały układ współrzędnych kropkami tworzącymi kwadraty o boku równym wielkości przyjętej podziałki.

Wciskając klawisz „t" siatka współrzędnych będzie wyświetlana do końca działania programu. Naciśnięcie klawisza „n" powoduje efekt przeciwny. Algorytm działania programu jest następujący:

1) Definiowanie figury.
2) Określenie wartości przesunięcia.
3) Obrazowanie figury podstawowej oraz po przesunięciu.
4) Wykonanie spacji umożliwiających:
   a) definiowanie figury od początku,
   b) kasowanie jednej z otrzymanych figur,
   c) dorysowanie nowych elementów figury,
   d) wykonywanie przesunięcia,
   e) zakończenie działania programu.

Identyfikację poszczególnych części programu źródłowego z przedstawionym algorytmem umożliwiają komentarze zawarte w programie.

Uwaga eksploatacyjna

Koniec definiowania figury uzyskuje się wpisując wartość współrzędnej $x>998$ oraz dowolną wartość współrzędnej y.

Modyfikacje

Zmianę wielkości podziałki uzyskamy modyfikując linie 1270 oraz 1510 programu. Konsekwencją zmiany podziałki jest konieczność modyfikacji opisu podziałki (linia 1370 i 1420) oraz siatki współrzędnych (linie 1460 i 1470).

Uwaga praktyczna

Wpisując tekst instrukcją PRINT na pozycję kursora graficznego należy umieścić średnik na końcu instrukcji, gdyż jego pominięcie powoduje uzyskanie nieoczekiwanego efektu — **Sprawdź**. Jest to rezultat przeniesienia tekstu do nowej linii.

Andrzej CETERA

Biblioteka C 64

„Biblioteka". W naszej domowej bibliotece zapanował ogromny bałagan. Pora zatem na zrobienie porządków. Pomoże nam w tym prezentowany poniżej program. Wszystkie zbiory książkowe możemy umieścić na kasecie magnetofonowej (wg obliczeń autora ok. 2000 pozycji na 100 obrotów licznika magnetofonu). Nasz zbiór zostanie uporządkowany według haseł (nazwisk autorów) o długości do 20 znaków. Pod każdym hasłem (nazwiskiem) możemy umieścić różne dane, jak np.: liczbę tomów, rok wydania, liczbę stron, na której półce znajduje się pozycja itp. Dane te jednak nie mogą zawierać więcej niż 75 znaków. Może przepływ informacji w naszej skomputeryzowanej bibliotece nie będzie zbyt szybki (to tylko przecież magnetofon), niemniej program umożliwia wykorzystanie mikrokomputera do bardzo poważnych celów. Program można łatwo dostosować do wielkości naszej biblioteki, a także, wykorzystać go do innych celów.

Opracował:

Tadeusz CISEK, Warszawa

```
4040 IF LEFT$(B$(Q),LEN(B$))()B$ THEN 40
80
4050 PRINT'AUTOR:';B$(Q):PRINT

4060 PRINTTAB(5)C$(Q):PRINT
4070 N=N+1:IF INT(N/3)=N/3 THEN INPUT'WI
ECEJ--NACISNIJ RETURN';A$
4080 NEXT Q
4090 GOTO 40
5000 PRINTCHR$(147):PRINT
5010 INPUT'PODAJ AUTORA ';B$:PRINT
5020 FOR Q=0 TO A
5030 IF LEFT$(B$(Q),LEN(B$))()B$ THEN 50
90
5040 PRINTCHR$(147)'AUTOR:';B$(Q):PRINT
5050 PRINTC$(Q):PRINT
5060 INPUT'CZY CHCESZ USUNAC TEN ZAPIS (
T/N)';A$:PRINT
5070 IF A$()'T' THEN 5090
5080 B$(Q)='':C$(Q)='':D=D+1
5090 NEXT Q
5100 GOTO 40
6000 INPUT'COFNIJ KASETE, NACISNIJ RETUR
N';A$
6010 OPEN 1,1,2,'BIBLIOTEKA'
6020 PRINT# 1,(A-D)

6030 FOR Q=0TOA:IF B$(Q)='' THEN 6050
6040 PRINT# 1,B$(Q);CHR$(32)
6045 PRINT# 1,C$(Q)
6050 NEXT Q:PRINT# 1,'END'
6060 CLOSE 1: GOTO 40
7000 PRINT'SORTOWANIE...':FORQ=1 TO A
7005 D$(Q)=B$(Q):E$(Q)=C$(Q)
7010 A$(Q)=D$(Q)
7020 NEXT Q
7030 K=0
7040 Q=1
7050 FOR J=2 TO A
7060 IF A$(Q)(A$(J) THEN 7080
7070 Q=J
7080 NEXT J
7090 A$(Q)='ZZZ'
7100 K=K+1
7110 B$(K)=D$(Q):C$(K)=E$(Q)
7120 IF K=A+1 THEN 7140
7130 GOTO 7040

7140 GOTO 40
8000 REM PROCEDURA INICJUJACA
8002 A$=''
8005 OPEN 1,1,2,'BIBLIOTEKA'
8010 PRINT# 1,0;A$;A$
8020 CLOSE 1:RETURN

1060 IF A$='N' THEN 1080
1070 A=A+1
1071 IF A-D(300 THEN 1000
1072 IF A-D=300 THEN PRINT'OSTATNI ZAPIS
 W PLIKU'
1074 IF A-D)=300 THEN PRINT'BLAD, PLIK P
ELNY':PRINT'KONIEC PLIKU'
1076 PRINT'URUCHOM PONOWNIE PROGRAM, INIC
JUJAC NOWA SEKCJE NA TASMIE'
1078 GOTO 1000
1080 GOTO 40
2000 PRINT'PODAJ AUTORA DLA KTOREGO NAST
API ZMIANA '
2010 INPUT'REDAKCJI';B$ :PRINT
2020 FOR Q=0TOA
2030 IF LEFT$(B$(Q),LEN(B$))()B$ THEN 21
10
2040 PRINTCHR$(147)
2050 PRINT'BIBLIOTEKA':PRINT
2060 PRINTC$(Q):PRINT
2070 INPUT'CZY CHCESZ REDAGOWAC TEN ZAPI
S(T/N)';A$:PRINT
2080 IF A$='N' THEN 2110:PRINT
2090 INPUT'PONOWNIE WPROWADZ WSZYSTKIE D
ANE';C$(Q):INPUT A$:PRINT
2100 PRINT'CZY POPRAWNIE (T/N)?';C$(Q):I
NPUT A$ :PRINT

2105 IF A$()'T'THEN PRINT'NIE MOZE BYC D
LUZSZY NIZ 75 LITER':GOTO 2090:PRINT
2107 INPUT'PODANIE INNEGO(T/N)';A$:IF A$
()'T' THEN 2120
2110 NEXT Q
2120 PRINT'KONIEC PLIKU':GOTO 40
3000 PRINT'WYBIERZ:1)WYLISTOWANIE
                KATALOGU ZAPISOW'
3005 PRINT'          2)WYLISTOWANIE ZAPISO
W                 WEDLUG AUTOROW'
3007 INPUT Y
3010 ON Y GOTO 3300,3020
3020 INPUT'PODAJ AUTORA ';H$ :PRINT
3040 PRINT'NUMER','AUTOR ':N=1:PRINT
3050 FOR Q=0 TO A
3060 IF LEFT$(B$(Q),LEN(H$))()H$ THEN 30
90
3070 PRINT N,B$(Q)
3080 N=N+1:IF INT(N/11)=N/11 THEN INPUT'
WIECEJ--NACISNIJ RETURN';A$

3090 NEXT Q
3100 PRINT'--------------------':PRINT:GOTO
 
3300 PRINT CHR$(147) :PRINT
3310 PRINT'NUMER','AUTOR ':N=1:PRINT
3320 FOR R=65 TO 90
3330 FOR Q=0 TO A
3340 IF LEFT$(B$(Q),1)()CHR$(R) THEN 336
0
3350 PRINTN,B$(Q):N=N+1:IF INT(N/11)=N/1
1THEN INPUT'WIECEJ--NACISNIJ RETURN';A$
3360 NEXT Q,R
3370 GOTO 40
4000 PRINTCHR$(147):PRINT
4010 INPUT'PODAJ AUTORA ';B$:PRINT
4020 N=1
4030 FOR Q=1TOA
5 PRINT'BIBLIOTEKA'
10 PRINTCHR$(147)TAB(55)'BIBLIOTEKA'
20 CLR:PRINT:DIM A$(300),B$(300),C$(300)
,D$(300),E$(300):FL=0:D=0
25 PRINT'WPROWADZ ODDZIELNA KASETE NA PL
IK DANYCH':PRINT
27 PRINT'COFNIJ KASETE;NACISNIJ RETURN '
;:INPUT'GDY JESTES GOTOW';A$:PRINT
30 PRINT'TEN PROGRAM MUSI BYC ZAINICJOWA
NY'
32 PRINT'KIEDY STOSOWANY JEST PO RAZ PIE
RWSZY, CZY CHCESZ GO '
33 INPUT'ZAINICJOWAC (T/N)';A$:IF A$()'T
' THEN 35
34 GOSUB 8000:PRINT:INPUT'PRZEWIN TASME,
 NACISNIJ RETURN ';A$
35 OPEN 1,1:INPUT# 1.Z:IF Z=0 THEN FL=1:
GOTO 40
36 IF Z)=295 THEN PRINT'PIEC REKORDOW DO
 KONCA PLIKU'
37 FOR A=0TOZ:INPUT# 1,B$(A):IF B$(A)='E
ND' THEN GOTO 40

38 INPUT# 1,C$(A):IF C$(A)='END'THEN GOT
O 40
39 NEXT
40 PRINT'WYBOR ROZKAZOW':PRINT
42 CLOSE 1,1
50 PRINT'ROZKAZ','FUNKCJA':PRINT:PRINT
60 PRINT'1','WPROWADZENIE NOWEJ POZYCJI'
:PRINT
70 PRINT'2','DRUKOWANIE LISTY POZYCJI':P
RINT
80 PRINT'3','SZUKANIE POZYCJI':PRINT
90 PRINT'4','ZMIANA REDAKCJI POZYCJI':PR
INT
100 PRINT'5','KASOWANIE POZYCJI':PRINT
110 PRINT'6','SORTOWANIE':PRINT
115 PRINT'7','NAGRANIE PLIKU':PRINT
120 INPUT A$
130 IF A$='1'THEN 1000
140 IF A$='2'THEN 3000
150 IF A$='3'THEN 4000
160 IF A$='4'THEN 2000
170 IF A$='5'THEN 5000

180 IF A$='6'THEN 7000
185 IF A$='7' THEN 6000
190 PRINT'ZLY ROZKAZ--PODAJ JEDEN Z '
195 PRINT'NASTEPUJACYCH:'
200 GOTO 50
1000 PRINT'PODAJ AUTORA (DLUGOSC  DO 20
':INPUT'LITER)';B$(A)
1010 IF LEN(B$(A)))20 THEN 1000
1020 PRINT'PODAJ TYTUL LUB INNE DANE (NA
JWYZEJ ';
1025 PRINT'75 ZNAKOW)':INPUT C$(A)
1030 PRINT'CZY POPRAWNIE (T/N)?';C$(A):I
NPUT A$:IF A$()'T' THEN 1020
1050 INPUT'WPROWADZENIE NASTEPNEGO(T/N)'
;A$
```

Jak narysować CPC 6128

Ten program na AMSTRAD-a może służyć jako pomoc w nauce początków geometrii. Pokazuje, jak narysować symetralną odcinka, dwusieczną kąta i trójkąt w trzech wersjach, gdy mamy dane: trzy boki, dwa boki i kąt, bok i dwa kąty, posługując się cyrklem i linijką bez podziałki. Program wykorzystuje tryb wysokiej rozdzielczości — MODE 2 — 560x400 punktów.

W programie można wyróżnić 10 rozdzielnych modułów. Pierwszy z nich to moduł sterujący (linie 10—180), 5 następnych realizuje kolejno funkcje. Moduły te nie wymagają dodatkowych opisów — teksty w nich zawarte dokładnie opisują wykonywane czynności. Następne trzy moduły to podprogramy realizujące funkcje wykorzystywane przez 5 modułów podstawowych i im należy poświęcić więcej uwagi, tym bardziej że mogą być one wykorzystane w innych programach.

Podprogram „PRZESUWANIE ODCINKA" — linie 1890—2000. Podprogram przesuwa odcinek w 30 krokach z jednego miejsca na ekranie w drugie, pokazując jego ruch. Należy podać mu współrzędne odcinka wyjściowego (zmienne x11, y11; x12, y12) i końcowego (zmienne x21, y21; x22, y22).

Podprogram „RYSOWANIE ŁUKU" — linie 2000—2050. Rysuje na ekranie łuk o zadanych parametrach, pokazując jednocześnie ruch promienia łuku. Wykorzystuje zmienne: x, y — współrzędne środka, r — promień, k, l — współrzędne kątowe początku i końca łuku, w stopniach, liczone zgodnie z ruchem wskazówek zegara, przyjęto, że 0 stopnie odpowiada godz. 12.

Podprogram „RYSOWANIE ODCINKA" — linie 2060—2100. Podprogram rysuje na ekranie odcinek o zadanych współrzędnych (x11, y11; x12, y12) dzieląc go na 400 części, dużo wolniej niż instrukcja DRAW.

Ostatni podprogram to definicje polskich liter nie występujących na klawiaturze AMSTRADA. Wykorzystano klawiaturę numeryczną (f1—f9). Małe litery otrzymujemy naciskając SHIFT i wybrany klawisz F1—f9, duże KONTROL i f1—f9 — patrz rysunek. Jest to system w tej chwili znany wśród wielu użytkowników AMSTRADA i wygodny.

Wpisywanie programu najwygodniej rozpocząć od wpisania podprogramu „POLSKIE LITERY" — litery 2110—2390. Po wpisaniu należy podprogram uruchomić (tak jak każdy program w BASIC-u) i od tej chwili posługujemy się polskim alfabetem co ułatwia wpisywanie tekstów zawartych w programie.

K. P.

```
10 MODE 1:DEG:CLS:FRAME
20 GOSUB 2110
30 LOCATE 9,1:PRINT "J A K   N A R Y S O W A Ć"
40 PRINT:PRINT:PRINT
50 PRINT:PRINT "   SYMETRALNA ODCINKA............1"
60 PRINT:PRINT "   DWUSIECZNA KĄTA...............2"
70 PRINT:PRINT "   TRÓJKĄT (3 boki)..............3"
80 PRINT:PRINT "   TRÓJKĄT (2 boki,kąt)..........4"
90 PRINT:PRINT "   TRÓJKĄT (bok,2 kąty)..........5"
100 LOCATE 1,20
110 PRINT "Do rysowania używamy linijki i cyrkla"
120 LOCATE 6,24:PRINT "Powrot do menu-dowolny znak"
130 a$=INKEY$:IF a$="1" GOTO 190
140 IF a$="2" GOTO 380
150 IF a$="3" GOTO 560
160 IF a$="4" GOTO 900
170 IF a$="5" GOTO 1330
180 GOTO 130
190 'rem "$$$$$ S Y M E T R A L N A $$$$$"
200 MODE 2:PRINT CHR$(22);CHR$(1)
210 LOCATE 30,1:PRINT "SYMETRALNA ODCINKA"
220 LOCATE 30,1:PRINT "__________________"
230 a=200:LET b=400:c=170:PLOT a,c
240 DRAW b,c:k=30:l=60:x=a:y=c:r=155
250 PRINT :PRINT:PRINT
260 PRINT "1-Rysujemy 2 łuki z"
270 PRINT "  lewego końca odcinka"
280 GOSUB 2000:k=120:l=150:x=a:y=c
290 GOSUB 2000:PRINT :PRINT:PRINT
300 LOCATE 53,7:PRINT "2-Rysujemy 2 łuki z"
310 LOCATE 55,8:PRINT "prawego końca odcinka"
320 k=210:l=240:x=b:y=c:GOSUB 2000
330 k=300:l=330:x=b:y=c:GOSUB 2000
340 LOCATE 2,22:PRINT "3-Przez punkty przecięcia"
350 LOCATE 4,23:PRINT "przechodzi symetralna odcinka"
360 x11=300:y11=30:x12=300:y12=300
370 GOSUB 2060:CLEAR INPUT:GOTO 540
380 ' "$$$$$ D W U S I E C Z N A   K Ą T A $$$$$"
390 MODE 2:PRINT CHR$(22);CHR$(1)
400 LOCATE 30,1:PRINT "DWUSIECZNA KĄTA"
410 LOCATE 30,1:PRINT "_______________"
420 PLOT 200,150:DRAW 400,100
430 MOVE 200,150:DRAW 400,200
440 LOCATE 3,14:PRINT "1-Z wierzchołka kąta"
450 LOCATE 5,15:PRINT "wykreślamy łuk"
460 x=200:y=150:l=120:k=70:r=150:GOSUB 2000
470 LOCATE 30,8:PRINT "2-Z punktów przecięcia wykreśl
amy dwa łuki"
480 x=346:y=187:k=90:l=130:r=150:GOSUB 2000
490 x=346:y=114:k=60:l=100:r=150:GOSUB 2000
500 LOCATE 3,23:PRINT "3-Przez wierzchołek kąta i pun
kt przecięcia"
510 LOCATE 46,23:PRINT " łuków przechodzi dwusieczna
kąta"
520 x11=170:y11=150:x12=510
530 y12=150:GOSUB 2060:CLEAR INPUT
540 IF INKEY$>=" " GOTO 10
550 GOTO 540
560 'rem "$$$$$ T R Ó J K Ą T (3 boki) $$$$$"
570 MODE 2:DEG:PRINT CHR$(22);CHR$(1)
580 LOCATE 30,1:PRINT "T R Ó J K Ą T"
590 LOCATE 30,1:PRINT "_________________"
600 LOCATE 40,5:PRINT "Mamy dane trzy boki trojkąta"
610 LOCATE 3,3:PRINT "a"
620 PLOT 50,356:DRAW 200,356
630 LOCATE 3,5:PRINT "b"
640 PLOT 50,324:DRAW 150,324
650 LOCATE 3,7:PRINT "c"
660 PLOT 50,296:DRAW 250,296
670 LOCATE 1,21:PRINT "1-Rysujemy odcinek np.'c'"
680 x11=50:x12=250:y11=296:y12=296
690 x21=200:x22=400:y21=100:y22=100
700 GOSUB 1890:LOCATE 39,20:PRINT "c"
710 LOCATE 3,15:PRINT "2-Z końca 'c' rysujemy"
720 LOCATE 3,16:PRINT "łuk o promieniu 'a'"
730 x11=50:x12=200:y11=356:y12=356
740 x21=200:x22=314:y21=100:y22=196
750 GOSUB 1890:LOCATE 35,16:PRINT "a"
760 x=200:y=100:r=150:k=55:l=70:GOSUB 2000
770 LOCATE 50,14:PRINT "3-Z drugiego końca 'c'"
780 LOCATE 52,15:PRINT "rysujemy łuk o promieniu 'b'"
790 x11=50:x12=150:y11=324:y12=324
800 x22=400:y22=100:x21=310:y21=142
810 GOSUB 1890:x=400:y=100:r=100
820 LOCATE 46,16:PRINT "b"
830 k=300:l=325:GOSUB 2000:LOCATE 15,24
840 PLOT 200,100:DRAW 314,196,1,1
850 PLOT 400,100:DRAW 310,142,1,1
860 PRINT "4-Łączymy punkt przecięcia łuków z końcami
odcinka 'c'"
870 x11=200:y11=100:x12=331:y12=172:GOSUB 2060
880 x11=400:y11=100:x12=331:y12=172:GOSUB 2060
890 CLEAR INPUT:GOTO 540
900 '$$$$$$  T R Ó J K Ą T ( 2 boki,kąt) $$$$$$$
910 MODE 2:DEG:PRINT CHR$(22);CHR$(1)
920 LOCATE 30,1: PRINT "T R Ó J K Ą T"
930 LOCATE 30,1:PRINT "_________________"
940 LOCATE 40,3:PRINT "Mamy dane dwa boki trójkąta"
950 LOCATE 40,5:PRINT "i kąt miedzy nimi zawarty"
960 LOCATE 3,3:PRINT "a"
970 PLOT 50,356:DRAW 250,356
980 LOCATE 3,5:PRINT "b"
990 PLOT 50,324:DRAW 200,324
1000 LOCATE 1,10:PRINT "kąt"
1010 PLOT 50,244:DRAW 200,244
1020 PLOT 50,244:DRAW 161,308
1030 LOCATE 1,21:PRINT "1-Rysujemy odcinek np.'a'"
1040 x11=50:x12=250:y11=356:y12=356
1050 x21=200:x22=400:y21=100:y22=100
1060 GOSUB 1890:LOCATE 39,20:PRINT "a"
1070 LOCATE 1,15:PRINT "2-Przenosimy kąt"
1080 x=50:y=244:r=100:k=55:l=95:GOSUB 2000
1090 x11=50:y11=244:x12=149:y12=236:x21=200
1100 y21=100:x22=276:y22=164:GOSUB 1890
1110 MOVE 200,100:DRAW 276,164,1,1
1120 x=200:y=100:r=100:k=50:l=100:GOSUB 2000
1130 x11=150:y11=244:x12=136:y12=294:GOSUB 2060
1140 x11=136:y11=294:x12=150:y12=244:x21=264
1150 y21=136:x22=300:y22=100:GOSUB 1890
1160 MOVE 300,100:DRAW 264,136,1,1
1170 x=300:y=100:r=52:k=320:l=360:GOSUB 2000
1180 LOCATE 48,12:PRINT "3-Rysujemy pomocniczą półpro
stą"
1190 LOCATE 50,13:PRINT "i odkładamy na niej odcinek
'b'"
1200 x11=200:y11=100:x12=373:y12=200:GOSUB 2060
1210 MOVE 200,100:DRAW 373,200,1,1
1220 x11=50:x12=200:y11=324:y12=324:x21=200
1230 y21=100:x22=306:y22=206:GOSUB 1890
1240 x=200:y=100:r=150:k=50:l=70:GOSUB 2000
1250 MOVE 200,100:DRAW 306,206,1,1
1260 LOCATE 31,16:PRINT "b"
1270 x11=200:y11=100:x12=329:y12=175:GOSUB 2060
1280 LOCATE 57,16:PRINT "4-Łączymy końce odcinków"
1290 LOCATE 59,17:PRINT "-rysujemy bok 'c'"
1300 x11=400:y11=100:x12=329:y12=175:GOSUB 2060
1310 LOCATE 47,16:PRINT "c"
1320 CLEAR INPUT:GOTO 540
1330 '$$$$$$ T R Ó J K Ą T ( bok,2 kąty) $$$$$$$
1340 MODE 2:DEG:PRINT CHR$(22);CHR$(1)
1350 LOCATE 30,1: PRINT "T R Ó J K Ą T"
1360 LOCATE 30,1:PRINT "_________________"
1370 LOCATE 40,3:PRINT "Mamy dane bok trójkąta"
1380 LOCATE 40,5:PRINT "i 2 kąty do niego przyległe"
1390 LOCATE 3,3:PRINT "a"
1400 PLOT 50,356:DRAW 250,356
1410 LOCATE 3,6:PRINT "kąt 1"
1420 PLOT 50,276:DRAW 200,276
1430 PLOT 50,276:DRAW 161,340
1440 LOCATE 3,14:PRINT "kąt 2"
1450 PLOT 50,140:DRAW 200,140
1460 PLOT 50,140:DRAW 136,246
1470 LOCATE 20,22:PRINT "1-Rysujemy odcinek 'a'"
1480 x11=50:x12=250:y11=356:y12=356
1490 x21=200:x22=400:y21=100:y22=100
1500 GOSUB 1890:LOCATE 39,20:PRINT "a"
1510 LOCATE 1,18:PRINT "2-Przenosimy kąty"
1520 x=50:y=276:r=100:k=55:l=95:GOSUB 2000
1530 x11=50:y11=276:x12=149:y12=264:x21=200
1540 y21=100:x22=276:y22=164:GOSUB 1890
1550 MOVE 200,100:DRAW 276,164,1,1
1560 x=200:y=100:r=100:k=50:l=100:GOSUB 2000
1570 x11=150:y11=276:x12=136:y12=326
1580 GOSUB 2060
1590 x11=150:y11=276:x12=136:y12=326
1600 x22=264:y22=136:x21=300:y21=100
1610 GOSUB 1890
1620 MOVE 300,100:DRAW 264,136,1,1
1630 x=300:y=100:r=52:k=320:l=360:GOSUB 2000
1640 '$$$$$$DRUGI KĄT$$$$$$$$$$$$
1650 x=50:y=140:r=90:k=40:l=95:GOSUB 2000
1660 x11=50:y11=140:x12=139:y12=133:x22=400
1670 y22=100:x21=311:y21=93:GOSUB 1890
1680 MOVE 400,100:DRAW 311,93,1,1
1690 x=400:y=100:r=90:k=270:l=330:GOSUB 2000
1700 x11=140:y11=140:x12=113:y12=203:GOSUB 2060
1710 x11=140:y11=140:x12=113:y12=203:x21=310
1720 y21=100:x22=333:y22=164:GOSUB 1890
1730 MOVE 310,100:DRAW 333,164,1,1
1740 x=310:y=100:r=68:k=15:l=35:GOSUB 2000
1750 LOCATE 48,12:PRINT "3-Rysujemy 2 pomocnicze półp
roste"
1760 x11=200:y11=100:x12=373:y12=200:GOSUB 2060
1770 MOVE 200,100:DRAW 373,200,1,1
1780 x11=400:y11=100:x12=300:y12=200:GOSUB 2060
1790 MOVE 400,100:DRAW 300,200,1,1
1800 LOCATE 54,16:PRINT "4-Łączymy końce odcinka 'a'"
1810 LOCATE 57,17:PRINT "z punktem przecięcia"
1820 LOCATE 57,18:PRINT "półprostych - rysujemy"
1830 LOCATE 57,19:PRINT "boki 'b' i 'c'"
1840 x11=200:y11=100:x12=328:y12=174:GOSUB 2060
1850 LOCATE 34,16:PRINT "b"
1860 x11=400:y11=100:x12=328:y12=174:GOSUB 2060
1870 LOCATE 47,16:PRINT "c"
1880 CLEAR INPUT:GOTO 540
1890 'rem "$$$$$ PRZESUWANIE ODCINKA $$$$$"
1900 dx1=(x21-x11)/30:dx2=(x22-x12)/30
1910 dy1=(y21-y11)/30:dy2=(y22-y12)/30
1920 FRAME:FOR i=1 TO 30
1930 x11=x11+dx1:x12=x12+dx2
1940 y11=y11+dy1:y12=y12+dy2
1950 MOVE x11,y11:DRAW x12,y12,1,1
1960 FOR p=0 TO 50:NEXT
1970 MOVE x11,y11:DRAW x12,y12,1,1
1980 NEXT:MOVE x11,y11
1990 DRAW x12,y12:RETURN
2000 'rem "$$$$$$$$ RYSOWANIE ŁUKU $$$$$$$$$"
2010 FOR i=k TO l STEP 5
2020 MOVE x,y:DRAW x+r*SIN(i),y+r*COS(i),1,1
2030 MOVE x,y:DRAW x+r*SIN(i),y+r*COS(i),1,1
2040 DRAW x+r*SIN(i-5),y+r*COS(i-5),1,3
2050 NEXT:RETURN
2060 ' "$$$ R Y S O W A N I E   O D C I N K A $$$"
2070 dx=(x12-x11)/400:dy=(y12-y11)/400
2080 MOVE x11,y11:FOR i=1 TO 400
2090 x11=x11+dx:y11=y11+dy
2100 DRAW x11,y11,1,3:NEXT:RETURN
2110 'rem "$$$$$ P O L S K I E   L I T E R Y $$$$$"
2120 SYMBOL AFTER 170
2130 SYMBOL 171,0,0,120,12,124,204,118,3
2140 SYMBOL 172,24,60,102,102,126,102,102,3
2150 SYMBOL 173,24,0,60,102,96,102,60,0
2160 SYMBOL 174,24,60,102,192,192,102,60,0
2170 SYMBOL 175,0,0,60,102,126,96,60,6
2180 SYMBOL 176,254,98,104,120,104,98,254,3
2190 SYMBOL 177,56,24,28,24,56,24,60,0
2200 SYMBOL 178,240,96,112,96,226,102,254,0
2210 SYMBOL 179,24,0,220,102,102,102,102,0
2220 SYMBOL 180,24,198,230,246,222,206,198,0
2230 SYMBOL 181,24,0,60,102,102,102,60,0
2240 SYMBOL 182,12,56,108,198,198,108,56,0
2250 SYMBOL 183,24,0,60,96,60,6,124,0
2260 SYMBOL 184,24,60,96,60,6,102,60,0
2270 SYMBOL 185,24,0,126,76,24,48,126,0
2280 SYMBOL 186,254,198,140,126,48,102,254,0
2290 SYMBOL 187,12,24,126,76,24,48,126,0
2300 SYMBOL 188,48,254,172,24,50,102,254,0
2310 KEY DEF 10,1,55,171,172
2320 KEY DEF 11,1,56,173,174
2330 KEY DEF 3,1,57,175,176
2340 KEY DEF 20,1,52,177,178
2350 KEY DEF 12,1,53,179,180
2360 KEY DEF 4,1,54,181,182
2370 KEY DEF 13,1,49,183,184
2380 KEY DEF 14,1,50,185,186
2390 KEY DEF 5,1,51,187,188
2400 RETURN
```

CELL – komórka, adres,
CELLAR – stos (w pamięci),
CELLULAR ARRAY – układ komórkowy,
CENTER – ośrodek,
CENTER FREQUENCY – częstotliwość nomina,
CENTRAL COMPUTER – komputer centralny,
CENTRAL PROCESSING UNIT (CPU) – jednostka centralna,
CENTRAL PROCESSOR – jednostka centralna,
CENTRAL STORAGE – pamięć operacyjna, pamięć centralna,
CERTIFIER – maszyna do regeneracji taśm magnetycznych,
CHAD(DED) TAPE – taśma dziurkowana całkowicie,
CHADLESS TAPE – taśma dziurkowana niecałkowicie,
CHADS – ścinki papierowe (powstałe przy dziurkowaniu taśmy lub karty),
CHAFF – konfetti, wycinki (kart lub taśm dziurkowanych),
CHAIN – łańcuch, szereg, zbiór liniowo uporządkowany,
CHAIN CODE – kod łańcuchowy,
CHAIN PRINTER – drukarka łańcuchowa,
CHAINED – także: doładowywany, dopinana (procedura),
CHAINED PROGRAM – program dołączany,
CHAINING – łączenie dotyczące adresowania pamięci masowych, łączenie (np. rozkazów), także: dzielenie programu na części, tworzenie łańcucha, wiązanie łańcuchowe,
CHANCHANGE – zmiana kanału,
CHANGE – zmiana, zmieniać,
CHANGE FILE – kartoteka zmian, kartoteka aktualizująca,
CHANGE OVER – przełączać,
CHANGE RECORD – zapis (rekord) zmian,
CHANGE TYPE – taśma zmian (dot. organizacji taśmy magnetycznej),
CHANGELABLE STORAGE – pamięć wymienna,
CHANGEOVER ONE-FOR-ONE – wprowadzenie nowego systemu pracy z jednoczesnym usunięciem starego,
CHANNEL – kanał,
CHANNEL ADDRESS WORD – słowo adresowe kanału,
CHANNEL CAPACITY – pojemność (zdolność przepustowa) kanału,
CHANNEL CONTROLLER – urządzenie sterujące kanałem,
CHANNEL INSTRUCTION – rozkaz kanałowy,
CHANNEL JUMP INSTRUCTION – rozkaz skoku kanałowy,
CHANNEL LOAD – zajętość kanału,
CHANNEL PROGRAM – program kanałowy,
CHAPTER – rozdział (programu),
CHARACTER – znak,
CHARACTER ADDRESSING – adresowanie pozycji, adresowanie znaku,
CHARACTER ADJUSTMENT – także: przesunięcie adresu,
CHARACTER ARITHMETIC – arytmetyka znakowa,
CHARACTER CODE MATRIX – matryca kodów znaków,
CHARACTER COMPUTER – komputer pozycyjny, komputer znakowy,
CHARACTER CONCENTRATOR – konwerter znaków,
CHARACTER CONVERTER – konwerter znaków,
CHARACTER DEFINITION TAG – znak definicji typu (na końcu nazwy zmiennej),
CHARACTER DENSITY – gęstość zapisu znaków,
CHARACTER ERROR PROBABILITY – stopa błędów znakowa,
CHARACTER ERROR RATE – wskaźnik błędów w znakach,
CHARACTER GENERATOR – generator znaków,
CHARACTER MACHINE – komputer pozycyjny, komputer znakowy,
CHARACTER PRINTER – drukarka znakowa, drukarka szeregowa,
CHARACTER READER – czytnik znaków,
CHARACTER RECOGNITION – rozpoznawanie znaków,
CHARACTER SET – zestaw znaków, także: zestaw przebitkowy,
CHARACTER STRING – ciąg znaków,
CHARACTER SUBSET – podzbiór znaków,
CHARACTER TRANSLATOR – konwerter znaków,
CHARACTER-AT-A-TIME PRINTER – drukarka znakowa,
CHARACTER-ORGANIZED STORAGE – pamięć o organizacji znakowej,
CHARACTERISTIC – cecha, cecha charakterystyczna, charakterystyka,
CHARGE-COUPLED DEVICE (CCD) – element ze sprzężeniem ładunkowym,
CHART – schemat, rysunek (techniczny), tabela (statystyczna), wykres,
CHARTING LAYOUT – wykres zamierzonej koncepcji (działania),
CHECK – także: analiza,
CHECK BIT – bit kontrolny,
CHECK CHARACTER – znak kontrolny,
CHECK COLUMN – kolumna kontrolna,
CHECK COMMAND – rozkaz weryfikacji,
CHECK DIGIT – cyfra kontrolna,
CHECK IN – wpisać się na listę, nawiązać łączność,
CHECK LIST – zestawienie (spis) kontrolne,
CHECK METER – przyrząd pomiarowy kontrolny,
CHECK NUMBER – numer kontrolny,
CHECK OUT – sprawdzać, kontrolować, test sprawdzający, także: wypisać się, zakończyć seans łączności,
CHECK POINT – punkt kontrolny,
CHECK READING – czytanie kontrolne, czytanie sprawdzające,
CHECK SUM – suma kontrolna,
CHECK TIME – czas sprawdzania,
CHECK TRACK – ścieżka kontrolna,
CHECKING CIRCUIT – układ kontrolny, układ sprawdzający,
CHECKING PROGRAM – program sprawdzający,
CHECKPOINT LABEL – etykieta punktu wznowienia,
CHIEF PROGRAMMER TEAM – zespół programisty wiodącego (zespół głównego programisty),
CHILD PROCESS – przebieg potomka (dot. ciągu programów),
CHIP – struktura półprzewodnikowa, moduł, kość (półprzewodnika),
CHIP BONDING – mikromontaż struktury,
CHIP BOX – pojemnik na ścinki (np. w dziurkarce kart, dziurkarce taśmy papierowej),
CHIP CIRCUIT – układ scalony pastylkowy,
CHIP ENABLE (CE) – sygnał wyboru struktury pamięci, uaktywnienie modułu,
CHIP ENABLE INPUT – wejście uaktywnienia,
CHIP SELECT – wybór modułu,
CHIP SELEKT INPUT – wejście wyboru,
CHIPPINGS – ścinki papierowe (powstałe przy dziurkowaniu taśmy lub karty),
CHOICE – wybór, dobór (np. maszyny),
CHROMA KEY – kluczowanie kolorem (TV),
CIM – patrz: COMPUTER INPUT MICROFILM,
CIPHER – cyfra, także: szyfr, szyfrować, kodować,
CIRCUIT – układ, obwód elektryczny,
CIRCUIT ALGEBRA – algebra schematów,
CIRCUIT ELEMENT – element przełączający,
CIRCULAR SHIFT – przesunięcie cykliczne,
CIRCULATING DECIMAL – ułamek dziesiętny okresowy,
CIRCULATING MEMORY – pamięć na liniach opóźniających,
CIRCULATING STORE – pamięć na liniach opóźniających,
CISC – patrz: COMPLEX INSTRUCTION SET COMPUTER,
CIU – patrz: COMPUTER INTERFACE UNIT,
CLAMPING – obcinanie, zakończenie wykreślania wektorów na brzegu ekranu,
CLASS – rodzaj,
CLASSIFIED INFORMATION – informacja zastrzeżona,
CLAUSE – klauzula, warunek,
CLEANER – filtr,
CLEAR – kasować, wyzerować (np. licznik, stan rejestru), wymazać (np. pamięć maszyny),
CLEAR BAND – miejsce na dokumencie, które nie może być pokryte drukiem,
CLEAR STATE – start czysty,
CLEAR THE REGISTER – wyzerować rejestr,
CLEARANCE – kasowanie, wymazywanie, przywracanie do położenia wyjściowego,
CLEARING – kasowanie, wymazywanie, przywracanie do położenia wyjściowego,
CLEARING ROUTINES – programy wymazywania danych,
CLIPPING – obcinanie, zakończenie wykreślania wektorów na brzegu ekranu,
CLOCK – zegar, licznik, czujnik zegarowy,
CLOCK CIRCUIT – układ generacji sygnałów zegarowych,
CLOCK DRIVER – zasilacz zegarowy,
CLOCK FREQUENCY – częstotliwość cyklu zegarowego,
CLOCK GENERATOR – generator zegarowy,
CLOCK PERIOD – takt zegarowy,
CLOCK RATE – częstotliwość cyklu zegarowego,
CLOCK SIGNAL – sygnał zegarowy,
CLOCK TRACK – ścieżka synchronizująca (zegarowa),
CLOCKING – synchronizowanie, taktowanie (impulsów, sygnałów),
CLOSED LOOP – pętla zamknięta (rozkazów),
CLOSED SHOP – w żargonie: niedostępny (zamknięty) ośrodek obliczeniowy,
CLOSED SUBROUTINE – podprogram zamknięty,
CLOSED-LOOP SYSTEM – praca sprzężona z procesem w układzie zamkniętym,
CLOSELY COUPLED MICROCOMPUTER SYSTEMS – systemy mikrokomputerowe silnie powiązane (komunikacja przez wspólną pamięć operacyjną),
CLOSING A FILE – zamykanie pliku (zbioru),
CLUSTER – niepodzielna jednostka allokacji (porcja), fragment pliku (zbioru), klaster,
CLUSTER SET – pochodna zbioru, zbiór punktów skupienia,
CMC-7 FONT – pismo CMC-7,
CNC – patrz: COMPUTER NUMERICAL CONTROL,
CO-PROCESOR – procesor dodatkowy (wykonuje tylko rozkazy przeznaczone dla siebie, resztę pomija),
COAXIAL CABLE – kabel koncentryczny, kabel współosiowy,
COAXIAL TRANSCEIVER INTERFACE (CTI) – sprzężenie nadajnika-odbiornika z kablem,
CODASYL – patrz: CONFERENCE ON DATA LANGUAGE,

CODE - kod źródłowy, kod, także: program, kod programu,
CODE CHECK - kontrola kodu,
CODE COMBINATION - ciąg kodowy,
CODE CONVERTER - przetwornik kodu,
CODE NUMBER - liczba zakodowana,
CODE SEGMENT - segment programu,
CODE SET - lista kodu,
CODE SWITCHOVER - przełączanie kodu,
CODE SYMBOL - symbol kodowy,
CODE SYMBOL POSITION - pozycja kodowa, rząd kodowy,
CODE TRANSLATION - przekodowanie,
CODE TREE - drzewo kodowe, schemat kodowy,
CODED GRAPHIC - grafika konturowa,
CODED IMAGE - obraz konturowy (rysunek liniowy),
CODER - koder, urządzenie kodujące,
CODER NETWORK - tablica kodowania, matryca kodowania,
CODEWORD - słowo kodu,
CODIFY - kodować,
CODING - kodowanie,
CODING FORM - formularz programowy, formularz kodowy,
CODING LANGUAGE - język kodowania,
CODING MATRIX - tablica kodowania, matryca kodowania,
CODING SHEET - formularz programowy, formularz kodowy,
COEFFICIENT - współczynnik,
COEFFICIENT FIELD - ciało współczynników,
COINCIDE - pokrywać się z ..., zbiegać się z ...,
COINCIDENT-CURRENT MEMORY (CCM) - pamięć koincydencyjna,
COLD START - zimny start także: proces ładowania i inicjacji systemu operacyjnego,
COLLECT - gromadzić, zbierać (np. dane),
COLLECTING - gromadzenie, zbieranie (np. danych),
COLLECTION - zbiór, także: zbieranie, gromadzenie,
COLLECTION OF INFORMATION - zbiór informacji,
COLON - dwukropek,
COLOR SUPRESSED MODE - tryb z wygaszonym kolorem,
COLUMN - kolumna (np. karty dziurkowanej), także: rubryka pionowa w tabeli,
COLUMNAL LAYOUT - rozmieszczenie rubryk (np. na zestawieniu),
COM - patrz: COMPUTER OUTPUT MICROFILM,
COM DEVICE - rejestrator mikrofilmowy (rejestruje obraz z komputera na mikrofilmie lub mikrofiszy),
COMB ACCESS - dostęp grzebieniowy,
COMBINABILITY - zdolność tworzenia połączeń, łączenia się w kombinacje,
COMBINATION - łączenie, wiązanie, kombinacja,
COMBINATIONAL CIRCUIT - sieć przełączająca,
COMBINED (READ/WRITE) HEAD - głowica uniwersalna,
COMMA - przecinek,
COMMAND - rozkaz, także: instrukcja programu,
COMMAND CHAINING - wiązanie łańcuchowe rozkazów,
COMMAND CYCLE - cykl rozkazu,
COMMAND LANGUAGE - język rozkazów,
COMMAND LEVEL - poziom procesora,
COMMENCE - zacząć, rozpocząć,
COMMENSURABILITY - współmierność,
COMMENT - komentarz,
COMMENT FIELD - pole komentarza,
COMMERCIAL DATA PROCESSING - przetwarzanie danych gospodarczych,
COMMON BUSINESS ORIENTED LANGUAGE (COBOL) - wysokopoziomowy język programowania (proceduralny),
COMMON DENOMINATOR - wspólny mianownik,
COMMON NODE - punkt wspólny,
COMMON-COLLECTOR CONNECTION - układ (tranzystorowy) o wspólnym kolektorze,
COMMON-EMITTER CONNECTION - układ (tranzystorowy) o wspólnym emiterze,
COMMONLY-USED - zwykle używany,
COMMUNICATION INSTRUCTION - rozkaz komunikacyjny,
COMMUNICATION LAYOUT - rozmieszczenie przewodów, plan połączeń,
COMMUNICATION SESSION - seans łączności,
COMMUNICATIONS - komunikacja, przesyłanie,
COMMUNICATIONS CONNECTOR - łącze komunikacyjne,
COMPACT - gęsty, zbity, zwarty, małych rozmiarów, także: zagęszczać, zbijać,
COMPACTION - upakowanie,
COMPARABLE - porównywalny,
COMPARATIVE - porównawczy,
COMPARATIVELY - porównawczo, w porównaniu z ...,
COMPARE - porównywać,
COMPARE-AND-SWAP - porównaj i zamień,
COMPARER - układ porównywania,
COMPARING - porównywanie,
COMPARING BRUSH - szczotka porównująca,
COMPARING INSTRUCTION - rozkaz porównania,
COMPARING OPERATION - operacja porównania,
COMPARISON - porównanie,
COMPASS - granica, skala, zasięg,
COMPATIBILITY - wymienność, kompatybilność, zgodność,
COMPATIBLE - kompatybilny, wymienny,
COMPENSATE - kompensować, wyrównywać,
COMPILATION - tłumaczenie (programu źródłowego na język wynikowy),
COMPILE - kompilować (tłumaczyć) czyli przekształcać program ujęty w języku problemowym na język maszyny (wynikowy), także: opracowywać, zestawiać,
COMPILER - program kompilujący, kompilator,
COMPILER COMPILER - kompilator kompilatorów,
COMPILING - tłumaczenie (programu źródłowego na język wynikowy),
COMPILING STATEMENT - instrukcja translacji,
COMPLEMENT - uzupełnienie, dopełnienie,
COMPLEMENTATION - dopełnienie, uzupełnienie,
COMPLETE - kończyć, zakończyć wykonywanie czynności,
COMPLETE CARRY - przeniesienie natychmiastowe w dodawaniu równoległym,
COMPLETION - zakończenie (np. czynności, pracy),
COMPLEX - liczba zespolona także: skomplikowany, zawiły, złożony, kompleksowy,
COMPLEX INSTRUCTION SET COMPUTER - komputer o złożonej liście rozkazów,
COMPLEX NUMBERS - liczby zespolone,
COMPLEXITY - złożoność,
COMPONENT - część składowa, element, składnik,
COMPONENT PART - część składowa,
COMPOSITE COLOR MONITOR - zespolony monitor kolorowy,
COMPOSITE NUMBER - liczba złożona,
COMPOSITE VIDEO - zespolone wyjście video,
COMPOSITION OF A RECORD - budowa zapisu (rekordu),
COMPOUND - złożony, także: łączyć ze sobą,
COMPOUND COMPUTERS - komputery sprzężone,
COMPOUND CONDITION - warunek złożony,
COMPREHEND - zrozumieć,
COMPREHENSIVE - obszerny, wszechstronny, wyczerpujący,
COMPRESSED FORMAT - format zmniejszony, format zagęszczony,
COMPRESSION - ściśnianie,
COMPRISE - zawierać,
COMPROMISING EMANATIONS - przypadkowe przekazywanie sygnałów odnoszących się do tajnych informacji,
COMPUNICATION - skomputeryzowane systemy łączności (computer communication),
COMPUTATION - liczenie, obliczanie,
COMPUTATION CENTER - centrum obliczeniowe,
COMPUTATIONAL - obliczeniowe,
COMPUTATIONAL LINGUISTICS - lingwistyka komputerowa,
COMPUTATIONAL SCHEME - schemat obliczeń,

COMPUTATIONS AIDS - środki obliczeniowe,
COMPUTE - liczyć, obliczać,
COMPUTED ENTRY TABLE - tablica wpisów o adresach obliczanych,
COMPUTER - komputer, maszyna matematyczna, maszyna licząca, elektroniczna maszyna cyfrowa,
COMPUTER-AIDED - skomputeryzowany,
COMPUTER AIDED DESIGN (CAD) - projektowanie układów wspomagane komputerem,
COMPUTER AIDED EDUCATION - nauczanie wspomagane komputerem,
COMPUTER AIDED INSTRUCTION - szkolenie wspomagane komputerowo,
COMPUTER AIDED LEARNING - nauczanie wspomagane komputerem,
COMPUTER AIDED MANUFACTURE - wytwarzanie wspomagane komputerowo,
COMPUTER AIDED MEASUREMENT AND CONTORL - komputerowo wspomagane pomiary i sterowanie,
COMPUTER ANALYST - analityk informatyk,
COMPUTER APPLICATION FOR MEASUREMENT AND CONTROL (CAMAC) - międzynarodowy skomputeryzowany system modułowy pomiarów i sterowania,
COMPUTER BULLETIN BOARD - komputerowa tablica informacyjna,
COMPUTER BUREAU - usługowe biuro obliczeniowe,
COMPUTER CENTRE - ośrodek obliczeniowy,
COMPUTER CODE - lista rozkazów komputera,
COMPUTER COMPATIBILITY - kompatybilność komputerów, wymienność komputerów,
COMPUTER CONFIGURATION - zestaw komputera,
COMPUTER CONTROL - sterowanie komputerowe,
COMPUTER CONTROLLED SYSTEM - układ automatyczny sterowany komputerem,
COMPUTER DESIGN LANGUAGE (CDL) - język do opisywania komputerów cyfrowych,
COMPUTER DIAGNOSTICS - diagnostyka komputerowa,
COMPUTER DISPLAY - monitor ekranowy,
COMPUTER EVALUATION - ocena komputerów,
COMPUTER FAMILY - rodzina komputerów,
COMPUTER GENERATIONS - generacje komputerów,
COMPUTER GRAPHICS - grafika komputerowa,
COMPUTER HARDWARE - sprzęt komputerowy,
COMPUTER INDEPENDENT LANGUAGE - język niezależny od komputera,
COMPUTER INPUT MICROFILM - urządzenie wejściowe mikrofilmowe,
COMPUTER INSTRUCTION - rozkaz komputera,
COMPUTER INSTRUCTION CODE - kod wewnętrzny maszyny, kod instrukcji wewnętrznych,
COMPUTER LANGUAGE - język komputera, język wewnętrzny,
COMPUTER LITERACY - znajomość komputerowego abecadła,

Generowanie liczb pierwszych — sito Eratostenesa

Generowanie liczb pierwszych nie jest zagadnieniem banalnym. Do dnia dzisiejszego brak jest równania rekurencyjnego lub jakiejkolwiek funkcji generującej, która pozwoliłaby na otrzymanie liczb pierwszych wg jednolitego przepisu. Jedyną znaną metodą generowania liczb pierwszych jest słynne **sito Eratostenesa**, wymyślone już w II w. p.n.e. Metodę tę można stosować tylko wówczas, gdy problem sformułowany jest następująco:

Znaleźć wszystkie *liczby pierwsze* ≤N, gdzie N jest z góry zadaną liczbą.

Jeżeli problem sformułowany jest w postaci:

znaleźć N kolejnych liczb pierwszych,

trzeba stosować inny algorytm polegający na sprawdzaniu podzielności kolejnych liczb naturalnych nieparzystych i = 3, 5, 7, 9,... przez wiele podzielników $\leq \sqrt{i}$. Wymaga to dużego nakładu obliczeń i jest bardzo czasochłonne (funkcja pierwiastkowania występująca wielokrotnie w algorytmie jest jedną z najdłużej obliczanych funkcji). Nie jest to właściwie metoda generowania liczb pierwszych, lecz metoda sprawdzenia, czy dana liczba jest liczbą pierwszą.

Algorytm sita Eratostenesa jest następujący.

1) tworzymy tablicę liczb naturalnych 1, 2, 3,..., N,
2) wiadomo, że 2 jest najmniejszą liczbą pierwszą, w związku z czym usuwamy z tablicy wszystkie wielokrotności liczby 2,
3) pierwsza po prawej niewykreślona liczba jest liczbą pierwszą,
4) usuwamy z tablicy wszystkie wielokrotności tej liczby,
5) powtarzamy czynności (3) i (4) do chwili, gdy niewykreślona liczba jest nie mniejsza od N/2,
6) wszystkie niewykreślone z tablicy liczby są liczbami pierwszymi.

Poniższy program napisany w języku BASIC na ZX Spectrum generuje liczby pierwsze metodą sita Eratostenesa.

```
 399 REM "Sito Eratostenesa"
     ************************
      (C)  Janusz  MORBITZER
     ************************
 400 BEEP .2,10: LET v=3: CLS :
PRINT AT v,4;"Program  znajduje
liczby";AT v+3,4;"pierwsze zawar
te w prze-";AT v+6,4;"dziale <1,
k> metoda sita";AT v+9,9;"Eratos
tenesa.": PRINT BRIGHT 1;AT v+13
,4;"########################"
 401 INPUT "             Podaj k:
"; LINE k$: IF LEN k$=0 THEN GO
TO 400
 402 FOR i=1 TO LEN k$: IF k$(i)
>="0" AND k$(i)<="9" THEN NEXT i
: LET k=VAL k$: GO TO 405
 403 GO TO 400
 405 IF k<=1 THEN CLS : PRINT BR
IGHT 1;AT 10,3;"Brak liczb pierw
szych w tym": PRINT BRIGHT 1;AT
12,11;"przedziale": GO SUB 475:
GO TO 400
 408 IF k>7000 THEN CLS : PRINT
BRIGHT 1;AT 10,5;"Za duża liczba
!(>7000)": BEEP .4,0: GO SUB 475
: GO TO 400
 410 DIM b(k)
 411 GO SUB 485
 412 FOR i=1 TO k: LET b(i)=i: N
EXT i
 415 LET i=0: LET p=2
 417 BEEP .5,10: BORDER 7: PAPER
 7: INK 0: CLS : PRINT BRIGHT 1;
AT 0,8;"LICZBY PIERWSZE.": PRINT
 AT 1,8;"________________": PRIN
T
 420 FOR j=2*p TO k STEP p
 425 LET b(j)=0
 430 NEXT j
 432 LET l$=STR$ p: LET dl=LEN l
$
 433 LET pp=i+1-INT ((i+1)/2)*2
 435 IF pp=1 THEN PRINT TAB 8-dl
;p;: GO TO 438
 437 PRINT TAB 25-dl,p
 438 LET i=i+1
 440 LET p=p+1
 442 IF p>k THEN GO TO 460
 445 IF b(p)<>0 THEN GO TO 420
 450 GO TO 440
 460 PRINT : GO SUB 475: CLS : P
RINT BRIGHT 1,AT 9,6;"W przedzia
le <1,";k;">";AT 11,8;"znajduje
sie ";i;AT 13,8;"liczb pierwszyc
h": GO SUB 475
 470 CLS : PAUSE 0: GO TO 400
 475 REM podprogram
 480 PRINT #1; BRIGHT 1; FLASH 1
,AT 0,4;"Naciśnij dowolny klawis
z": PAUSE 0: RETURN
 485 REM podprogram
 490 BORDER 0: PAPER 0: INK 7: C
LS : PRINT AT 8,9; FLASH 1;"PROS
ZE CZEKAC!": PRINT AT 12,7;"(tr
waja obliczenia)": RETURN
 495 REM koniec programu
```

Czas, jaki upływa od podania komputerowi wartości parametru N do pojawienia się na ekranie komunikatu LICZBY PIERWSZE jest niezbędny do inicjalizacji tablicy A/N/ i wypełnienia jej elementów kolejnymi liczbami naturalnymi /A/I/ = I dla I = 1, 2, ..,N/. Na ekranie obserwujemy pojawienie się kolejnych liczb pierwszych. Jest rzeczą charakterystyczną, że zaczynają się one pojawiać coraz szybciej. Wynika to z faktu, że rozmieszczenie liczb pierwszych w zbiorze liczb naturalnych jest nierównomierne: początkowo jest ich stosunkowo dużo, a następnie coraz mniej (np. w przedziale /1, 1000/ jest 168 liczb pierwszych, w przedziale /1001, 2000/ jest ich 135, a w przedziale /2001, 3000/ tylko 127). Większa ilość liczb pierwszych w początkowej fazie implikuje większy nakład obliczeń (większą liczbę wykreśleń, które w programie zostały zastąpione zerowaniem elementów tablicy A), a co za tym idzie dłuższy czas pomiędzy pojawieniem się na ekranie monitora kolejnych liczb pierwszych.

Największą dotychczas znaną liczbą pierwszą jest $2^{11213} - 1$. Odkryta została w roku 1963 przez **D.B. Gilliesa**, oczywiście przy użyciu maszyny cyfrowej. Znane są również tzw. liczby bliźniacze, tj. liczby pierwsze różniące się od siebie o 2, np. 5 i 7 czy 881 i 883. Oczywiście im większe liczby, tym trudniej znaleźć parę liczb bliźniaczych. Czytelnikom proponuję zmodyfikowanie przedstawionego algorytmu, tak aby program znajdował liczby bliźniacze.

Janusz MORBITZER

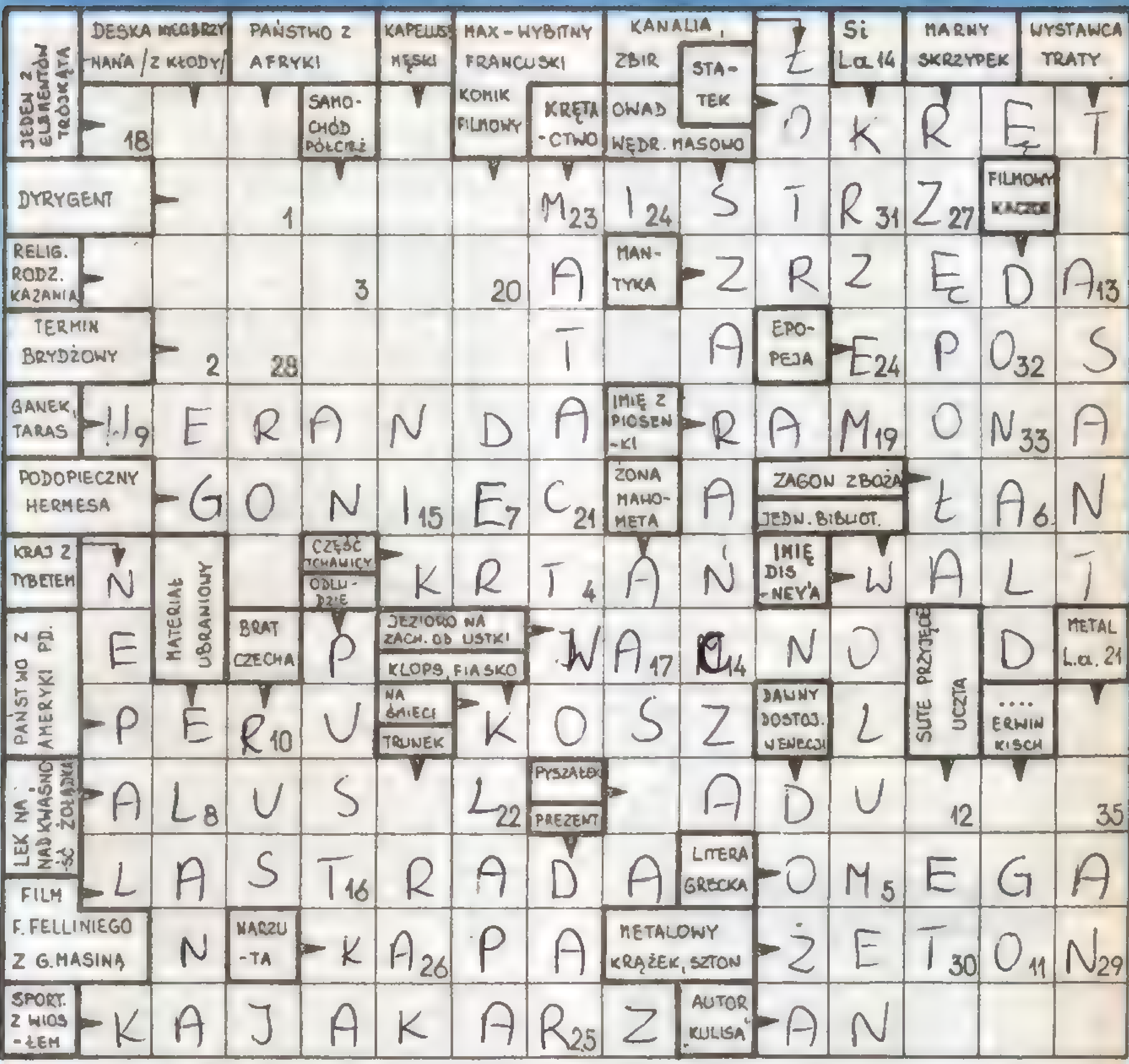
Literatura

1) Corge C. „Elementy informatyki. Informatyka a myśl ludzka", PWN, 1981.

2) Tassel D. „Praktyka programowania", WNT, 1982.

KRZYŻÓWKA NR 1

KUPON



Litery z kratek ponumerowanych od 1 do 35 utworzą nazwy 7 producentów sprzętu komputerowego, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji na kartkach pocztowych w terminie do końca lutego naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 9554. Nr ind. 361682.

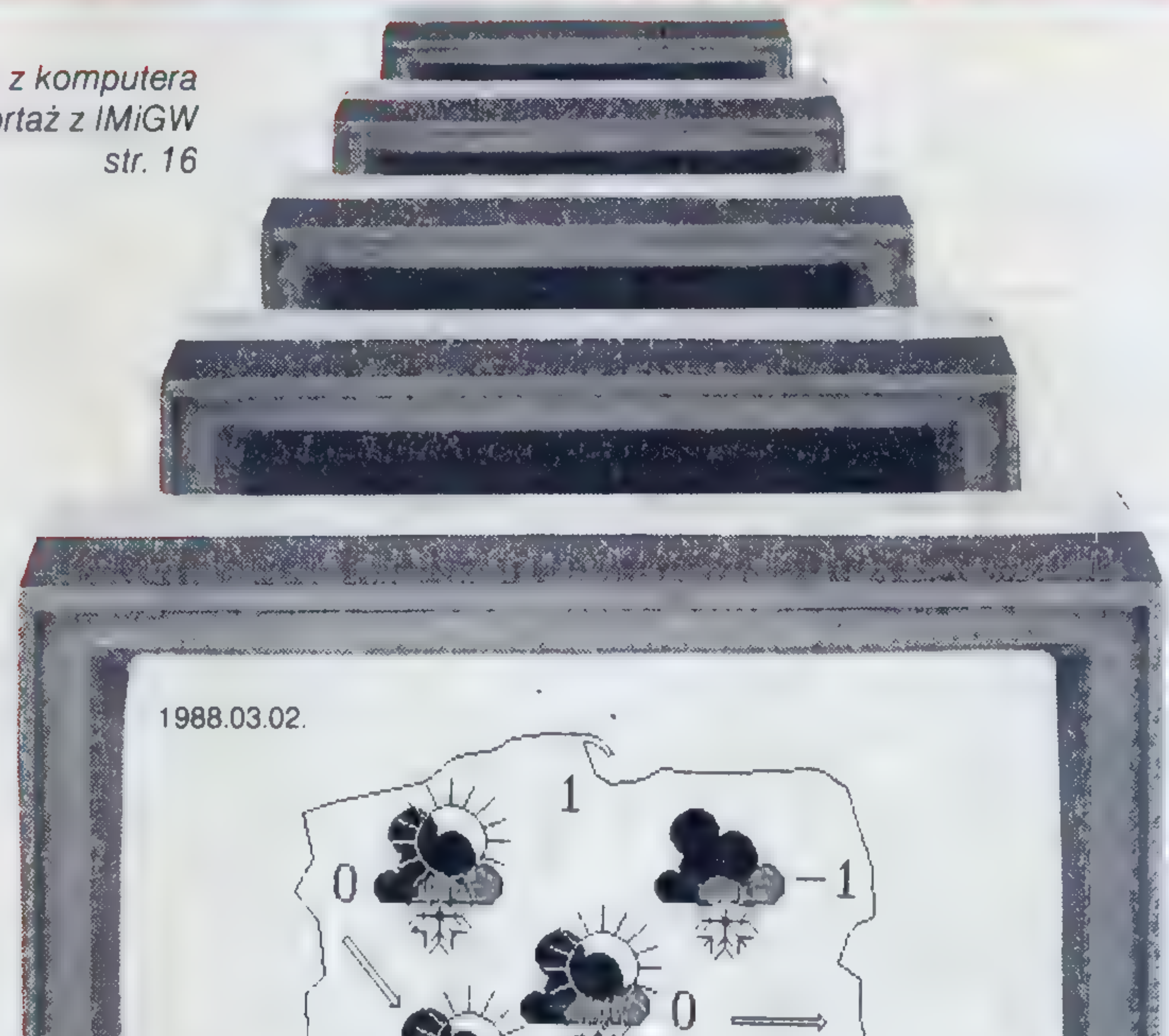
INFORMATYKA KOMPUTERY SYSTEMY



CENA 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" Nr 2/1988 ISSN 0860—2794

Pogoda z komputera — reportaż z IMiGW *str. 16*



PC STANDARD '88

W dniach 9—12 lutego br. odbyła się kolejna konferencja informatyczna. Tym razem aż pod dziewięcioma niezależnymi tytułami. Między innymi omawiane były takie tematy, jak: sieci, wielodostęp, wspomaganie pomiarów, prac wydawniczych, zarządzania i edukacji

Zważywszy na to, że konferencja była imprezą towarzyszącą wystawie pod tym samym tytułem, jej uczestnicy mogli zobaczyć wiele ciekawego sprzętu i oprogramowania (stanowić to będzie przedmiot samodzielnych publikacji, zapewne nie tylko w „IKS-ie")

Miałem opłacone dwie konferencje (a' 13 000 zł): „Sieci lokalne i wielodostęp" oraz „Zastosowania edukacyjne mikrokomputerów".

„Sieci" odbywały się już od świtu pierwszego dnia. Konkretnie zgodnie z planem konferencja ta miała się rozpocząć o godz. 10.30. Niestety rozdawanie materiałów i „przepustek" na salę konferencyjną („wizytówki na pierś") zmieniło nieco ten plan. Biada tym co nie pobrali „wizytówek" i chcieli wejść na salę — Cerber w postaci uroczej dziewczyny delikatnie, acz stanowczo zagradzał przejście na salę nie mającym „przepustki". Natomiast na sali prowadzący co kilka minut szacował, jaka część uczestników jest obecna. Siedemdziesięciopięcioprocentowa frekwencja miała być „progiem" rozpoczęcia, pozostali uczestnicy nadal czekali po „wizytówki".

O godzinie 11.10 konferencja rozpoczęła się. Na scenie sali konferencyjnej zgromadzono około 10 monitorów (w tym jeden ciekłokrystaliczny) oraz dwa ekrany — jeden do powielania obrazu LCD oraz wyświetlania slajdów, drugi — wideoskop ukazujący to, co „widziała" kamera — ekrany monitorów. Potem okazało się, że monitory te stanowiły wyposażenie kilku komputerów z terminalami połączonymi w sieć

Wykład rozpoczął pan doktor (nazwisko nie zostało podane). Było to autentyczne wprowadzenie na poziomie przeliczania bodów na bajty i odwrotnie, definiowania sieci i wielodostępu. Mój sąsiad zdrzemnął się nieco. Osobiście uniknąłem tego nietaktu, wobec prelegenta, za przyczyną supermocnych lamp panów z Teleexpressu.

Ożywiło nieco atmosferę wystąpienie Ryszarda Kajkowskiego. Dyrektor (właściciel) CSK z Gdyni zwrócił uwagę na pochopne wyrzucanie pieniędzy na sprzęt i oprogramowanie kupowane w celu budowy sieci lub wielodostępu. Okazuje się bowiem, że znaczna część transakcji nie jest przemyślana (nie konsultowana w CSK?). Firmy kupują sprzęt lub oprogramowanie bez systemowej analizy problemu. Pan Kajkowski niemal apelował o kompleksowe podejście do kupowania tego typu systemów. Należy mieć na uwadze całość: odpowiedni sprzęt, oprogramowanie użytkowe oraz system operacyjny. Egzemplifikacją tego problemu jest XENIX, który może pracować tylko przy instalacji niektórych kart interfejsu szeregowego

Dwa terminale podłączone do XT, a cztery do AT to maksimum znacznie przekraczające moce obliczeniowe tych mikrokomputerów. Są one substytutami m i n i k o m p u t e r ó w. Niestety nie mamy doświadczeń w eksploatacji tego typu sprzętu. Pozwala on na uzasadnione dzielenie swego potencjału na wiele, bardzo wiele terminali

Interesującą propozycję (handlową) przedstawił gość z Holandii — zwartą sieć lokalną MINSTREL. Idea tego systemu polega na zgromadzeniu całej „ważnej" elektroniki w komputerze centralnym (po jednym procesorze dla każdego terminala). Wątpliwą atrakcją wystąpienia była fatalna jakość obrazów na dużych ekranach. Miały one ukazywać to, co się dzieje na ekranach monitorów sieci

Potem świadkami kampanii reklamowej sieci (konferencja?) w atrakcyjnej formie pytań, np.: „dlaczego ta sieć jest taka wspaniała?", odpowiedź: „bo ...", itp

Następnie były inne firmy, inne coraz mniej ciekawe wysiłki kolejnych prelegentów, wybrałem oglądanie wystawy

Najbardziej utkwiła mi w pamięci wizyta na stoisku prezentującym efekty cyfrowej syntezy obrazów. Jeden z wystawców (okazał się artystą-kreatorem komputerowych obrazów) opowiadał mi wrażenia z pracy na sprzęcie, który odtwarza obrazy na monitorze w postaci mozaiki 8000X8000 pikseli, przy czym każdy piksel opisywany jest 24 bitami, ale to już inna historia.

Nie sposób było nie zauważyć wszechobecnych drukarek firmy STAR. Niemal każdy komputer (były ich setki) dysponował tym sprzętem. Przebojem stają się obecnie drukarki 24-igłowe o doskonałej jakości wydruków. Firma ABC DATA proponowała ponadto między innymi nowość LC10 — drukarkę która zastępuje o 20 proc. wolniejszą NL10

Druga konferencja, w której mogłem uczestniczyć („Zastosowanie edukacyjne techniki mikrokomputerowej") odbyła się w pięknej sali Rudniewa na IV piętrze PKiN Gospodarze — Władysław Majewski, znany także jako redaktor naszej konkurencji („Komputera") wygłosił słowo wstępne, dokonując pewnych uogólnień i klasyfikacji Wystąpienie to potwierdziło, iż jesteśmy na początku długiej drogi efektywnego wykorzystania sprzętu informatycznego w szkołach wszystkich szczebli Według prelegenta proces ten uwarunkowany jest poprawnością przyjętej koncepcji dydaktycznej, poniesionych nakładów oraz odpowiednich przygotowań. Przygotowania dotyczą szczególnie kadry dydaktycznej (jej szkolenia), którą wg głosu z sali pana z ministerstwa, dokształca w tej materii aż 12 wyższych uczelni na podyplomowych kursach.

Redaktor Majewski zwrócił ponadto uwagę na brak kryteriów oceny komputerowych programów dydaktycznych oraz na ucieczkę autorów programów dydaktycznych do firm zagranicznych i spółek z o.o. Prelegent wyraził zadowolenie z „pluralizmu" rynku sprzętu komputerowego, wytykając jednocześnie szereg uchybień naszemu byłemu monopoliście — ELWRO (mając na uwadze komputer marki „Junior"). Co mamy obecnie? Co mogą kupować szkoły? Komputer ACORN wymieniony został jako pierwszy (alfabetycznie). Ten brytyjski komputer, firmowany przez BBC, jest obecnie, wg red Majewskiego, najlepiej edukacyjnie oprogramowanym komputerem na świecie. Nie zmienia to faktu, że komputer ten z całą pewnością nie jest dostępny — o czym prelegent nie wspomniał — zbyt wysoka cena tego sprzętu doprowadziła niemal do bankructwa jego producenta (niezbędna była pomoc firmy OLIVETTI, zob, Byte 85/10/9 s 386). Kolejna propozycja to AMSTRAD Zasadniczy argument jego stosowania to niska cena. Podobnie rzecz się ma z ATARI który jednak jest już starym („moralnie") komputerem. Następnie wymieniony był BOSMAN — jeszcze mało znany nasz komputer rodem z Wybrzeża. MERITUM III został wymieniony jako ostatni komputer szkolny. Podobno od 01 03.88 rozpoczęta będzie masowa produkcja tego komputera (powiedział o tym pan z ministerstwa). Jeśli idzie o klany IBM PC są one po prostu dla szkół za drogie

Następnie dwie niewiasty, reprezentujące British Council, uzupełniły nieco słowo wstępne red. Majewskiego, podkreślając między innymi, że człowiek w nowoczesnym społeczeństwie powinien dysponować minimalną wiedzą i kulturą informatyczną Edukacja informatyczna w Wielkiej Brytanii uzyskuje bardzo wymierną pomoc od władz czego efektem jest (był) komputer BBC oraz jego wspaniałe oprogramowanie. Stwierdzenia te poparte zostały przykładami programów do nauki „ekonomicznego myślenia" oraz poprawności wytapiania stali

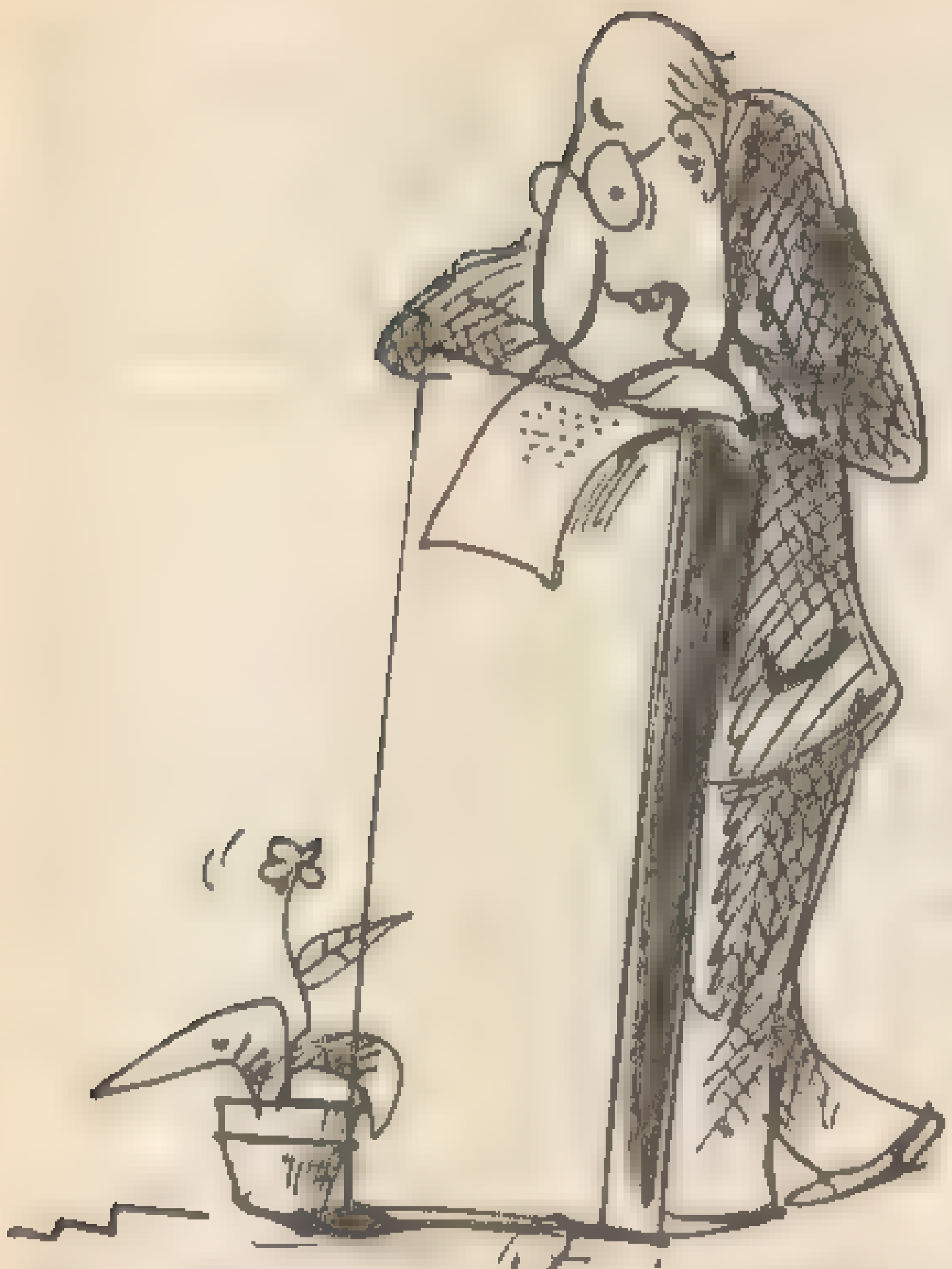
Wśród uogólnień, ciekawy wydaje się podział programów edukacyjnych na programy: z treścią, narzędziowe i symulacyjne. Oddzielne zagadnienie to programy „zastępujące nauczyciela". Doświadczenia brytyjskie dowodzą, że najpopularniejsze w szkołach podstawowych są proste programy narzędziowe np.: edytory tekstów i bazy danych

Następnie został „wywołany do tablicy" doradca ministra (niestety nie był przedstawiony) ds. edukacji informatycznej (wspomniany wcześniej pan z ministerstwa Stwierdził, iż Meritum III jest szansą dla szkół, bowiem produkcja (planowana) 400 sztuk miesięcznie niemal bilansuje, w skali 2 lat, potrzeby szkolnictwa. Komputer ten ma mieć możliwość pracy pod CPM, a nawet pod DOS-em!

Podnoszona była także kwestia praw autorskich dla twórców oprogramowania. Brak

W NUMERZE:

Komputery na start — str. 4
Programowanie strukturalne — str. 5
W szponach Atari — str. 7—11
Informatyka w wojsku — str. 14
Prognoza pogody — str. 16—18
Lilavati — str. 19
Basic — str. 21—26
Słownik — str. 29



— Jeżeli będę mówił z sensem, dajcie mi jakiś znak.

tych praw zniechęca ambitnych programistów. Podobno PTI podjęło w tej kwestii odpowiednie kroki. Ponadto, w tym kontekście, wspomniano, że na Zachodzie kupujący nasze oprogramowanie żądają dowodów legalnego zakupienia programów narzędziowych, które służyły do budowy przedmiotu transakcji. Na dodatek nasi informatycy mają coraz więcej kłopotów z otrzymywaniem wizy do krajów dysponujących atrakcyjnym programami

Pan dr O. Gajl wygłosił ciekawy wykład, zgodny z tytułem konferencji, dokonując między innymi wielu uogólnień oraz klasyfikacji (bariery, co robić, dlaczego, formy działania, warunki realizacji, standardy itp.)

Następnie wystąpił pan profesor, który co kilka lat jeździ za granicę — najczęściej do Kalifornii. Dostrzegł, że kiedyś informatyka (tam) była zajęciem elitarnym, a teraz komputery stoją (tam) na korytarzach. Osobiście sądzę, że proces ten można dostrzec także dużo bliżej — nad Wisłą

Następnie pan profesor uzasadnił autentyczność komputerowego wspomagania zdolności intelektualnych człowieka. Komputer ma wyraźny wpływ na mentalność współczesnego człowieka — zmusza do większej precyzji myślenia, ułatwia zobrazowanie (dotychczas tylko werbalnych) informacji. Pytań nie było, tylko jeden z uczestników wyraził w sposób bardzo kategoryczny swoje zdziwienie nad przerostem treści nad formą wystąpienia pana profesora. Większość informacji wyświetlanych na ekranie oraz na monitorach była nieczytelna, przeczyło to zamierzeniu, które wcześniej pan profesor sprecyzował, a także wykazało brak przydatności nowoczesnej techniki w procesie nauczania — wszak to był wykład! Adwersarz pana profesora widział w tym (wystąpieniu) nieświadome odstraszanie potencjalnych użytkowników informatyki. Potwierdzało to (ilościowo) ciągłe ubywanie początkowo licznego audytorium

Konferencję zakończyła oferta Zakładu Edukacji Komputerowej. Jednak i ona, podobnie jak wstęp wygłoszony przez red. Majewskiego nie tylko nie zmieniła, ale utwierdziła garstkę słuchaczy (około 5 proc. początkowej frekwencji) w przekonaniu, że informatyka w naszych szkołach jest przed długą, bardzo długą i wyboistą drogą

UCZESTNIK

NAUKA • TECHNIKA • NAUKA

Od kilku lat NASA planuje wysłanie specjalnej sondy międzyplanetarnej o nazwie GALILEUSZ w kierunku Jowisza. Nie posiadając wystarczająco silnej rakiety nośnej przewidywano wstępnie wykorzystanie wahadłowca kosmicznego, lecz katastrofa CHALLENGERA odsunęła realizację tego pomysłu. Ostatnio jednak w labolatorium NASA w Pasadenie (Kalifornia), za pomocą komputera, wytyczono bardzo złożoną i oryginalną trasę lotu, na której w celu rozpędzenia aparatu do niezbędnej prędkości, po wprowadzeniu go w kosmos, wykorzysta się pola grawitacyjne Wenus i Ziemi. Podróż na Jowisza zajmie 6 lat, z czego połowę - nabieranie prędkości. Aparat ma krążyć wokół Jowisza przez około 22 miesiące przesyłając na Ziemię informacje naukowe i zdjęcia planety oraz jej czterech satelitów. Zdaniem specjalistów lot GALILEUSZA zapoczątkuje nowy "złoty wiek" badania planet.

■ ■ ■

Na amerykańskim rynku samochodowym pojawił się ostatnio najmniejszy model z rodziny Fordów - FESTIVA. Został on opracowany we współpracy z japońską firmą MAZDA, a produkowany jest w zakładach Kia Motors w Korei Płd. Samochód napędzany jest 4-cylindrowym silnikiem o pojemności 1,3 l i mocy 43 kW. FESTIVA jest kilka centymetrów krótsza od znanej, także na polskim rynku, FIESTY, ma taki sam rozstaw osi i jest nieco szersza.

■ ■ ■

W Stanach Zjednoczonych, po ponad 30 lat trwających pracach badawczych, udało się zbudować maszynę, która łączy w sobie zalety helikoptera i normalnego, napędzanego silnikami turbośmigłowymi samolotu. Ten nowy statek powietrzny, który nazwano OSPREY ma być produkowany w dużych seriach najpierw na potrzeby amerykańskiego lotnictwa wojskowego, a później, co się już rozważa - cywilnego. Jego realizacja była możliwa przede wszystkim dzięki postępowi w technice obliczeniowej (stosowano CAD), w dziedzinie materiałów a zwłaszcza kompozytów. Konstrukcja OSPREY-a składa się w 60% z kompozytu włókna węglowe - żywica epoksydowa, w 20% z żywicy wzmacnianej włóknem szklanym i tylko 20% stanowią detale metalowe, co znacznie obniża jego masę i zwiększa wytrzymałość . W nowej maszynie nie zrezygnowano oczywiście z hydraulicznych elementów poruszających stery, klapy itp., ale komendy pilota docierają do nich w postaci sygnałów cyfrowych za pośrednictwem pokładowych komputerów. Dostawy ponad 900 maszyn tego typu mają się rozpocząć od 1991 roku.

TECHNIKA • NAUKA • TECHNIKA

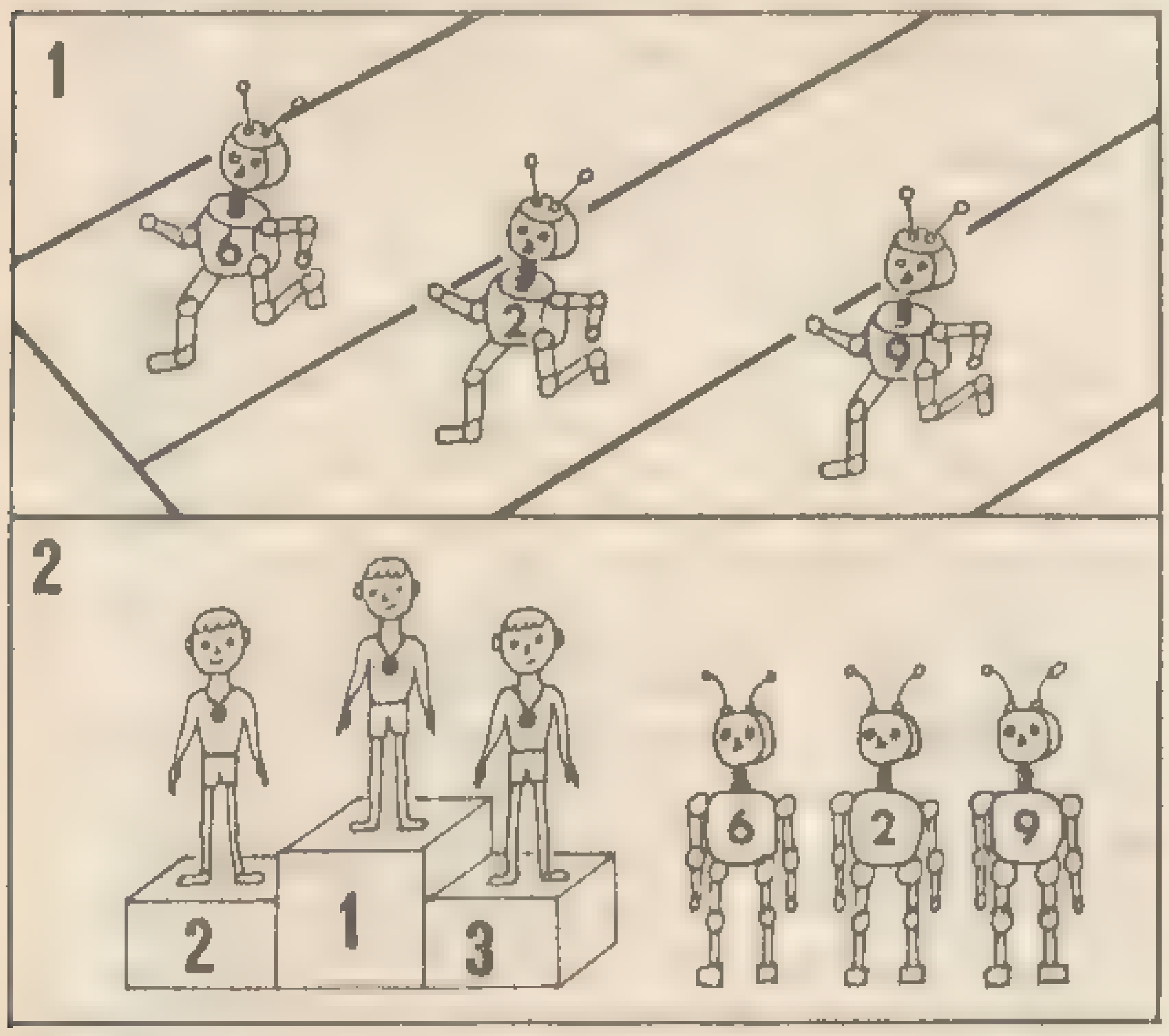
MOIM ZDANIEM

KOMPUTERY NA START

Czyli głos mają Czytelnicy. Pod tym hasłem publikować będziemy opinie i poglądy naszych Czytelników. Nie muszą one być zgodne z poglądami redakcji, ważne jest to, że ich oryginalność zmusza do refleksji, a może nawet pobudzać do twórczego działania.

Porównanie sportu do piramidy jest bardzo trafne: jej podstawa to sport masowy, a wierzchołek — wyczynowy. Nawet nie stosując wzorów matematycznych, łatwo przewidzieć, że im podstawa większa, tym wyżej można wznieść wierzchołek, nie naruszając stabilności układu. Niestety, w naszym sporcie proporcje nie zostały zachowane, nic więc dziwnego, że w większości dyscyplin piramida runęła lub chyli się ku upadkowi. Jest to widoczne gołym okiem nie tylko dla fachowców i kibiców, błędem jednak jest upatrywanie źródeł słabości w wierzchołku, czyli w sporcie wyczynowym. Bez solidnego zaplecza młodzików i juniorów nie można śnić o potędze opartej na wąskiej grupie zawodników wyczynowych.

Do osiągnięcia sukcesów nie wystarczy też tylko młodzież, nawet najbardziej utalentowana. Niezbędne są jeszcze rzesze solidnie przygotowanych trenerów i instruktorów oraz odpowiednie warunki materialne. Jeśli realnie myślimy o przezwyciężeniu sportowego kryzysu, to zarówno trenerom, jak i warunkom materialnym musimy poświęcić dużo uwagi. Na pograniczu tych dwóch elementów znajduje się moim zdaniem, komputer. Na pograniczu, bo chociaż komputer jako taki jest wytworem technicznym, to jednak może, (i w wielu krajach już odgrywa rolę „prawej ręki" trenera. Nie chcę tu gloryfikować techniki komputerowej, uważam, że nawet najlepsza maszyna w pewnych dziedzinach nie dorówna nawet gorszemu wydaniu homo sapiens. Dlatego też komputer jeszcze długo pozostanie tylko technicznym środkiem, wspomagającym pracę trenera, zawodnika i działacza. W czym to wspomaganie się przejawia?



Informatykę w sporcie można wykorzystać w procesie szkoleniowym, pracy naukowo-badawczej, przy organizacji imprez sportowych oraz w zarządzaniu działalnością organizacji sportowych.

Proces szkoleniowy

Mówiąc o procesie szkoleniowym, mam na myśli przygotowania trenerów w Akademiach Wychowania Fizycznego i szkolenie zawodników w klubach. Panuje przy tym opinia, że po pierwsze, bez informatyki trening staje się coraz mniej efektywny, zatem trzeba uczyć studentów pracy z komputerem, po drugie, uważa się, że obecne pokolenie szkoleniowców jest dla informatyki stracone. Tym samym istnieje potrzeba kształcenia studentów AWF z udziałem komputerów i wytworzenia u nich nawyku i umiejętności korzystania z nich w procesie treningowym. Umiejętność tę powinni również zdobyć lekarze, naukowcy i w pewnym stopniu zawodnicy. Do czego może być przydatny komputer w procesie treningowym?

Przede wszystkim do jego doskonalenia oraz do doskonalenia techniki. Komputer umożliwia stworzenie na podstawie istniejącej wiedzy optymalnego procesu treningowego dla każdej dyscypliny sportowej. Porównanie go z treningiem wybitnych sportowców pozwoli wyciągnąć stosowne wnioski. Więcej, znając psychofizyczne parametry swoich zawodników można opracować dla nich trening zmierzający do osiągnięcia określonych wyników. Prowadzenie ciągłej analizy obciążenia treningowego i uzyskiwanych rezultatów (też za pomocą komputera) pozwala szybko korygować trening, eliminować błędy (przeciążenia, lenistwo itp.) i prognozować wyniki. W zakresie techniki komputer umożliwia jej wzorcowe określenie, porównanie z nią technik poszczególnych zawodników, dokonanie stosownych korekt i określenie odpowiedniego treningu.

Komputer umożliwia również zgromadzenie informacji o zawodnikach, uzyskiwanych przez nich rezultatach, wynikach badań wydolnościowych, testowych i lekarskich. Taki zbiór informacji jest niezbędny do doskonalenia procesu treningowego.

Praca naukowo-badawcza

Konieczność nadążania za postępem w sporcie jest oczywista. Wymaga to stworzenia systemu informatycznego zapewniającego ciągłą aktualizację informacji o stanie nauki, metodach treningu, ewolucji sprzętu sportowego itp. Również prowadzenie własnych badań i doświadczeń bez tej wiedzy jest mało efektywne, a opracowanie ich wyników bez komputera pracochłonne i niekiedy dezaktualizujące je. Niestety, u nas taki system jeszcze nie istnieje.

Organizacja imprez sportowych

Trudno dziś wyobrazić sobie zorganizowanie zawodów sportowych dużej rangi bez wykorzystania komputera. Jego zastosowanie nie tylko ułatwia, ale i przyśpiesza ich organizację i prowadzenie. Przyjmowanie zgłoszeń zawodników, tworzenie list startowych, protokołów konkurencji, ustalanie kolejności startów, rejestracja wyników, ustalanie miejsc, wyświetlanie wyników na tablicy, drukowanie końcowych komunikatów, to jedynie niewielki wycinek pracy związanej bezpośrednio z zawodami. Akredytacja dziennikarzy, dysponowanie bazą hotelową, środkami transportu itp., to inne formy działania, w których komputer odgrywa nieocenioną rolę

Zarządzanie działalnością organizacji sportowych

Zastosowanie informatyki w zarządzaniu związkami sportowymi, klubami, to klasyczne już dziś wykorzystanie komputera.

Przedstawione możliwości zastosowania informatyki w sporcie są rozległe i wymagają wielu przedsięwzięć organizacyjnych. Istniejąca w stosunku do komputerów bariera psychologiczna jest trudna do pokonania. Opory te wynikają głównie z niewiary w możliwość przyswojenia sobie wiedzy i umiejętności posługiwania się komputerem. Bez pokonania tej bariery trudno jest przekonać kogokolwiek o potrzebie stosowania komputerów w procesie treningowym, czy zarządzaniu. Nie mniej istotne są problemy natury organizacyjnej. Wydanie ogólnoresortowych wytycznych w zakresie tworzenia stosownych komórek informatycznych, ujednolicenie sprzętu, szkolenie kadr, to tylko niektóre aspekty, bez których opracowania jednolity system informatyczny jest niemożliwy do wdrożenia

Jednak komputer nie jest panaceum na nasze bolączki. Podmiotem działania w sporcie jest, i mam nadzieję, że pozostanie człowiek. Jesteśmy w stanie dokonać komputerowej symulacji nie tylko wybranych fragmentów treningu, ale i czynności psychicznych. Nie oddadzą one jednak atmosfery zawodów, swoistości treningu, a więc tych nieuchwytnych, ale jakże ważnych dla końcowego wyniku czynników. Przeżycia emocjonalne towarzyszą nam na każdym kroku, odzwierciedlają nasz stosunek do otaczającego nas świata i nie pozostają bez wpływu na to, co robimy, i jak robimy. Te procesy psychiczne trudno przedstawić w postaci algorytmu, jeszcze trudniej je przewidzieć i zaprogramować. I dopóki ten stan rzeczy będzie istniał, dopóty walka sportowa będzie miała swój sens i urok.

Ryszard RADZIEJEWSKI

Programowanie strukturalne

Janusz MORBITZER

Koncepcja programowania strukturalnego po raz pierwszy pojawiła się w 1968 r. w artykule holenderskiego informatyka E. W. Dijkstry, pt. „Szkodliwe GOTO" *(2)*. Rozwinięciem idei Dijkstry były późniejsze prace N. Wirtha *(5)*, *(6)*, *(7)*.

Znajomość zasad programowania strukturalnego jest niezbędna każdemu, kto pragnie samodzielnie rozwiązywać problemy i pisać dobre programy.

Programowanie strukturalne często utożsamiane jest z jednym z jego elementów — kodowaniem strukturalnym. Jest to podstawowy błąd, programowanie strukturalne oznacza bowiem pewną filozofię, podstawę przy rozwiązywaniu problemów i zapisywaniu tych rozwiązań w postaci programów. Podstawowymi założeniami są:

a) programy mają być w pełni zrozumiałe przede wszystkim dla człowieka, a następnie dopiero dla maszyny,
b) struktura programu powinna być odzwierciedleniem struktury problemu, który jest przez ten program rozwiązywany,
c) program ma się dać czytać „z góry na dół" (unikanie instrukcji skoku).

Stosowanie zasad programowania strukturalnego przyczynia się w znacznym stopniu do tworzenia przejrzystych i czytelnych programów o mniejszej liczbie i łatwiej wykrywalnych w porównaniu z innymi metodami błędów logicznych. Zasady te mogą być stosowane w odniesieniu do każdego języka wyższego poziomu, także do języka „niestrukturalnego", jakim jest np. BASIC.

Programowanie strukturalne obejmuje trzy elementy:

— projektowanie zstępujące (ang. top-down),
— programowanie modularne,
— kodowanie strukturalne.

Projektowanie zstępujące — zwane również projektowaniem metodą kolejnych uściśleń (ang. step-wise refinement) — polega na tworzeniu programu w najogólniejszym zarysie, a następnie stopniowym uściśleniu struktury w celu wypełnienia jej szczegółami. Pracę rozpoczyna się od spisania celu i założeń projektu.
Jednym zdaniem określa się, co program ma robić. W każdym następnym kroku zadanie rozkładane jest na szereg podzadań, aż do momentu, gdy prostota podzadania pozwala na łatwe napisanie go w postaci fragmentu programu — odrębnego modułu. Każdy moduł powinien zawierać opis wykonywanej funkcji i użytych struktur danych oraz opis danych służących do testowania modułu. Zaletą takiego podejścia jest wczesne tworzenie dokumentacji programu. Bardzo ważne jest również stworzenie projektu przed rozpoczęciem kodowania. Napisanie fragmentu programu stwarza bowiem barierę psychologiczną uniemożliwiającą ulepszenie projektu; łatwiej jest zmienić projekt niż napisany już fragment programu. Wykonując projekt systemu metodą zstępującą mamy pewność, że oprogramowanie będzie odpowiadać celom oraz że popełnione błędy logiczne zostaną wcześnie wykryte. Równoległe sporządzenie dokumentacji zmusza do starannego przemyślenia struktury programu i danych.

Programowanie modularne polega na podzieleniu programu na niezależne części zwane modułami lub segmentami. Podział na moduły staje się widoczny już w fazie projektowania zstępującego — poszczególne podzadania wyznaczają kolejne moduły. Podział dużego programu na mniejsze jednostki służy do ograniczenia złożoności programu, ułatwia jego analizę i zrozumienie.
Poszczególne moduły to *„cegiełki"*, z których budowany jest program. Najważniejszymi cechami modułów są:

— wielkość modułu,
— niezależność.

Nie istnieją ścisłe wytyczne, określające wielkość modułu, panuje jednak zgodne przekonanie, że moduły mniejsze są lepsze od dużych. Wielkość modułu powinna być taka, aby łatwo można było jego treść pojąć i zapamiętać. **Dla mikrokomputerów rozsądne jest założenie, że moduł powinien mieścić się w całości na ekranie.** Ułatwia to znacznie analizę i wyszukiwanie błędów.

Przez niezależność modułu należy rozumieć brak ubocznego wpływu na inne części programu, tj. jego niezależność od pochodzenia danych, od przeznaczenia wyników i od wcześniejszego wykorzystania modułu.

Nieprzestrzeganie postulatu modułów prowadzi do tzw. efektu falowego (ang. ripple effect): zmiana w jednym module pociąga za sobą konieczność dokonania zmian w innych częściach programu. Prostym sposobem ograniczenia efektu falowego jest unikanie zmiennych globalnych i używanie małych modułów. W języku BASIC wszystkie zmienne są zmiennymi globalnymi. W celu uniknięcia kolizji warto założyć katalog używanych w programie nazw, a w razie potrzeby stosować nazwy dłuższe, mnemoniczne (tj. kojarzące się z pełnioną funkcją, np. zmienna o nazwie LICZ używana jest jako licznik itp.).

Dobry moduł powinien mieć cechy podobne do standardowych funkcji matematycznych, takich jak EXP, SQR czy SIN. Są to następujące cechy:

— algorytm rozwiązania zadania,
— zakres dopuszczalnych wartości wejściowych (dziedzina),
— zakres dopuszczalnych wartości wyjściowych (przeciwdziedzina),
— efekty uboczne (komunikaty o błędzie, gdy argument nie należy do dziedziny).

Dobry moduł powinien sprawdzać, czy przekazane mu dane wejściowe należą do dziedziny. W razie rozpoznania błędu moduł nie powinien przerywać wykonywania programu, lecz przekazać odpowiednią informację do modułu bezpośrednio go wywołującego. Każdy moduł powinien mieć jeden punkt wejścia i jeden punkt wyjścia. Należy pamiętać, że czytelność poszczególnych modułów decyduje o czytelności i przejrzystości całości programu.

Kodowanie strukturalne oznacza technikę pisania programów opartą na podstawowych strukturach programowania. Zgodnie z postulatem programowania strukturalnego o pisaniu programów dla ludzi, a nie dla maszyn, wśród podstawowych struktur nie ma instrukcji skoku. Człowiek przetwarza bowiem informacje sekwencyjnie. Im większa liczba odstępstw od sekwencyjności, tym trudniej jest śledzić kompozycję programu. Szczególnym utrudnieniem ze względu na różnorodność możliwych dróg, są skoki warunkowe.

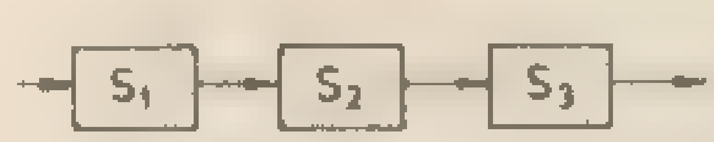
Matematyczną podstawą kodowania strukturalnego jest następujące twierdzenie o strukturze:
każdy program mający jedno wejście i jedno wyjście, nie zawierający nieskończonych pętli i nieosiągalnych fragmentów, można zapisać przy użyciu tylko trzech podstawowych struktur:

— sekwencji,
— selekcji (IF THEN ELSE),
— iteracji (WHILE DO).

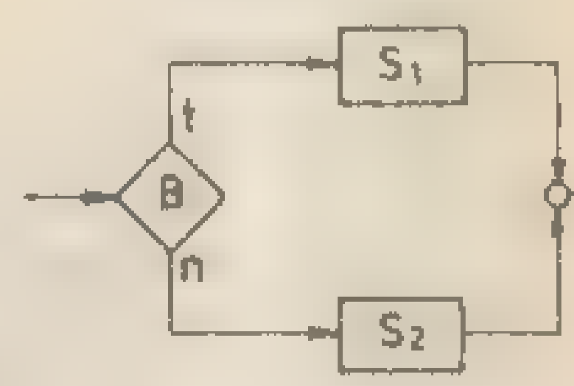
Zwykle w celu ułatwienia programowania dopuszcza się dwie dodatkowe struktury:

— iteracji REPEAT UNTIL
— operacji selektywnego wyboru (CASE).

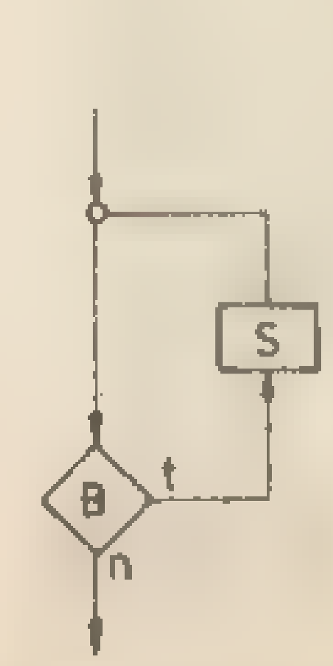
Wprowadzone tu instrukcje strukturalne przedstawione zostały na poniższym rysunku.



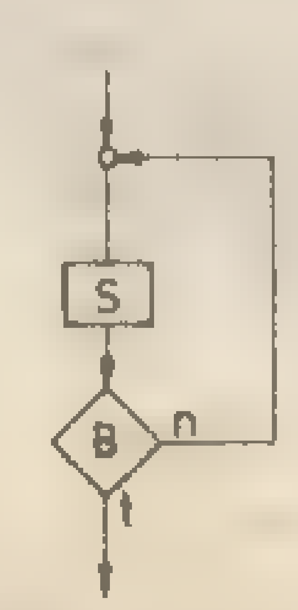
SEKWENCJA



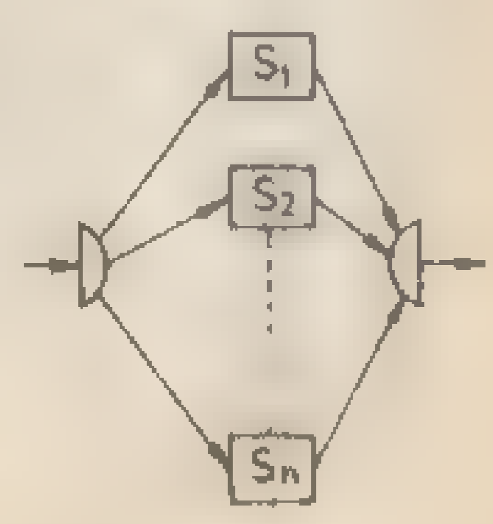
IF B THEN S_1 ELSE S_2
(jeżeli B to S_1 w przeciwnym razie S_2)



WHILE B DO S
(dopók B wykonuj S)



REPEAT S UNTIL B
(wykonuj S aż zajdzie B)



CASE

Poprawność programu zostaje zachowana, gdy dowolny prostokąt symbolizujący operację zostanie zastąpiony inną strukturą podstawową. Pętle WHILE DO i REPEAT UNTIL są podobne do siebie. Różnica polega na tym, że w pętli WHILE DO warunek badany jest przed wykonaniem instrukcji, a w pętli REPEAT UNTIL po jej wykonaniu oraz że inny jest warunek wyjścia z pętli — pętla WHILE DO kończy się, gdy warunek staje się fałszywy, natomiast pętla REPEAT UNTIL, kiedy warunek staje się prawdziwy. Wynika z tego, że pętla REPEAT UNTIL wykonana będzie — niezależnie od warunku — przynajmniej jeden raz, podczas gdy pętla WHILE DO może nie zostać wykonana ani razu.

Instrukcja selektywnego wyboru CASE służy do wykonania jednej z operacji S_1, S_2... S_n w zależności od wartości (całkowitej) wyrażenia zwanego selektorem.

Wszystkie przedstawione konstrukcje programowania strukturalnego występują jedynie w języku PASCAL. W innych językach można — jeśli dana konstrukcja nie występuje — symulować ją. W artykule przedstawione zostaną sposoby zapisywania konstrukcji strukturalnych w języku BASIC dla mikrokomputera ZX Spectrum. Niezbędne tu będzie użycie instrukcji skoku — ale wyłącznie skoku w przód. Nie narusza to postulatu czytelności programu z „góry na dół".

Konstrukcję strukturalną IF THEN ELSE można symulować w BASIC-u za pomocą instrukcji IF THEN:

```
10 REM [illegible] REPEAT UNTIL
20 FOR i=1 TO 2 STEP 0
30 ...tu instrukcja (ciąg) S...
40 IF warunek THEN GO TO 60
50 NEXT i
60 REM [illegible] REPEAT UNTIL
```

W wypadku, gdy po THEN i ELSE wykonywane są pojedyncze instrukcje (lub sekwencje pisane po dwukropku), konstrukcję IF THEN ELSE można zapisać przy użyciu dwóch instrukcji IF THEN:

```
10 IF warunek THEN S1
20 IF NOT warunek THEN S2
```

Konstrukcje strukturalne WHILE DO i REPEAT UNTIL można symulować przy użyciu pętli FOR z bardzo dużą wartością końcową lub — lepiej — z krokiem Ø (zapewnia to nieskończone wykonywanie pętli).

Test opuszczenia pętli dla konstrukcji WHILE DO umieszczony jest na początku, a warunek jest zanegowany:

```
10 [illegible]
20 FOR i=1 TO 2 STEP 0
30 IF NOT warunek THEN GO TO 60
40 ...tu instrukcja (ciąg) S...
50 NEXT i
60 [illegible]
```

W konstrukcji REPEAT UNTIL test znajduje się na końcu, a warunek jest prosty (nie zanegowany):

```
10 [illegible]
20 IF warunek THEN GO TO 60
30 •wykonywane, gdy fałsz (S2)•
40 ..........
50 GO TO 80
60 •wykonywane, gdy prawda (S1)•
70 ..........
80 [illegible]
```

Do zapisania ostatniej z konstrukcji strukturalnych (CASE) można wykorzystać instrukcję GOTO (w niektórych wersjach języka BASIC znajduje się bardzo wygodna instrukcja ON GOTO). Istotne jest sprawdzenie, czy wartość selektora S_1 nie wykracza poza dopuszczalny zakres.

```
1 [illegible]
5 IF s1<1 OR s1>4 THEN LET s1=5
7 GO TO 10*s1
10 ...tu instrukcja (ciąg) S1...
15 GO TO 60
20 ...tu instrukcja (ciąg) S2...
25 GO TO 60
30 ...tu instrukcja (ciąg) S3
35 GO TO 60
40 ...tu instrukcja (ciąg) S4...
45 GO TO 60
50 ...selektor spoza zakresu...
55 ...(sytuacja awaryjna)......
60 [illegible]
```



— Panno Ziutko, proszę zakodować moje uczucie do pani.

Konsekwentne stosowanie zasad programowania strukturalnego prowadzi do tworzenia czytelnych, dobrze przemyślanych programów. Daje możliwość tzw. strukturalnej przechadzki (ang. structured walkthrough), tj. czytanie kodu z góry na dół, bez konieczności skakania po tekście. Program podzielony jest na mniejsze jednostki (moduły), podobnie jak książka podzielona jest na rozdziały. Stosowanie wcięć w tekście programu (akapitowanie) w celu uwidocznienia struktur sterowania dodatkowo zwiększa czytelność. Dopuszczalne jest (oszczędne!) używanie instrukcji skoku — tylko do przodu. Skok w tył oznacza pętlę — należy wówczas użyć konstrukcji pętli. Należy pamiętać, że najwięcej skoków pojawia się w programie przy nanoszeniu poprawek i uzupełnień. Zapobiega temu konieczność sporządzenia szczegółowego projektu (uwzględniającego wszystkie możliwe sytuacje) przed przystąpieniem do kodowania.

Jak wynika z przedstawionych rozważań, programowanie strukturalne składa się z trzech elementów, z których dwa (projektowanie zstępujące i programowanie modularne) są mało zależne od języka programowania. Trzeci element — kodowanie strukturalne — może być, jak pokazano w artykule, symulowane w innych językach.

Programowanie strukturalne koncentruje się na kompozycji programu. Warto znać i stosować przedstawione zasady. Unikniemy wówczas mozolnej analizy własnego programu, kiedy po miesiącu od napisania go, tracimy w nim orientację (któż z nas robi porządny projekt i dokumentację?!), a nasze programy staną się lepsze, bardziej czytelne i zrozumiałe — dla nas i dla innych.

Janusz MORBITZER

Literatura

(1) *Dahl O.J., Dijkstra E.W., Hoare C.A.R. „Structured programming". Londyn, Academic Press 1972.*

(2) *Dijkstra E.W. „GOTO Statement considered harmfurl". CACM, 11/1968.*

(3) *Myers G.J. „Projektowanie niezawodnego oprogramowania". WNT 1980*

(4) *Tassel D. „Praktyka programowania". WNT 1982*

(5) *Wirth N. „Program derelopment by step — wise refinement". CACM 14/1971*

(6) *Wirth N. „Wstęp do programowania systematycznego". WNT 1979*

(7) *Wirth N. „Algorytmy i struktury danych". WNT 1980*

W szponach Atari (12)

Jak wykorzystać przerwania?

Tomasz MROWIEC, Ludwik PIELA

Elastyczność działania współczesnych komputerów, również domowych, wynika z ich zdolności selektywnego reagowania na specjalne zdarzenia układowe i programowe, zwane przerwaniami. Spośród kilku rodzajów przerwań występujących w Atari, dwa z nich są szczególnie interesujące: VBI — czyli przerwanie powrotu pionowego oraz DLI — przerwanie programu wyświetlania. Pierwsze z nich służy głównie do zsynchronizowania działania układów komputera z sygnałem telewizyjnym wysyłanym do monitora (telewizora) i pojawia się regularnie 50 razy na sekundę. Występuje ono niezależnie od woli użytkownika. Drugie, DLI, jest przerwaniem pochodzącym od specjalizowanego procesora graficznego — ANTIC'u. O jego pojawieniu się i momencie wystąpienia decyduje użytkownik.

Przerwanie powrotu pionowego

W celu uzyskania czystego i wyraźnego obrazu na ekranie monitora lub telewizora niezbędne jest stałe jego odnawianie. Dlatego w pamięci RAM znajduje się obszar, zwany pamięcią ekranu, w którym przechowywane są informacje aktualnie wyświetlane na ekranie. Samo wyświetlanie realizowane jest przez procesor ANTIC, zgodnie z dostarczonym progamem wyświetlania, niezależnie od mikroprocesora 6502.

Dla jednoznaczności opisu przypomnijmy definicje pewnych pojęć. Pojedyncze przejście strumienia elektronów przez ekran nazywamy *„poziomą linią ekranu"*. Okres, w którym strumień wraca od prawego do lewego brzegu, nazywamy *„powrotem poziomym"*. Okres, w którym strumień elektronów wraca na górę ekranu (bez kreślenia), nazywamy *„powrotem pionowym"*. Rozkazy programu wyświetlania określają sposób wyświetlania jednej linii. Nie jest to linia ekranu, ale tak zwana linia obrazu, która w zależności od używanego trybu składa się z kilku linii ekranu (w trybie 8 BASIC'u z jednej).

Proces kreślenia obrazu (wykonywany 50 razy na sekundę) zajmuje około 20 000 mikrosekund. W tym czasie następuje wykreślenie 312 linii, od góry do dołu ekranu, tworzących półobraz oraz przemieszczenie strumienia elektronów (po jego uprzednim wygaszeniu) na górę ekranu. Bezpośrednio po wykreśleniu ostatniej linii ekranu i wygaszeniu strumienia elektronów powstaje przerwanie.

Po rozpoznaniu przyczyny przerwania system operacyjny umieszcza na stosie zawartość rejestrów A, X i Y oraz przechodzi do odpowiedniej procedury według adresu umieszczonego w komórce VVBLKI (546,547). Procedura obsługi przerwania, rozpoczynająca się od adresu $E45F, zwiększa stan zegara (komórka o adresie 20), zmniejsza zawartości liczników (zegarów systemowych), dokonuje zmiany kolorów, kopiuje zawartości rejestrów „cieni" do rejestrów układowych i aktualizuje wartości na wyjściach jednostek sterujących. Jej działanie kończy się przejściem do wektora VVBLKD, umieszczonego w komórkach o adresach 548,549. Wskazuje on adres procedury zakończenia przerwania.

Adresy komórek VVBLKI oraz VVBLKD umożliwiają nam śledzenie procedury obsługi przerwania powrotu pionowego oraz wykorzystanie przerwania do własnych celów. W tych dwóch miejscach możliwe jest dołączenie własnej procedury. Będzie ona wtedy realizowana albo przed (natychmiastowa) albo po (opóźniona) wykonaniu systemowej procedury obsługi przerwania. W pewnych przypadkach nie jest obojętne miejsce podłączenia własnej procedury obsługi przerwania. Pierwszy z nich ma miejsce, kiedy nasza procedura odczytuje lub zmienia zawartości rejestrów, które mają swoje „cienie" w pamięci. Może być konieczna zmiana ich zawartości po wykonaniu tego przez procedurę **GO**. O zawartości decyduje ostatni zapis.

Drugi przypadek występuje, gdy nasza procedura zużywa zbyt dużo czasu procesora. Realizacja procedury obsługi przerwania VBI może być opóźniona poza koniec okresu powrotu pionowego. Lecz wtedy może się zdarzyć, że pewne rejestry związane z grafiką będą zmieniane podczas kreślenia obrazu na ekranie. Można uyskać nieoczekiwane wyniki. W takim przypadku musi być realizowana po wykonaniu systemowej procedury obsługi, jako tak zwana opóźniona. Limit czasowy dla natychmiastowej procedury obsługi VBI wynosi około 2000 cykli maszynowych, dla opóźnionej — około 20 000 cykli. Jednak wiele z nich realizowanych jest w czasie, gdy strumień elektronów kreśli obraz na ekranie. Dlatego procedury obsługi VBI, związane ze zmianą parametrów obrazu nie powinny być zbyt długie. **Ponadto musimy pamiętać, że czas obsługi przerwań skraca czas procesora dostępny dla realizacji podstawowego programu.**

Po zadecydowaniu, czy procedura VBI ma być natychmiastowa, czy opóźniona, musimy umieścić ją w pamięci (dobrym miejscem jest strona 6 RAM), połączyć z odpowiednią procedurą systemową i zmodyfikować zawartość pewnych komórek pamięci w celu jej wywołania. Natychmiastową procedurę VBI kończymy rozkazem JMP $E45F, a opóźnioną JMP $E462. Może się zdarzyć, że chcemy całkowicie ominąć systemową procedurę obsługi VBI (na przykład w celu połączenia czasów przetwarzania), wtedy procedurę natychmiastową kończymy rozkazem JMP $E462.

Podczas dołączania własnej procedury obsługi przerwania konieczna jest zmiana zawartości wektora przerwania. Jest to wielkość dwubajtowa, zatem potrzebne są dwa rozkazy przechowania. Mogłoby się zdarzyć wystąpienie przerwania po zmianie pierwszego bajtu, przed aktualizacją drugiego. Może to spowodować upadek systemu. Dlatego należy zabezpieczyć się przed taką sytuacją poprzez wykorzystanie procedury SO SETVBV od adresu $E45C. W tym celu wprowadzamy do rejestru Y młodszy bajt adresu, do rejestru X — starszy bajt, a do akumulatora 6 dla natychmiastowej procedury VBI lub 7 dla opóźnionej. Następnie wykonujemy JSR SETVBV. Nowa procedura VBI rozpocznie działanie w ciągu 0,02 s.

Własna procedura obsługi przerwania VBI umożliwia wykonanie różnorodnych operacji:

1. W czasie powrotu pionowego strumienia elektronów mogą być wykonywane różnorodne manipulacje ekranem. Mamy pewność, że nie wystąpią one w trakcie kreślenia obrazu, lecz dotyczyć będą całego ekranu. W szczególności możliwa jest rytmiczna animacja, w której zmiany muszą występować w tempie niezależnym od innego przetwarzania.

2. Przerwania powrotu pionowego umożliwiają efektywne sterowanie częstotliwością, głośnością i zniekształceniami dźwięku. Dzięki temu możemy dodać do dźwięku fazy nabrzmiewania i wybrzmiewania.

3. Wykorzystanie VBI umożliwia stałe kontrolowanie wejść użytkownika (50 razy na sekundę) bez zakłócania realizacji programu. Wejścia te wymagają zwykle mało przetwarzania, lecz stałej uwagi. Jest to idealne rozwiązanie problemu zachowania ciągłości obliczeń bez ignorowania użytkownika

4. Wykorzystanie VBI umożliwia uzyskanie wielozadaniowości w najczystszej postaci. Program pierwszoplanowy może działać pod VBI, podczas gdy program drugoplanowy działa w głównej linii kodu.

Program nr 1 jest przykładem praktycznego wykorzystania przerwań. Umożliwia on uzyskanie dziesięciu kolorów w drugim trybie graficznym (zamiast standardowych czterech). Uzyskane kolory, podobnie jak wszystko na ekranie, są w rzeczywistości iluzją. Następuje mieszanie się kolorów ponieważ obrazy zmieniane są tak szybko, że nasze oczy nie mogą zauważyć zmian. Dlatego widzimy tylko jeden kolor, który jest mieszaniną kolorów we wszystkich obrazach. Możemy mieszać więcej niż dwa kolory jednocześnie, lecz w miarę wzrostu liczby obrazów, wzrasta także ilość migotań ekranu. Praktyczną granicą jest mieszanie czterech kolorów.

Efekt ten uzyskujemy poprzez użycie kilku obszarów RAM ekranu i zmianę bajtów LMS w programie wyświetlania w czasie obsługi przerwania powrotu pionowego.

Szybka zmiana bajtów adresu rozkazu LMS umożliwia przełączenie z jednego obrazu na drugi i z powrotem. Nie można tego zrobić w języku BASIC — jest zbyt wolny. Bajty LMS zmieniane są przez krótką procedurę w języku maszynowym, która działa 50 razy na sekundę, w czasie przejścia strumienia elektronów na górę ekranu.

```
FK 1 REM ****************
FJ 2 REM *              *
YH 3 REM * Program nr 1 *
FL 4 REM *              *
FL 5 REM ****************
NL 6 REM
KI 10 ? CHR$(125):GRAPHICS 2+16:B
   REAK=1000
BI 15 REM
IE 20 FOR I=0 TO 39:READ A:POKE 1
   536+I,A:NEXT I
LI 30 DATA 173,51,6,73,1,141,51,6
   ,240,15,173,40,6,141,49,6,173,
   41,6,141,50,6,76,98,228
YE 40 DATA 173,42,6,141,49,6,173,
   43,6,141,50,6,76,98,228
EL 45 REM
SX 50 DL=PEEK(560)+256*PEEK(561)
BM 55 REM
ZC 60 BYTE=DL+4:GOSUB BREAK
NM 70 POKE 1550,LOW:POKE 1565,LOW
IY 80 POKE 1551,HIGH:POKE 1566,HI
   GH
ZR 90 BYTE=DL+5:GOSUB BREAK
HI 100 POKE 1556,LOW:POKE 1571,LO
   W
GS 110 POKE 1557,HIGH:POKE 1572,H
   IGH
NZ 120 BYTE=DL+20:GOSUB BREAK
EE 130 POKE 1577,HIGH:POKE 1576,L
   OW
ZY 140 BYTE=DL-250:GOSUB BREAK
HM 150 POKE 1579,HIGH:POKE 1578,L
   OW
RL 155 REM
GZ 160 FOR I=0 TO 10:READ A:POKE
   1600+I,A:NEXT I
GK 170 DATA 104,162,6,160,0,169,7
   ,32,92,228,96
ZA 180 X=USR(1600)
GR 220 REM
OC 230 BYTE=DL+20:GOSUB BREAK
TQ 235 POKE 88,LOW:POKE 89,HIGH
IQ 240 POSITION 0,4
Y  250 ? #6;"   INformaTYka       "
PM 260 ? #6;"   KOMpuTEry         "
LP 270 ? #6;"   sysTEmy           "
ZT 280 BYTE=DL-250:GOSUB BREAK
TN 290 POKE 88,LOW:POKE 89,HIGH
PD 305 REM
DF 312 SETCOLOR 0,12,6
VL 313 SETCOLOR 1,4,6
ID 314 SETCOLOR 2,15,8
ZF 315 SETCOLOR 3,8,6
IN 320 POSITION 0,4
JT 330 ? #6;"   INforMATYka       "
ZF 340 ? #6;"   KOMpuTEry         "
PY 350 ? #6;"   SYSteMY           "
KQ 359 REM X=USR(1600)
PC 360 GOTO 360
SN 999 REM
EB 1000 HIGH=INT(BYTE/256)
CK 1010 LOW=BYTE-HIGH*256
AF 1020 RETURN
```

Program nr 2 umożliwia zainstalowanie i wykorzystanie procedury, która wygrywa melodię bez naruszania ciągłości działania programu. Generatory dźwięku wysterowywane są w czasie obsługi przerwania VBI na podstawie informacji zawartych w zmiennych tekstowych. Każda ze zmiennych zawiera dane, sterujące jednym generatorem, w postaci par znaków. Kod pierwszego znaku określa wysokość dźwięku, a drugiego — czas trwania w jednostkach równych 0,02 s. Na przykład para: CHR$(121) i CHR$(25) zdefiniuje nutę C o czasie trwania 0,5 s. Każdy z ciągów par musi być zakończony dwoma znakami o kodzie 0(CHR$(0)).

W celu uzyskania dźwięku musimy użyć następującej instrukcji.

x = USR (1536, ADR (M1$), ADR (M2$), ADR (M3$), ADR (M4$))

gdzie 1536 jest adresem początku procedury (umieszczonej na stronie 6), a M1$, M2$, M3$ i M4$, są nazwami zmiennych tekstowych zawierających dane sterujące poszczególnymi generatorami. W instrukcji wywołania używamy tylu nazw zmiennych, ilu generatorów dźwięku chcemy używać. Na przykład, dla dwóch generatorów instrukcja wywołania ma następującą postać:

x = USR (1536, ADR(M1$), ADR (M2$))

Sposób użycia opisywanej procedury dźwiękowej stanie się oczywisty po przeanalizowaniu tekstu programu.

W celu uzyskania pauzy należy podać wysokość tonu i (nie może być 0) i żądany czas trwania

Przerwania programu wyświetlania

Przerwania programu wyświetlania są jedną z bardziej przydatnych własności komputera Atari. Wprawdzie same nie dostarczają dodatkowych możliwości, lecz używane łącznie z innymi własnościami systemu, jak: grafika gracz—pocisk, wymiana zestawów znaków lub zmiana zawartości rejestrów kolorów, umożliwiają pełne ich wykorzystanie.

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Program nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
   10 GRAPHICS 5
TU 15 GOTO 70
   20 C=INT(RND(0)*16)
FT 30 COLOR C
OL 40 PLOT INT(RND(0)*80),INT(RND
   (0)*192)
   50 GOTO 20
NV 70 GOSUB 32000
RY 75 REM SPOSOB WYWOLANIA
   77 REM X=USR(1536,(ADR M1$),AD
   R M2$),ADR M3$),ADR M4$))
KN 80 DIM M1$(255),M2$(255):I=0
JI 85 RESTORE 110
LL 90 READ D:I=I+1
   95 IF D=-1 THEN 100
   97 M1$(I,I)=CHR$(D)
   98 GOTO 90
   100 I=0
   102 READ D:I=I+1
JT 104 IF D=-1 THEN 108
GH 105 M2$(I,I)=CHR$(D)
MW 106 GOTO 102
   108 X=USR(1536,ADR(M1$),ADR(M2
   $))
KF 109 GOTO 20:REM GLOS PIERWSZY
JG 110 DATA 96,25,108,25,121,25,1
   28,25,144,25,128,25,121,25,128
   ,25,144,25,128,25
SL 111 DATA 121,25,128,25,144,25,
   128,25,121,25,108,25,96,25,108
   ,25
DN 112 DATA 121,25,128,25,144,25,
   128,25,121,25,128,25,144,25,12
   8,25
SR 113 DATA 121,25,128,25,144,25,
   128,25,121,25,108,25,96,25,108
   ,25
DT 114 DATA 121,25,128,25,144,25,
   128,25,121,25,128,25,144,25,12
   8,25
ID 115 DATA 121,25,128,25,144,25,
   128,25,121,25,108,25,96,25,91,
   25
NL 116 DATA 96,25,108,25,121,25,1
   28,25,144,25,153,25,144,50,193
   ,150,96,25,91,25
IU 117 DATA 96,50,108,50,121,50,1
   28,50,121,50,144,25,128,25,121
   ,25,108,25,96,25,91,25
KU 118 DATA 81,50,91,50,96,50,108
   ,50,96,50,121,25,108,25,96,25,
   85,25,81,25,72,25
LS 119 DATA 64,50,72,50,81,50,85,
   50,81,50,96,50,60,50,72,50,72,
   25,81,25
EA 120 DATA 85,25,96,25,64,50,128
   ,50,96,50,128,50,133,50,1,300,
   0,0,-1
MG 129 REM GLOS DRUGI
NA 130 DATA 60,25,53,25,47,50,45,
   50,47,50,53,50,47,50,72,100,76
   ,50,72,50,45,50,47,50,53,50
MJ 131 DATA 47,50,72,100,76,50,72
   ,50,45,50,47,50,53,50,35,50,40
   ,25,45,25,47,50,53,50
KN 132 DATA 60,50,64,25,60,25,53,
   25,64,25,60,25,72,25
FX 133 DATA 64,25,72,25,76,25,72,
   25,64,25,60,25,53,25,64,25
ZD 134 DATA 60,25,53,25,47,25,45,
   25,47,25,53,25,60,25,53,25
YH 135 DATA 47,25,53,25,60,25,64,
   25,72,25,64,25,60,25,53,25
YJ 136 DATA 47,25,45,25,40,25,35,
   25,40,25,45,25,47,25,45,25
WN 137 DATA 40,25,45,25,47,25,53,
   25,60,25,53,25,47,25,42,25
IH 138 DATA 40,25,35,25,31,25,29,
   25,31,25,35,25,40,25,35,25
PW 139 DATA 31,25,35,25,40,25,42,
   25,47,25,50,25,47,25,42,25
HI 140 DATA 64,50,40,50,40,25,42,
   25,47,25,50,25
BB 141 DATA 47,150,1,300,0,0,-1
```

Wprawdzie obraz kreślony jest na ekranie 50 razy na sekundę, niebywale szybko jak na nasze możliwości, jednak komputer ma mnóstwo czasu na zmianę parametrów wyświetlania. Istotne jest, aby dokonywać zmiany w czasie każdego kreślenia obrazu, to znaczy 50 razy na sekundę oraz zmieniać interesujący nas parametr dokładnie w tym samym momencie kreślenia obrazu. Oznacza to, że cykl modyfikowania parametrów musi być zsynchronizowany z cyklem kreślenia. Doskonałym sposobem rozwiązania tego problemu może być pojawienie się sygnału przerwania tuż przed momentem zmiany parametrów obrazu. Mikroprocesor rozpoznaje przerwanie, zmienia odpowiednie parametry i wraca do swojej normalnej pracy. Moment pojawienia się przerwania musi być precyzyjnie odmierzany, aby występowało ono dokładnie w tej samej chwili w czasie kreślenia obrazu. Takie specjalnie taktowane przerwanie dostarczane jest przez ANTIC i nazywa się przerwaniem programu wyświetlania (DLI).

Jak wiadomo, ANTIC posiada własny program wyświetlania, którego rozkazy kolejno pobiera i realizuje. Pod koniec wyświetlania linii obrazu pobiera następny rozkaz wyświetlania. Jeśli stwierdzi, że jest w nim ustawiony bit przerwania (D7), to po rozpoczęciu wyświetlania ostatniej linii ekranu aktualnej linii obrazu sprawdza, czy w rejestrze NMIEN (o adresie 54286) dozwolone jest przerwanie programu wyświetlania. Jeśli bit zezwolenia (D7) jest wyzerowany, ANTIC ignoruje żądanie przerwania i kontynuuje swoje normalne zadania. Jeśli bit zezwolenia jest ustawiony, ANTIC wysterowuje linię NMI mikroprocesora 6502, który przechodzi do procedury obsługi przerwania według wektora NMI. Po określeniu przyczyny przerwania przechodzi do procedury obsługi DLI (na podstawie zawartości komórek \$200, \$201 tj. 512 i 513), która może zmienić zawartość jednego lub kilku rejestrów graficznych sterujących wyświetlaniem. Następnie 6502 wraca do realizacji programu głównego.

Jak widać z tego krótkiego opisu, przed wykonaniem procedury obsługi przerwania DLI należy wykonać następujące czynności:

1. *Wprowadzić procedurę obsługi przerwania do bezpiecznego obszaru pamięci (idealnym miejscem jest strona 6 RAM).*
2. *Zmodyfikować pewne bajty programu wyświetlania w celu wykonania przerwań.*
3. *Do komórek o adresach 512 i 513 wstawić adres pierwszego rozkazu procedury obsługi przerwania programu wyświetlania.*
4. *Zezwolić na pojawienie się przerwań za pomocą POKE 54286, 192.*

Z kolei procedura obsługi przerwania powinna wykonywać następujące czynności:

1. *Przechować zawartości rejestrów, które będą używane, poprzez umieszczenie ich na stosie 6502.*
2. *Wykonać odpowiednie działania.*
3. *Odtworzyć zawartości przechowywanych rejestrów poprzez zdjęcie ich ze stosu i umieszczenie w odpowiednich rejestrach.*
4. *Wykonać rozkaz RTI (powrót z przerwania).*

Powyższe czynności muszą być wykonywane podczas każdej realizacji procedury obsługi przerwania. W przeciwnym przypadku mogą być zniszczone zawartości rejestrów mikroprocesora (i prawdopodobnie nasz program także).

Podobnie, jak w każdej procedurze obsługi przerwania musimy uwzględnić zależności czasowe. ANTIC nie wysyła sygnału przerwania do 6502 bezpośrednio po napotkaniu odpowiedniego rozkazu; opóźnia to do chwili rozpoczęcia kreślenia ostatniej linii ekranu aktualnej linii obrazu, która ma być przerwana. Dalsze opóźnienia wynikają z czasów działania procedury rozpoznawania przyczyny przerwania oraz przejścia do naszej procedury. Rozpocznie ona zatem działanie, gdy strumień elektronów przebył już część drogi po ekranie. Jeśli procedura DLI zmienia zawartość rejestru koloru, to stary kolor będzie wyświetlony w lewej połowie ekranu, a nowy kolor pojawi się w prawej połowie ekranu w tej samej linii. Z powodu niezbyt precyzyjnego odmierzania czasu odpowiedzi na przerwanie, granica między nimi nie będzie ostra, lecz denerwująco drgająca.

W celu rozwiązania tego problemu musimy wykorzystać rejestr WSYNC (czekanie na synchronizację poziomą). Jeśli wstawimy rozkaz STA WSYNC tuż przed rozkazem, który zmienia zawartość rejestru koloru, to kolor ulegnie zmianie wtedy, gdy strumień elektronów jest wyłączony (powrót poziomy). Zmiana koloru wystąpi o jedną linię ekranu niżej, lecz będzie czysta i ostra.

Właściwym użyciem programu wyświetlania jest ustawienie bitu D7 w rozkazie programu wyświetlania poprzedzającym linię obrazu, w której chcemy dokonać zmiany.

Prosty przykład procedury obsługi przerwania programu wyświetlania prezentujemy poniżej:

PHA	*przechowanie zawartości akumulatora*
TXA	
PHA	*przechowanie zawartości rejestru X*
LDA #$50	*ciemny kolor dla znaków*
LDX #$58	*różowy kolor dla tła*
STA WSYNC	*czekanie*
STA COLPF1	*zmiana zawartości rejestrów kolorów*
STX COLPF2	
PLA	
TAX	*odtworzenie zawartości rejestrów X i A*
PLA	
RTI	*powrót z obsługi przerwania*

Powyższa procedura, zrealizowana w **programie nr 3**, zmienia kolor tła z niebieskiego na różowy. Zmienia także kolor znaków. Okazuje się, że po jej wykonaniu górna część ekranu pozostaje niebieska, pomimo tego, że procedura ustawia kolor różowy w odpowiednim rejestrze. Dzieje się tak dlatego, że procedura obsługi przerwania powrotu pionowego (VBI) wstawia do tego rejestru kod koloru niebieskiego. Pochodzi on z rejestru *„cienia"* tego rejestru koloru. Każdy z układowych rejestrów koloru ma swój „cień" w komórkach RAM o adresach 708 do 712. W większości przypadków wygodnie jest zmieniać kolory poprzez zmianę zawartości rejestrów *„cieni"*. Jeśli zmienimy zawartości rejestrów układowych, to w ciągu 1/50 s ustawione kolory zostaną zmienione przez system operacyjny. Jednak w procedurze obsługi DLI musimy zmieniać zawartości rejestrów układowych. Zauważmy również, że nie możemy zmieniać koloru pierwszej wyświetlanej linii obrazu.

```
FK  1 REM ****************
FJ  2 REM *              *
YO  3 REM * Program nr 3 *
FL  4 REM *              *
FO  5 REM ****************
NL  6 REM
ST 10 DL=PEEK(560)+256*PEEK(561)
FJ 20 POKE DL+15,130
LH 30 FOR I=0 TO 19
UI 40 READ A:POKE 1536+I,A:NEXT I
MA 50 DATA 72,138,72,169,80,162,8
   9
VV 60 DATA 141,10,212,141,23,208
FN 70 DATA 142,24,208,104,170,104
   ,64
TX 80 POKE 512,0:POKE 513,6
CX 90 POKE 54286,192
```

Wstawianie kodów kolorów bezpośrednio do rejestrów układowych powoduje pojawienie się innego problemu: kasujemy automatyczny tryb zwracania uwagi (attract mode), który jest własnością dostarczaną przez system operacyjny. Po dziewięciu minutach bez naciśnięcia jakiegokolwiek klawisza, kolory na ekranie zaczynają się zmieniać, przyjmując przypadkowe barwy przy obniżonej jaskrawości. Gwarantuje to, że komputer pozostawiony bez kontroli na kilka godzin nie wypali obrazu na ekranie telewizora. Musimy zatem rozważyć możliwość utrzymania tej własności. Można to zrobić poprzez wstawienie do procedury obsługi przerwania programu wyświetlania dwóch dodatkowych rozkazów:

stare	*nowe*
LDA NEWCOL	**LDA NEWCOL**
STA WSYNC	**EOR COLRSH**
STA COLPF2	**AND DRKMSK**
	STA WSYNC
	STA COLPF2

DRKMSK i COLRSH są komórkami pamięci na stronie zerowej (78 i 79) ustawianymi i aktualizowanymi przez system operacyjny w czasie obsługi przerwania powrotu pionowego. Gdy nie ma trybu zwracania uwagi, COLRSH przyjmuje wartość ØØ a DRKMSK — $FF. Po wymuszeniu tego trybu, co 4 sekundy do COLRSH wstawiana jest nowa wartość losowa, a DRKMSK przechowuje wartość $F6. Zatem COLRSH, zmienia kolor, a DRKMSK obcina najstarszy bit jaskrawości.

Wprowadzenie trybu zwracania uwagi do procedury obsługi DLI zaostrza istniejący już problem czasu działania procedury. Szczegółowe rozważania czasowe uczynią go bardziej oczywistym. W realizacji przerwania programu wyświetlania można wyróżnić trzy fazy. Faza 1 obejmuje okres od pojawienia się przerwania do rozkazu STA WSYNC. W czasie jej trwania strumień elektronów kreśli ostatnią linię ekranu przerywanej linii obrazu. Faza 2 obejmuje okres od rozkazu STA WSYNC do momentu pojawienia się strumienia elektronów na ekranie. Odpowiada to przerwie poziomej; wszystkie zmiany obrazu powinny być robione w tym czasie. Faza 3 obejmuje okres od pojawienia się strumienia na ekranie do zakończenia procedury obsługi DLI.

Kreślenie jednej linii ekranu trwa 114 cykli zegara procesora. Przerywanie dochodzi do mikroprocesora w cyklu numer 2. Zużywa on od 8 do 14 cykli na rozpoznanie przyczyny przerwania. Procedury: obsługi przerwania SO i skierowania go do procedury obsługi DLI zużywa 11 cykli maszynowych. W tym czasie od 1 do 3 cykli zostanie zabranych na odświeżenie pamięci DMA. Zatem procedura obsługi DLI nie jest osiągana przed upływem 28—36 cyklu zegara. Dla celów planowania musimy przyjąć najgorszy przypadek i programować tak, jakby procedura obsługi DLI osiągana była w 36 cyklu. Dodatkowo rozkaz STA WSYNC musi być osiągnięty w cyklu numer 100; redukuje to czas dostępny w fazie 1 o 14 cykli. Ponadto DMA ANTIC'u będzie zabierał pewne cykle maszynowe. Dziewięć cykli będzie straconych na odświeżanie pamięci DMA. Pozostawia to maksymalnie 55 cykli dostępnych dla fazy 1. To maksimum uzyskiwane jest tylko w przypadku pustej linii obrazu. Rozkazy trybu znakowego i graficznego powodują utratę jednego cyklu dla każdego bajtu wyświetlanych danych. Najgorszy przypadek występuje dla trybów BASIC'u 0, 7, 8, 14 i 15, które wymagają 40 bajtów na linię. Zatem procedura fazy 1 będzie miała na realizację od 15 do 55 cykli maszynowych.

Faza 2, krytyczna, rozszerza się ponad 27 cykli zegara. Jak w fazie 1, pewne z nich są tracone na rzecz DMA. Jeśli używana jest grafika P-M, tracimy dodatkowe 5 cykli. Rozkaz wyświetlania zużywa jeden cykl, jeśli używana jest opcja LMS, to tracimy dwa dodatkowe cykle. W końcu, jeden lub dwa cykle mogą być stracone na odświeżenie pamięci lub wyszukiwanie wyświetlanych danych. Zatem dla fazy 2 dostępnych jest od 17 do 26 cykli maszynowych.

Problemy czasowe obsługi przerwania programu wyświetlania stają się teraz oczywiste. Pobranie, zmiana i przechowanie pojedynczego koloru zużywa 4 cykle.

Przechowanie na stosie zawartości rejestrów A, X i Y, a następnie wprowadzenie, zmiana i przechowanie trzech kolorów w A, X i Y zużywają 47 cykli, to jest większość, jeśli nie wszystkie, fazy 1. Programista, który chce używać DLI dla rozbudowanych zmian graficznych, zużywa sporo czasu na kalkulacje czasowe. Na szczęście początkujący programista nie musi koncentrować się na obliczeniach czasowych. Jeśli zmienia się tylko jeden kolor lub wykonywane są proste operacje graficzne, zliczanie cykli i optymalizacja procedury nie są konieczne.

Nie ma żadnych prostych opcji dla programisty, który potrzebuje zmienić więcej niż trzy rejestry koloru w pojedynczym przerwaniu. Może być możliwa zmiana czwartego koloru na początku fazy 3, jeśli nie jest on wyświetlany na lewym brzegu ekranu. Podobnie, kolor, który nie występuje przy prawym brzegu ekranu, może być zmieniony w fazie 1. Innym sposobem podejścia jest rozdzielenie jednej procedury obsługi DLI o nadmiernej aktywności na dwie mniej ambitne, z których każda wykonuje połowę pracy oryginału. Drugie DLI można uzyskać przez wstawienie pojedyńczego rozkazu pustej linii ekranu (z ustawionym bitem DLI) do programu wyświetlania tuż poniżej przerywanej linii obrazu. Zajmuje to jednak pewien obszar ekranu.

Innym częściowym rozwiązaniem jest dokonanie zmiany kolorów w czasie przerwy pionowej. W tym celu w RAM muszą być przechowywane dwie tablice kolorów. Pierwsza z nich zawiera kody kolorów, które mają być wyświetlane przez procedury obsługi DLI, druga zawiera wartości umożliwiające zmianę kolorów. W czasie powrotu pionowego dostarczona przez nas procedura obsługi przerwania pobiera kod koloru z pierwszej tablicy, zmienia go i przechowuje w drugiej tablicy. Procedura obsługi DLI pobiera wtedy wartości bezpośrednio z drugiej tablicy, bez straty dodatkowego czasu na zmianę.

Często pożądane jest posiadanie kilku procedur obsługi przerwań programu wyświetlania występujących w kilku miejscach ekranu. Niestety, istnieje tylko jeden wektor DLI. Jeśli chcemy zastosować kilka procedur obsługi przerwania, to w samej procedurze musimy umieścić kierowanie do odpowiedniej obsługi. Można to zrobić na kilka sposobów. Jeśli procedura obsługi DLI wykonuje taki sam proces z różnymi wartościami, wtedy może być sterowana tablica. Po każdej realizacji procedury zwiększany jest licznik i używany jako indeks tablicy wartości. Poniższy przykład prezentuje ten sposób.

PHA	
TXA	*przechowanie zawartości rejestrów*
PHA	
INC LICZ	*zwiększenie licznika*
LDX LICZ	
LDA TABCKOL, X	*użycie strony 2 dla tablicy kolorów*
STA WSYNC	*oczekiwanie*
STA COLBAK	
CPX #$4F	*czy ostatnia linia?*
BNE KON	*nie, wyjście*
LDA #$00	*tak, wyzerowanie licznika*
STA LICZ	
KON PLA	
TAX	*odtworzenie zawartości rejestrów*
PLA	
RTI	

W celu wykorzystania powyższej zasady należy wprowadzić **program nr 4**, który umieszcza na ekranie 80 kolorów jednocześnie.

Innym sposobem zastosowania wielokrotnych DLI jest użycie licznika przerwań jako testu dla przejścia do odpowiedniej procedury obsługi. Jednak lepszym będzie rozwiązanie, w którym każda procedura obsługi DLI wpisuje adres następnej do komórek $200, $201. Może to być wykonywane w fazie 3. Jest to najbardziej uniwersalne rozwiązanie problemu wielokrotnych DLI. Ma tę dodatkową zaletę, że rozkazy sterowania wyborem wykonywane są po części krytycznej obsługi, a nie przed.

```
FY  1 REM ****************
FJ  2 REM *              *
ZI  3 REM * Program nr 4 *
FL  4 REM *              *
FO  5 REM ****************
NL  6 REM
[illegible] 10 GRAPHICS 7
[illegible] 20 DL=PEEK(5[illegible]0)+2[illegible]*PEEK(561)
[illegible] [illegible] F[illegible]R J=5 T[illegible] [illegible]4
KH  40 POKE DL+J,141
[illegible] 50 NEXT J
HI  60 FOR J=0 TO 30
[illegible] 70 READ A:POKE 15[illegible]+J,A:NEXT J
[illegible] 80 DATA 72,138,72,2[illegible]8,32,6,1[illegible]5
    ,32,6
[illegible] 90 DATA 189,0,240,141,10,212,1
    41,26,208
[illegible] 100 DATA 224,[illegible],[illegible],9,1[illegible]3,0
[illegible] 110 DATA 141,32,6,104,1[illegible]0,104,
    [illegible],4
[illegible] 120 POKE 512,0:POKE 513,6
[illegible] 130 POKE 54[illegible],1[illegible]
```

Procedura systemowa trzasku (klekotu) klawiatury interferuje z procedurą obsługi DLI. Po naciśnięciu i rozpoznaniu klawisza tworzony jest trzask. Jego czas trwania odmierzany jest przez kilka rozkazów STA WSYNC. Może to naruszyć taktowanie procedury obsługi DLI i spowodować, że kolory przeskoczą w dół (o jedną linię ekranu) na ułamek sekundy. Rozwiązanie tego problemu nie jest łatwe. Jeden ze sposobów wykorzystuje rejestr VCOUNT, który informuje, która linia ekranu jest aktualnie wyświetlana przez ANTIC. Procedura obsługi DLI może badać jego zawartość w celu podjęcia decyzji kiedy zmienić kolor. Innym rozwiązaniem może być wyłączenie systemowej procedury obsługi klawiatury i dostarczenie własnej, lecz wymaga to sporo pracy. Ostatecznym rozwiązaniem jest brak akceptacji wejść z klawiatury. Jeśli naciśnięty klawisz nie jest rozpoznany, wstrząsy ekranu nie występują.

Zastosowania

Dopiero teraz staje się oczywista ogromna wartość grafiki pośredniej i rejestrów układowych, których zawartość możemy zmieniać. Za pomocą przerwań programu wyświetlania każdy z tych rejestrów może być zmieniany „w locie". Najbardziej oczywistym zastosowaniem DLI jest umieszczenie na ekranie większej liczby kolorów. Każdy rejestr koloru może być zmieniany po wystąpieniu przerwania. Dotyczy to zarówno rejestrów koloru pola gry, jak i rejestrów koloru graczy. Mamy zatem dziewięć rejestrów, z których każdy może wyświetlać jeden z 256 różnych kolorów. Oczywiście normalny program nie będzie używał wszystkich kolorów. Z kolei zbyt wiele przerwań spowalnia cały program. W praktyce tuzin kolorów jest łatwy do uzyskania, dwa tuziny wymagają uważnego planowania, a więcej — wymaga wymyślonej sytuacji.

Przerwania programu wyświetlania mogą także zwiększyć możliwości grafiki P-M. Pozycja pozioma gracza może być zmieniana podczas DLI. W ten sposób gracz może mieć kilka wcieleń (różnych kształtów) na ekranie.

Innym sposobem wykorzystania DLI jet zmiana szerokości lub priorytetu gracza. Ostatnim zastosowaniem przerwań programu wyświetlania jest zmiana zestawu znaków w dolnej części ekranu. Umożliwia to używanie grafiki znakowej w dużym oknie i zwykłego tekstu w oknie tekstowym. Możliwe są wielokrotne zmiany zestawu znaków; program może używać jednego zestawu znaków u góry ekranu, innego w środku i normalnego zestawu na dole.

Właściwe wykorzystanie przerwań programu wyświetlania wymaga uważnego zaprojektowania obrazu. Musimy pamiętać, że obraz ma „bardziej" pionową strukturę niż poziomą. Wynika to z systemu tworzenia obrazu, w którym tempo kreślenia w poziomie jest 262 razy szybsze od tempa kreślenia w pionie. Komputerowy obraz nie jest czystą kartką papieru, na której kreślimy, jest to stos cienkich pasków, z których każdy może być inny. I musi to być brane pod uwagę podczas projektowania obrazu.

Prezentowane pogramy demonstrują kilka z możliwych zastosowań przerwań programu wyświetlania.

Program nr 5 umożliwia dokonanie zmiany koloru tła w dowolnej części ekranu. Ze względu na to, że zmieniamy zawartość rejestru układowego, zmienia się kolor części ekranu poniżej miejsca przerwania. Górna część ekranu pozostaje bez zmian, ponieważ rejestr układowy aktualizowany jest zawartością rejestru „cienia" podczas realizacji procedury obsługi VBI.

Dla dolnej części ekranu możemy wybrać dowolny kolor. W tym celu określamy kod koloru i umieszczamy jako trzecią liczbę w instrukcji DATA (linia 150). Możemy również zmieniać ten kolor bezpośrednio, za pomocą POKE z adresem 1538.

Możemy również zmienić kolor tła w górnej części ekranu, poprzez wstawienie odpowiedniej wartości do rejestru „cienia" o adresie 712

Możliwa jest również zmiana koloru dowolnego z kreślonych prostokątów. W tym celu musimy wstawić żądaną wartość koloru do odpowiedniego rejestru „cienia" (dla górnej części prostokąta) lub zmienić zawartość odpowiedniego rejestru układowego (dla dolnej części prostokąta). Lewy prostokąt kreślony jest przy użyciu COLOR 1, środkowy — przy użyciu COLOR 2, a prawy — COLOR 3.

Jeśli chcemy dokonać tych zmian w samym programie musimy zmienić ósmą liczbę w instrukcji DATA w linii 150 dla młodszego bajtu adresu odpowiedniego rejestru układowego. Dla prostokąta uży-

wającego COLOR 1 wstawimy 22, dla COLOR 2 — 23 i dla COLOR 3 — 24.

W celu wybrania różnych pozycji pionowych dla zmiany koloru, dodajmy 128 do różnych rozkazów programu wyświetlania. Wykorzystamy do tego linię 240 która umieszcza żądanie przerwania programu wyświetlania w odpowiedniej linii. W używanym trybie 5 BASIC'u otrzymujemy 40 linii obrazu, możemy zatem umieszczać żądania przerwania w komórkach od START + 6 do START + 44.

```
FK 1 REM ***************
FJ 2 REM *             *
AC 3 REM * Program nr 5 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
QM 100 REM
QC 110 FOR I=0 TO 10
SK 120 READ A
RB 130 POKE 1536+I,A
FZ 140 NEXT I
ZY 150 DATA 72,169,6,141,10,212,1
   41,26,208,104,64
QY 160 REM
VZ 170 POKE 512,0:POKE 513,6
VB 210 GRAPHICS 5
FI 220 START=PEEK(560)+256*PEEK(5
   61)
BC 240 POKE START+24,10+128
RQ 310 FOR X=1 TO 3
OC 320 COLOR X
LF 330 PLOT 20*X+5,30:DRAWTO 20*X
   +5,10
NY 340 DRAWTO 20*X-5,10:POSITION
   20*X-5,30
GM 350 POKE 765,X
SJ 360 XIO 18,#6,0,0,"S:"
MB 370 NEXT X
RE 380 REM
JT 390 FOR D=1 TO 300:NEXT D
QP 400 REM
OI 410 POKE 54286,192
NX 420 END
```

Program nr 6 umożliwia zademonstrowanie reklamowanych możliwości komputera Atari, to znaczy uzyskanie na ekranie 128 kolorów jednocześnie. Oprócz dostarczenia procedury obsługi przerwania, wymagało to także zmodyfikowania programu wyświetlania w celu otrzymania odpowiedniej liczby wyświetlanych linii.

Procedura obsługi przerwania tak została napisana, że za każdym jej użyciem wartość kolor/jaskrawość zwiększana jest o 2. Na końcu obrazu otrzymamy 128 kolorów

```
FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Program nr 6 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
UR 110 FOR I=0 TO 17
SU 120 READ B
RS 130 POKE 1536+I,B
FZ 140 NEXT I
VX 160 POKE 512,0:POKE 513,6
WR 210 GRAPHICS 3
FQ 220 DL=PEEK(560)+256*PEEK(561)
XQ 230 COLOR 1
NW 240 FOR I=1 TO 5
TH 250 READ X1,Y1,X2,Y2,X3,Y3,X4,
   Y4
FD 260 PLOT X1,Y1:DRAWTO X2,Y2:DR
   AWTO X3,Y3:POSITION X4,Y4
LP 270 POKE 765,1
SM 280 XIO 18,#6,0,0,"S:"
GK 290 NEXT I
JW 310 POKE DL+3,78
WK 320 FOR I=6 TO 33:POKE DL+I,14
   :NEXT I
CN 330 FOR I=34 TO 98:POKE DL+I,1
   4+128:NEXT I
RA 340 POKE DL+99,78+128
CT 350 FOR I=102 TO 164:POKE DL+I
   ,14+128:NEXT I
FC 360 FOR I=165 TO 198:POKE DL+I
   ,14:NEXT I
EC 370 POKE DL+199,65
SM 380 POKE DL+200,80
EW 390 POKE DL+201,128
[illegible] 410 POKE 54286,192
[illegible] 500 REM
[illegible] 510 DATA 72,173,139,2,24,105 1
[illegible] 520 DATA 141,10,212,141,198,2
   141,24,208,104,64
[illegible] 530 REM
[illegible] 540 DATA 139,155,116,27,77,27,
   100,155
GA 550 DATA 219,155,142,27,203,27
   ,173,155
[illegible] 560 DATA 183,99,160,35,150,35,
   129,99
TZ [illegible] DATA 140,155,153,99,130,99
   ,139,155
[illegible] 580 DATA 173,155,189,99,159,99
   ,178,155
```

Program nr 7 demonstruje sposób wydrukowania komunikatu, w trybie GR. 1 lub 2, zawierającego wielokolorowe znaki (do siedmiu kolorów jednocześnie). Każdy kolor występuje w jednej linii ekranu.

```
FK 1 REM ***************
FJ 2 REM *             *
80 3 REM * Program nr 7 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
WF 40 DIM MSS$(120)
[illegible] 50 TRAP 1000
OP 60 ? "PODAJ TRYB 1 LUB 2";:INP
   UT A:IF A<>1 AND A<>2 THEN 60
[illegible] 70 GOSUB 100:GRAPHICS A+16
DA 80 DL=PEEK(560)+256*PEEK(561)+
   6
[illegible] 90 FOR I=0 TO 6:POKE DL+I,[illegible]+
   A:NEXT I:GOTO 200
QM 100 REM
[illegible] 110 POKE 82,10:POKE 83,29:?
[illegible] 120 REM
OP 130 ? :? "PODAJ TEKST"
[illegible] 140 INPUT MSS$:RETURN
[illegible] 200 REM
[illegible] 210 FOR I=0 TO 42:READ B:POKE
   1553+I,B:NEXT I
QE 220 DATA 72,138,72,152,72,162,
   0,141,10,212
MD 230 DATA 189,1,6,188,9,6,141,2
   4,208,140,25,208
CE 240 DATA 232,173,0,6,201,1,144
   ,3,141,10,212
[illegible] 250 DATA 224,6,144,226,104,16[illegible]
   104,170,104,64
QO 300 REM
[illegible] 310 FOR I=0 TO 73:READ L:POKE
   [illegible]+I,L:NEXT I:POKE 1596,0
IX 320 DATA 72,138,72,173,60,6,20
   [illegible],1,14,[illegible]55,152,[illegible]
[illegible] 330 DATA 1[illegible]3,[illegible],6,141,61,6,173,
   9,6,141,62,6
[illegible] 340 DATA 202,199,1,6,157,2,6
[illegible] 350 DATA 224,0,208,245,173,61,
   [illegible]
QO 360 DATA 141,1,6,162,0,189,10,
   6,157,9,6,232,224,7,208,245,17
   3,62,6,141,16,6,169,0
[illegible] 370 DATA 141,60,6,[illegible],60,6,104
   ,170,104,76,98,228
[illegible] 400 REM
[illegible] 410 FOR I=0 TO 7:POKE 1537+I,(
   [illegible]+I)*16+I*2:NEXT I
HO 420 FOR I=0 TO 7:POKE 1545+I,(
   1+I)*16+I*2:NEXT I
VN 430 POKE 1536,A
MJ 490 POSITION 0,0:? #6," "
[illegible] 500 REM
MS 510 POKE 512,17:POKE 513,6:POK
   E 54286,192
[illegible] 520 POKE 548,63:POKE 549,6:POK
   E 54286,192
[illegible] 600 POSITION 1,2:? #6;MSS$
[illegible] 610 GOTO 610
[illegible] 1000 TRAP 40000:GOTO 50
```

Po wprowadzeniu programu i uruchomieniu go będziemy proszeni o wybranie trybu (1 lub 2). Następnie musimy podać komunikat, który chcemy wyświetlać. Wielkie i małe litery będą miały swoje normalne kolory: pomarańczowy i jasnozielony. Negatywy wielkich i małych liter będą wielokolorowe. Poszczególne kolory będą przesuwać się w przeciwnych kierunkach. Maksymalna długość komunikatu wynosi 120 znaków, lecz może być zmieniona (zmienna MSS$). Podczas wprowadzania marginesy są tak ustawione, symulują szerokość obrazu w trybie 1 lub 2, aby łatwo było rozmieścić komunikat.

Spójrzmy na linię 400. Dwie pętle FOR...NEXT przechowują różne wartości kolorów w tablicy, umieszczonej na stronie szóstej, dla użycia ich przez procedury obsługi DLI i VBI. Zauważmy, zawartości koloru i jaskrawości wzrastają dla każdej komórki, w rezultacie kolor i jaskrawość zmieniają się dla każdej linii ekranu.

Jeśli chcemy, łatwo możemy zmienić te wartości dla otrzymania innych efektów. Na przykład, zachowując tę samą wartość jaskrawości, uzyskamy ostry kontrast między kolorami. A dla otrzymania większego zakresu kolorów należy zmieniać numer koloru o więcej niż jeden dla każdej linii ekranu (np.: I*16*1.2). Jest duża liczba możliwości, które możemy brać pod uwagę dla otrzymania pożądanego efektu.

Procedura obsługi VBI zawarta jest w instrukcjach Data od linii 300. Jej funkcją jest cykliczna zmiana dwóch tablic kolorów, w różnych kierunkach, dla otrzymania efektu ruchomej kurtyny. Jeśli chcemy mieć obraz z nieruchomymi kolorami, możemy wyeliminować procedurę obsługi VBI przez usunięcie linii 520 i dodanie POKE 54286, 192 do linii 510.

Inną łatwą modyfikacją jest zmiana szybkości rotacji kolorów. Liczba 4 po liczbie 201 w linii 320 steruje szybkością zmian. Zmniejszenie tej liczby zwiększa szybkość rotacji; zwiększenie jej — zmniejsza. Zauważmy, że maksymalną liczbą jet 255. Jeśli przekroczymy tę wielkość, zostanie zarejestrowany błąd, który zostanie przechwycony przez TRAP (linia 50); spowoduje to rozpoczęcie działania programu od początku. I tak będzie do chwili poprawienia błędnej wartości dla POKE.

Podczas przygotowywania tego artykułu korzystaliśmy z opisów i przykładów zawartych w „De Re Atari", „Second book of Atari graphics" oraz „Second book of Atari".

UWAGA!

Dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną linii dla Edytora BASIC'u („IKS" nr 4 z 87 r.). Nie wprowadzamy ich do komputera.

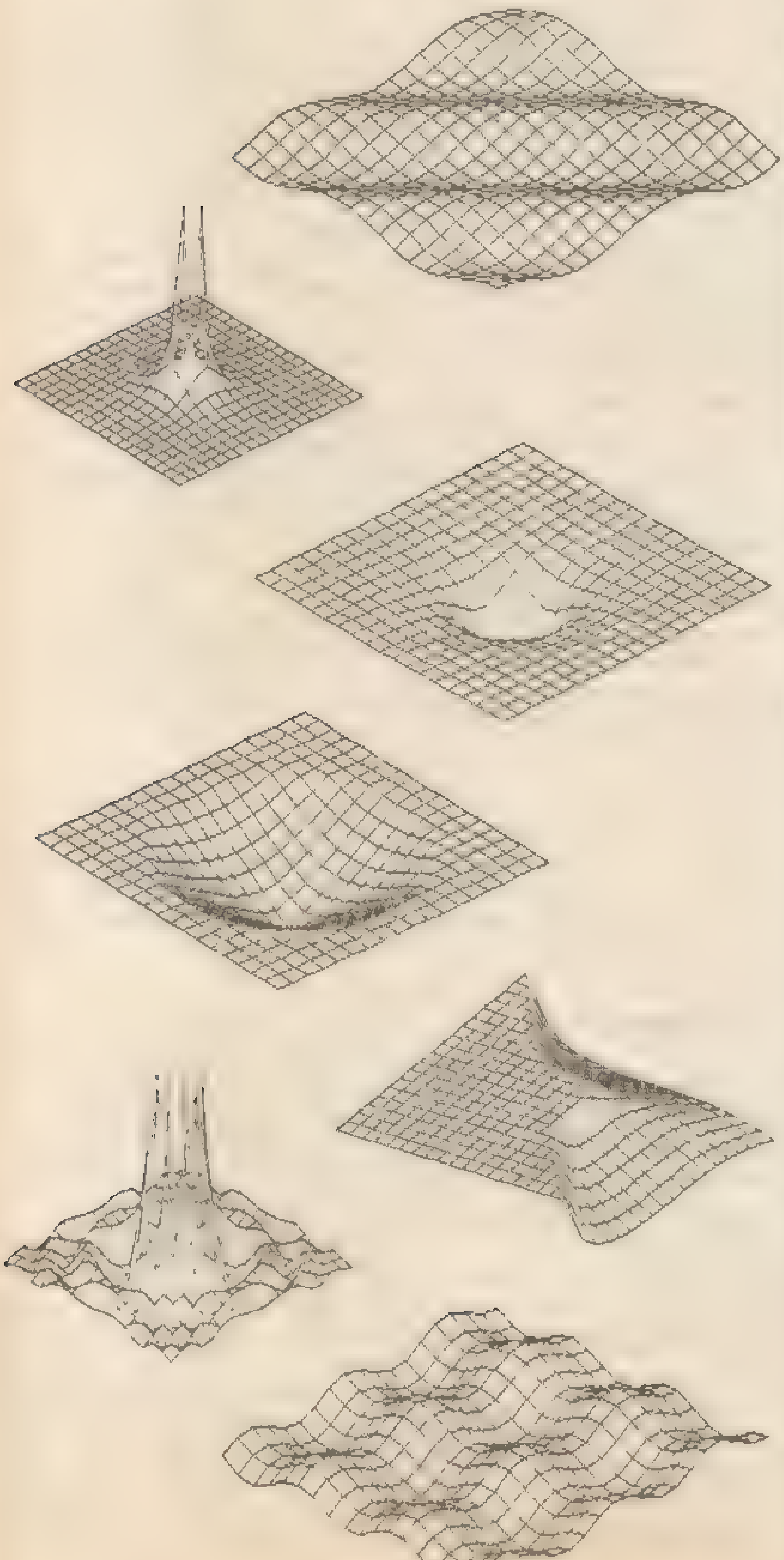
Występujące w programach podkreślone znaki należy wprowadzić w negatywie.

Tomasz MROWIEC
Ludwik PIELA

Grafika trójwymiarowa

```
40 REM **********************************
45 REM grafika  trojwymiarowa
47 REM **********************************
50 BORDER 0:INK 0,1:INK 1,26:INK 2,21:IN
K 3,17:MODE 1:DEG
60 sc=18:tilt=0.45:mag=400/sc:da=45:hlr=
7:vex=10
70 DIM z(sc+1,sc+1),scrd(sc+1,sc+1,2)
80 PRINT"    obliczanie wspolrzednych baz
owych"
90 FOR x=sc TO 0 STEP -1:LOCATE 19,12:PE
N 1:PRINT x:LOCATE 15,25:PEN 2:PRINT "PR
OSZE  CZEKAC":FOR y=0 TO sc
100 a=ATN((x+1)/(y+1)):IF SGN(a)=-1 THEN
 a=180-a
110 r=SQR((x+1)^2+(y+1)^2)
120 scrd(x,y,1)=mag*r*COS(da+a)
130 scrd(x,y,2)=mag*r*SIN(da+a)*tilt
140 NEXT y,x
150 REM
160 WHILE 1:MODE 1:BORDER 0
170 ERASE z:DIM z(sc+1,sc+1)
180 PEN 2:LOCATE 9,2:PRINT CHR$(24);" GR
AFIKA  TROJWYMIAROWA ";CHR$(24)
190 INK 1,0:WINDOW 11,30,4,21:PAPER 1:CL
S:WINDOW 12,29,5,20:PAPER 0:CLS:INK 1,26
200 PRINT:PEN 3
210 PRINT" WYBIERZ  FUNKCJE"
220 PRINT
230 PRINT
240 PRINT:PEN 1
250 PRINT" 1-czarna dziura"
260 PRINT" 2-biala dziura"
270 PRINT" 3-falowanie"
280 PRINT" 4-uderzenia"
290 PRINT" 5-plusk"
300 PRINT" 6-zalamania"
310 PRINT" 7-wzgorze"
320 PRINT" 8-falda"
330 INK 1,26:WHILE INKEY$<>"":WEND
340 j=0:WHILE j<1 OR j>8 :j=VAL(INKEY$):
WEND
350 PEN 3:PRINT:PRINT" obliczenia...":PE
N 1
360 ON j GOSUB 610,670,710,760,810,870,1
080,1130
370 REM
380 MODE 2:DRAW 639,0,1:DRAWR 0,399:DRAW
 0,399:DRAW 0,0
390 BORDER 10 :ORIGIN 320,4,1,638,2,396
400 REM
410 FOR x=sc TO 0 STEP -1:FOR y=sc TO 0
STEP -1
420 sx0=scrd(x,y,1):sy0=scrd(x,y,2)+vex*
z(x,y)
430 sx1=scrd(x+1,y,1):sy1=scrd(x+1,y,2)+
vex*z(x+1,y)
440 sx2=scrd(x,y+1,1):sy2=scrd(x,y+1,2)+
vex*z(x,y+1)
450 IF x<> sc THEN GOSUB 520
460 IF y<> sc THEN GOSUB 560
470 NEXT y,x
480 WINDOW 1,1,1,25:PRINT CHR$(24);"naci
snij dowolny klawisz ";CHR$(24);
490 WHILE INKEY$ <> "":WEND:CALL &BB18
500 WEND
510 REM
520 PLOT 1000,1000,0
530 FOR n=1 TO hlr:MOVE sx0,sy0-n*2:DRAW
 sx1,sy1-n*2:NEXT
540 MOVE sx0,sy0:DRAW sx1,sy1,1
550 RETURN
560 PLOT 1000,1000,0
570 FOR n=1 TO hlr:MOVE sx0,sy0-2*n:DRAW
 sx2,sy2-n*2:NEXT
580 MOVE sx0,sy0:DRAW sx2,sy2,1
590 RETURN
600 REM
610 REM
620 FOR x=0 TO sc:FOR y=0 TO sc
630 z(x,y)=-((20/((x-sc/2+0.25)^2+(y-sc/
2+0.25)^2)))
640 NEXT y,x
650 RETURN
660 REM
670 FOR x=0 TO sc:FOR y=0 TO sc
680 z(x,y)=(15/((x-sc/2+0.25)^2+(y-sc/2+
0.25)^2))
690 NEXT y,x
700 RETURN
710 REM
720 FOR x=0 TO sc:FOR y=0 TO sc
730 z(x,y)=COS((x+y)*600/sc)
740 NEXT y,x
750 RETURN
760 REM
770 FOR x=0 TO sc:FOR y=0 TO sc
780 z(x,y)=COS(x*1000/sc)+SIN(y*1000/sc)
790 NEXT y,x
800 RETURN
810 REM
820 FOR x=0 TO sc:FOR y=0 TO sc
830 q=(x-sc/2)^2+(y-sc/2)^2
840 z(x,y)=COS(q*8)/(0.01*(q+0.1))
850 NEXT y,x
860 RETURN
870 REM
880 RANDOMIZE TIME:FOR i=1 TO 35
890 s1=RND*3+1:s2=RND*3+1
900 IF s2=s1 THEN 890
910 d1=RND*sc:d2=RND*sc
920 d0=d1:ON s1 GOSUB 1040,1050,1060,107
0:ec1(0)=a:ec1(1)=b
930 d0=d2:ON s2 GOSUB 1040,1050,1060,107
0:ec2(0)=a:ec2(1)=b
940 IF ec1(0)=ec2(0) AND ec1(1)=ec2(1) T
HEN 910
950 IF ec1(0)>ec2(0) THEN t0=ec1(0):t1=e
c1(1):ec1(0)=ec2(0):ec1(1)=ec2(1):ec2(0)
=t0:ec2(1)=t1
960 xd=ec2(0)-ec1(0):IF xd=0 THEN m=0:GO
TO 980
970 m=(ec2(1)-ec1(1))/xd
980 c=ec1(1)-m*(ec1(0))
```

```
990 d=ROUND(RND)*2-1
1000 FOR x=0 TO sc:FOR y=0 TO sc
1010 IF y>m*x+c THEN z(x,y)=z(x,y)+d
1020 NEXT y,x,i
1030 RETURN
1040 a=d0:b=0:RETURN
1050 a=d0:b=sc:RETURN
1060 a=0:b=d0:RETURN
1070 a=sc:b=d0:RETURN
1080 REM
1090 FOR x=0 TO sc:FOR y=0 TO sc
1100 z(x,y)=MAX((0.4*sc-SQR((x-sc/2)^2+(
y-sc/2)^2)),0)
1110 NEXT y,x
1120 RETURN
1130 REM
1140 FOR x=0 TO sc:FOR y=0 TO sc
1150 q=(x-sc/2)^3-(y-sc/2)^3-(sc/6)^3
1160 z(x,y)=MAX(ABS(q)^(1/3)*SGN(q),0)
1170 NEXT y,x
1180 RETURN
```



SPECTRUM

Spectrum pomoże dobrze gospodarować materiałem

Jan RUBINOWICZ

```
 100 FOR n=65368 TO 65383: READ
bt: POKE n,bt: NEXT n: RESTORE :
 DATA 0,72,80,96,64,192,126,0,0,
126,66,66,66,66,126,0: REM ab=AB
 110 CLS : INPUT "DAUGOSC MATERI
AAU (3 do 32) ",dlm
 120 IF dlm>32 OR dlm<3 OR dlm<>
INT dlm THEN BEEP 1,-10: GO TO 1
10
 130 INPUT "SZTUK MATERIAAU (3 d
o 22) ";sztm
 140 IF sztm>22 OR sztm<3 OR szt
m<>INT sztm THEN BEEP 1,-10: GO
TO 130
 150 DIM m$(sztm,dlm): DIM r(szt
m)
 160 PRINT AT 5,11; FLASH 1; BRI
GHT 1;" CZEKAJ ! "; FOR n=1 TO s
ztm: BEEP .03,26: FOR m=1 TO dlm
: LET m$(n,m)="B": NEXT m: NEXT
n: CLS
 170 FOR n=1 TO sztm: LET r(n)=d
lm: NEXT n
 180 LET k=1
 190 INPUT "DAUGOSC ELEMENTU ";(
CHR$ (64+k));" (MAX. ";(dlm);")
";dle
 200 IF dle>dlm OR dle<1 OR dle<
>INT dle THEN BEEP .5,2: GO TO 1
90
 210 INPUT "SZTUK ELEMENTU ";(CH
R$ (64+k));" ";szte
 220 IF szte<1 OR szte<>INT szte
 THEN BEEP .5,2: GO TO 210
 230 DIM e$(dle)
 240 FOR n=2 TO dle-1: LET e$(n)
=CHR$ (64+k): NEXT n: LET e$(1)=
"<": LET e$(dle)=">"
 250 FOR n=1 TO sztm: IF r(n)>=d
le AND szte>0 THEN LET m$(n,1+dl
m-r(n) TO dle+dlm-r(n))=e$: LET
r(n)=r(n)-dle: LET szte=szte-1:
GO TO 250
 260 NEXT n
 270 IF szte>0 THEN BEEP 1,-10:
PRINT AT 3,5; BRIGHT 1; FLASH 1.
" ZABRAKAO MATERIAAU ! ": LET sz
tm=sztm+1: GO TO 140
 280 PRINT AT 0,0;: FOR n=1 TO s
ztm: BEEP .01,n: PRINT TAB (16-d
lm/2); PAPER 6; BRIGHT 1;m$(n):
NEXT n
 290 LET reszta=0
 300 FOR n=1 TO sztm: LET reszta
=reszta+r(n): NEXT n: PRINT #0;A
T 0,0;" RESZTA = ";reszta;AT 1,0
;" WYKORZYSTANIE MATERIAAU ";.01
*INT (10000*(dlm*sztm-reszta)/(d
lm*sztm));" %"
 305 IF reszta=0 THEN PAUSE 0: G
O TO 100
 310 PAUSE 300: LET k=k+1: GO TO
 190
 800 CLS : PRINT '''''"
  Niniejszy program pozwala na
  optymalny rozkroj materialow
   o dlugosci do 32 jednostek
  na krotsze odcinki z wykorzy-
  staniem pozostalych odpadow."
 810 PAUSE 0: GO TO 100
 900 SAVE "Rozkroj" LINE 800: BE
EP .2,6: CLS : PRINT '"Verify:
Rozkroj": VERIFY "": GO TO 100
```

Informatyka w wojsku

Wciśnięcie właściwych cyfr zamka szyfrowego sprawia, że ciężkie, obite blachą drzwi, otwierają się automatycznie. Do środka, mogą dostać się tylko ci, którzy znają ten szyfr i... obowiązujący tu język. Odmienny od tych, których nauczają w szkołach, rzec można wyjątkowy — język komputerów.

To prawda już dziś banalna. Współczesna nauka i technika odmieniły nasze życie. Nigdy jeszcze w historii człowiek nie wymyślił tak wiele, w tak krótkim czasie. Dziś każdy dzień nadaje tym słowom szczególny sens. Nie można sobie wyobrazić obecnie żadnej dziedziny ludzkiej aktywności bez najnowocześniejszych osiągnięć współczesnej techniki czy elektroniki. Dlatego trzeba, na co dzień, być z nią za pan brat.

Ba, ale czy w naszych warunkach jest to obecnie możliwe? Zwłaszcza, jeśli na te nowości elektroniki patrzy się przez pryzmat cen komputerów i oprzyrządowania do nich. To prawda. W cywilu wciąż jeszcze nie dla wszystkich jest ten sprzęt dostępny. A jak jest w wojsku?

Utechnicznienie, komputeryzacja, postęp, racjonalizacja. W armii ich znaczenie jest trudne do przecenienia. Nie jest dziś dla nikogo tajemnicą, że każda nowinka techniczna, zanim zostanie wprzęgnięta na usługi ludności, trafia najpierw pod wojskowy kapelusz. Poza tym współczesne uzbrojenie, którym posługują się codziennie żołnierze, to skomplikowany, nafaszerowany elektroniką, jak dobry placek rodzynkami, sprzęt.

Przykład pierwszy z brzegu. We współczesnym czołgu, albo armato-haubicy, najtrudniejsze, najbardziej czasochłonne prace, wykonuje elektronika. Komputer, na spółkę z laserem, w kilka sekund ustalą odległości do celów i podadzą właściwe nastawy do prowadzenia celnego ognia. Specjalne urządzenia samoczynnie załadują działo i wyrzucą łuskę po wystrzale, potrafią ostrzegać o zagrożeniu bronią masowego rażenia, same — jeśli zajdzie taka potrzeba — uruchomią system gaszenia ognia w wozie bojowym.

Dlatego właśnie technika nie może mieć dla żadnego żołnierza tajemnic. Podczas służby wojskowej młodzi ludzie poznają jej tajniki i uczą innych posługiwania się nią. Bo nawet najnowocześniejszy sprzęt pozostanie jedynie kupą żelastwa, jeśli nie będą go obsługiwać specjaliści najwyższej klasy. Rozumieć i znać technikę, to w wojsku wymaganie dotyczące wszystkich — żołnierzy i dowódców.

Właśnie tu, za obitymi żelazem drzwiami, w ośrodku przetwarzania danych Pomorskiego Okręgu Wojskowego, widać to najlepiej. Wszystko, co się dzieje w środku, ma ściśle tajny charakter. Dlatego tak bardzo dba się tutaj o ochronę tajemnicy wojskowej. Wszędzie elektroniczne zamki oraz przeróżne zabezpieczenia, które uniemożliwiają praktycznie dostanie się do środka komuś niepowołanemu.

I czystość. Niewiarygodna czystość. Płócienne ochraniacze na buty, klimatyzacja zapewniająca stałą temperaturę w pomieszczeniach. Kurz i pył są bowiem wrogami komputerów. Zwłaszcza że pracują bardzo długo. W ośrodku komputerowym mają zatrudnienie przez cały, dopuszczalny instrukcją, czas. Wystarczy zobaczyć, jak ogromna ilość dysków magnetycznych składa się na tutejszy bank informacji, aby mieć wyobrażenie o rozmachu działalności wojskowych elektroników, głównie absolwentów wydziału cybernetyki Wojskowej Akademii Technicznej.

Na tych dyskach zapisane zostały dane do wszystkich systemów informatycznych. Chor. Andrzej Rohuka twierdzi, że mają tutaj nieograniczoną możliwość przesyłania danych za pomocą urządzeń transmisji danych do bardzo wielu abonentów. I to z każdej dziedziny życia i pracy żołnierzy okręgu. Od racjonalnego gospodarowania poczynając, a na kontroli podzespołów i części zamiennych kończąc.

Bo komputery mogą dosłownie wszystko. W ich pamięci zapisano dane umożliwiające efektywne kierowanie i dowodzenie wojskami w koszarach i w polu. Wystarczy „nacisnąć guzik", a komputer równie łatwo wydrukuje plakat zespołu młodzieżowego, jak udzieli aktualnych informacji o wyposażeniu i uzbrojeniu pododdziału z najdalej położonego garnizonu. Ba, jeśli jest taka potrzeba, może być ona wydrukowana z szybkością np. tysiąca wierszy na minutę i dostarczona w tej formie adresatowi.

W ośrodku obsługujący komputery specjaliści i specjalistki twierdzą, że jedynej rzeczy, której jeszcze nie potrafią robić, to... gotować. Ale za książkę kucharską mogą służyć znakomicie. Ale poważnie.

— Komputery u nas ciężko pracują — mówi operator systemu informatycznego. — Oprócz dostarczania danych, dokonują także analizy różnych zagadnień taktycznych, wskazują najbardziej efektywne sposoby przemieszczania wojsk w terenie, obliczają siły i środki niezbędne do wykonania określonego zadania, wskazują ilość i jakość zapasów.

Dlatego też oprócz stacjonarnego, jest także polowy zestaw komputerowy, niezbędny, jak się okazuje, podczas działania wojsk w polu. Bez komputerów praca dowódców oraz oficerów sztabu byłaby żmudna i uciążliwa. A tak? Życzy sobie szef służby jakieś dane — proszę bardzo! Komputer się nie męczy, jest na każde polecenie.

— Nie ma praktycznie w naszych jednostkach dziedziny służby i pracy — tłumaczył oficer kierujący ośrodkiem — w której komputer nie miałby zastosowania. Nawet jadłospisy układa. Posiłek żołnierza zawierać przecież musi określoną ilość kalorii, odpowiednio muszą być użyte — i we właściwych ilościach — najróżniejsze składniki. Dla człowieka to kawał roboty, a dla komputera — fraszka.

One programują trasy przejazdów samochodów, aby nie przewoziły powietrza, liczą pieniądze i kontrolują konta, pomagają gospodarować mieszkaniami oraz kierować zaopatrzeniem, a nawet... kontrolować obecność i efektywność pracy kadry i żołnierzy w jednostkach i instytucjach wojskowych.

Specjalny system informacyjny pozwala służbie dyżurnej na wyjątkowo szybkie otrzymanie informacji na ten temat. Wystarczy tylko podnieść słuchawkę. Komputerowy system kontroli gotowości bojowej i nadzoru pracy potrafi także rozliczyć każdego z wykonanych przez niego zadań dziennych, a także tygodniowych i miesięcznych.

Nowoczesna technika mocno zakorzeniła się w wojsku. Dziś, bez pomocy elektroniki, trudno byłoby skutecznie dowodzić i efektywnie się szkolić. I będzie ona zmieniać obraz naszego życia w dalszym ciągu. W zespole informatyków POW zainstalowano znowu mikrokomputery nowszej generacji. Opracowuje się oprogramowanie dla kolejnych systemów informacyjnych. Elektronika w żołnierskiej rogatywce czuje się dobrze i służy armii skuteczną pomocą.

Włodzimierz KALETA

SPECTRUM

Semigrafika

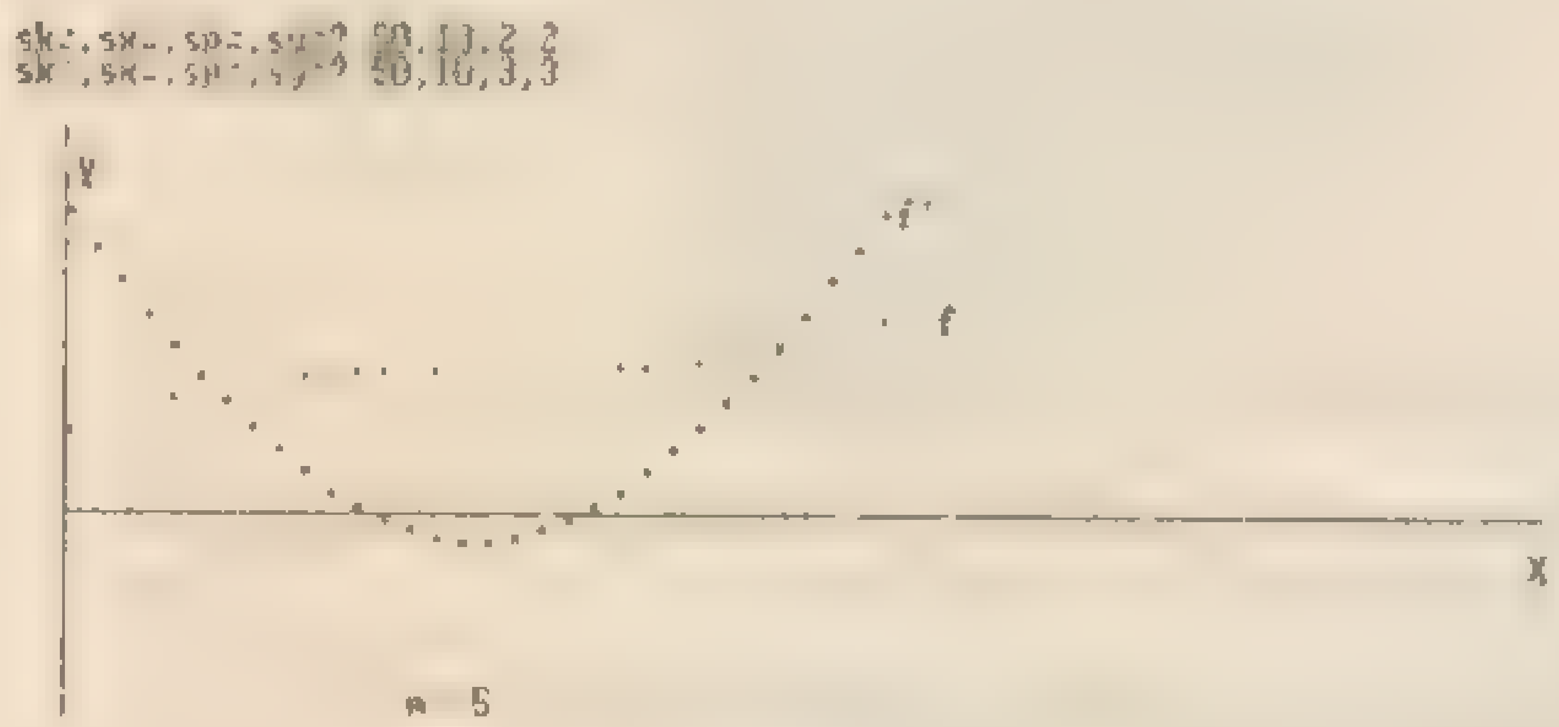
Jan RUBINOWICZ

```
   1 REM SEMIGRAFIKA
   5 BORDER 6: CLS : FOR n=97 TO
 112: PRINT #0; BRIGHT 1;AT 0,2*
(n-97);CHR$ (31+n);: NEXT n
   6 FOR n=97 TO 112: PRINT #0;A
T 1,2*(n-97);CHR$ (n);: NEXT n
  10 PAUSE 0: LET a=CODE INKEY$:
 IF a>95 AND a<113 THEN BEEP .01
,26: PRINT BRIGHT 1;CHR$ (31+a);
  15 IF a=12 OR a=46 THEN BEEP .
02,-10: PRINT AT 24-PEEK 23689,3
2-PEEK 23688;" ";AT 24-PEEK 2368
9,32-PEEK 23688;
  17 IF PEEK 23688=33 AND PEEK 2
3689<24 THEN POKE 23689,PEEK (23
689)+1: POKE 23688,1
  20 IF a=115 THEN PRINT #0;AT 0
,0,"
    ": SAVE "semigrafik"SCREEN$
  25 GO TO 10
  30 SAVE "SEMIGRAFIK" LINE 1
```

Funkcja i jej pochodna

Zdzisław ŁUKASZEWICZ

Niejednokrotnie każdy z Czytelników „IKS-a" rozwiązywał bądź też będzie rozwiązywał w przyszłości zadania z matematyki, które zaczynają się od słów: zbadaj przebieg funkcji. Prezentowany poniżej program, napisany w języku BASIC na mikrokomputer AMSTRAD pomoże w rozwiązaniu tego typu zadań. Program ten umożliwia wykreślenie funkcji pierwotnej i jej pierwszej pochodnej. Aby otrzymać te wykresy należy przed uruchomieniem programu w linię o numerze 80 wpisać postać funkcji pierwotnej. Daną wejściową jest liczba *m*, która ze względu na przyjętą metodę wyznaczania wartości pochodnej funkcji jest liczbą naturalną. Należy w tym miejscu zaznaczyć, że dokładność wyznaczenia wartości pochodnej zadanej funkcji rośnie wraz ze wzrostem liczby *m*. Przykładowo dla $m = 2$ uzyskamy $n = 2$ i $m = 4$, a dla $m = 5$ $n = 32$ wartości funkcji i jej pochodnej, trzeba o tym pamiętać przy skalowaniu wykresu. W programie wartości funkcji pierwotnej zawiera tablica *a(i)*, zaś wartości jej pochodnej tablica *c(i)*. Do programu dołączony jest podprogram WYKRES. Daje on możliwość wykonania na ekranie monitora wykresu funkcji pierwotnej i jej pierwszej pochodnej. Jest on mało wygodny w użyciu, gdyż nie zawiera automatycznego skalowania, bardziej wybrednym polecam programy wykonywania wykresów prezentowane na łamach ogólnodostępnych pism informatycznych. W przytoczonym podprogramie zmienne do skalowania mają następujące znaczenie — sk — skala współrzędnej Y, sx — skala współrzędnej X, sp i sy określają rozmiar punktu, którym wykonywany jest wykres, co ma znaczenie ze względu na nakładanie się dwóch wykresów



W celu zilustrowania efektu, jaki można uzyskać posługując się prezentowanym programem, na rysunku pokazano wykres funkcji F(x) = PI * x + COS (PI * x) dla m = 5.

Metoda wyznaczania wartości pochodnej funkcji, którą posłużyłem się przy opracowywaniu przedstawionego programu, oparta jest na teorii szeregów ortogonalnych.

Życzę przyjemnej zabawy

```
10 MODE 2
20 INPUT"m=";m
30 n=2^m
40 DIM a(n+1),c(n+1) ,b(n+1)
50 FOR i=1 TO n+1
60 x=(i-1)/n
70 '*********          W linii 80
   wpisz postac funkcji      *********'
80 a(i)=PI*x+COS(PI*x)
90 NEXT i
100 FOR i=1 TO n:PRINT a(i);:NEXT i
110 FOR i=1 TO n
120 b(i)=((a(i)+a(i+1))/2)-a(i)
130 NEXT i
140 z=2^(m+1)
150 g=2^(m+2)
160 FOR i=1 TO n
170 FOR j=1 TO n
180 IF j>i THEN GOTO 240
190 IF i=j THEN 220
200 c(i)=c(i)+((-1)^(i+j))*g*b(j)
210 GOTO 230
220 c(i)=c(i)+((-1)^(i+j))*z*b(j)
230 NEXT j
240 NEXT i
250 FOR i=1 TO n
260 PRINT c(i);
270 NEXT i
280 FOR k=1 TO 5000
290 NEXT k
300 GOSUB 500
310 GOSUB 700
320 GOSUB 570
330 GOTO 330
490 '******** PODPROGRAM WYKRES *******'
500 CLS
501 d=1
510 FOR i=20 TO 580
520 PLOT i,175
530 NEXT i
540 FOR i=20 TO 350
550 PLOT 20,i
560 NEXT i
570 INPUT"sk=,sx=,sp=,sy=";sk,sx,sp,sy
580 FOR i=0 TO n-1
590 FOR j=1 TO sp
600 FOR k=1 TO sy
610 PLOT sx*i+20+j,sk*a(i+1)+150+k
620 NEXT k
630 NEXT j
640 NEXT i
650 RETURN
700 FOR i=1 TO n
710 a(i)=c(i)
720 NEXT i
730 RETURN
```

Jak jest pogoda? Przeważnie widzimy sami. Jednak chyba najbardziej interesuje nas, jaka będzie pogoda? A ponieważ jest ona bardzo zmienna i nie mamy na jej zmianę żadnego wpływu, przeto staramy się przewidywać — prognozować. Dawniej zajmowali się tym wróżbici. Gdy udało się im przewidzieć stan pogody — potwierdzali tym samym swoją moc magiczną. Jeśli nie, wtedy zrzucali winę na moce nadprzyrodzone.

A dzisiaj? Niekiedy odwołujemy się jeszcze do wróżb, np. górali, albo „czujemy" pogodę w kościach. Najczęściej jednak słuchamy radia, patrzymy w telewizor, ufając, że nazajutrz nie musimy brać parasola do pracy, bądź wyjmować akumulatora z samochodu. Coraz bardziej ufamy prognozom naukowym i jest już niewielu ludzi, którzy na podstawie zjawisk zachodzących w atmosferze starają się na własny użytek przewidzieć pogodę. Ale też obrywa się synoptykom za niesprawdzone prognozy, bo takie się zdarzają — niestety. Mimo bardzo dokładnych i skomplikowanych metod pomiaru i określania pogody. Przyjrzyjmy się zatem, jak się to robi w świecie i u nas w kraju. Komu i czemu służą systemy meteorologiczne?

wywanie prognoz po prostu się opłaca. Już dziś informacje meteorologiczne są towarem. W Polsce jeszcze daleko do stanu, jak jest na Zachodzie, ale wszystko wskazuje na nieuchronność tego zjawiska. Przyjrzyjmy się zatem technicznym aspektom opracowywania informacji o pogodzie.

Instytut Meteorologii i Gospodarki Wodnej w Warszawie. Placówka naukowa a zarazem instytucja zajmująca się sprawami pogody, o której codziennie słyszymy w komunikatach prasowych, radiowych i telewizyjnych. W pomieszczeniu synoptyków bezustannie trwa opracowywanie danych. Pochodzą one z trzech źródeł: pomiarów naziemnych, satelitarnych, radarów meteorologicznych. A więc informacje punktowe, globalne i przestrzenne w zasięgu radaru.

— *Podstawą do prognozy jest zebranie materiału obserwacyjnego* — wyjaśnia mgr inż. **Ryszard Klejnowski,** zastępca dyrektora IMiGW do spraw Prognoz. — *I to co najmniej z całej półkuli północnej w czasie rzeczywistym. Jest właśnie godzina dziewiąta. O tej porze na całym świecie dokonuje obserwacji w klatkach pomiarowych około 8 tysięcy ludzi jednocześnie.*

SŁUŻBA POGODY. Organizacja, której zadaniem jest zapewnienie informacji o bieżących i prognozowanych warunkach atmosferycznych całej gospodarce narodowej i obronie kraju. Służba pogody IMGW liczy w Polsc 62 stacje meteorologiczne oraz 8 biur: Białystok, Gdynia, Kraków, Poznań, Szczecin, Wrocław, Warszawa.

W ciągu 10 minut wszystkie dane z Polski przekazuje się natychmiast drogą teleksową do GTS (Globalny System Telekomunikacyjny) przez regionalne centrum meteorologiczne w Pradze. Przesyła się je także do sąsiadów w Moskwie i Berlinie. Praktycznie

POGODA Z KO

Marek

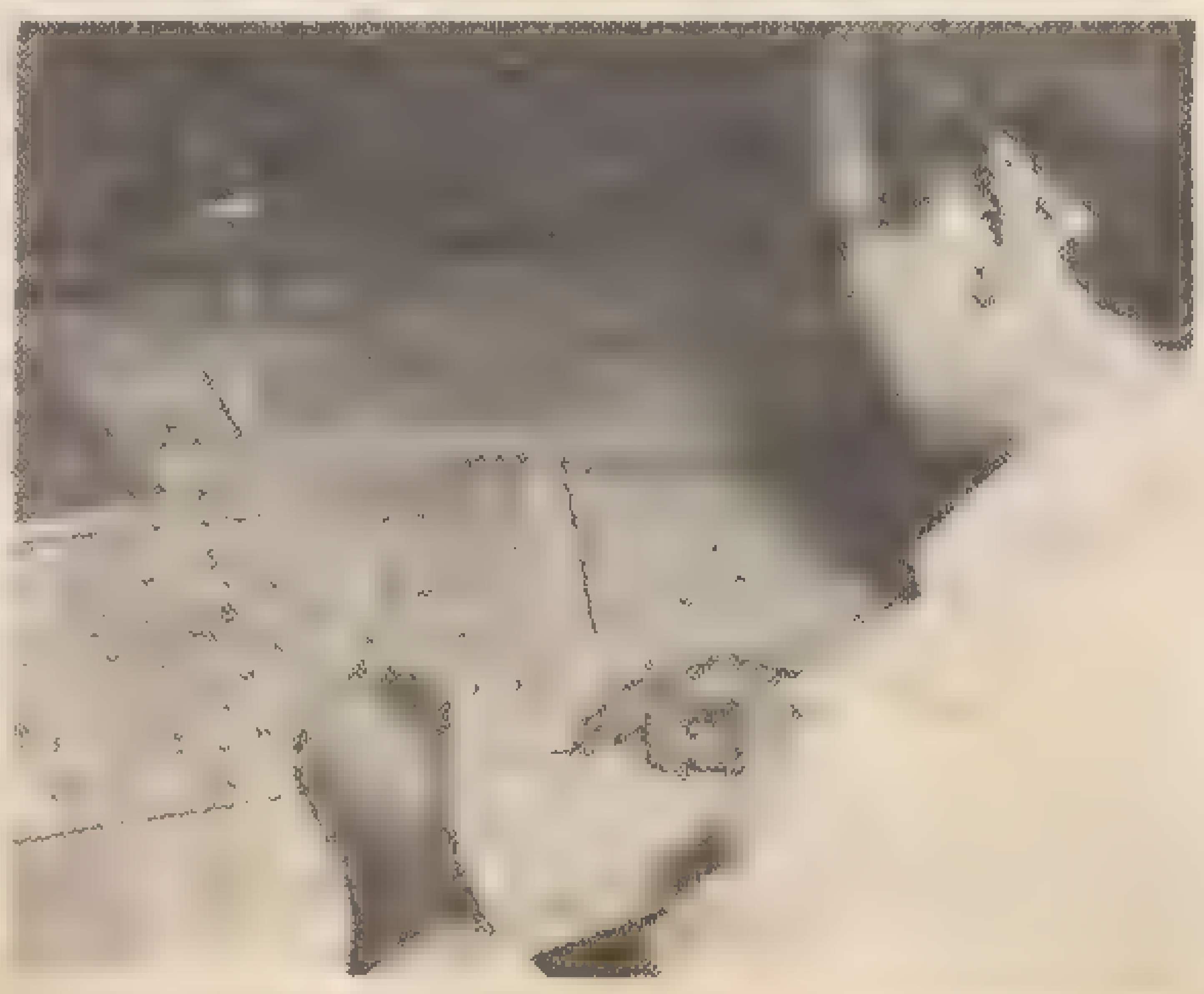
METEOROLOGIA — część geofizyki. Nauka o atmosferze, jej budowie, właściwościach i przebiegających w niej procesach fizycznych. Dużą część meteorologii, poświęconą klimatowi, wydziela się w samodzielną dyscyplinę nauki — klimatologię.

Z prognoz pogody najczęściej korzystają lotnictwo, żegluga morska, rolnictwo oraz wiele przedsiębiorstw takich jak: stocznie, porty, bazy przeładunkowe, oczyszczania miast. Przykładów, gdy zlekceważono ostrzeżenia synoptyków o zbliżających się burzach, opadach, mrozach, wiatrach, mgłach lub innych zjawiskach atmosferycznych można podać bardzo dużo. Z naszych, polskich doświadczeń, najbardziej pamiętnym będzie tzw. zima stulecia (chociaż wcale nią nie była, ale z uwagi na straty, tak ją określono) w roku 1978/79. Ściślej biorąc noc sylwestrowa z 31 na 1 stycznia, gdy nagły powiew chłodu sparaliżował większość kraju. Albo ostatnie zaskoczenie huraganem w Wielkiej Brytanii. Mimo ostrzeżeń francuskich służb meteorologicznych i wskutek zlekceważenia własnej informacji z wybrzeża, kataklizm pokonał wszystkie systemy zabezpieczeń. A właśnie Wielka Brytania należy do państw szczycących się posiadaniem bardzo nowoczesnych systemów meteorologicznych. Jeśli jesteśmy już przy kraju wyspiarzy, przypomnijmy przykład właściwego wykorzystania pogody podczas inwazji w Normandii. Pewien major ze służby meteo przewidział niewielką poprawę pogody, co spowodowało zupełne zaskoczenie u Niemców. Prognoza ta bowiem, choć bardzo ryzykowna, sprawdziła się wtedy.

To tylko znane i głośne przykłady, kiedy człowiekowi udało się lub też nie, zapobiec skutkom żywiołów spowodowanych warunkami atmosferycznymi. Niezaprzeczalne jest jednak to, że bardzo dobra znajomość stanu pogody i coraz doskonalsze opraco-



Uzyskaliśmy stosunkowo wysoką, bo 86 procent, sprawdzalność codziennych prognoz meteorologicznych — podkreśla mgr inż. Ryszard Klejnowski.



Na razie mapy pogody kreślą u nas jeszcze ludzie. Ale niedługo będą to robiły komputery.

(Foto: Jan Zelman)

już w 15 minut po pomiarze znane s
pomiarów z całego świata. Co god
konuje się tych pomiarów. Infor
Europy otrzymujemy co godzinę, a z
co 3 godziny regularnie przez całą

Na stole głównego synoptyka
przypadku był to **Jerzy Bednarek,**
się wszystkie potrzebne dane, w tym
satelitarne. Z satelity geostacjo
METEOSAT-2 oraz orbitującego
Oprócz tego liczne komunikaty ze s
darowych, od sąsiadów, poprzez m
noptyczne, tj. sprzed kilku godzin
przednich dni. Ale okazuje się to jes
wystarczające. Im więcej bowiem
godnych informacji w zasięgu rąk s
ka, tym lepiej. Bo tym lepiej można
dzieć pogodę. Jednak natłok inform
obecnie tak duży, że trudno je prz
opracować metodą ręczną. Niezbęd
się wprzęgnięcie techniki kompute
prawdę znają wszyscy meteorolo
nak względy czysto finansowe pow
nie stać nas na najnowszy sprzęt.

Cały świat przechodzi już na zau
zowane systemy gromadzenia i op
wania danych, ich selekcję, anali
prognozowanie, a następnie dystryb
formacji. Dziś mówi się już o fabryc
macji dotyczących stanu atmosfer
kto potrafi je wszystkie wykorzysta
mieć trafniejsze prognozy.

Zostańmy więc jeszcze przez c
sali głównego synoptyka kraju w IM
opracowanie mapy synoptycznej po
on około dwóch godzin. Wykresa
na podstawie danych nanoszonych
pę przez innych ludzi z nim współ
cych. Obszar obejmuje całą Euro
nocną część Afryki, Azję i Atlantyk p
landę.



...k się robi mapę pogody dla prasy? Pokazał nam to Michał ...worski, który tego dnia pełnił dyżur.

...MPUTERA

...SIENIAWSKI, Tadeusz BERLIŃSKI

...są wyniki ...dzinę do- ...rmacje z ...ze świata ...dobę

..., w tym ...znajdują ...n zdjęc a ...onarnego ... NOAA ...stacj ra- ...napy sy- ...oraz po- ...szcze nie ...n wiary- ...synopty- ...a przewi- ...nacji jest ...zerobić i ...dne staje ...owej. Tę ...ży Jed- ...dują, że

...utomaty- ...pracowy- ...lizę oraz ...bucę in- ...ce infor- ...ry. I ten ...ać, może

...chwilę w ...MGW. Na ...otrzebuje ... zobary ...h na ma- ...pracują- ...opę, pół- ...po Gren-

Obecnie już niewiele państw na świecie zajmuje się ręcznym wykreślaniem map synoptycznych. Właściwie wszystko robią za człowieka komputery. Potrafią one bardzo szybko wykreślić obrazy...



Wyże, niże, zjawiska i elementy meteorologiczne, to wszystko widać na mapie pogody. Kpt. Andrzej Waloryszak, synoptyk wojskowy z CBH wyjaśnia jaka będzie nazajutrz pogoda.

ne dane bez pomocy człowieka. Synoptyk otrzymuje już gotowe przetworzone informacje i na tej podstawie oraz opierając się na swoim doświadczeniu, znanych zjawiskach, mając dzięki temu przestrzenne wyobrażenie o stanie atmosfery — prognozuje. Najczęściej piórem świetlnym na ekranie monitora. Określa kierunki przemieszczania mas powietrza i rozwoju sytuacji.

Dane te koreluje na podstawie zdjęć satelitarnych. I natychmiast przesyła odbiorcom.

Rola synoptyka ogranicza się do znajomości sytuacji i zgodnie z tym ostrzegania, podawania prognoz natychmiastowych oraz średnio- i długoterminowych. Zapotrzebowanie na informacje meteorologiczne wzrasta. Głównie zaś na informacje dla lotnictwa i sił zbrojnych. To właśnie względy militarne, a na drugim miejscu potrzeby meteorologiczne tworzą siłę napędową do budowy superszybkich komputerów. Obecnie dochodzi jeszcze fizyka jądrowa.

Najszybszy komputer CRAY-1 pracuje w Światowym Centrum Meteorologicznych Prognoz w Waszyngtonie oraz Regionalnym Europejskim Centrum Prognoz Średnioterminowych w BRACKNEL (Wielka Brytania). Jest to wspólna inwestycja EWG. CRAY-1 musi wykonać 500 miliardów operacji arytmetycznych, aby postawić prognozę na 10 dni. Przy szybkości 80 mln operacji na sekundę trwa to 1 godzinę i 45 sekund. ODRA-1305 robiłaby to 2 miesiące.

Światowa Organizacja Meteorologiczna (WMO), której członkiem jest także Polska, składa się z 3 podsystemów.

— Obserwacja. 8 tys. stacji synoptycznych, 1500 stacji aerologicznych, 350 radioliniowych, 4 satelity geostacjonarne, 3—5 satelitów orbitujących, 6500 stacji morskich. Przekazuje na dobę 40 mln bajtów informacji.

— Telekomunikacja. Przesyła informacje przetworzone do centrów narodowych, regionalnych i światowych.

— Przetwarzanie. Analiza, diagnozowanie i prognozowanie pogody na podstawie napływających informacji.

W WMO główną rolę odgrywają centra światowe połączone telekomunikacyjną magistralą światową. Zasadniczą prędkością jest obecnie 2400 bodów, a na niektórych kierunkach stosuje się już 9600 dzięki światłowodom i łączności satelitarnej. Do Warszawy na dobę docierać może 14 mln bajtów. Cóż z tego, skoro nie jesteśmy w stanie przetworzyć tej ilości informacji.

ŚWIATOWA SŁUŻBA POGODY. Sieć stacji meteorologicznych i aerologicznych oraz innych środków obserwacji (satelity, sondy) działających według globalnego programu oraz centra meteorologiczne. Sieć łączności do wymiany i przetwarzania danych., Program badań naukowych. .

Światowe centra meteorologiczne: Moskwa, Waszyngton, Melbourne. IMGW jest krajowym centrum meteorologicznym. Połączone z centrami w: Moskwie, Pradze, Poczdamie. Przez Regionalną Zbiornicę Telekomunikacyjną w Pradze połączone z Regionalnym Centrum Meteorologicznym w Offenbach.

W IMGW wkracza, powoli co prawda, ale jednak, najnowsza technika komputerowa do wspomagania pracy synoptyka. W gabinecie dyrektora Klejnowskiego możemy zobaczyć, na razie program próbny, komputerowego zobrazowania danych, przetworzonych na podstawie napływających informacji ze świata i kraju. Natomiast w sali głównego synoptyka już pracuje rejestrator obrazów satelitarnych na barwnym monitorze telewizyjnym. Dzięki współpracy z Politechniką Wrocławską zbudowano i zainstalowano w 1985 roku w Krakowie pierwszy uniwersalny zestaw do odbioru satelitarnych danych meteorologicznych o podwyższonej zdolności rozdzielczej z satelitów TIROS, NOAA wraz z procesorem obrazów pogody. W ten sposób uzyskano możliwości odbioru obrazu satelitarnego, pozwalającego na precyzyjniejsze określenie temperatury powierzchni lądów, mórz i chmur.

Jarosław Mojzych na monitorze komputera IBM AT prezentuje możliwości barwnego przedstawiania sytuacji meteorologicznej pochodzącej z satelity geostacjonarnego METEOSAT. Są to obrazy temperaturowe, wykonane techniką podczerwieni, a więc bez względu na porę dnia. Obrazy docierają co 1 godzinę systemem fotografii, a na komputerze co 30 minut. Komputer animuje te zmiany, co pozwala zobrazować ruch mas powietrza.

Obok **Michał Jaworski** pokazuje, jak się robi na komputerze mapki pogody dla prasy codziennej. Mapa taka powstaje w niespełna godzinę. Jedną z nich, aktualną w dniu oddawania „IKS-a" prezentujemy na pierwszej stronie pisma.

PROGNOZA POGODY. Sporządzanie naukowo uzasadnionych przewidywań o przyszłym stanie pogody. Sporządza się metodą analizy i prognozy makroprocesorów w atmosferze oraz warunków pogody na dużych obszarach za pomocą map synoptycznych i dodatkowych środków (diagramy aerologiczne, przekroje pionowe itd.).

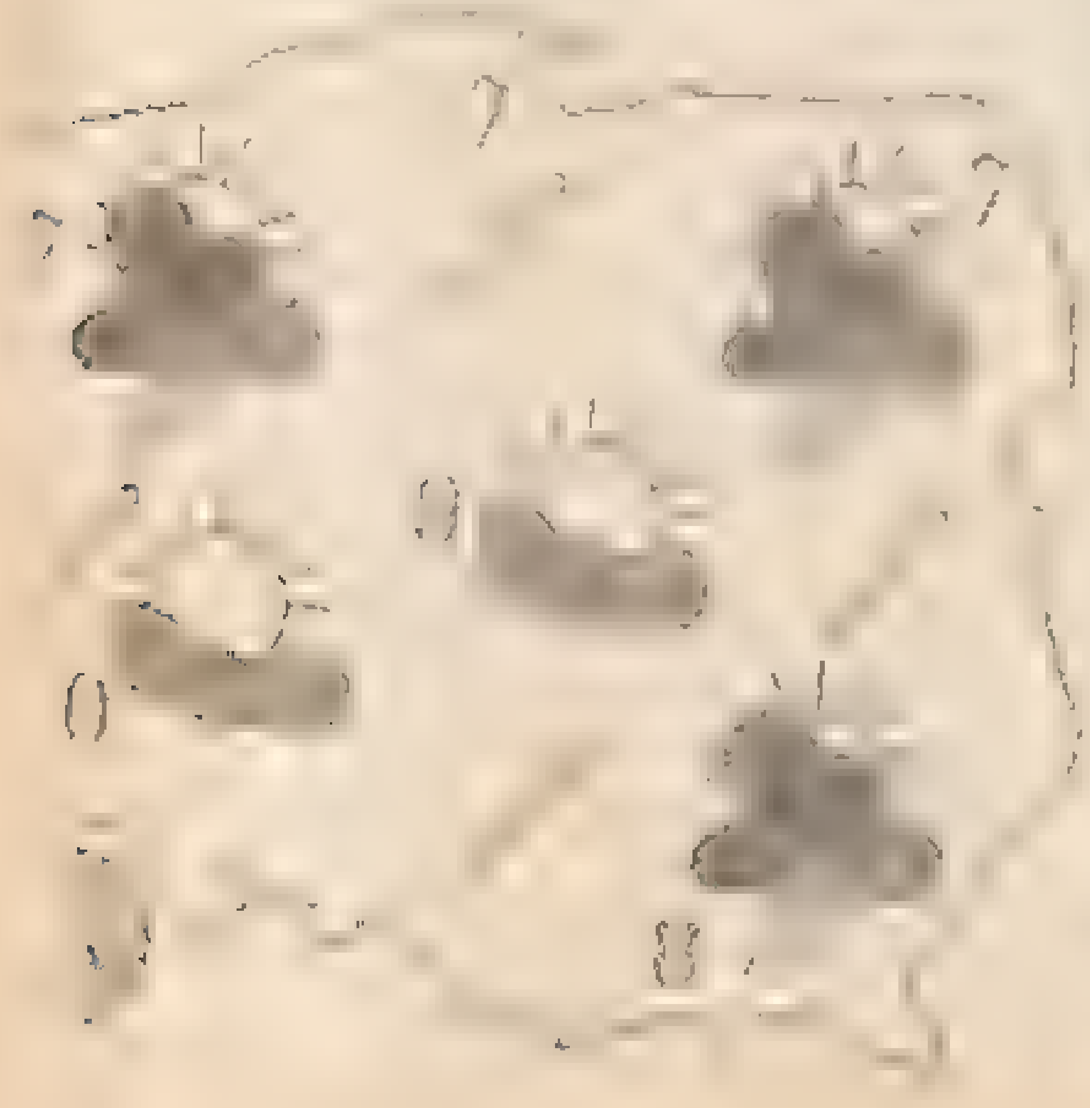
Mgr inż. **Ryszard Klejnow...** IMGW: *Mimo nietypowego w ubiegłym ro...*

FRONT ATMOSFERYCZNY. Strefa przejściowa lub teoretyczna powierzchnia rozdziału między dwiema masami powietrza w atmosferze.
NIŻ (cyklon) zaburzenie w atmosferze o obniżonym ciśnieniu atmosferycznym (minimalne ciśnienie w centrum) i cyrkulacji powietrza dokoła centrum. W lewo na półkuli północnej i w prawo na półkuli południowej.
WYŻ (antycyklon) wyż baryczny — obszar podwyższonego ciśnienia atmosferycznego, który na przyziemnych (dolnych) mapach synoptycznych ma kształt układów zamkniętych, w przybliżeniu koncentrycznych izobar o kształtach nieregularnych, lecz na ogół kolistych lub owalnych.

na początku bieżącego — przebiegu pogody, uzyskaliśmy stosunkowo wysoką sprawdzalność codziennych prognoz meteorologicznych — około 86 proc. W pełni wywiązaliśmy się z funkcji ostrzegania gospodarki narodowej. Nie zanotowano żadnej katastrofy komunikacyjnej, czy też strat gospodarczych związanych z brakiem ostrzeżenia prognostycznego

Rok 1987 był przełomowym w zakresie modernizacji służby prognoz IMGW. Wprowadzano technikę mikrokomputerową. Pozwoliła na to samodzielność finansowa instytutu. Uzyskaliśmy 5 mld zł ze sprzedaży naszych usług. Pracujemy nad systemami przetwarzania i odbioru danych satelitarnych oraz metodyki ich wykorzystania w służbie prognoz. Przełomem stało się zastosowanie komputerów IBM PC. Dysponujemy unikatową aparaturą, wykonaną w kraju, pozwalającą na zastosowanie danych satelitarnych nie tylko w meteorologii, ale również w hydro- i oceanografii, ochronie środowiska, rolnictwie i leśnictwie.

— *Zespół w składzie: Marek i Ludmiła Kozłowscy (małżeństwo), Przemysław Ignatowicz oraz Stanisław Kozłowski od kwietnia 1987 roku zajmuje się opracowaniem systemu odbioru i opracowania informacji meteorologicznych zawartych w depeszach SYNOP, TEMP, GRID przy użyciu mikrokomputerów kompatybilnych z IBM PC/AT/XT* — mówi dr **Anna Olszowicz**



[illegible] w Warszawie.

Efekty już widać na monitorach. Komputer sam kreśli izobary, nanosi dane odbierane ze stacji pomiarowych. Barwy pozwalają na rozróżnianie istotnych elementów pogody.

— *Najsłabszym elementem systemu są sieci do transmisji danych między komputerami* — dodaje dr Olszowicz. — *Próbowaliśmy sieć TRANS-NET i się nie sprawdziła. Obecnie robimy eksperymenty z siecią D-LINK, także amerykańską.*

Zajrzyjmy jeszcze do meteorologów z Wojsk Obrony Powietrznej Kraju.

— *Bardzo istotne jest przekazywanie informacji meteorologicznych odbiorcom, czyli na lotniska wojskowe* — mówi mjr dr inż. **Leonard Kaczanowski** z dowództwa WOPK. — *Sytuacje pogodowe zmieniają się z godziny na godzinę. Piloci wojskowi muszą mieć bardzo aktualne i wiarygodne informacje. Dlatego komputeryzujemy służbę. Warunkują to także prace międzynarodowe, skąd również czerpiemy informacje i dane.*

Na przykład tuż po wojnie obowiązywał system przekazywania informacji alfabetem Morse'a i operatorzy wprost z radia nanosili dane na mapy. O błąd nietrudno, a dane nie były już powtarzane. Potem przeszło się na łączność dalekopisową i w 1981 roku Europa zrezygnowała z łączności dalekopisowej na korzyść teletransmisji. Mówi się już poważnie o przejściu na łączność w systemie cyfrowym.

W wojsku przeważnie sporządza się prognozy krótkoterminowe. Gdy zbliża się termin lotów lub ćwiczeń, wówczas rozpoczyna się gorączka w służbie meteo. Zresztą tu pracuje się bez przerwy. Prognozy dla lotnictwa wojskowego muszą być aktualne w każdej chwili.

Elementy meteorologiczne. Charakterystyki stanu powietrza i niektórych procesów atmosferycznych: ciśnienie, temperatura, wilgotność, wiatr, zachmurzenie (ilość, rodzaj), widzialność, rodzaj i ilość opadów.
Zjawiska meteorologiczne. Są to: opady i mgły różnych typów, zamiecie, zjawiska natury elektrycznej takie jak: burze, błyskawice, zorze polarne, szkwały, burze pyłowe, wiry, trąby powietrzne, igły lodowe, gołoledź, pokrywa śnieżna.

Bywa i tak niekiedy, że za złą pogodę wini się meteorologów. Sypią się też gromy, gdy nad poligonem się przejaśni i mogłoby działać lotnictwo, ale służba meteo nie zezwala. Tylko że nikt nie rozumie, iż na odległych lotniskach pogoda jest zła zarówno do startu, jak i przede wszystkim do lądowania. No cóż, bywa i tak, że meteorologów dowódcy oceniają nie tyle za prognozę, ile za pogodę. Najczęściej też widzi się tych 10 proc. niesprawdzonych prognoz — stwierdza jeden z meteorologów wojskowych. — Na zasadzie czarnej owcy w stadzie.

A praca ta nie jest łatwa. Zajmuje się nią Służba Meteorologiczna WOPK. Informacje odbierane są różnymi systemami i przekazywane do komputera MERA-60. Również z IMGW, Pragi, Moskwy. Komputer ten przyjmuje i rozsyła informacje do wszystkich lotnisk wojskowych w Polsce i sojuszniczych. Kiedyś wszystkie prace wykonywało 5 osób. Teraz komputer pracuje sam. Ppłk mgr inż. **Maciej Ostrowski** blokuje na chwilę komputer i ten po minucie emituje sygnał alarmowy. Natychmiast zgłasza się technik chor. **Zbigniew Paćkowski**, który usuwa przyczynę awarii. Możemy spytać o pogodę na dowolnym lotnisku w Europie. Wybieram z mapy kod lotniska w Neapolu — IY LIRN. Podpułkownik wypisuje daną na klawiaturze mikrokomputera sprzężonego z MERA-60. W niespełna minutę mamy dane za pośrednictwem biura regionalnego w Pradze: wiatr zachodni 7 węzłów, widzialność 8 km, zamglenie, zachmurzenie 3/8, stratocumulus 750 metrów. I dalej dane o przewidywanych zmianach.

— Ten komputer załatwia sprawy związane z telekomunikacją — mówi kierownik biura. — Obecnie pracujemy nad wdrożeniem do pracy plotera przy użyciu komputera IBM. Będzie on wpisywać na mapę te wszystkie dane, które dotąd zapisywali ludzie.

Także tutaj wykorzystuje się faksymilowe zdjęcia satelitarne i wiele innych informacji ze źródeł zagranicznych. Dyżurny synoptyk, ppłk **Klemens Bartysiewicz**, zaznajamia z wykreśloną mapą synoptyczną. Wiemy więc o wiele więcej o stanie pogody i prognozie na najbliższych 10 godzin, aniżeli z telewizji. Możemy potwierdzić, że prognoza na następny dzień się sprawdziła.

Służba meteo opracowała i uruchomiła system NIMBUS, obsługujący wejścia i wyjścia: dalekopisowe, transmisji danych i urządzeń zewnętrznych. Zapewniło to rozpoznanie typu informacji (wg światowego standardu służby meteo), czasu nadawania, numeru stacji i następnie przesyłanie informacji do urządzeń zewnętrznych oraz mikrokomputerów. Zastosowano przy tym 15 wniosków racjonalizatorskich i nowatorskich. Szybko też wdrożono obsługi biur lotniskowych, nadające depesze do komputera. Ten bowiem bezlitośnie traktuje wszystkie niedokładności, a błędne informacje zwraca nadawcy, do czasu ich poprawienia. NIMBUS zastąpił pracę 10 ludzi, zmniejszył opóźnienie zobrazowania stanu pogody na lotniskach do zera. Dotąd opóźnienie wynosiło 15—20 minut. Pracuje się jednak nad tym, aby wspomagać proces opracowywania prognoz pogody bez udziału ludzi, formułować komunikaty na przeloty, ostrzeżenia o niebezpiecznych zjawiskach pogodowych.

Biuro Prognoz Lotniczych w Kansas City (USA) zastąpiło 9 biur regionalnych, daje prognozy na obszar 30 razy większy od Polski i pracuje z częstotliwością 3—4 razy większą. Zatrudnia połowę jego stanu osobowego. Jest więc 200 razy wydajniejszy. Zastosowano w nim 4 systemy komputerowe, 2 zewnętrzne (łączność i prognozy tornad) oraz do wspomagania synoptyka i redagowania prognozy pogody.

A jaką pogodę będziemy mieli jeszcze tej kapryśnej zimy, jaka będzie wiosna? Wierzcie, to naprawdę nie zależy od meteorologów.

Marek SIENIAWSKI
Tadeusz BERLIŃSKI

Lilavati (cz. II)

Rozstawienie żołnierzy na warcie

Mieczysław SKONIECZNY

W drugim odcinku cyklu poświęconego anegdotom matematycznym, chciałbym przedstawić rozstawianie żołnierzy na warcie („LILAVATI" — anegdota nr 16). Treść historyjki jest następująca:

„Na kwadratowe podwórze zbrojowni wszedł porucznik z oddziałem żołnierzy w celu rozstawienia warty. Postawił wzdłuż każdej ściany 4 żołnierzy i się oddalił. Po chwili przybył kapitan i uważając, że warta nie jest dostateczna, umieścił wzdłuż każdej ściany 5 wartowników. Wreszcie na podwórzu zbrojowni zjawił się major i wzdłuż ściany umieścił 6 żołnierzy. Jakie było rozstawienie żołnierzy w 1, 2 i 3 przypadku, jeśli wszyscy trzej oficerowie rozporządzali tym samym oddziałem".

Na ekranie mikrokomputera podwórze to zielony prostokąt, a budynek zbrojowni, to czarna ramka. Na podwórzu rozstawionych jest 8 ponumerowanych posterunków (budki wartownicze). Zasady gry są następujące:

1. *Gra składa się z 3, kolejno po sobie następujących, etapów:*
   I — rozprowadzający: porucznik;
   II — rozprowadzający: kapitan;
   III — rozprowadzający: major;
2. *Posterunki są ponumerowane od 1 do 8, a wartownicy mają nadane kody literowe od A do L (12 żołnierzy).*
3. *Maksymalnie na jednym posterunku można postawić 5 wartowników, a minimalnie dopuszcza się, że posterunek jest pusty.*
4. *Umieszczenie wartownika na wybranym posterunku polega na:*
   — podaniu numeru posterunku;
   — podaniu kodu literowego żołnierza, którego chcemy rozstawić.
   — Po podaniu numeru posterunku na pierwszym wolnym miejscu w tej budce wartowniczej pojawia się czerwony kwadrat.
5. *Rozstawiać można wartownika z dwuszeregu, dopuszczalne jest przestawienie go z posterunku na inny posterunek.*
6. *Kolejny etap gry kończy się sukcesem, gdy grający rozmieści na posterunkach wszystkich wartowników z dwuszeregu, a liczba żołnierzy chroniących każdą ścianę będzie zgodna z wymaganiami tego etapu.*
7. *Użytkownik wygrywa grę, gdy zakończy sukcesem wszystkie 3 etapy (koniec pracy programu).*

Wersja źródłowa programu jest następująca:

```
1 ranga$(1)=" Porucznik":ranga$(2)=" Kapitan":ranga$(3)=" Major"
5 DIM szereg(12):DIM budka(8,5):DIM s(ka 8)
10 BORDER 18:INK 0,18:INK 1,0:INK 2,26:INK 3,6:PAPER 0:CLS:GOSUB 9000
50 DATA 7,7,20,3,35,3,3,11,35,11,3,19,20,19,36,19
70 PAPER 0:CLS:RUWEF 18:PEN 1
80 PRINT"Na podworze zbrojowni wszedl porucznik  z oddzialem 12 zolnierzy. Rozstawil     wzdluz kazdej sciany
4 zolnierzy i       odd
alil sie po wykonaniu zadania."
90 PRINT:PRINT"Po chwili przybyl kapitan i uwazajac, zewarta nie jest dostateczna, rozstawil   oddzial w ten s
posob, ze wzdluz kazde
j  sciany stalo 5 zolnierzy."
100 PRINT:PRINT"Wreszcie na podworzu zjawil sie major i wzdluz kazdej sciany umiescil 6 zolnie- rzy."
110 PRINT:PRINT"Jakie bylo rozstawienie zolnierzy w 1,2 i 3 przypadku ?
120 PEN 2:LOCATE 1,25:PRINT"NACISNIJ DOWOLNY KLAWISZ":c$="":IF INKEY$=c$ THEN GOTO 130
300 etap=1
310 BORDER 0:RESTORE 50:PAPER 0:CLS: FOR c=1 TO 8:READ a,b:GOSUB 1000:NEXT c
315 PAPER 1:LOCATE 1,1:FOR i=1 TO 9:PRINT" ";:NEXT i
320 LOCATE 1,1:PEN 2:PRINT"Etap:";USING "£";etap;:LOCATE 8,1:PRINT"Rozprowadzajacy:";ranga$(etap)
330 GOSUB 2000
360 FOR t=1 TO 2000:NEXT t
370 PAPER 1:PEN 2:LOCATE 1,1:PRINT"Kazda sciane ma chronic ";USING "£";etap+3;:PRINT" wartownikow"
380 ruch=1:FOR i=1 TO 12:szereg(i)=64+i:NEXT i
390 FOR i=1 TO 8:FOR j=1 TO 5:budka(i,j)=0:NEXT j:NEXT i
400 PAPER 0:PEN 2:LOCATE 9,4:PRINT"Ruch: ";ruch
410 LOCATE 21,2:PAPER 1:PRINT"                    ":LOCATE 1,2:INPUT "Podaj nr posterunku: ",nrb
420 IF nrb<1 OR nrb>8 GOTO 470
430 blad=0:FOR i=1 TO 5
440 IF budka(nrb,i)=0 THEN blad=1
450 NEXT i
460 IF blad=1 GOTO 490
470 LOCATE 1,2:PRINT"Blad. Wprowadz poprawny parametr.       ":FOR T=1 TO 1500:NEXT T
480 GOTO 410
490 GOSUB 3000:LOCATE X1,Y1:PEN 3:PRINT CHR$(143)
500 PEN 2:LOCATE 21,2:PRINT"                    ":LOCATE 1,2:INPUT "Podaj kod wartownika: ",kod$
510 kodw=ASC(kod$):IF kodw<65 OR kodw>77 GOTO 560
520 blad=0:FOR i=1 TO 5
530 IF budka(nrb,i)=kodw THEN blad=1
540 NEXT i
550 IF blad=0 GOTO 580
560 LOCATE 1,2:PRINT"Blad. Wprowadz poprawny parametr.       ":FOR T=1 TO 1500:NEXT T
570 GOTO 500
580 IF SZEREG(kodw-64)=0 GOTO 700
590 y2=11:FOR i=66 TO 76 STEP 2
600 IF kodw=i THEN y2=15
610 NEXT i
620 FOR i=1 TO 6
630 IF kodw=63+2*i OR kodw=64+2*i GOTO 650
640 NEXT i
650 PAPER 0:x2=12+2*i:FOR i=0 TO 2:LOCATE x2,y2+i:PRINT"   ":NEXT i
660 szereg(kodw-64)=0
670 PEN 1:LOCATE x1,y1:PRINT kod$:budka(nrb,b)=kodw:GOTO 800
```

```
7.. FOR i=1 TO 8:FOR j=1 TO 5
710 IF budka(i,j)=kodw GOTO 730
720 NEXT j:NEXT i
730 BUDKA(I,J)=0:XX1=4:IF i=2 OR i=7 THEN XX1=2
732 IF i=3 OR i=5 OR i=8 THEN XX1=37
734 YY1=4+j:IF i=4 OR i=5 THEN YY1=YY1+8
736 IF i=6 OR i=7 OR i=8 THEN YY1=YY1+16
740 LOCATE X1,Y1:PAPER 0:PRINT" "
750 PEN 1:LOCATE x1,y1:PRINT kod$:budka(nrb,h)=kodw
8.0 RUCH=RUCH+1
805 SUK=1:FOR I=1 TO 12
810 IF SZEREG(I) < 0 THEN SUK=0
815 NEXT I
8.. FOR P=1 TO 8
8.. suma(p)=0
8.. FOR p1=1 TO 5
8.. IF budka(p,p1)<>0 THEN suma(P)=suma(p)+1
85. NEXT p1:NEXT p
8.. s1=suma(1)+suma(2)+suma(3):s2=suma(...)+suma(5)+suma(8):s3=suma(8)+suma(7)+suma(6):s4=suma(6)+suma(4)+suma(1)
... IF etap<>1 GOTO 9.0
... IF s1=s2 AND s2=s3 AND s3=s4 AND s4=4 AND SUK=1 THEN GOSUB 4000:etap=etap+1:...
..4 GOTO 400
... IF etap<>2 GOTO 920
... IF SUK=1 AND s1=s2 AND s2=s3 AND s3=s4 AND s4=5 THEN GOSUB 4000:etap=etap+1:...
915 GOTO 400
... IF SUK=1 AND s1=s2 AND s2=s3 AND s3=s4 AND s4=6 THEN GOSUB 4000:CLS:STOP
... GOTO 400
```

Do programu dołączono kilka podprogramów, które spełniają następujące funkcje:

1. Rysowanie jednego posterunku w obszarze zadanym parametrami a, b (współrzędne znakowe szczytu daszku).

```
... PAPER 0:PEN 2:LOCATE a,b:PRINT CHR$(189);CHR$(...);CHR$(190)
... LOCATE a-1,b+1:PRINT CHR$(19.);CHR$(19.);CHR$(194);CHR$(195);CHR$(190)
... LOCATE a-2,b+2:PRINT CHR$(...);CHR$(...);CHR$(...);" ";CHR$(...);CHR$(...);CHR$(...)
... PAPER 3:LOCATE a-1,b+3:PRINT CHR$(189):LOCATE a+.,b+3:PRINT CHR$(190):LOCATE a-.,b+5:
PRINT CHR$(189):LOCATE a+3,b+5:PRINT CHR$(...)
...
... PAPER 2:PEN 3:LOCATE a-.,b+2:PRINT CHR$(189):LOCATE a+3,b+.:PRINT CHR$(190):LOCATE a-
...:PRINT CHR$(189):LOCATE a+.,b+4:PRINT
...
... LOCATE a-.,b+6:PRINT CHR$(189):LOCATE a+2,b+6:PRINT CHR$(190)
... LOCATE a+1,b:PRINT USING "&";...:PAPER 0
... RETURN
```

2. Wyrysowanie na ekranie oddziału 12 żołnierzy z kodami literowymi oraz rozprowadzającego (ranga rozprowadzającego jest uzależniona od etapu gry).

```
... PAPER 0:PEN 1:n=65
... RESTORE 2000:FOR a=5 TO 30 STEP ...:FOR b=11 TO 15 STEP 4
... GOSUB 2.00:LOCATE a+1,b+1:PAPER 1:PEN 2:PRINT CHR$(n):n=n+1:PAPER 0:PEN 1:NEXT b:NEXT
...
... a=...:b=...:GOSUB 2100:GOSUB 220.
2040 RETURN
2100 LOCATE a,b:PRINT CHR$(198);CHR$(199);CHR$(200)
2110 LOCATE a,b+1:PRINT CHR$(201);CHR$(143);CHR$(202)
2120 LOCATE a,b+2:PRINT CHR$(203);CHR$(143);CHR$(204)
2130 RETURN
... GRAPHICS PEN 2:ON etap GOTO 2210,...,2400
... FOR wy=... TO 170 STEP 14
... PLOT 166,wy:DRAWR 2,0:DRAWR 2,-2:DRAWR -6,0:DRAWR 0,-2:DRAWR 6,0:DRAWR -2,-2:DRAWR -2
... NEXT wy
2230 GOTO 2500
... FOR wy=198 TO 168 STEP -.0
... PLOT 166,wy:DRAWR 2,0:DRAWR 2,-2:DRAWR -6,0:DRAWR 0,-2:DRAWR 6,0:DRAWR -2,-2:DRAWR -2
..:NEXT wy
... GOTO 2500
... PLOT 166,188:DRAWR 2,0:DRAWR 2,-2:DRAWR -6,0:DRAWR 0,-2:DRAWR 6,0:DRAWR -2,-2:DRAWR -
.
... PLOT 154,1.4:DRAWR 26,0:DRAWR 0,-2:DRAWR -26,0:PLOT 152,16.:DRAWR 30,0:DRAWR 0,-2:DRA
R -30,0
... RETURN
```



3. Ustalenie współrzędnych znakowych pierwszego wolnego miejsca w budce wartowniczej (x1, y1).

```
30.. ...
30.0 IF NRB=... OR NRB=... OR NRB=... THEN ...
3020 FOR H=1 TO 5
3030 IF BUDKA(NRB,H)=0 GOTO 3050
3040 NEXT H
30.. Y1=4+H:IF NRB=4 OR NRB=5 THEN Y1=Y1+8
30.. IF NRB=6 OR NRB=7 OR NRB=8 THEN ...
3070 RETURN
```

4. Zakończenie sukcesem dowolnego etapu gry.

```
4000 PAPER 0:PEN ...:LOCATE 15,...:PRINT"...":LOCATE 19,...:PRINT"...";USING "..
..."...
4..0 PAPER ...:PEN ...:LOCATE ...:PRINT"      B R A W O"

40.0 FOR I=1 TO ...
4040 FOR T=1 TO 100:NEXT T:NEXT I
4050 RETURN
```

5. Definiowanie własnych symboli graficznych wykorzystanych do rysowania budki i oddziału żołnierzy.

```
9... ...
94.. SYMBOL ...
9410 SYMBOL 1..,255,...,255,255
9420 SYMBOL 195,255,...,255,255,...
9430 SYMBOL 196,...
9440 SYMBOL 197,...
9450 SYMBOL 198,...
9460 SYMBOL 199,...
9470 SYMBOL 19.,...
9480 SYMBOL 197,240,...,192,...
9490 SYMBOL 198,0,0,0,0,1,...
9500 SYMBOL 199,...
95.. SYMBOL ...
95.. SYMBOL 201,...
95.. SYMBOL 2..,...
9540 SYMBOL 203,...
9550 SYMBOL 204,224,224,224,240,240,240,240,240
9600 RETURN
```

Życzę Czytelnikom sukcesów w kolejnych etapach gry.

Bardziej dociekliwym proponuję rozwinięcie gry o etap 4. Łatwo bowiem zauważyć, że istnieje jeszcze jedno możliwe rozstawienie wartowników, w którym każdej ściany chroni 3 żołnierzy. Jakie? Poszukajcie sami. Rozprowadzającym w tym etapie mógłby być porucznik.

Jeżeli modyfikacja programu zakończy się powodzeniem, będzie to dodatkowa satysfakcja.

Mieczysław SKONIECZNY

B A S I C

Powróćmy jeszcze do „poczciwego" Basica

W wykładzie tym omówimy operacje na tekstach oraz związane z tym odpowiednie instrukcje i funkcje. Zajmować będziemy się także tablicami alfanumerycznymi

Wykład ósmy

> stałe, i zmienne proste alfanumeryczne, nadawanie „wartości" zmiennej alfanumerycznej, łączenie tekstów (konkatenacja), wybieranie tekstów (za pomocą TO), funkcje standardowe:
> LEN, STR$, VAL, VAL$, CHR$, CODE, porównywanie znaków (operatory relacji, instr. IF) tablice alfanumeryczne

Istnieje grupa problemów, których rozwiązanie opiera się na przetwarzaniu tekstów Dla tych problemów tworzone są specjalne języki programowania zorientowane tylko na przekształcenie tekstów, natomiast pozostałe języki są wzbogacane o różne operacje tekstowe. I tak jak w innych reprezentacjach języka BASIC, w BASIC-u na ZX-Spectrum wprowadzono zmienne alfanumeryczne proste i indeksowane, rozszerzono działania funkcjonalne instrukcji podstawienia i warunkowej, wprowadzono funkcje standardowe wykonujące operacje na tekstach.

Tekstem w języku BASIC jest ciąg znaków. W wykładzie 2 i 3 omawialiśmy stałe i zmienne proste alfanumeryczne, powtórzmy wybrane definicje

Stała alfanumeryczna (tekstowa) ma postać
„ciąg znaków"

— ciąg znaków: dowolny ciąg znaków
— znaki cudzysłowu ograniczają stałą tekstową, nie należą do stałej, lecz jedynie wyznaczają jej początek i koniec
— tekst "" bez żadnego znaku zwany jest tekstem pustym
— tekst " " nie jest tekstem pustym, oznacza jedną spację: spacje wewnątrz cudzysłowu są znakami tekstu

Długością tekstu jest liczba znaków (ze znakami spacji włącznie) znajdujących się w ciągu. Maksymalna długość tekstu wynosi 255 znaków

Poniższe teksty mają następujące liczby znaków

„ZX SPECTRUM" — 11 znaków
„WYNIK = a+b•c−d" — 15 znaków

Analogicznie do pojęcia zmiennej prostej numerycznej istnieje pojęcie zmiennej prostej alfanumerycznej, której „wartością" jest tekst. Nazwa takiej zmiennej składa się z litery (tylko jednej), po której musi nastąpić znak dolara ($), np. a$, p$, r$ są nazwami zmiennych prostych alfanumerycznych

Z dotychczasowych wykładów wynika, że

— tekst może być zapamiętany pod nazwą zmiennej alfanumerycznej

Nazwom zmiennych numerycznych można nadawać „wartości", używając instrukcji LET lub INPUT lub ze zbioru DATA instrukcją READ np

20 LET m $ — "mały"

lub

10 INPUT m $

(z klawiatury wprowadzamy: „mały")

lub

30 READ m$

100 DATA "mały"

— tekst może być wydrukowany instrukcją PRINT: „bezpośrednio" np :

40 PRINT "mały"

można też wydrukować tekst zapamiętany pod zmienną alfanumeryczną np .

20 LET m $ = "mały"

40 PRINT m $

Do tekstów można stosować wszystkie operatory relacji, a tym samym instrukcję IF

Kolejne operacje na tekstach wraz z dotychczas poznanymi zostaną omówione na przykładach. Zakładamy, że instrukcje wprowadzania (INPUT, READ, DATA) i wyprowadzania (PRINT) tekstów są już opanowane przez czytelnika. A więc włączamy komputer i zaczynamy

Przykład 8.1.

a) Dany jest program

```
   5 REM Przykład 8.1
  10 INPUT "a$ =";a$: PRINT "a$
=";a$
  20 INPUT "b$ =";b$: PRINT "b$
=";b$
  30 INPUT "c$ =";c$: PRINT "c$
=";c$
  40 INPUT "d$ =";d$: PRINT "d$
=";d$
  50 PRINT  a$;b$;c$;d$
  60 LET e$=a$+" "+b$+" "+c$+" "
+d$
  70 PRINT '"e$ = ";e$
  80 PRINT '"Wybieranie fragment
ow tekstu     przy uzyciu TO"
  90 PRINT 'e$(1 TO 5),":";e$( T
O 12)
 100 PRINT 'e$(3 TO 10),":";e$(3
 TO )
 110 PRINT 'e$( TO ),":";e$(1 TO
 15)
 120 PRINT 'e$(),":";e$(9)
 130 STOP
```

b) wyniki dla przykładowych danych

```
a$ =As
b$ =to
c$ =jest
d$ =pies

Astojestpies

e$ = As to jest pies

Wybieranie fragmentow tekstu
przy uzyciu TO

As to               As to jest p

 to jest            : to jest pies

As to jest pies :As to jest pies

As to jest pies :s
```

Do przenoszenia tekstów między zmiennymi alfanumerycznymi oraz do przypisywania zmiennym „wartości" stałych alfanumerycznych służy instrukcja podstawienia postaci

n LET V = e

n — nr wiersza
V — nazwa zmiennej alfanumerycznej.
e — wyrażenie tekstowe

Wyrażenie tekstowe może być stałą lub zmienną alfanumeryczną — prostą lub indeksowaną, standardową lub definiowaną funkcję alfanumeryczną. Elementy wyrażenia tekstowego można łączyć operatorem konkatenacji, oznaczanym symbolicznie znakiem plus (+) tworząc jeden tekst. Postać i wykonanie tej instrukcji jest analogiczne do arytmetycznej instrukcji podstawienia (por wykład 2)

I tak wykonanie instrukcji

60 LET e$ = a$ + b$ + c$ + d$

spowoduje przypisanie (nadanie) zmiennej alfanumerycznej e$ wartości będącej tekstem utworzonym z 4 tekstów (przez sumowanie)

Mając dany tekst można wyróżnić jako jego fragment dowolny tekst składający się z części jego kolejnych znaków ustawionych w tym samym porządku.

Instrukcja zwana wybieraniem służąca do określenia fragmentu dowolnego tekstu (w BASIC-u na ZX Spectrum) ma postać wyrażenie tekstowe (*p* TO *k*)

— wyrażenie tekstowe jak wyżej

— *p*, *k* — stałe lub zmienne numeryczne całkowite), określające odpowiednio: początek (*p*), koniec (*k*), wybieranego tekstu i muszą odnosić się do istniejących części tekstu

"abcdef" (2 TO 5) = *"bcde"*

Obowiązuje tu kilka reguł

— jeżeli pominiemy początek (TO *k*), to zakłada się, że jest on równy 1 i wybrane zostaną znaki od 1 do *k* — tego — danego wyrażenia tekstowego

— jeżeli pominiemy koniec (*p* TO), to zakłada się, że jest on równy ostatniemu znakowi i wybrane zostaną znaki od *p* — tego do ostatniego danego wyrażenia tekstowego

— napisanie (TO) lub () powoduje wybranie wszystkich znaków tekstu,

— napisanie (*m*) (*m* — stała lub zmienna określająca liczbę całkowitą), powoduje wybranie m — tego znaku tekstu ($m \leq$ od liczby znaków tekstu),

— jeżeli $p > k$ (początek > koniec), to efektem jest tekst pusty,

— jeżeli $k > 1$ (koniec > liczby znaków wyrażenia tekstowego) to będzie sygnalizowany błąd.

3 subscript wrong
(zły indeks)

— jeżeli początek lub koniec są liczbami ujemnymi ($p < 0$ lub $k < 0$) to będzie sygnalizowany błąd

B integer out of range
(liczba całkowita poza zakresem)

Prostą ilustracją obowiązujących reguł są instrukcje o nr. 80 ÷ 120 w programie z przykładu 8.1

Analizę wykonania tych instrukcji pozostawiamy czytelnikowi, zalecając wykonanie tego programu dla różnych tekstów (zwracając uwagę na różną długość tych tekstów).

W celu ugruntowania nowych wiadomości wprowadźmy i wykonajmy następujące programy:

Przykład 8.2.

a) Dany jest program

```
10 REM Przykład 8.2
20 LET w$="katarynka"
30 FOR n=1 TO 9
40 PRINT w$(n TO 9)
50 NEXT n
60 STOP
```

b) wyniki

```
katarynka
atarynka
tarynka
arynka
rynka
ynka
nka
ka
a
```

Przykład 8.3.

a) dany jest program

```
 10 REM Przykład 8.3
 20 LET p$="warszawianka"
 30 FOR n=1 TO 12
 40 PRINT p$(n TO 12),p$((12-n+
  1) TO 12)
 50 NEXT n
 60 STOP
```

b) wyniki

```
warszawianka    a
arszawianka     ka
rszawianka      nka
szawianka       anka
zawianka        ianka
awianka         wianka
wianka          awianka
ianka           zawianka
anka            szawianka
nka             rszawianka
ka              arszawianka
a               warszawianka
```

Programy z przykładów 8.2 i 8.3 radzimy wykonać kilka razy, zmieniając stałą tekstową w instrukcji o nr. 20

Przykład 8.4.

a) Dany jest program

```
  10 REM Przykład 8.4
  20 INPUT "Twoje imie? ";i$
  30 INPUT "Twoje nazwisko? ";n$
  40 LET x$=i$+n$
  50 LET y$=i$+" "+n$
  60 PRINT "Nazywam sie: "
  70 PRINT 'x$
  80 PRINT 'y$
  90 LET k=LEN (i$)
 100 LET m=LEN (n$)
 110 LET p=LEN (i$+n$)
 120 LET t=LEN (i$+" "+n$)
 125 PRINT '"Twoje imie: ";i$;" 
zawiera ";k;" znaki(ow)"
 130 PRINT '"Twoje nazwisko: ";n
$;" zawiera ";m;" znaki(ow)"
 140 PRINT '"tekst w zmiennej x$
: ";x$;" zawiera ";p;" znaki(ow)
"
 150 PRINT '"tekst w zmiennej y$
: ";y$;" zawiera ";t;" znaki(ow)

 160 STOP
```

b) wyniki dla przykładowych danych

```
Nazywam sie:

KazimierzAbacki

Kazimierz Abacki

Twoje imie: Kazimierz zawiera 9
znaki(ow)

Twoje nazwisko: Abacki zawiera 6
 znaki(ow)

tekst w zmiennej x$: KazimierzAb
acki zawiera 15 znaki(ow)

tekst w zmiennej y$: Kazimierz
Abacki zawiera 17 znaki(ow)
```

W wykładzie 3 omawialiśmy funkcje standardowe, służące do obliczania wartości podstawowych funkcji matematycznych. Teraz omówimy funkcje standardowe służące do „wykonywania obliczeń" na tekstach

W programie z przykładu 8.4 w instrukcjach o nr. 90, 100, 110 120 występuje funkcja o nazwie LEN

Funkcja LEN oblicza długość tekstu, jej argumentem jest tekst, którego długość chcemy obliczyć, a wartością (wynikiem) — jego długość.

Weźmy pod uwagę instrukcję:

110 LET p = LEN (X$)
115 LET t = LEN (Y$)

i zapiszemy je w następujący sposób

110 LET p = LEN (I$) + LEN (N$)
(i) 115 LET t = LEN (I$) + LEN (" ") + LEN (N$)

lub

110 LET p = LEN (I$ + N$)
(ii) 115 LET t = LEN (I$ + " " + N$)

Dokonajmy zmian w programie i sprawdźmy podane kombinacje tych instrukcji. Za każdym razem otrzymamy ten sam wynik (dla tego samego imienia i nazwiska).

Zobrazowaliśmy na przykładzie następującą regułę: — jeśli w jednym wyrażeniu wystąpią funkcje i operacje, wówczas funkcje będą obliczane przed operacjami. Regułę tę można ominąć przez użycie nawiasów.

I tak w przypadku *(i)* obliczane są długość (liczba znaków) odpowiednich tekstów, a potem sumowane są te długości, natomiast w przypadku *(ii)* tworzony jest nowy tekst (przez sumowanie), a potem obliczana jest jego długość.

Zastosujmy funkcję LEN w programach z przykładów 8.2 i 8.3.

```
 10 REM Przyklad 8.2.1
 20 LET w$="konstantynopolitank
a"
 25 LET k=LEN (w$)
 30 FOR n=1 TO k
 35 REM nie powinien wystapic
        zaden blad - znamy dlu-
        gosc tekstu
 40 PRINT w$(n TO k)
 50 NEXT n
 60 STOP
```

oto wyniki dla przykładowych danych

```
konstantynopolitanka
onstantynopolitanka
nstantynopolitanka
stantynopolitanka
tantynopolitanka
antynopolitanka
ntynopolitanka
tynopolitanka
ynopolitanka
nopolitanka
opolitanka
politanka
olitanka
litanka
itanka
tanka
anka
nka
ka
a
```

```
 10 REM Przyklad 8.3.1
 20 LET p$="to jest bajka"
 25 LET m=LEN (p$)
 30 FOR n=1 TO m
 40 PRINT p$(n TO m),p$(m-n+1
TO m)
 50 NEXT n
 60 STOP
```

oto wyniki dla przykładowych danych

```
to jest bajka    a
o jest bajka     ka
 jest bajka      jka
jest bajka       ajka
est bajka        bajka
st bajka          bajka
t bajka          t bajka
 bajka           st bajka
bajka            est bajka
ajka             jest bajka
jka               jest bajka
ka               o jest bajka
a                to jest bajka
```

Przykład 8.5.

a) Dany jest program:

```
  5 REM Przyklad 8.5
 10 LET x=354710
 20 LET x$=STR$ (x)
 30 PRINT x,x$
 50 PRINT 'x$(2 TO 4)
 60 PRINT 'x$(1 TO 3)
 70 LET a$=x$(4 TO )
 80 LET b=VAL (a$)
 90 PRINT 'a$,b
100 LET c$=x$(3)
110 LET d=VAL (c$)
120 PRINT 'c$,d
130 LET y=VAL ("15*3")
140 LET z=VAL ("15"+"*3")
150 PRINT y,z
160 LET w$=VAL$ ("""to jest tek
st""")
170 LET q$=VAL$ ("""i to tez"""
)
180 PRINT 'w$,q$
190 LET p$="inny tekst"
200 LET r$="i jeszcze inny"
210 PRINT 'VAL$ ("p$"),VAL$ ("r
$")
220 STOP
```

b) wyniki:

```
354710          354710

547

354

710             710

4               4
45              45

to jest tekst   i to tez

inny tekst      i jeszcze inny
```

W powyższym programie (przykład 8.5) zastosowano inne funkcje standardowe.

Funkcja STR$ zmienia liczbę na tekst; jej argumentem jest liczba, a wartością (wynikiem) tekst, innymi słowy przekształca liczbę do postaci tekstowej.

Umożliwia ona stosowanie instrukcji wybierania (TO) w celu wydzielenia poszczególnych cyfr liczby.

Instrukcje (przykład 8.5) o numerach 10 ÷ 70 ilustrują wykonanie funkcji STR$.

I tak wykonanie instrukcji.

10 LET x = 354710
20 LET x$ = STR$ (x)

da taki sam rezultat jak wykonanie instrukcji

25 LET x$ = 354710

Funkcja VAL zamienia tekst na liczbę, jej argumentem jest tekst, a wartością (wynikiem) — wartość numeryczna tekstu (liczba).

Argumentem funkcji VAL jest wyrażenie numeryczne podane w postaci ciągu znaków. Jeżeli wyrażenie to nie przedstawia poprawnego (w języku BASIC) wyrażenia numerycznego, sygnalizowany jest błąd.

Instrukcje (przykład 8.5) o numerach 80÷120 ilustrują wykonanie funkcji VAL, gdy argumentem jest tekst będący stałą numeryczną. Natomiast instrukcje

130 LET y = VAL ("15 * 3")
140 LET z = VAL ("15" + " * 3")

ilustrują wykonanie funkcji VAL, gdy argumentem jest tekst zawierający wyrażenia numeryczne podane w postaci ciągu znaków. W wyniku wykonania tych instrukcji otrzymamy y = z = 45.

Wykonanie instr. 140 przebiega w następujący sposób:

— argument traktowany jest jako tekst, tzn. z wyrażenia tekstowego " 15 " + " * 3 " otrzymujemy tekst " 15 * 3 "

— następnie po odrzuceniu cudzysłowów i wykonaniu mnożenia oraz funkcji VAL otrzymujemy liczbę 45
— zmiennej z zostaje przypisana wartość 45

Funkcja VAL$ jest podobna do funkcji VAL, zarówno jej argument, jak i wartość (wynik!) są tekstami. Wykonanie tej funkcji (podobnie jak funkcji VAL) przebiega w dwóch krokach: najpierw argument obliczany jest jako tekst, a po odrzuceniu cudzysłowów, cokolwiek by to było, jest traktowane jako inny tekst

Argument funkcji VAL$ jest tekstem, a więc musi być napisany w cudzysłowach (" ") Stąd też w instrukcjach 160, 170 mamy potrójne cudzysłowy np.: VAL$ (" " " (to jest tekst) " " ")

```
VAL$ ( " " (to jest tekst) " " )
       |                  |
       |   ______ (•) ____|
       |                    |
       |___  ..             |
```

(•) — to są cudzysłowy ograniczające stałą tekstową: to jest tekst
— to są cudzysłowy argumentu funkcji VAL$

Jeżeli wewnątrz cudzysłowu określającego stałą tekstową chcę napisać tekst w cudzysłowach, to te cudzysłowy muszę napisać podwójnie itd. np. wykonując instrukcję

PRINT """przykład"""

otrzymamy napis "przykład"
Natomiast w instrukcji postaci

210 PRINT VAL$ (" p$ "), VAL$ (" r$ ")

występujące cudzysłowy są cudzysłowami argumentu funkcji VAL$, którym jest zmienna alfanumeryczna

Litery, cyfry znaki przestankowe itd pojawiające się w tekstach ogólnie zwiemy znakami. Tworzą one alfabet, jest to zestaw znaków, którymi ZX Spectrum może się posługiwać. Większość z tych znaków to znaki pojedyncze, lecz istnieją także symbole złożone jak INPUT, FOR, STOP, <> itd

Istnieje 256 znaków i każdemu z nich przypisana jest liczba zwana kodem, z przedziału 0 do 255. Kompletna ich lista zawarta jest w dodatku A podręcznika ZX Spectrum BASIC programming

Do zamiany znaku na kod i z powrotem używane są dwie funkcje CODE i CHR$

Funkcja CODE podaje kod pierwszego znaku danego tekstu, jej argumentem jest tekst, a wartością (wynikiem) liczba będąca kodem pierwszego znaku (jeżeli tekst jest pusty, to wynik równa się zero (0)).

Funkcja CHR$ podaje znak odpowiadający danemu kodowi, jej argumentem jest liczba z zakresu < 0, 255 > (kod), a wartością (wynikiem) jest znak, którego kodem jest dana liczba

Poniższy program wypisuje zestaw znaków o kodach od 32 do 164

Przykład 8.6.

a) Dany jest program

```
10 REM Przykład 8.6
20 FOR i=32 TO 164 STEP 2
30 PRINT i;"  ";CHR$ (i),i+1;"
",CHR$ (i+1)
40 NEXT i
50 STOP
```

b) przykładowe wyniki dla i = 32, 33... 75

```
32                33   !
34   "            35   #
36   $            37   %
38   &            39   '
40   (            41   )
42   *            43   +
44   ,            45   -
46   .            47   /
48   0            49   1
50   2            51   3
52   4            53   5
54   6            55   7
56   8            57   9
58   :            59   ;
60   <            61   =
62   >            63   ?
64   @            65   A
66   B            67   C
68   D            69   E
70   F            71   G
72   H            73   I
74   J            75   K
```

Na wynikach widzimy liczby dziesiętne, a obok odpowiadający im znak. Są to spacja, 15 symboli i znaków przestankowych, dziesięć cyfr, siedem dalszych symboli, duże litery, sześć symboli, małe litery oraz pięć symboli. Wszystkie (oprócz £ i ©) należą do powszechnie stosowanego zestawu znaków graficznych znanego jako ASCII (**A**merican **S**tandard **C**odes for **I**nformation **I**nterchange — amerykańskie kody standardowe do wymiany informacji). ASCII określa również kody tych znaków i one właśnie stosowane są w ZX Spectrum.

Pozostałe znaki nie są częścią ASCII i są specyficzne dla ZX Spectrum. Jest to 16 znaków symboli graficznych („grafiki"), które mogą być używane do rysowania obrazków (por. wykład. 7 i 8) oraz 21 symboli, które wyglądają na kopię liter alfabetu od A do V Są to symbole, które możemy zdefiniować — grafiki użytkownika

Zastępując instr. 20 w przykładzie 8 6 instrukcją

20 FOR i = 165 TO 255 STEP 2

(po wykonaniu programu) otrzymamy symbole złożone ZX Spectrum wraz z odpowiadającymi im kodami. Pierwsze 32 znaki o kodach od 0 do 31 określają znaki kontrolne. Nie dają one niczego, co dałoby się wypisać, lecz tworzą pewien mniej uchwytny efekt na ekranie, albo są używane do kontrolowania czegoś poza ekranem, a na ekranie zobaczymy znak ? napisany na znak, że ich komputer nie rozumie

Trzy z tych, które używane są przez ekran, mają kody 6, 8, 12 CHR$ 16 pisze spacje na tej samej zasadzie, co przecinek w instrukcji PRINT, np..

PRINT 10; CHR$ (6); 15

da taki sam efekt co

PRINT 10, 15

Taki sam efekt otrzymamy, gdy wykonamy poniższe instrukcje

```
10 LET a$ = "10" + CHR$ (6) + "15"
20 PRINT a$
```

CHR$ (8) to „bachspace" — cofa pozycję, w której będzie wydruk następnego znaku o jedno miejsce, np

PRINT "12345"; CHR$ (8); "6"

da w wyniku napis

12346

CHR$ (13) to „newline" powoduje, że następny tekst będzie pisany od nowej linii, np.

PRINT CHR$ (13); "alfa"

da taki sam efekt co

PRINT '"alfa"

Korzystanie ze znaków (INK, PAPER, FLASCH, BRIGHT, INVERSE, OVER, AT, TAB) o kodach od 13 do 23 zostało omówione w poprzednich wykładach

Tak więc każdemu znakowi występującemu w tekstach przyporządkowana jest stała wartość numeryczna reprezentująca go w pamięci maszyny. Możemy więc rozszerzyć pojęcie kolejności alfabetycznej, tak aby objęło wszystkie znaki, nie tylko litery, zwiększając alfabet do 256 znaków, ułożonych w kolejności wartości ich kodów

Np. poniższe zdania są w porządku alfabetycznym:

CHR$ 3 + "KOWALSKI"
CHR$ 8 + "ABACKI"
"AAABBB"
"(Park)"
"201"
"ZOO"
"zoologia"

Sprawdzenie, który z podanych dwóch tekstów jest pierwszy (w kolejności alfabetycznej) przebiega w następujący sposób: porównujemy kolejno parami znaki z obu tekstów od strony lewej do

prawej. Jeżeli porównane znaki są różne, to jeden z nich ma kod mniejszy od drugiego i tekst rozpoczynający się tym znakiem jest wcześniejszy, (mówimy, że jest mniejszy). Jeżeli są takie same, to porównujemy następne znaki. Tak więc określenia relacji między tekstami następuje po znalezieniu pierwszej pary nieidentycznych znaków. Jeżeli w porównywanych znakach nie pojawi się różnica w kodach kolejnych znaków, ale jeden z tekstów jest krótszy, to jest on wcześniejszy. Przy takiej samej długości są one równe.

Operatory relacji =, < , >, <=, >= oraz <> mogą być użyte do tekstów tak samo jak do liczb (np. < oznacza „jest przed", a > — „jest po"). Wynikiem obliczenia jest jedna z dwóch wartości — 1 oznaczającej wartość prawda lub 0 oznaczającą fałsz.

Dla porównania testów rozszerzoną warunkową instrukcją sterującą o testowanie wyniku relacji. Postać i wykonanie tej instrukcji jest analogiczne do instrukcji IF... THEN... Tak więc powyższa instrukcja umożliwia wyszukiwanie i analizę tekstów.

Poniższe programy obrazują wykonanie funkcji CODE oraz instrukcji IF... THEN..

Przykład 8.7.

a) Dany jest program:

```
  10 REM Przyklad 8.7
  15 PRINT  "tekst";"           ";"ko
d pierwszego znaku": PRINT
  20 FOR i=1 TO 12
  30 READ a$
  40 PRINT a$,CODE (a$)
  50 NEXT i
  60 STOP
  70 DATA "ALFA","BETA","C","D",
"EWA",""," ","alfa","beta","c","
d","ewa"
```

b) wyniki

```
tekst        kod pierwszego znaku

ALFA              65
BETA              66
C                 67
D                 68
EWA               69
                  0
                  32
alfa              97
beta              98
c                 99
d                 100
ewa               101
```

Przykład 8.8.

a) Dany jest przykład

```
  10 REM Przyklad 8.8
  20 PRINT "ustawianie dwoch tek
stow          w kolejnosci alfabet
ycznej"
  30 PRINT '"teksty dane :"
  40 INPUT "wprowadz dwa teksty
: ",x$,y$
  50 PRINT 'x$;"    ";y$
  60 IF x$>y$ THEN LET z$=x$:
     LET x$=y$: LET y$=z$
  70 PRINT '"teksty po porownani
u :"
  80 PRINT 'x$;" ";
  90 IF x$<y$ THEN PRINT "<";: G
O TO 110
 100 PRINT "=";
 110 PRINT " ";y$
 120 STOP
```

b) Wyniki dla przykładowych danych:

```
ustawianie dwoch tekstow
w kolejnosci alfabetycznej

teksty dane :

katarynka   katar

teksty po porownaniu :

katar < katarynka
```

Przykład 8.9.

Napisać program znajdowania w zestawie DATA nazwisk o zadanym z klawiatury imieniu.

a) program

```
  10 REM Przyklad 8.9
  20 PRINT "  ***  wyszukiwanie
nazwisk   ***"
  30 INPUT "podaj imie : ";a$
  40 PRINT '"podane imie : ";a$
  50 READ n
  60 LET k=0
  70 FOR i=1 TO n
  80 READ b$,c$
  90 IF b$<>a$ THEN GO TO 120
 100 PRINT '"znalezione nazwisko
:"'"          ";c$
 110 LET k=k+1
 120 NEXT i
 130 IF k=0 THEN PRINT "w zestaw
ie danych nie bylo       nazwisk
o podanym imieniu": STOP
 140 PRINT '"w zestawie danych b
ylo ";k'"nazwisk(a)";
 150 PRINT " o imieniu ";a$
 160 STOP
 170 REM n-liczba par (imie,naz-
         wisko)
 180 DATA 5
 190 DATA "Agata","Kowalska","Ew
a","Wisniewska","Ewa","Klobukows
ka","Irena","Polkowska","Anna","
Xksinska"
```

b) Wyniki dla przykładowych danych

```
  ***  wyszukiwanie nazwisk   ***

podane imie : Ewa

znalezione nazwisko :
     Wisniewska

znalezione nazwisko :
     Klobukowska

w zestawie danych bylo 2
nazwisk(a) o imieniu Ewa
```

Zalecamy czytelnikowi poćwiczenie z programami z przykładów 8.7, 8.8, 8.9 dla różnych danych wejściowych.

W podanych przykładach używaliśmy zmiennych prostych alfanumerycznych. Takiej zmiennej mogliśmy nadać „wartość" będącą tekstem o długości od 0 do 255 znaków. Analogicznie do zmiennych indeksowanych i tablic numerycznych wprowadzamy zmienne indeksowane i tablice alfanumeryczne.

Teksty w tablicy różnią się od zmiennych prostych alfanumerycznych (tekstowych) tym, że mają określoną długość i są dopasowane zgodnie z zasadą Prokrutesa — obcinane z prawej strony albo dopełniane spacjami do wymaganej długości. Nazwa tablicy tekstowej jest pojedynczą literą ze znakiem dolara ($) i przeciwnie do tablic numerycznych zmienna prosta i tablica tekstowa nie mogą mieć tej samej nazwy.

Deklaracja tablicy postaci np.

DIM X$ (3, 15)

rezerwuje w pamięci komputera tablicę 3 X 10 znaków, może zawierać ona np. trzy wiersze tekstu, a każdy ma 15 znaków.

X$ (1) = X$ (1, 1) X$ (1, 2) ..... X$(1, 15)

X$ (2) = X$ (2, 1) X$ (2, 2) ..... X$(2, 15)

X$ (3) = X$ (3, 1) X$ (3, 2) ..... X$(3, 15)

Jeżeli napiszemy zmienną indeksowaną postaci X$ (1) lub X$ (2) lub X$(3), to rozumiemy, że odnosi się ona do całego wiersza tablicy X$ (3, 15), (otrzymamy tekst o określonej długości (15 znaków), jeżeli natomiast napiszemy X$ (2, 8), to otrzymamy jeden znak, tzn. ósmy znak wiersza drugiego (X$ (2))

Tak więc w tablicy alfanumerycznej (tekstowej) wszystkie wiersze mają stałą określoną długość, a deklaracja tablicy DIM zawiera dodatkową liczbę (ostatnią) określającą tę długość.

Zamiast ostatniego indeksu (który z zasady opuszczamy) możemy użyć instrukcji do wybierania fragmentów tekstu np

```
10 DIM x$(3,15)
20 LET x$(2)="abcdefghijklmno"
30 LET y$=x$(2,9 TO 14)
40 LET z$=x$(2)(9 TO 14)
50 PRINT y$,z$
60 REM y$=z$="ijklmn"
```

Deklaracja tablicy postaci np

DIM a$ (6)

deklaruje nam tablicę tekstową bezwymiarową, tzn. a$ zachowuje się jak zmienna prosta alfanumeryczna, przy czym zawsze ma długość równą 6 znaków i znaki dopasowane są wg zasady Prokrutesa

Uwaga: w innych reprezentacjach BASICU tablice alfanumeryczne mogą mieć inną organizację, np. taką samą, jak tablice numeryczne, tzn. że w każdy element tablicy mogą wpisać tekst o dowolnej długości (długość — określona max. liczbą znaków tekstu dla danej reprezentacji)

Omówione własności zmiennych numerycznych oraz ich funkcje są często stosowane w programach graficznych, np. w rysowaniu wykresu funkcji. Wykorzystanie ich przy przetwarzaniu tekstów obrazują poniższe dwa przykłady

Przykład 8.10.

Zadanie do wykonania.

1. Założyć książkę telefoniczną (przykładową)
2. Na żądanie
   — wyświetlić nr telefonu dla nazwiska abonenta podanego z klawiatury,
   — wyświetlić nazwisko abonenta dla numeru telefonu podanego z klawiatury

a) program

```
  10 REM Przyklad 8.10
  20 REM n$ - nazwiska abonentow
     o dlugosci (maksymalnej)
     nazwiska 20 znakow
  25 REM t - numery telefonow
  30 INPUT "podaj liczbe wprowad
zanych        nazwisk : ";n
  35 PRINT '"Ksiazka telefoniczn
a             zawiera : ";n;" na
zwisk"
  40 DIM n$(n,20): DIM t(n): DIM
 k$(20)
  45 FOR i=1 TO n
  50 INPUT "nazwisko : ";n$(i)
  55 INPUT "nr.telefonu : ";t(i)
  60 NEXT i
  65 PRINT '"  wybieramy wedlug
:                 - nazwiska     n
acisnij - 1       - nr.telefonu  n
acisnij - 2"
  70 INPUT z
  75 IF z<>1 AND z<>2 THEN GO TO
 65
  80 IF z=2 THEN GO TO 135
  85 INPUT "podaj nazwisko : ";k
$
  90 FOR i=1 TO n
  95 IF k$=n$(i) THEN GO TO 110
 100 NEXT i
 105 PRINT '"brak nazwiska w ksi
azce": GO TO 115
 110 PRINT 'k$'"nr.tel. : ";t(i)
 115 INPUT "szukamy dalej ?(t/n)
 :";x$
 120 IF x$="T" OR x$="t" THEN GO
 TO 65
 125 PRINT '"koniec pracy"
 130 STOP
 135 INPUT "nr.telefonu = ";y
 140 FOR i=1 TO n
 145 IF y=t(i) THEN GO TO 160
 150 NEXT i
 155 PRINT '"brak nr.telefonu ";
y;" w ksiazce": GO TO 115
 160 PRINT 'n$(i),"nr.tel. : ";t
(i)
 165 GO TO 115
```

b) wyniki dla przykładowych danych

```
Ksiazka telefoniczna
zawiera : 5 nazwisk

  wybieramy wedlug :
   - nazwiska      nacisnij - 1
   - nr.telefonu   nacisnij - 2

XYZ
nr.tel. : 333333

  wybieramy wedlug :
   - nazwiska      nacisnij - 1
   - nr.telefonu   nacisnij - 2

Abacki
nr.tel. : 102200

koniec pracy
```

Przykład 8.11.

Napisać program, który wprowadza do tablicy n — tekstów o max. długości 20 znaków każdy, następnie w każdym z wprowadzonych tekstów zlicza częstotliwość występowania podanych liter (umieszczonych w DATA)

a) program

```
  10 REM Przyklad 8.11
  15 REM a$ tablica tekstow o
       dlugosci 20 znakow
  20 INPUT "podaj liczbe wprowad
zanych        tekstow : ";n
  25 PRINT "wyszukiwanie i zlicz
anie odpo-  wiednich liter w dan
ych tekstach"
  30 DIM a$(n,20)
  35 REM litery ktorych szukamy
umieszczone sa w zestawie DATA
  40 FOR i=1 TO n
  45 INPUT (i);" -tekst : ";a$(i
)
  50 NEXT i
  60 FOR i=1 TO n
  65 READ m
  70 PRINT "tekst : ";i;" :"
  75 PRINT a$(i)
  80 FOR k=1 TO m
  85 READ z$
  90 LET p=0
  95 FOR j=1 TO 20
 100 IF a$(i,j)=z$ THEN LET p=p+
1
 105 NEXT j
 110 PRINT '"w tekscie : ";i;"
litera : ";z$'"wystapila : ";p;"
 razy"
 115 NEXT k
 120 RESTORE : COPY : CLS
 125 PRINT '
 130 NEXT i
 135 STOP
 140 DATA 6
 145 DATA "a","c","o","p","x","z
"
```

C-64

Test czytania

Po uruchomieniu programu na ekranie pojawia się na krótko wyraz, który należy następnie napisać. Jeżeli wyrazy pisane są poprawnie, to oczywiście przybywa nam punktów w tej rozgrywce z komputerem. Czas trwania wyrazu na ekranie można zmniejszyć lub zwiększyć, zmieniając długość pętli w linii 170. Może być to zatem pożyteczna zabawa dla przedszkolaków i dla bardziej zaawansowanych. Oczywiście, w programie mogą być użyte inne wyrazy.

```
10 REM TEST CZYTANIA
27 POKE 5 ,0,13:POKE 5 80,1
30 AT$=CHR$(17):FOR T=1TO6:AT$=AT$+AT$:N
EXT T:AT$=CHR$(19)+AT$
35 GOSUB 10000
40 SC=0
60 HS=0
70 DIMW$(50)
80 FOR J=1TO50
90 READW$(J)
100 NEXT J
110 FOR N=1TO 0
120 R=INT(RND(1)*50)+1
130 PRINT CHR$(147)
140 PRINTLEFT$(AT$,2)SPC(4)'WYNIK: ';SC
150 PRINTLEFT$(AT$,2)SPC(18)'NAJWYZSZY W
YNIK: ';HS
160 PRINTLEFT$(AT$,8)SPC(18-(LEN(W$(R))/
2))W$(R):WW$=W$(R)
165 IF LEN(W$(R))<10 THEN W$(R)=W$(R)+'
':GOTO 165
170 FOR DL=1TO500 :NEXT DL

180 PRINTLEFT$(AT$,8)SPC(9)'
                         '
190 PRINTLEFT$(AT$,16)SPC(9)'JAKIE TO BY
LO SLOWO  '
195 PRINTLEFT$(AT$,18)SPC(9);
200 INPUT A$
202 IF LEN(A$)<10 THEN A$=A$+' ':GOTO 20
2
210 IF A$=W$(R) THEN 270
220 POKE S0+1,20:POKE S0+4,17:FOR QQ=1TO
30:NEXT QQ:POKE S0+4,16
230 PRINTLEFT$(AT$,8)SPC(11)'TYM SLOWEM
BYLO: '
235 PRINTLEFT$(AT$,10)SPC(17-(LEN(WW$)/2
))WW$
244 PRINTLEFT$(AT$,16)SPC(3)'
                         '
245 PRINTLEFT$(AT$,18)SPC(3)'
                         '
250 FOR DL=1TO 400:NEXT DL
260 GOTO 300

270 POKE S0+1,  :POKE S0+4,17:FOR QQ=1TO
30:NEXT QQ:POKE S0+4,16
280 SC=SC+10
300 NEXTN
310 IF SC>HS THEN HS=SC
312 PRINTCHR$(147)
314 PRINTLEFT$(AT$,8)SPC(3)'TWOJ KONCOWY
 WYNIK........'SC
316 IF SC >HS THEN PRINTLEFT$(AT$,10)SPC
(3)'TO JEST NAJLEPSZY WYNIK DZISIAJ.'
320 INPUT'     JESZCZ RAZ (T/N)';Q$
330 IF LEFT$(Q$,1)='N' THEN 380
340 SC=0
360 GOTO110
380 END
400 DATA 'OGROD','CHLOPIEC','DZIEWCZYNA'

410 DATA 'STRAWA','OGAREK','KOBIETA'
420 DATA 'LAMPA','PIES','KOT'
430 DATA 'OKNO','DRZWI','SCIANA'

440 DATA 'DZIS','JUTRO','WCZORAJ'
450 DATA 'KWIAT', 'ROLA', 'WAZA'
460 DATA 'LAS',,'TRAWA','BZOSA'
470 DATA 'KASIA','OBRAZ','SZARY'
480 DATA 'SZYMPANS','KROWA','ZEBRA'
490 DATA 'KOZA','KOGUT','KACZKA'
500 DATA 'MONIKA','MARCIN','DANUTA'
510 DATA 'ADAM','TOMEK','ZOSIA'
520 DATA 'PIERWSZY', DRUGI', 'TRZECI'
530 DATA 'OSTATNI','NAJLEPSZY','DOBRY'
540 DATA 'PAPIER','ALA','AS'
550 DATA 'WINO','SZKLANKA','WODA'
560 DATA 'LITERA','SYLABA'
10000 REM  DZWIEK
10010 S0=54272
10020 FORT=0TO28:POKES0+T,0:NEXT
10030 POKES0+1,20
10040 POKE S0+5,9
10050 POKE S0+6,240
10060 POKE S0+24,15
10070 POKE S0+4,16
10080 RETURN
```

T. CISEK

PROGRAM 40 41 ATARI

Histogram

Program służy do otrzymywania histogramów. Po jego uruchomieniu otrzymamy pytanie o liczbę słupków (maksymalnie 9). Gdy komputer uzyska odpowiedź, zostanie uruchomione przyjmowanie danych o poszczególnych słupkach. Kolejnym etapem jest kreślenie histogramu. Pod każdym słupkiem pojawia się kolejny numer; wartość danej w zaokrągleniu (liczby większe od stu są zredukowane do dwóch pierwszych cyfr); oraz procentową zawartość danej w stosunku do sumy wszystkich składników histogramu.

Ostatni słupek odwzorowuje sumę składników.

Największa podana wartość słupka uzyskuje pełną wysokość ekranu, pozostałe są jej procentowym odwzorowaniem.

Naciśnięcie klawisza START powoduje ponowne uruchomienie programu.

Artur NOWAKOWSKI

```
AP 10 REM *****  HISTOGRAM  *****
MR 20 REM *** wersja 1.0 POPR.***
ZM 30 REM **** (c) 1987-08-11****
KC 40 REM ** Artur Nowakowski  **
AO 50 REM ul. W. Wasilewskiej 5/8
GF 60 REM 08-110 Siedlce
LF 70 REM POLSKA
VG 80 REM /tel. 209-02/
AK 90 DIM A$(1),B$(9),ILOSC(9)
GN 100 REM STRONA TYTULOWA
SW 110 GRAPHICS 1
BX 120 B$="histogram"
VK 130 FOR X=1 TO 9
VJ 140 A$=B$(X,X) WYS=INT(RND 0)*13
   +1)
AA 150 FOR Y=1 TO WYS
HL 160 POSITION 2*X,14-Y:? #6;A$
MJ 170 NEXT Y
MB 180 NEXT X
CK 190 POSITION 0,14:? #6;"     WER
   SJA 1.0"
SW 200 ? #6;"   (C) 1987-08-11"
TQ 210 ? #6;"  artur nowakowski"
CJ 220 ? #6;" W.WASILEWSKIEJ 5/8"
UD 230 ? #6;"    08-110 SIEDLCE"
JA 240 ? #6;"        POLSKA"
RT 250 IF SPR=1 THEN GOTO 300
MV 260 ? "            ";CHR$(145);CHR
   $(146);CHR$(146);CHR$(146);CHR$(
   146);CHR$(146);CHR$(133)
IB 270 ? "Nacisnij  ";CHR$(252);CHR
   $(211);CHR$(212);CHR$(193);CHR$(
   210);CHR$(212);CHR$(252)
GG 280 ? "            ";CHR$(154);CHR
   $(146);CHR$(146);CHR$(146);CHR$(
   146);CHR$(146);CHR$(131)
PW 290 IF PEEK(53279)<>6 THEN POKE
   755,0 GOTO 290
OR 300 REM PODAWANIE DANYCH
WQ 310 POKE 755,0:? CHR$(125);"Ile
   slupkow (1-9) ",
KQ 320 TRAP 300
HL 330 INPUT SKLAD
PO 340 IF SKLAD>9 OR SKLAD<1 THEN G
   OTO 300
AC 350 SUM=0:REM *****************
PA 360 ? CHR$(125);"Podawaj kolejno
   dane.":? :?
PO 370 FOR X=1 TO SKLAD:TRAP 870
IU 380 ? CHR$(28);"
          "
XH 390 ? CHR$(28);"       ";X;" - ";
   INPUT ILOSC:IF ILOSC<0 THEN GOT
   O 880
TF 400 ILOSC(X)=ILOSC:SUM=SUM+ILOSC
   (X)
LA 410 NEXT X
UK 420 REM SZUKANIE NAJWIEKSZEJ DAN
   EJ

EZ 430 FOR X=1 TO SKLAD
FK 440 FOR Y=1 TO SKLAD
MV 450 IF ILOSC(X)>=ILOSC(Y) THEN N
   EXT Y:MAX=ILOSC(X):GOTO 470
MA 460 NEXT X
RG 470 REM RYSOWANIE SLUPKOW
TL 480 IF MAX<=0 THEN SPR=1:GOTO 10
   0
BV 490 IF MAX>=100 THEN SUM=0:FOR X
   =1 TO SKLAD:ILOSC(X)=ILOSC(X)/10
   :SUM=SUM+ILOSC(X):NEXT X:GOTO 42
   0
UA 500 GRAPHICS 3
KH 510 POKE 755,0 .
CI 520 RESTORE
FA 530 FOR X=1 TO SKLAD
SD 540 ? X;"  ";:NEXT X
OC 550 ? " SUMA"
FG 560 FOR X=1 TO SKLAD
YF 570 A$=""
IN 580 IF ILOSC(X)<10 THEN A$=" "
ND 590 ? INT(ILOSC(X));" ";A$;:NEXT
   X
CU 600 ? "-ILOSC"
EX 610 FOR X=1 TO SKLAD
UD 620 WYS=INT(ILOSC(X)*20/MAX)
PT 630 A$="":IF ILOSC(X)*100/SUM<10
    THEN A$=" "
ZI 640 ? INT(ILOSC(X)*100/SUM);" ";
   A$;
KR 650 IF ILOSC(X)*20/MAX<0.5 AND X
   <SKLAD THEN NEXT X
HI 660 IF ILOSC(X)*20/MAX<0.5 THEN
   GOTO 720
IF 670 READ A$
AL 680 FOR Y=1 TO WYS
PP 690 POSITION 3*X-1,20-Y:? #6;A$
MB 700 NEXT Y
LT 710 NEXT X
KB 720 REM OSTATNIE SLUPKI
OC 730 KOL=INT(SUM/MAX):WYS=INT(SUM
   *20/MAX)-20*KOL
UL 740 READ A$:? " 100 %"
RA 750 IF KOL<1 THEN GOTO 800
8D 760 FOR X=4 TO KOL+3
XJ 770 FOR Y=1 TO 20
JJ 780 POSITION 3*SKLAD+X-1,20-Y:?
   #6;A$
UW 790 NEXT Y:NEXT X
OJ 800 FOR X=0 TO WYS:POSITION 3*SK
   LAD+KOL+3,20-X
SP 810 ? #6;A$
LW 820 NEXT X
TV 830 ? "           Nacisnij ";CHR$(
   160);CHR$(211);CHR$(212);CHR$(19
   3);CHR$(210);CHR$(212);CHR$(160)
   ;
DG 840 IF PEEK(53279)<>6 THEN 840
FN 850 SPR=1:GOTO 100
CM 860 REM KONIEC PROGRAMU


JD 870 REM OBSLUGA BLEDU
HZ 880 ? CHR$(253);CHR$(29);"  Zla
   dana! - popraw"
OF 886 FOR Y=1 TO 1000:NEXT Y:? CHR
   $(28);"                       "
DP 888 ? CHR$(28);CHR$(28);CHR$(28)
   ;"              ":X=X-1
MK 890 NEXT X
BV 900 REM DANE KOLOROW
LH 910 DATA A,B,C,A,B,C,A,B,C,A
```

Giełda pomysłów

LICZBY

Przedstawiony program wykonuje cztery podstawowe działania na liczbach naturalnych (plus 0) o maksymalnej długości 510 cyfr

W celu jego uruchomienia należy wpisać do komputera **program nr 2** (kod maszynowy) i wykonać zleceniem RUN. W wyniku otrzymasz dwa pliki binarne (kod 1, kod 2), które zostaną umieszczone na dyskietce

Teraz skasuj pamięć komputera i wprowadź **program nr 2**, po czym nagraj go komendą SAVE „LICZ", na tę samą dyskietkę, na której są pliki kod 1 i kod 2. Program jest gotowy. Uruchom go zleceniem RUN „LICZ"

Wprowadzana liczba jest na monitorze samoczynnie formatowana (tworzone są grupy składające się z trzech cyfr). Ułatwia to jej odczyt. Puste miejsca od lewej strony (jeśli takie są), można wypełnić spacjami

R. FOLTYNIEWICZ

```
10 'program +,-,$,/ na lb. o dl. do 510 cyfr
20 'real. [illegible]niewicz
30 MODE 2:DEFINT a,d,i,n,p,x,z
40 MEMORY &9FFF:LOAD "kod1":LOAD "kod2":DIM a$(510)
50 [illegible]:CLS
60 PRINT "znak '.' konczy wprowadzanie; podaj pierwszy skladnik":w=1:GOSUB 210
70 PRINT:PRINT "podaj drugi skladnik":w=0:GOSUB 210:PRINT
80 PRINT "                    (+)dodaj (-)odejmij ($)mnoz (/)dziel"
90 d$=INKEY$
100 IF d$="+" THEN PRINT d$:POKE &A001,MAX(x,z):CALL &A000:GOTO 170
110 IF d$="-" THEN PRINT d$:POKE &A01F,MAX(x,z):CALL &A01E:GOTO 170
120 IF d$="$" THEN IF x+z>255 THEN PRINT"za duze liczby !":GOTO 60
130 IF d$="$" THEN PRINT d$:POKE &A03D,x+z:CALL &A03C:GOTO 170
140 IF d$="/" THEN IF x=1 AND a$(x)="0" THEN PRINT"/0 !!":GOTO 60
150 IF d$="/" THEN PRINT "/":POKE &A314,z:CALL &A313:GOTO 170
160 GOTO 90
170 GOSUB 550
180 PRINT "LICZYSZ DALEJ (t/n)":INPUT o$
190 IF UPPER$(o$)="T" THEN 50
200 END
210 'wprowadzanie liczby
220 CURSOR 1,1:d=1:r=0
230 FOR a=1 TO 510
240 b$=INKEY$:IF b$="" THEN b$="?"
250 IF INKEY(8)<>-1 THEN 260 ELSE 300
260 GOSUB 670
270 IF a<=1 THEN SOUND 1,100,1,12:GOTO 300
280 IF ((a+r) MOD 3)=1 THEN PRINT CHR$(8);
290 PRINT CHR$(8);:a=a-1
300 IF INKEY(1)<>-1 THEN 310 ELSE 350
310 GOSUB 670
320 IF a>=d THEN SOUND 1,100,1,12:GOTO 350
330 IF ((a+r) MOD 3)=0 THEN PRINT CHR$(9);
340 PRINT CHR$(9);:a=a+1
350 IF INKEY(79)<>-1 THEN 360 ELSE 400
360 GOSUB 670
370 IF (a<>d OR d<=1) THEN SOUND 1,100,1,12:GOTO 400
380 IF ((a+r) MOD 3)=1 THEN PRINT CHR$(8);
390 PRINT CHR$(8);" ";CHR$(8);:d=d-1:a=d
400 IF b$="." THEN 460
410 IF b$=" " THEN PRINT " ";:r=(r+1) MOD 3:GOTO 240
420 IF ASC(b$)<48 OR ASC(b$)>57 THEN 240
430 a$(a)=b$:PRINT b$;:IF a=d THEN d=d+1
440 IF ((a+r) MOD 3)=0 THEN PRINT " ";
450 NEXT a
460 IF w=1 THEN adres=&9000 ELSE adres=&9200
470 adresa=adres
480 FOR i=d-1 TO 1 STEP -2
490 POKE adres,VAL("&"+a$(i-1)+a$(i)):adres=adres+1
500 NEXT i
510 FOR j=adres TO adresa+255:POKE j,0:NEXT j
520 pl=(d-1)/2:rz=pl-FIX(pl):IF rz=0 THEN p=FIX(pl) ELSE p=FIX(pl)+1
530 IF w=1 THEN z=p ELSE x=p
540 RETURN
550 'wydruk wyniku
560 baza=&[illegible]
570 FOR i=255 TO 0 STEP -1
580 adres=i+baza
590 IF PEEK(adres)=0 THEN NEXT i
600 FOR n=i TO 0 STEP -1:adres=n+baza
610 s$=HEX$(PEEK(adres) AND &F):t$=HEX$((PEEK(adres) AND &F0)/16)
620 IF n=i AND t$="0" THEN t$=" "
630 IF (n MOD 3)=2 THEN PRINT t$+s$;ELSE GOSUB 680
640 NEXT n
650 IF i=-1 THEN PRINT "0"
660 PRINT:RETURN
670 FOR n=1 TO 80:NEXT n:RETURN
680 IF (n MOD 3)=0 THEN PRINT t$+s$+" ";ELSE PRINT t$+" "+s$;
690 RETURN
```

```
100 REM kod maszynowy
110 MEMORY &9FFF:adres=&A000
115 ilosclinii=26
120 RESTORE 310
130 FOR i=1 TO ilosclinii
140 suma=0:READ cc$,ww$
150 FOR j=1 TO 39 STEP 2
160 bb=VAL("&"+MID$(cc$,j,2))
170 POKE adres,bb
180 suma=suma+bb:adres=adres+1
190 NEXT j
200 IF suma<>VAL("&"+ww$) THEN PRINT "Blad w linni nr ";300+i*10:STOP
205 IF adres=&A118 THEN adres=&A313
210 NEXT i
220 SAVE "kod1",b,&A000,280:SAVE "kod2",b,&A313,240
300 REM dane
310 DATA 0600210090110092 78a7c81a8e2777231305c20b,58f
320 DATA a0c9000000000000000000600210090110092 78b7,3f2
330 DATA c83feb3e99ce00964f1a812712231305c22ba0c9,7e1
340 DATA 0600210090110092cd48a0c978b7c8320ba1320d,6ec
350 DATA a1220fa1eb2211a12114a22213a1eb0e007e1271,6d9
360 DATA 132305c261a02115a13a0ba1477123 05c271a03e,6ac
370 DATA 01320ca197320ea13a0ca1b72a13a17ec28fa00f,752
380 DATA 0f0f0fe60fcab5a04f2a11a11115a13a0ba147b7,717
390 DATA 1a8e2712231305c2e0a0d2b1a0210ea1340dc295,7a9
400 DATA a03a15a1e60f473a0ca1b778c2c7a00f0f0f0f2a,771
410 DATA 0fa1b6773a0ba1475f16002115a1193a0ea1572b,5da
420 DATA 7e4feb6f0b20f0f0f0f7779e60f5705c2dba0210c,83c
430 DATA a135ca7ca02a13a1232213a12a0fa123220fa121,683
440 DATA 0da135c277a0c900000000000000000000000000,385
450 DATA 0600210090110092cd1fa3c922f8a3eb22faa378,891
460 DATA 32f5a3b7cad1a32101a522fea3eb2102a422fca3,abc
470 DATA 977712231305c23ca33af5a36f260029232200a4,675
480 DATA 2afaa33af5a34797b62305c257a3b7cad3a32afa,b2c
490 DATA a33af5a3473e9996772305c268a39732f6a32af8,a19
500 DATA a3cdd5a32a00a42b2200a47cb5cad1a32afca3cd,aac
510 DATA d5a39732f6a32afca3eb2afea34d442afaa33af5,be0
520 DATA a332f7a3371a8e27022313033af7a33d32f7a3c2,84f
530 DATA a4a3d275a33af6a3c61032f6a32afca3eb2afea3,c24
540 DATA 22fca3eb22fea3c391a3b7c937c93af5a3473af6,c2f
550 DATA a35f7e57e60fb30f0f0f0f777aebf05f2305c2dd,8a8
560 DATA a37b32f6a3c900000000000000000000000000000,3b2
```

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

COMPUTER MEMORY – pamięć komputera,
COMPUTER MICROGRAPHICS – mikrografika komputerowa,
COMPUTER NETWORK – sieć komputerowa, zespół (połączonych ze sobą) komputerów,
COMPUTER NUMERICAL CONTROL – sterowanie numeryczne za pomocą komputera,
COMPUTER ORIENTED LANGUAGE – język (programowania) zorientowany maszynowo,
COMPUTER OUTPUT MICROFILMER – rejestrator mikrofilmowy (rejestruje obraz z komputera na mikrofilmie lub mikrofiszy),
COMPUTER OUTPUT MICROFILM – dane wyjściowe z komputera w postaci mikrofilmu,
COMPUTER OUTPUT MICROFILM – dane wyjściowe na mikrofilmie,
COMPUTER PROGRAM – program wewnętrzny,
COMPUTER READABLE INFORMATION – informacja w postaci czytelnej dla komputera,
COMPUTER SCIENCE – informatyka,
COMPUTER SCIENTIST – informatyk,
COMPUTER SERVICES CENTRE – centrum komputerowych usług obliczeniowych,
COMPUTER STORE – pamięć komputera,
COMPUTER SYSTEM – zestaw komputera,
COMPUTER THEORY – informatyka,
COMPUTER WORD – słowo komputerowe,
COMPUTER-BASED INFORMATION SYSTEM – skomputeryzowany system informacyjny,
COMPUTERIZATION – komputeryzacja,
COMPUTERIZED – skomputeryzowany,
COMPUTERIZED MANAGEMENT INFORMATION SYSTEM – system informatyczny zarządzania,
COMPUTING CENTER – ośrodek obliczeniowy,
CONCATENATE OPERATOR – operator sklejania, operator spinania, operator konkatenacji,
CONCATENATION – scalanie (łączenie ciągów znaków za pomocą operatora),
CONCATENATION – wiązanie łańcuchowe,
CONCENTRATOR – koncentrator,
CONCEPTUAL – pojęciowy,
CONCEPTUAL DICTIONARY – słownik pojęć,
CONCLUDE – wnioskować, także: zakończyć, sfinalizować,
CONCLUSION – wniosek,
CONCURRENCY – zbieżność,
CONCURRENT – równoczesny, współdziałający, zbieżny,
CONCURRENT PROCESSES – procesy współbieżne,
CONCURRENT PROCESSING – wieloprogramowość,
CONCURRENT USER STACKS – stosy użytkowników (w systemie współbieżnym),
CONDENSATION – skrócenie (np. programu), skondensowanie,
CONDENSING ROUTINE – procedura zagęszczania,
CONDITION – warunek, także: stan, okoliczność,
CONDITION BOX – figura na schemacie blokowym symbolizująca warunek,
CONDITION CODE REGISTER – rejestr wskaźników,
CONDITION NAME – nazwa warunku,
CONDITIONAL – warunkowy,
CONDITIONAL BRANCH – rozgałęzienie warunkowe,
CONDITIONAL FIELD – pole warunków,
CONDITIONAL INSTRUCTION – rozkaz warunkowy,
CONDITIONAL JUMP – skok warunkowy,
CONDITIONAL JUMP INSTRUCTION – rozkaz skoku warunkowego,
CONDITIONALLY – warunkowo,
CONDUCT – przewodzić (np. prąd elektryczny), także: wykonywać (np. działania arytmetyczne),
CONDUCTOR TRACK – ścieżka przewodząca,
CONFERENCE ON DATA LANGUAGE – język programowania dostosowany do potrzeb zarządzania,
CONFIGURATION – dobór, zestaw, komplet, konfiguracja, także: układ, ukształtowanie, kombinacja,
CONFORMITY CERTIFICATE – atest,
CONFUSION – zamieszanie, chaos,
CONJUGATE – sprzężony,
CONJUNCTION CIRCUIT – układ koniunkcji,
CONNECT TOGETHER – łączyć ze sobą razem,
CONNECTING LINK – element łączący,
CONNECTION – połączenie, także: związek (pomiędzy),
CONNECTION LAYOUT – schemat montażu,
CONNECTION SET-UP – nawiązywanie połączenia,
CONNECTIVE – łącznik (logiczny), element łączący,
CONNECTIVE LINE – linia łącząca,
CONNECTOR – łącznik, wtyk, także: symbol na schemacie blokowym, odsyłacz na schemacie blokowym,
CONNOTATION OF A CONCEPT – treść pojęcia,
CONSECUTIVE – kolejny, następny,
CONSECUTIVE COMPUTER – komputer sekwencyjny,
CONSECUTIVELY – kolejno,
CONSEQUENT DRIVEN – wnioskowanie wsteczne (w systemach ekspertowych),
CONSIDERABLE DIFFERENCE – znaczna różnica, istotna różnica,
CONSISTENCE CHECK – sprawdzian logicznej zgodności,
CONSOLE – konsola,
CONSOLE COMMAND PROCESSOR – moduł, który interpretuje i wykonuje polecenia wprowadzane przez użytkownika za pomocą klawiatury,
CONSOLE MODE – bezpośredni tryb pracy "z konsoli",
CONSOLIDATION – scalanie, łączenie, konsolidacja,
CONSOLIDATOR – program scalania (łączenia), konsolidator,
CONSTANT – stała,
CONSTANT AREA – część pamięci zawierająca dane stałe,
CONSTITUENT – element, składnik,
CONSTRAINT – ograniczenie, warunek ograniczający,
CONTACT – styk, zestyk,
CONTENT – zawartość, także: stan licznika, rejestru,
CONTENT ADDRESSABLE MEMORY – pamięć adresowana zawartością, pamięć asocjacyjna, pamięć skojarzeniowa,
CONTENTS OF THE COUNTER – zawartość licznika, stan licznika,
CONTEXT-FREE LANGUAGE – język bezkontekstowy,
CONTIGUOUS – przyległy, sąsiedni,
CONTINUOUS – ciągły, nieprzerwany,
CONTINUOUS DRAWING – rysunek ciągły (elementy składowe są opisane współrzędnymi względnymi z wyjątkiem punktu początkowego),
CONTINUOUS FORM – papier ciągły, formularz ciągły,
CONTINUOUS FORM SET – zestaw formularzy ciągłych,
CONTINUOUS STATIONARY – papier rolowy do drukarek,
CONTINUOUS TAPE – pętla taśmy dziurkowanej,
CONTRACTING A FILE – zmniejszanie pliku (zbioru),
CONTRAST – kontrast,
CONTRAST WITH ... – przeciwstawiać (coś czemuś),
CONTROL – sterowanie, kierowanie, także: kontrola, regulowanie,
CONTROL AND READ-ONLY MEMORY – pamięć stała ze sterowaniem,
CONTROL AND SIMULATION LANGUAGE – symulacyjny język programowania,
CONTROL AREA – programowy obszar sterowania pliku (zbioru),
CONTROL BALL – manipulator kulowy,
CONTROL BLOCK – blok sterujący,
CONTROL BOARD – pulpit sterowniczy,
CONTROL BRUSH – szczotka sterująca,
CONTROL CHANNEL – kanał sterujący,
CONTROL CHARAKTER – znak sterujący,
CONTROL COMMAND – rozkaz sterujący,
CONTROL CONSOLE – pulpit sterowania,
CONTROL DEVICE – urządzenie sterujące,
CONTROL INFORMATION – informacja sterująca, rozkaz sterujący,
CONTROL JET – strumień sterujący,
CONTROL KEY – klucz sterujący, klawisz sterujący,
CONTROL LOOP – pętla sterowania, pętla regulacji,
CONTROL MEMORY – pamięć sterująca,
CONTROL MODE – tryb sterowania,
CONTROL PROGRAM FOR MICROKOMPUTERS – program sterujący dla mikrokomputerów,
CONTROL QUANTITY – wielkość sterująca,
CONTROL RECORD – rekord sterujący,
CONTROL ROUTINE – część sterująca (programu organizacyjnego),
CONTROL SIGNAL – sygnał sterujący,
CONTROL STATEMENT – instrukcja sterująca,
CONTROL SYSTEM – układ sterowania, układ regulacji,
CONTROL TOTAL – suma kontrolna,
CONTROL TRANSFER INSTRUCTION – rozkaz skoku,
CONTROL UNIT (CU) – układ sterowania,
CONTROLLABILITY – sterowność, sterowalność,
CONTROLLABLE – regulowany, sterowany,
CONTROLLED CONDITION – wielkość regulowana, parametr regulowany,
CONTROLLED PLANT – obiekt regulacji,
CONTROLLED VARIABLE – wielkość regulowana, parametr regulowany,
CONTROLLER – regulator, urządzenie sterujące, sterownik, kontroler,
CONTROLLING – sterowanie,
CONVENTIONAL – konwencjonalny, klasyczny, typowy, powszechnie stosowany,
CONVENTIONAL DATA PROCESSING – przetwarzanie danych konwencjonalne,
CONVENTIONS – symbole, umowne znaki, także: typowe procedury w programach,
CONVERGENCE – zbieżny,
CONVERSATIONAL LANGUAGE – język konwersacyjny,
CONVERSATIONAL MODE – konwersacyjny tryb współpracy,
CONVERSATIONAL OPERATION – praca konwersacyjna, praca interakcyjna,
CONVERSATIONAL STATION – stacja konwersacyjna,
CONVERSATIONAL TERMINAL – terminal konwersacyjny,
CONVERSION – konwersja, przekształcenie, także: przechodzenie z jednego systemu na drugi,
CONVERSION PROGRAM – program konwersji,
CONVERSION TIME – czas konwersji,

CONVERT - przekształcać, przeliczać, także: przetwarzać,
CONVERTER - konwerter, przemiennik, przetwornik,
CONVERTER CONTROL PROGRAM - program sterujący konwertera,
CONVERTING - przekształcanie,
CONVEY - komunikować, przenosić,
CONVEYER - przenośnik,
COORDINATE GRAPHIC - grafika konturowa,
COORDINATE INDEXING - indeksowanie koordynujące,
COORDINATE ORIGIN - początek układu współrzędnych,
COORDINATE PLOTTER - pisak X Y, kreślak,
COORDINATE SYSTEM - układ współrzędnych,
COORDINATES - współrzędne,
COPY - kopia, odpis, także: kopiować, przepisz (jako instrukcja),
COPY PROTECTION PROGRAM - program zabezpieczający przed kopiowaniem,
COPYABLE SOURCE DOCUMENT - oryginał przystosowany do powielania,
COPYING - kopiowanie,
COPYING ROUTINES - programy kopiowania,

CORD - sznur, także: przewód elektryczny,
CORE - rdzeń, rdzeń ferrytowy, także: rdzeniowa pamięć operacyjna komputera,
CORE ARRAY - matryca rdzeni,
CORE IMAGE LIBRARY - biblioteka programów przystosowanych do załadowania,
CORE LAYOUT - schemat rozdysponowania pamięci,
CORE MEMORY - pamięć rdzeniowa, pamięć magnetyczna ferrytowa,
CORE STORE - pamięć rdzeniowa, pamięć magnetyczna ferrytowa,
CORNER OUT - ścięty rożek (karty),
CORRECT - prawidłowy, poprawny, także: poprawiać, wprowadzać poprawkę,
CORRECTABLE - dający się poprawiać (sprostować),
CORRECTABLE CODE - kod umożliwiający korygowanie,
CORRECTION - poprawka, wniesienie poprawki,
CORRECTION STATEMENT - instrukcja korygowania,
CORRECTIVE MAINTAINING - usuwanie awarii w maszynie,
CORRECTOR - korektor, urządzenie prostujące,
CORRELATION - korelacja, współzależność, wzajemne powiązanie,
CORRELATION NET - siatka korelacji,
CORRESPONDENCE - odpowiedniość, także: stosowność,
CORRESPONDING - odpowiedni,
CORRUPTION - zepsucie,
COSTS PER BYTE - koszty zapamiętywania bajtu,
COUNT - liczenie, obliczanie, zliczanie, także: stan licznika, liczyć, obliczać,
COUNT-DOWN COUNTER - licznik poprzednikowy (zliczający "do tyłu"),
COUNT-UP COUNTER - licznik następnikowy (zliczający 'w przód'),
COUNTABLE - policzalny, przeliczalny,
COUNTABLY INFINITE SET - zbiór przeliczalny,
COUNTDOWN - praca licznika powodująca odejmowanie,
COUNTER - licznik, mechanizm zliczający,
COUNTER CAPACITY - pojemność licznika,
COUNTER REGISTER - rejestr liczący, rejestr zliczający,
COUNTER TIMER CIRCUIT - programowany zegar,
COUNTERCLOCKWISE - odwrotnie do ruchu wskazówek zegara,
COUNTERPART - odpowiednik,
COUNTING - liczenie, obliczanie, zliczanie,
COUNTING BOARD - liczydło,
COUNTING CIRCUIT - układ liczący,
COUPLING - sprzężenie, połączenie,
COUPLING ELEMENT - element sprzęgający,
CPM - patrz: CRITHICAL PATH METHOD,
CPU - patrz: CENTRAL PROCESSING UNIT,
CRAM - patrz: CARD RANDOM ACCESS MEMORY,
CRC - patrz: CYCLIC REDUNDANCY CHECK CHARACTER,
CREATING ADDITIONAL SYSTEM CONTROL AREA ROUTINES programy poszerzania systemowego obszaru sterowania,
CREATION DATE - data zapisu,
CRISS-CROSS - krzyżujące się (przecinające się) linie,
CRITERION - kryterium,
CRITICAL PATCH METHOD - metoda krytycznej ścieżki,
CROM - patrz: CONTROL AND READ-ONLY MEMORY,
CROSS FOOTS - działanie arytmetyczne polegające na dodawaniu i odejmowaniu,
CROSS PROTECTION - zabezpieczanie krzyżowe,
CROSS SOFTWARE - oprogramowanie skrośne,

CROSS-REFERENCE - odsyłacz,
CROSSBAR SWITCH - przełącznik krzyżowy,
CROSSCHECK - sprawdzanie poziome, sprawdzanie dwoma różnymi sposobami,
CROSSWISE - krzyżowo,
CRT - patrz: CATHODE-RAY TUBE,
CRT DISPLAY - monitor ekranowy, ekranopis,
CRT RASTER DISPLAY - monitor rastrowy,
CRYOGENIC MEMORY - pamięć kriogeniczna, pamięć nadprzewodnikowa,
CS (CODE SEGMENT) - segment programu,
CSL - patrz: CONTROL AND SIMULATION LANGUAGE,
CSMA-CD - patrz: CARRIER-SENSE MULTIPLE-ACCESS WITH COLISION-DETECTION,
CTI - patrz: COAXIAL TRANSCEIVER INTERFACE,
CU - patrz: CONTROL UNIT,
CUBE - sześcian, także: trzecia potęga,
CUMULATE - kumulować, nagromadzać,
CUMULATIVE DATA - dane zbiorowe,
CURRENCY SIGN - znak waluty,
CURRENT - prąd elektryczny, także: bieżący,
CURRENT AWARENESS SERVICE - bieżąca służba informacyjna,
CURRENT DIRECTORY - katalog bieżący (aktualnie otwarty),
CURRENT INFORMATION - informacja bieżąca, informacja ciągła,
CURRENT LINE - linia bieżąca (aktualnie zapisywana lub błędna),
CURRENT POINT - bieżący punkt odniesienia,
CURRENT POSITION - pozycja bieżąca,
CURRENT PROGRAM - program bieżący (aktualnie w pamięci),
CURRENT SEGMENT - segment bieżący (zdefiniowany ostatnią instrukcją DEF)
CURRENT VIEWPORT - wziernik bieżący (aktualnie używany),
CURSOR - kursor, wskaźnik pozycji,
CURSOR CONTROL KEYS - klawisze sterujące kursorem,
CURTATE LOWER - dolna część karty 80-kolumnowej z wierszami 1-9,
CURTATE UPPER - górna część karty 80-kolumnowej z wierszami 10,11,12,
CURVE - krzywa, krzywa charakterystyczna,
CURVE GENERATOR - generator krzywych,
CURVE PLOTTER - pisak krzywych,
CUSTOMER - nabywca,
CUSTOMIZED - także: przyjęty
CUT DOWN - redukować, zmniejszać,
CUT OFF - zatykać, odcinać,
CX (COUNT REGISTER) - rejestr zliczający (pętli),
CYAN - niebiesko-zielony,
CYBERNATION - komputeryzacja,
CYBERNETIC ENGINEERING - cybernetyka techniczna,
CYBERNETIC MODEL - model cybernetyczny,
CYBERNETICS - cybernetyka,
CYCLE - cykl, także: okres,
CYCLE CARRY - przeniesienie cykliczne (z najwyższej pozycji liczby do najniższej),
CYCLE COUNT - liczenie cykli,
CYCLE COUNTER - licznik cykli,
CYCLE STEALING - zabieranie cykli,
CYCLE TIME - czas cyklu,
CYCLES PER SECOND - liczba cykli na sekundę,
CYCLIC - cykliczny, okresowy,
CYCLIC CODE - kod cykliczny,

CYCLIC PERMUTED CODE - kod cykliczny permutowany,
CYCLIC PROGRAMMING - programowanie cykliczne,
CYCLIC REDUNDANCY CHECK CHARACTER (CRC) - znak CRC, znak kontroli cyklicznej,
CYCLIC SHIFT - przesunięcie cykliczne,
CYCLICAL REDUNDANCY CHECK - cykliczna kontrola redundancji,
CYLINDER - cylinder,
CYLINDER ADDRESS - adres cylindra,
CYLINDER INDEX - indeks cylindra,
CZ-13 FONT - pismo CZ-13,

D

DAILY DATA - dane dzienne,
DAISY-CHAIN - układ łańcuchowy, rozproszony układ przerwań skierowanych,
DAISY-WHEEL PRINTER - drukarka z wirującą głowicą, drukarka rozetkowa,
DAMAGE - uszkodzenie, awaria, także: uszkodzić,
DAMAGEABLE - podatny na uszkodzenia,
DARK TRACE TUBE - lampa z ciemnym śladem (strumień elektronów daje ciemny ślad na ekranie),
DAS - patrz: DATA ACQUISITION SYSTEM,
DASH - kreska, znak łamania,
DASHED LINE - linia kreskowa,
DATA - dane, także: wartość,
DATA ACCEPTED - potwierdzenie odczytu danych,
DATA ACQUISITION - gromadzenie danych, zbieranie danych,
DATA ACQUISITION SYSTEM - układ gromadzenia danych,
DATA AVAILABLE - sygnał gotowości danych,

OGŁOSZENIA

Commodore 16/116/+4 — programy — wymienię. Jacek Rubinkowski, 09-400 Płock, ul. Orlińskiego 3 m 20

Wideo niesprawne zamienię na mikrokomputer lub sprzedam. Andrzej Peta, Okumino, 77-203 Dretyń, woj. Słupsk

Programy na C-64 sprzedam. Marek Rudnicki, 51-376 Wrocław, ul. Kłokoczycka 154

Commodore 64 — chętnie wymienię się oprogramowaniem. Posiadam około 400 programów i gier. Alexander Stompel, 40-160 Katowice 2, ul. Armii Czerwonej 104/1

Kupię TIMEX 2020. Michał Pracel, 81-116 Gdynia, ul. Bosmańska 28/12

Sprzedam Commodore 64 ze stacją 1541 i magnetofonem 1530 lub samą stację. Piotr Węcławik, 71-015 Szczecin, ul. Osikowa 26, tel. 82-41-82

Wymienię programy na Commodore 64. Radosław Dubiella, 75-347 Koszalin, ul. Władysława IV 58a/37, tel. 529-36

Pocztowa giełda

Rozwiązanie krzyżówki z nr. 7/8 „IKS-a". Hasło brzmi: „Kluby komputerowe młodych mistrzów techniki".
Bony pieniężne (1000 zł) wylosowali: Irena Pietruszewicz — Poznań, Dariusz Sideł — Kielce, Dorota Noworyta — Zielonki, Konrad Kwiatkowski — Jedlicze, Sławomir Załęcki — Łask.

Nagrody książkowe otrzymują: Wojciech Stefaniak — Włocławek, Klementyna Dembek — Toruń, Maria Ramian — Tarnów, Małgorzata Kaliszan — Murowana Goślina, Wiesław Kołder — Ustroń, Maciej Trąd — Zielona Góra, Marian Borawski — Lubaczów, Wojciech Abramczuk — Sokołów Podlaski, Andrzej Piela — Zabrze, Artur Rapacz — Legnica.

Rozwiązanie krzyżówki z nr. 9/87 „IKS-a". Hasło brzmi: „Pamiętajmy o komputerze nie tylko na co dzień ale i w święta".
Bony PKO (1000 zł) wylosowali: Jerzy Trzaska — Zawadzkie, Jarosław Kędrak — Częstochowa, Andrzej Tadewicz — Lublin, Anna Rynkiewicz — Dolistowo, Grażyna Stefanek — Lublin.
Nagrody książkowe otrzymują: Danuta Bamber — Margonin, Mariusz Zacharkiewicz — Buraków k. W-wy, Weronika Kowalska — Sosnowiec, Sewer Nyckowski — Gostynin, Tadeusz Dziugieł — Ruda Śl., Jacek Szczepański — Zielona Góra, Alicja Kokot — Częstochowa, Wojciech Pilski — Gdańsk, Jarosław Murenia — Gorzów, Piotr Bakun — Janów Lub.

Rozwiązanie krzyżówki z nr. 10/87 „IKS-a". Bony pieniężne (1000 zł) wylosowali: Elżbieta Mazurowska — Szczecin, Dariusz Lubawy — Gniezno, Edward Chmielak — Kłodzko, Agnieszka Jastrząb — Dąbrowa Górnicza, Aleksander Wierzbicki — Białystok.
Nagrody książkowe otrzymują: Jacek Wychowaniec — Sosnowiec, Wacław Skrzypczak — Warszawa, Krzysztof Chudy — Szczecin, Janusz Midura — Pszczyna, Henryka Kaczmarska — Radom, Tomasz Żak — Wrocław, Maciej Ziętek — Poznań, Jolanta Pogorzelska — Gdańsk, Stefan Kubowicz — Kraków, Jan Strzałka — Stargard Szczeciński.

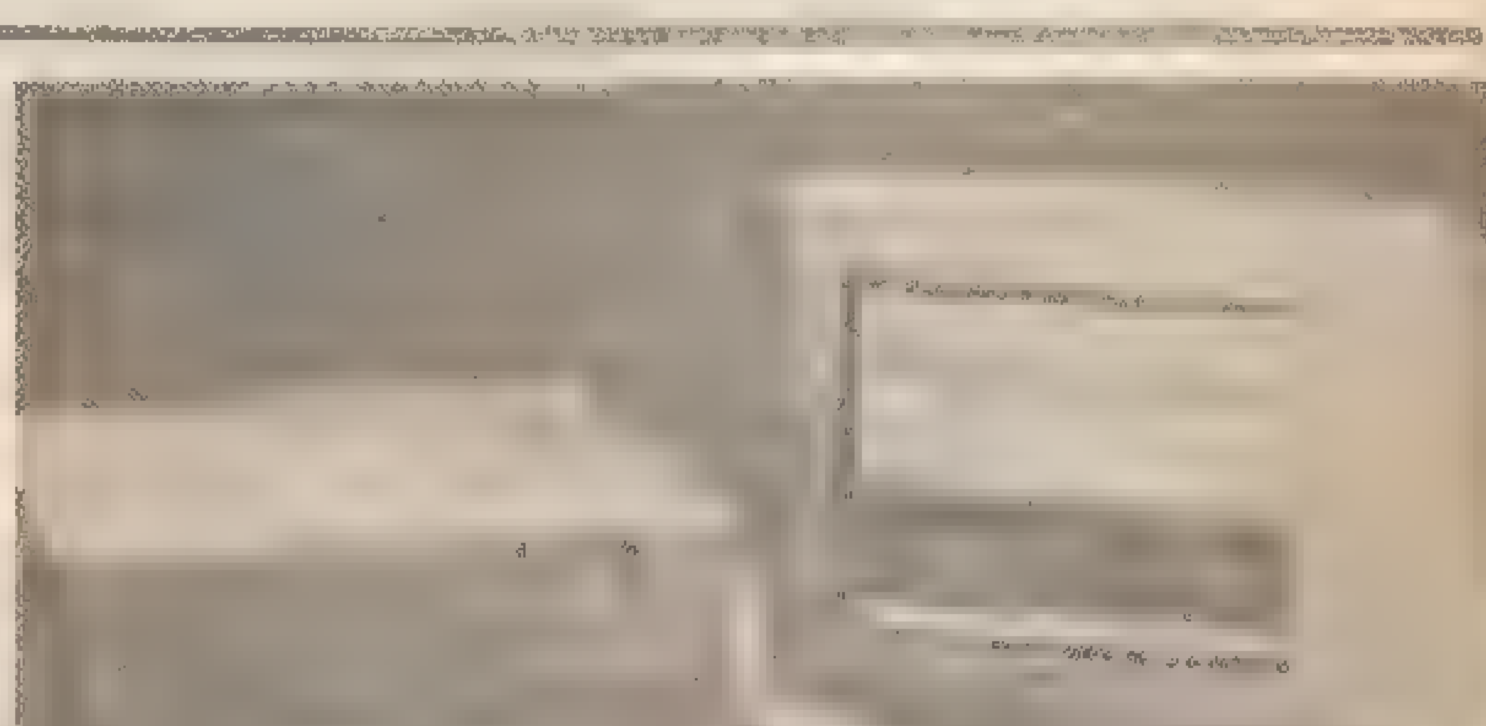
Rozwiązanie krzyżówki z nr. 11/87 „IKS-a". Bony pieniężne (1000 zł) wylosowali: Michał Zimnicki — Malbork, Włodzimierz Plewa — Gdańsk, Grzegorz Romanowski — Łomża, Jerzy Wieloch — Poznań, Krzysztof Michalski — Ełk.
Nagrody książkowe otrzymują: Leszek Pniak — Siemianowice Śl., Leszek Bakalarczyk — Bydgoszcz, Tomasz Woźny — Poznań, Marek Wachel — Kraków, Radosław Łepek — Gdynia, Marek Lasecki — Warszawa, Tadeusz Mamak — Kraków, Piotr Kalewicz — Gdynia, Andrzej Balec — Olkusz, Aleksander Stochalski — Kraków.

Rozwiązanie krzyżówki z nr. 12/87 „IKS-a". Hasło brzmi: „Poznaj komputer — zyskasz przyjaciela".

Bony pieniężne (1000 zł) wylosowali: Sławomir Bukowski — Pruszków, Grażyna Nowak — Kraków, Andrzej Pieszalski — Słupsk, Andrzej Herla — Głogówek, Dorota Radziwon — Torzyn.

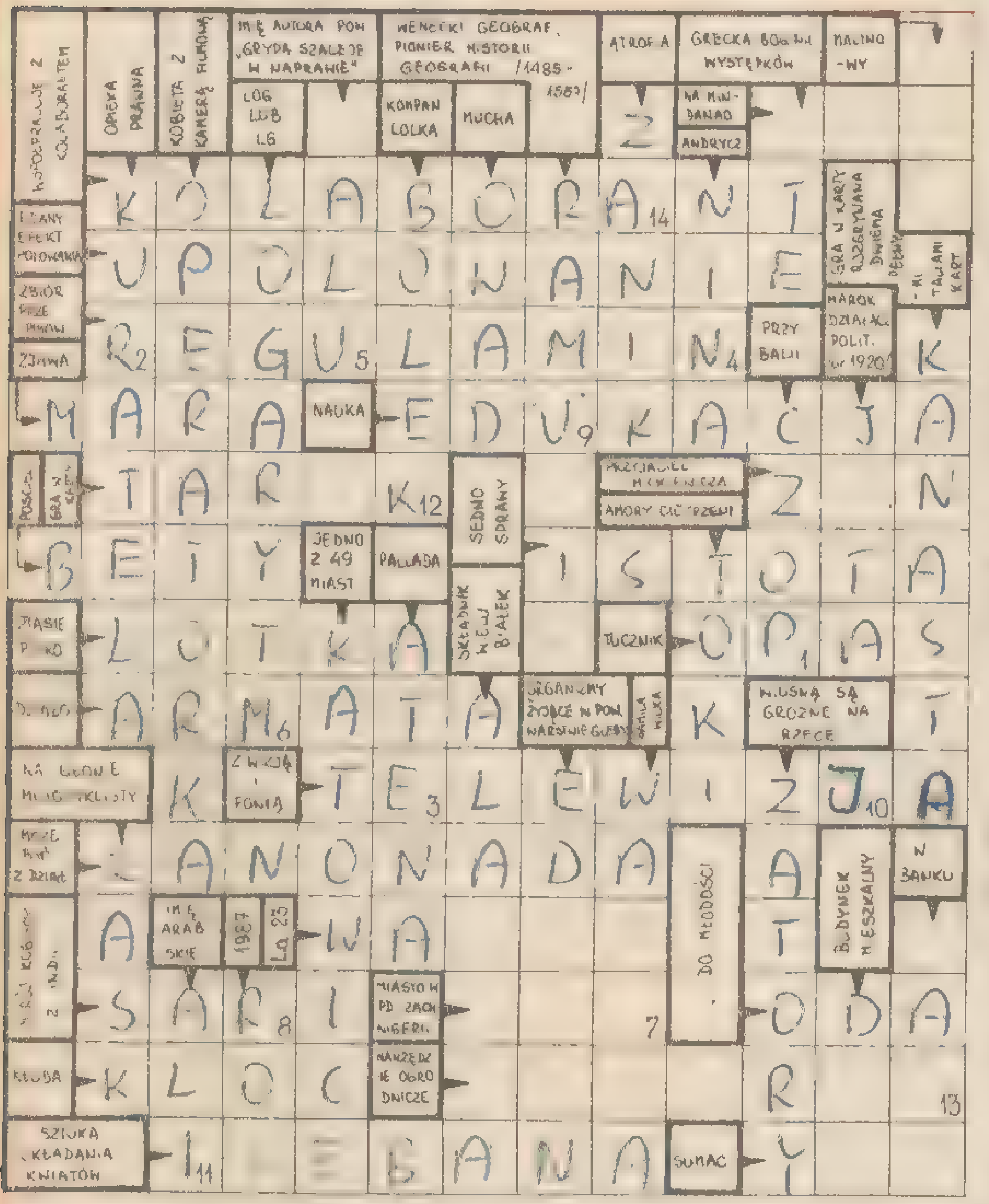
Nagrody książkowe otrzymują: Janina Stefanowska — Bielsko-Biała, Danuta Stryczek — Kluczbork, Ewa Dobrucka — Jastrzębie-Zdrój, Andrzej Ćwikliński — Wrocław, Wiesław Luks — Poznań, Maciej Trólka — Czeladź, Barbara Raczkowska — Szczecin, Maciej Dusza — Warszawa, Robert Perłowski — Ziębice, Tomasz Salachna — Częstochowa.

OLIVETTI PC M380



Ten komputer pracuje na 32-bitowym mikroprocesorze 80386. Podobnie jak inne modele tej firmy jest programowo zgodny ze swoim odpowiednikiem z IBM. M 380 produkowany jest w trzech wersjach: **M 380 C** (Compact), **M 380 S** (Standard) i **M 380 T** (Tower) W ostatniej z nich pamięć RAM może być rozbudowana do 64 Mb

KRZYŻÓWKA NR 2

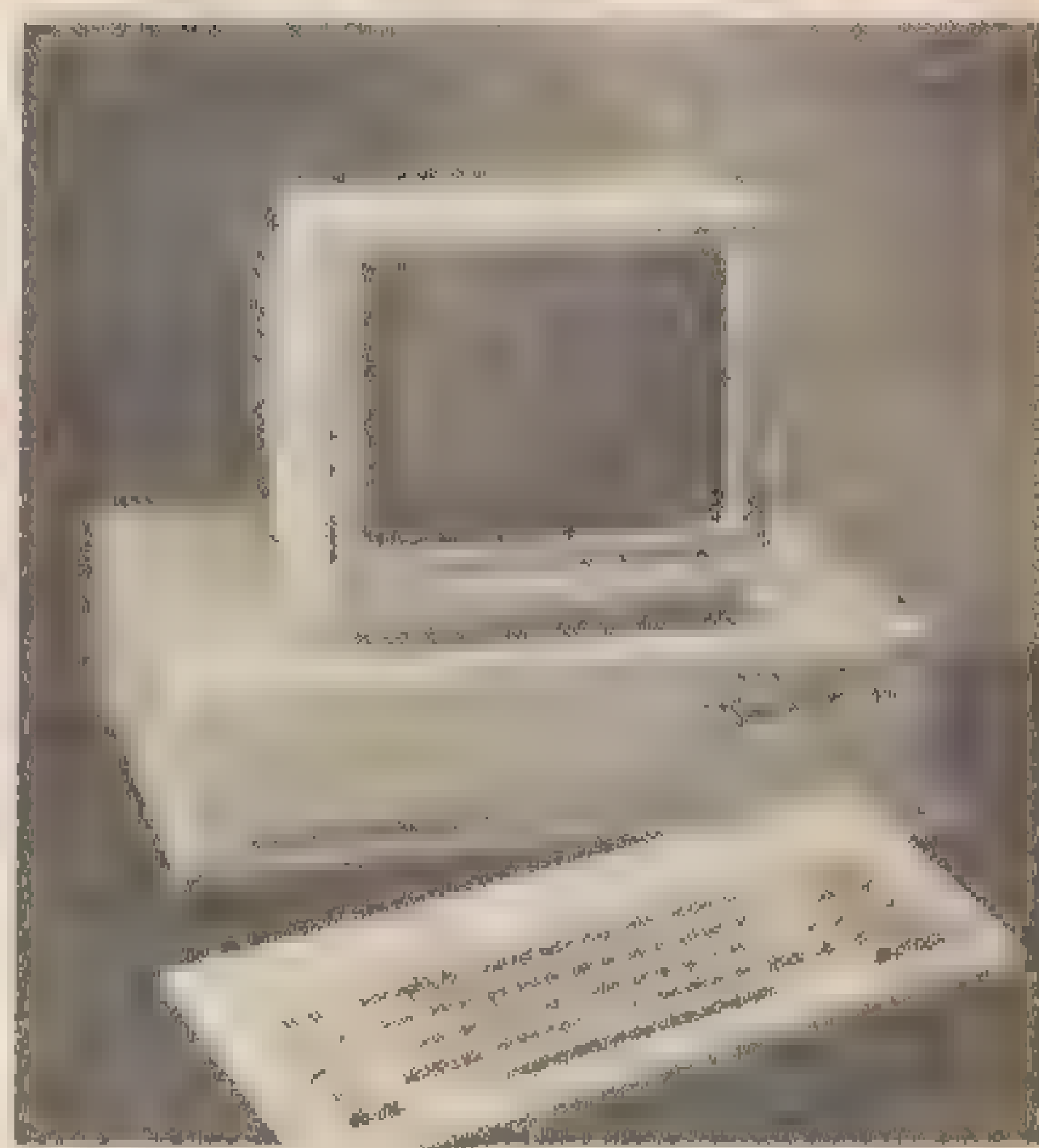


Litery z kratek ponumerowanych od 1 do 14 dadzą hasło, które wystarczy nadesłać (jako rozwiązanie zadania) pod adresem redakcji na kartach pocztowych w terminie do końca kwietnia, naklejając kupon „IKS"-a. Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

KUPON ▶

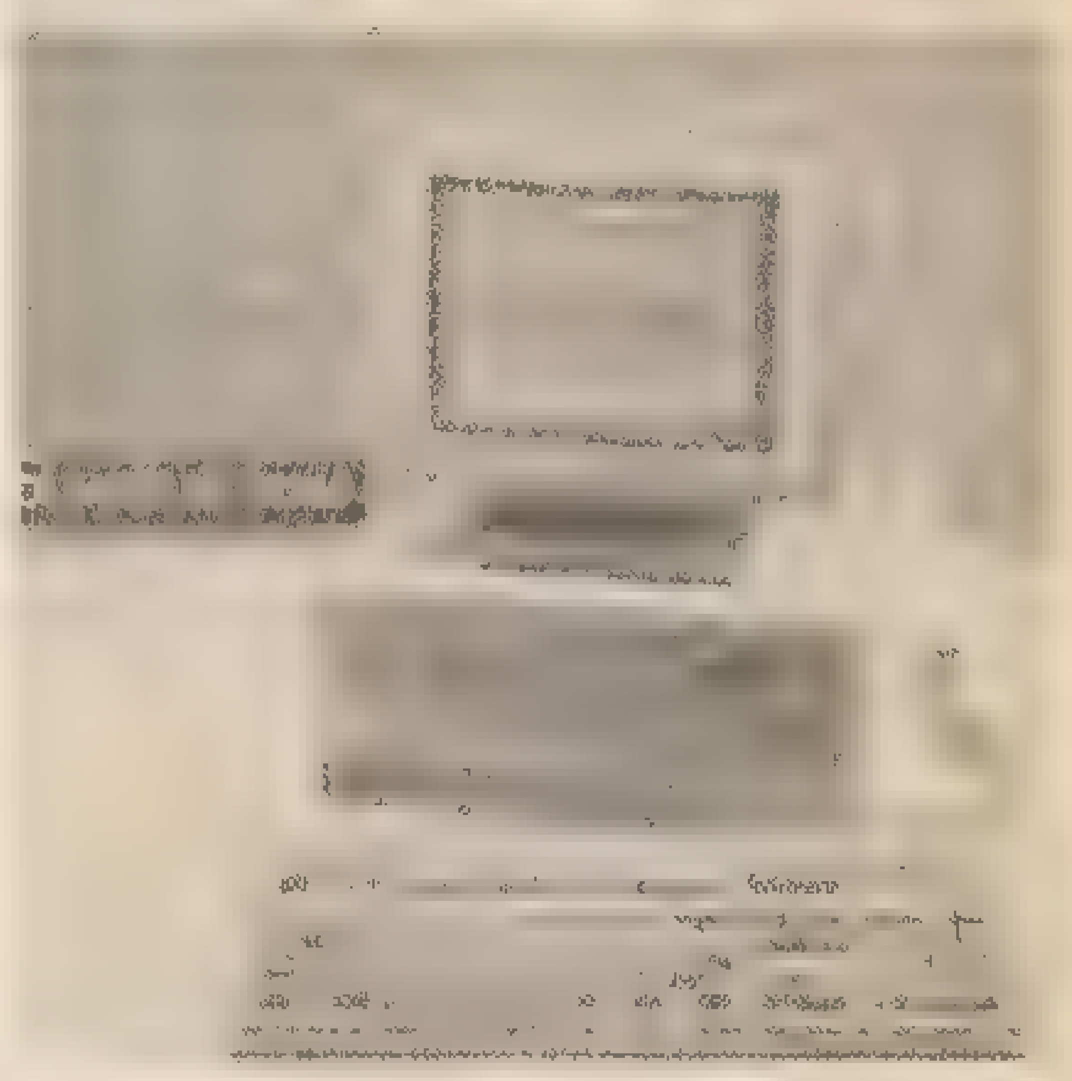


COMMODORE PC 10 i PC 20



PC 10 i PC 20 zbudowane zostały na Intelu 8088. Pierwszy z nich ma RAM 512 Kb drugi 640 Kb. Oba pracują pod systemem operacyjnym MS DOS 3.2. Zasadniczą różnicą jest dysk twardy (20 Mb) zainstalowany w modelu PC 20

OLIVETTI PC M240



Zbudowano go na mikroprocesorze 8086. **M240** dysponuje pamięcią RAM 640 Kb. Standardowo wyposażony jest w jedną stację dysków miękkich (5 1/4 cala). Istnieją oczywiście możliwości rozbudowy konfiguracji podstawowej. Można ją wzbogacić między innymi o dyski miękkie i 20 Mb dysk twardy. Zaletą tego modelu Olivetti jest jego programowa zgodność ze standardem IBM.

Systemem operacyjnym jest MSDOS 3.2. M240 pobiera w czasie pracy moc 183 W (napięcie sieci zasilającej 220 V ± 10%).

„IKS" — dodatek „Żołnierza Wolności". [illegible]

INFORMATYKA KOMPUTERY SYSTEMY

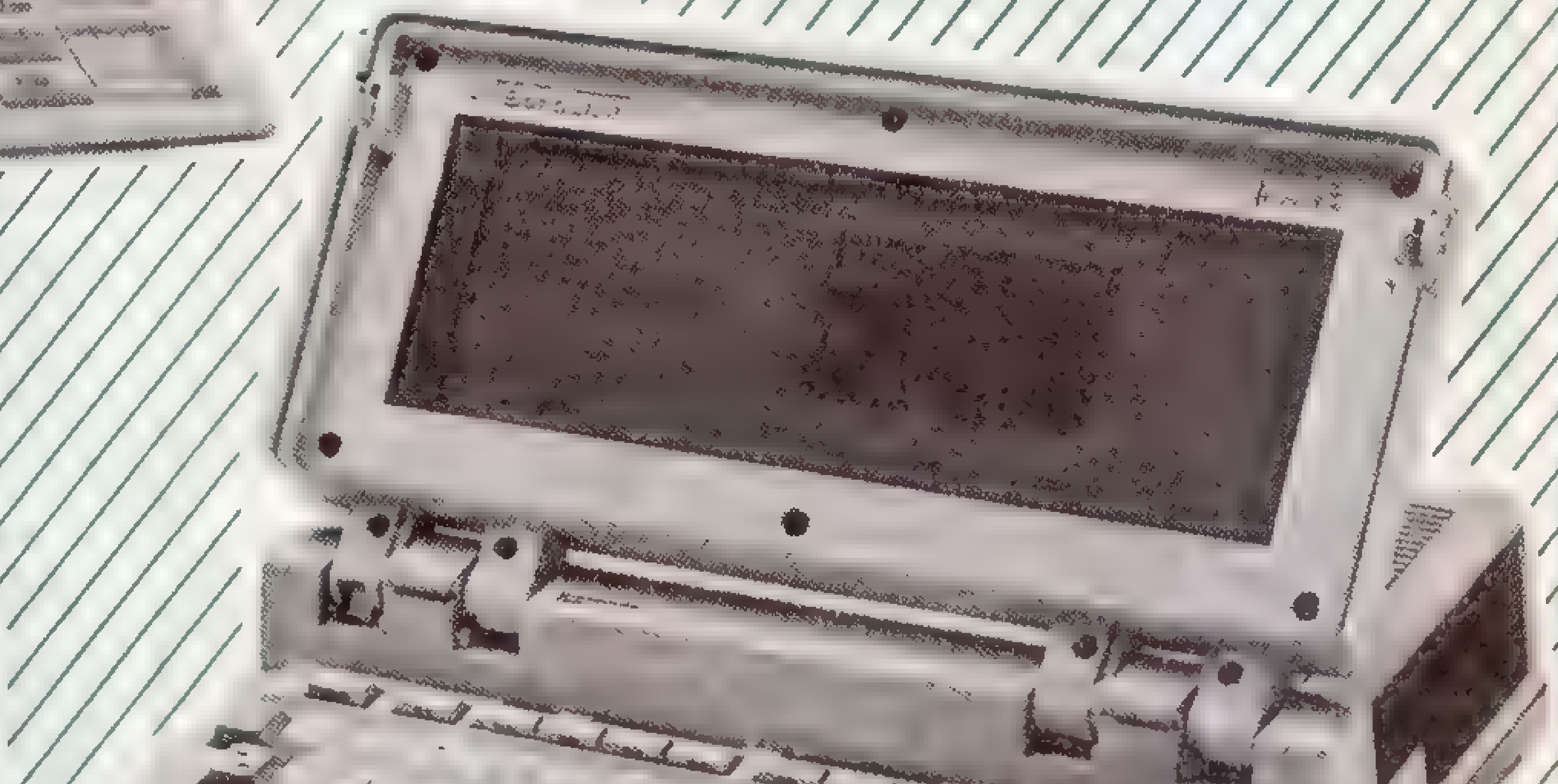


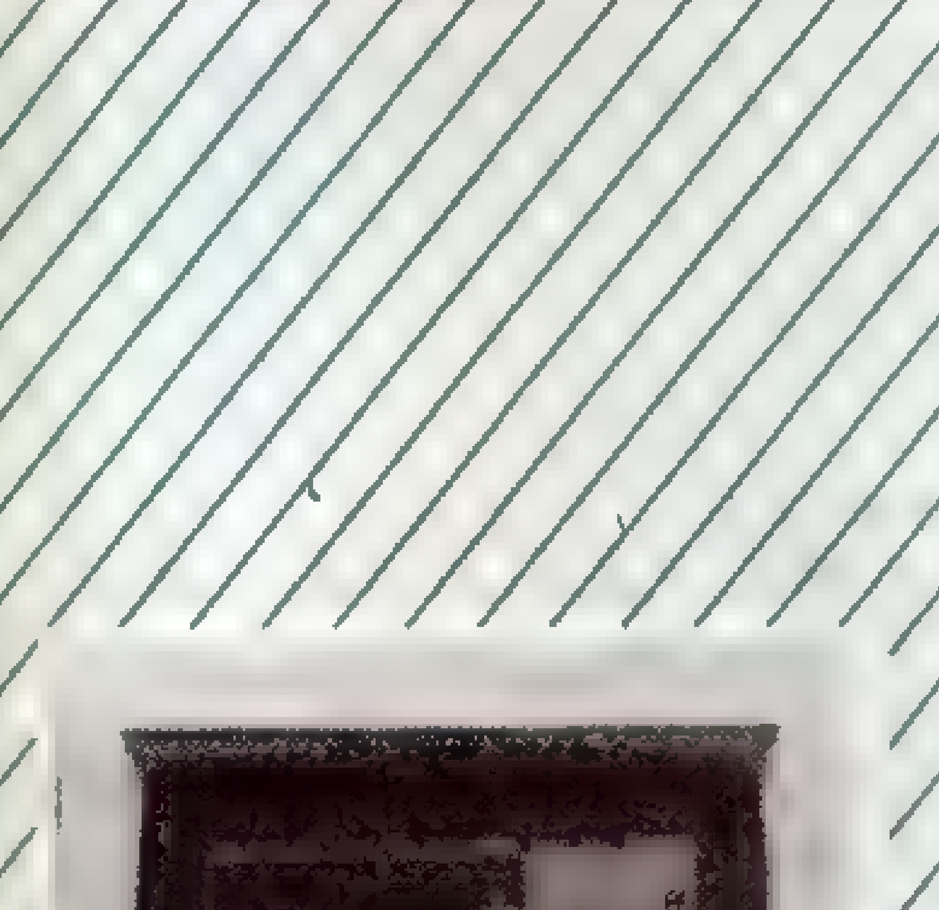
CENA 80 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 3/1988 ISSN 0860-2794



W szponach Atari
Interfejs magnetofonu
Compact Disc
Grafika na ekranie





W NUMERZE:

Doczekaliśmy się zdrowej konkurencji — str. 4
ODRA — Amstrad — str. 5
W szponach Atari (13) — str. 6—10
Tabelaryzacja — str. 10—11
Interfejs magnetofonu — str.12—13
Lilavati (cz. 3) — str. 14—15
Laureaci „Ligi Myślących" — str. 15
Compact Disc — str. 16—18
Standard MSX, a mikrokomputer SVI-738 X'PRESS — str. 19
Grafika na ekranie (cz. 4) — str. 20—21
Animacja kwadratu — str. 22
Budżet domowy — str. 23—25
Krzywe Sierpińskiego — str. 26—28
Informatyczny słownik angielsko-polski — str. 29—30
Giełda pomysłów — str. 31

Komputerowy biznes Centralnej Składnicy Harcerskiej

Od roku 1986 Centralna Składnica Harcerska handluje również sprzętem komputerowym. Osiem wyspecjalizowanych placówek w całym kraju sprzedało w ubiegłym roku sprzęt wartości 6.5 miliarda złotych. CSH związała się z jednym dostawcą — jest nią austriacka firma PROSYSTEM, a warunki, jakie zaproponowała są na tyle atrakcyjne, że polski kontrahent nie myśli o zmianie partnera. Szczegóły umowy są oczywiście tajemnicą handlową. Muszą być jednak atrakcyjne dla obu stron, skoro na polski rynek trafiają za sprawą Austriaków kolejne generacje sprzętu komputerowego. Rozmowy partnerów muszą być jednak długie, efektem są zapełnione półki w sklepach CSH i... coraz lepsza znajomość języka polskiego przez właściciela Prosystemu Hansa Hofera.

W najbliższej przyszłości na Mokotowskiej będzie można kupić komputery zgodne z IBM PC/XT i IBM PC/AT, zbudowane według koncepcji wspólnej magistrali, szybkie, 32-bitowe komputery z zegarem 24 MHz (!), a także wielkoekranowe projektory grafiki kolorowej, sieci komputerowe, systemy edycji składu i komputerowego wspomagania projektowania inżynierskiego.

Sklepy CSH prowadzące sprzedaż sprzętu mikrokomputerowego

BYDGOSZCZ	ul. Chodkiewicza 68,	telefon 41 44 30
KATOWICE	ul. Rozdzieńskiego 88a,	telefon 58 28 33
KRAKÓW	ul. Rynek Główny 5	telefon 22 73 12
LUBLIN	ul. Kowalska 14,	telefon 29 472
ŁÓDŹ	ul. Lutomierska 12,	telefon 57 15 27
[illegible]	ul. Mariana Buczka 24c,	telefon 35 596
WARSZAWA	ul. Mokotowska 26	telefon 21 76 55
WROCŁAW	plac Grunwaldzki 6a	telefon 21 92 61

Pascal w drodze do Czytelników

Już wkrótce drukarnię opuści kolejny zeszyt programów „IKS-a". Pierwszy w tym roku poświęcony będzie w całości pascalowi. Ten język należy do jednych z najefektywniejszych narzędzi programowania strukturalnego. Programy napisane w pascalu działają o wiele szybciej niż te, które do tej pory zwykliśmy pisać w basicu.

I jeszcze jedno: pascal jest jeden — program napisany w tym języku na przykład na Spectrum można przepisać bez żadnych przeróbek w postaci źródłowej i uruchomić na IBM-ie. W naukę pascala warto zainwestować.

POCZTOWA GIEŁDA

Rozwiązanie krzyżówki z nr. 1/88 „IKS-a".

Nagrody książkowe otrzymują: Sławomir Łysz — Włocławek, **Andrzej Erd** — Radom, **Marcin Kobiela** — Świerklaniec, **Małgorzata Nowak** — Kraków, **Andrzej Niedbała** — Chorzów, **Dorota Baranowska** — Oleśnica, **Adam Dubicki** — Gdynia, **Jarosław Rosiński** — Radom, **Maria Rabczak** — Gdańsk, **Adam Kuryłowski** — Szczecin.

Rozwiązanie krzyżówki z nr. 2/88 „IKS-a",

Nagrody książkowe otrzymują: Wacław Skrzypczak — Warszawa, **Małgorzata Czapla** — Tarnowskie Góry, **Eugeniusz Chaszczewski** — Kłodzko, **Hanna Kornatowska** — Warszawa, **Przemysław Niekrasz** — Wrocław, **Krzysztof Małek** — Zwoleń, **Michał Urbaniak** — Turek, **Tomasz Skalski** — Mrągowo, **Jacek Wolski** — Łódź, **Łukasz Gumkowski** — Ostrołęka.

LAUREACI DRUGIEJ EDYCJI „LIGI MYŚLĄCYCH"

Zwycięzcami drugiej edycji „Ligi Myślących" zostali, uzyskując maksymalną liczbę 64 punktów, za prawidłowe rozwiązanie wszystkich zadań, następujący czytelnicy:

Zdzisław BEDYNEK
03-982 Warszawa
ul. Zawadzkiego 18/68

Wiesław RYCHLICKI
38-604 Hoczew
Średnia Wieś (szkoła)

O jeden punkt mniej, tj. 63 uzyskali następujący czytelnicy:

Magdalena ROJEK
01-006 Warszawa
ul. Nowolipki 19/15

Andrzej BARCEWICZ
18-400 Łomża
ul. Broniewskiego 2/39

Adam DOMIŃSKI
80-395 Gdańsk
ul. Rokossowskiego 11/49

O terminie wręczenia nagród laureaci zostaną powiadomieni listownie, natomiast od następnego numeru rozpoczynamy kolejną turę zmagań Myślących.

NAUKA — TECHNIKA

Firma MARTIN MARIETTA przystępuje do modyfikacji międzykontynentalnych rakiet balistycznych typu TITAN II w celu przystosowania ich do roli nosiciela satelitów. Pierwsza z czterech zamówionych, zmodyfikowana rakieta tego typu została już przetransportowana do bazy lotniczej Vandenberg w Kaliforni, gdzie w kwietniu tego roku ma nastąpić jej start. Umieści ona na orbicie okołoziemskiej kolejnego amerykańskiego satelitę przeznaczonego do celów wojskowych.

x x x

Nowym produktem zachodniemieckiej firmy ILC-DDC Elektronik GmbH jest, oznaczona symbolem MEM-8400, statyczna pamięć RAM o pojemności 64Kx16 wykonana technologią CMOS. Zbudowana jest z czterech bloków 32Kx8. Może ona pracować w zakresie temperatur od -55°C do +125°C, co w pełni odpowiada nowej wojskowej normie MIL-STD-883. Mały pobór prądu umożliwia zapamiętywanie danych przy zewnętrznym zasilaniu baterią 2V. Czas dostępu wynosi 100 nanosekund. Pamięć ta, w połączeniu z 16- lub 32-bitowym mikroprocesorem znajduje zastosowanie między innymi w układach grafiki komputerowej .

x x x

Chemikom z amerykańskiego uniwersytetu RHODE ISLAND udało się wynaleźć polimer przewodzący prąd elektryczny i zmieniający swą barwę pod wpływem zmian jego natężenia. W stanie normalnym polimer jest bezbarwny, dopiero pod działaniem prądu staje się najpierw zielony, a później niebieski i fioletowy. Materiałem tym zainteresowali się już specjaliści planując skonstruowanie polimerowego ekranu telewizyjnego. Nowy materiał zostałby umieszczony pomiędzy dwiema szklanymi taflami pokrytymi drobną siatką przewodów elektrycznych. Dostarczany za jej pośrednictwem prąd zmieniałby w określonych miejscach barwę takiego ekranu.

Tegoroczny model japońskiego samochodu MAZDA 626 ma zostać wyposażony w system przeciwdziałający poślizgowi kół podczas hamowania. System ten jest efektem pięciu lat wspólnych prac badawczych firm MAZDA, SUMITOMO, LUCAS GIRLING i został wypróbowany zarówno w Japonii, jak i Europie. Jest to układ czterokanałowy nadzorujący indywidualnie każde z czterech kół pojazdu. Stałe ciśnienie w układzie zapewnia pompa napędzana silnikiem elektrycznym. Każdy z członów sterujących wyposażony jest w dwa mikroprocesory sterujące pracą układu i kontrolujące wzajemnie swoje działanie.

■ ■ ■

Dużym zaskoczeniem dla uczonych, była analiza danych o komecie HALLEYA dostarczonych przez zachodnioeuropejską sondę kosmiczną GIOTTO, króra przeleciała w odległości 600 km od jej jądra. Okazało się, że w atmosferze komety występuje trzy razy więcej izotopu wodoru - deuteru, niż w atmosferze Jowisza i Saturna. Zawartość deuteru jest w komecie taka sama jak w wodorze spotykanym na Ziemi, w skałach księżycowych oraz meteorytach. Jest to ważne spostrzeżenie z punktu widzenia kosmochemii. Wysunięto hipotezę, że planety grupy ziemskiej mogły formować się w tej samej części protosłonecznego obłoku, co komety. Odkrycie to rzuca nowe światło na problem powstania i ewolucji naszego Układu Słonecznego.

■ ■ ■

Na uniwersytecie w Birmingham (Wielka Brytania) opracowano nową technologię jednoczesnego wytapiania kilku metali z rud wieloskładnikowych. Wytapianie odbywa się w specjalnym wielopoziomowym piecu, w którym podgrzewa się rudę do temperatury około 1300°C. Podstawą nowej technologii jest recyrkulacja stopionej rudy pomiędzy kolejnymi poziomami pieca, która, dając ciągłą reakcję chemiczną, pozwala na wyodrębnienie poszczególnych metali z rudy. Przewiduje się, że wdrożenie tej technologii spowoduje ponowny rozkwit hutnictwa metali nieżelaznych w XXI wieku. Umożliwi bowiem wykorzystanie olbrzymich pokładów rud wieloskładnikowych o strukturze drobnoziarnistej, króre nie mogły być dotychczas eksploatowane.

Doczekaliśmy się zdrowej konkurencji

Rozmowa z dyrektorem Biura Organizacji Importu i Eksportu — Waldemarem ZNAMIEROWSKIM.



● Rynek informatyczny zapełniony już jest podobnymi, jak wy firmami, czy uważa Pan, że zdołacie się przebić

— To chyba zdrowy objaw prawdziwej konkurencji, nawet tu na Chłodnej działają obok siebie trzy spółki zajmujące się informatyką. Nie przeszkadzamy sobie. Nasi kontrahenci stają się wybredniejsi, zmuszeni zatem jesteśmy nie tylko do rozszerzania oferty, ale i do doskonalenia jakości naszych usług i to decyduje o powodzeniu

● Czy konkurencja wpływa także na ceny?

— Sprzętu już tak, choć nie zawsze ceny idą w dół. W zasadzie utrzymują się na stałym poziomie, co przy zmiennym przecież kursie złotówki jest dowodem względnego spadku cen. Ale klienci na ogół poszukują nowości

● To znaczy?

— Coraz częściej odnotowujemy zainteresowanie sieciami. To zjawisko dobrze świadczy o rozwoju naszej informatyki. Myślę, że moda na „komputer" przemija już bezpowrotnie. Częściej informatyka ma być tą dziedziną działalności, która ułatwia funkcjonowanie przedsiębiorstw, a tego nie załatwi przecież spektakularnie zainstalowany mikrokomputer

● Co zatem proponujecie?

— Przede wszystkim oczywiście sprzęt zgodny ze standardem IBM. Oferujemy sieci Ethernet, Delink, Novell, Arcnet w zależności od przeznaczenia i możliwości, a także wielodostęp. Pod tym względem nie różnimy się od innych tego rodzaju firm

● I to ma skłonić potencjalnego klienta do wyboru właśnie was?

— W tej branży decyduje przede wszystkim jakość obsługi, sprawność serwisu, niezawodność proponowanego sprzętu. Staramy się sprzedawać komputery jednego producenta — to znacznie ułatwia sprawy serwisowe. Nie bez znaczenia jest też fakt, że za rok, dwa nasz pierwszy klient wróci do nas po kolejne zakupy

● Jaką dajecie gwarancję, że będziecie istnieć

— My chcemy istnieć, ale gwarancji dać nie możemy, bo za dwa miesiące mogą zmienić się przepisy finansowe i zbankrutujemy

● Krucha ta perspektywa

— Ale prawdziwa

● Co zatem w najbliższej przyszłości?

— Instalujemy kilka sieci komputerowych wraz z oprogramowaniem, ale są to instalacje lokalne. Trzeba pamiętać, że im sieć bardziej rozbudowana, a poszczególne jej elementy od siebie bardziej odległe, jej cena szybko rośnie. W tym przypadku decyduje o tym cena... kabla — jeden metr 15 dolarów

● Przecież można wykorzystać istniejącą już sieć telekomunikacyjną

— Tak, ale nie tej jakości...

● Rozumiem, że ubogim klientom nie dajecie żadnych szans

— Jesteśmy na etapie, w którym efekty wprowadzania informatyzacji już widać. Nie będzie w tej dziedzinie rewolucji, ale i od informatyki nie uciekniemy. Dlatego mało kto boi się informatyki i inwestycji w tej dziedzinie — to konieczność — a poza tym zmieni się wkrótce jakość świadczonych usług

● To znaczy?

— Myślę, że już niedługo większą uwagę przywiązywać się będzie do oprogramowania. Fala inwestycji sprzętowych już przechodzi punkt kulminacyjny. Zresztą na całym świecie koszty sprzętu są jedynie mniej więcej jedną trzecią wydatków na informatykę — reszta to oprogramowanie

● U nas oprogramowanie jest za darmo

— Albo kosztuje krocie. Nie ma tu reguły, choć obowiązują dwa krańcowo różne podejścia. Prawo autorskie nie chroni oprogramowania, a zatem kopiuje się je bezkarnie i za darmo, albo dokonuje się drobnych przeróbek i przyznaje do jego autorstwa

● Tak jest w przypadku oprogramowania narzędziowego i standardów

— Prawo autorskie nie chroni również programów naszych polskich informatyków. Sądzę jednak, że wkrótce sprawy te zostaną uregulowane

● Macie jakieś asy w rękawie

— Stacje teleksowe sprzężone z komputerem personalnym klasy IBM. Handlujemy też sprzętem wideo, ostatnio sprowadziliśmy ekrany wizyjne (Cine Vision)

● Przyznaję, że jest to duży rozrzut zainteresowań

— Tak, ale my proponujemy zawsze najwyższą jakość usług

● Czego Wam i klientom życzę

Wiesław Cetera



(Foto: J. Zelman)

ODRA — Amstrad

Wyposażenie uczelni w mikrokomputery ośmiobitowe, nawet takie jak AMSTRAD CPC-6128, w niewielkim stopniu rozwiązuje informatyzację (z punktu widzenia zaopatrzenia w sprzęt) zarówno dydaktyki, jak i procesów zarządzania — ze względu na małą pojemność pamięci (operacyjnej i dyskowej) oraz niezależną pracę każdego zestawu mikrokomputerowego.

Także eksploatowany dotychczas (rozbudowany) sprzęt komputerowy ODRA — 1305 pracujący pod kontrolą Systemu Operacyjnego (SO) GEORGE — 3 nie spełniał całkowicie związanych z nim oczekiwań. Powodem było skromne (jest to chyba problem wszystkich ośrodków obliczeniowych) wyposażenie w sprzęt teletransmisji — Urządzenia Przetwarzania Danych (UPD), modemy, a przede wszystkim w t e r m i n a l e — drukarki (DZM) lub monitory z klawiaturą.

W tej sytuacji zdecydowano się połączyć mikrokomputery z ODRĄ, nie tylko w celu uzyskania dodatkowych terminali do pracy w wielodostępie pod kontrolą SO GEORGE-3, ale także z zachowaniem niezależnej pracy AMSTRADA — z możliwością przepisywania danych między dyskietką a pamięciami masowymi ODRY.

Sprzężenia takiego dokonano przez szeregowy interface oraz odpowiednie oprogramowanie — emulator „LIDIA". AMSTRAD CPC-6128 stał się standardową asynchroniczną dialogową końcówką ODRY — taką jak DZM. Wymuszenie (w czasie pracy pod kontrolą SO GEORGE-3) przerwania ([CONTROL] Λ K) umożliwia wybranie kierunku przesyłania danych, z dyskietki AMSTRADA do Pamięci Zbiorów Systemu (PZS) GEORGE-3 — komendą INPUT lub w drugą stronę — z PZS na dyskietkę — komendą LISTFILE.

Takie rozwiązanie umożliwia przygotowanie danych do przetwarzania na dyskietkach z wykorzystaniem mikrokomputerów rozproszonych (w pomieszczeniach wykonawców, użytkowników lub w domu), a następnie przepisywanie zawartości dyskietki — z wydzielonych stanowisk — do pamięci masowych ODRY. Także wyniki przetwarzania (na ODRZE) mogą być przekazywane użytkownikom — przez transmisję w drugą stronę — z PZS na dyskietkę AMSTRADA, a nie jak dotychczas — w postaci wydruku.

Podstawową formą wykorzystania AMSTRADA będzie jednak traktowanie tego urządzenia jako terminalu do typowej pracy pod kontrolą SO GEORGE-3:

— realizacja zadań programowych (edycje zbiorów, kompilacje, testowania programów)

— inicjowanie i kontrola wykonania prac użytkowych.

Uruchomienie pracy AMSTRADA pod kontrolą SO GEORGE-3 następuje przez:

— uruchomienie systemu CPM+

— ustawienie parametrów transmisji (SETSIO, BITS, STOP, PARITY)

— uruchomienie emulatora.

Praktycznie ładowany jest tylko CPM+. Pozostałe operacje przebiegają zgodnie z komendami założonego na dyskietce zbioru PROFILE.SUB.

Informacje praktyczne

— zbiory do zapisu (odczytu dyskietki nie mogą przekraczać jej pojemności 180 kbajtów)

— długość rekordu zbiorów do zapisu (odczytu z dyskietki nie może przekraczać 255 bajtów

— czas przepisywania danych zależy od obiążenia SO GEORGE-3, jest jednak długi. Przepisywanie pełnej dyskietki (180 kbajtów) trwa od 20 do 30 min (przy prędkości 4800 bodów)

— wykorzystując mikrokomputer jako typowy terminal SO GEORGE-3, nie odczwa się różnicy w szybkości pracy — w porównaniu z pracującymi jednocześnie innymi urządzeniami (DZM, monitory)

— dostępne prędkości przesyłania informacji — od 150 do 9600 bodów, zależą w praktyce od parametrów technicznych stosowanego sprzętu. Sprzężenie zrealizowano na prędkościach 1200 i 2400 bodów.

— formaty zbiorów:

— na dyskietce — zbiory tekstowe

— w PZS — zbiory formatu graficznego (CR, LP, TP, TR)

— sprzęt i oprogramowanie:

— komputer ODRA-1305 pracujący pod kontrolą SO GEORGE-3

— urządzenia teleprzetwarzania dla komputera ODRA (multiplekser, UPD i lub modemy)

— mikrokomputer AMSTRAD CPC-6128 z minimum jedną (wbudowaną) stacją dysków pracujący pod kontrolą systemu CPM+

— interface RS 232-AC produkowany przez AGROKOMPUTER w Warszawie

— emulator „LIDIA" wykonany przez mgr inż. Lidię Stasiewicz i mgr. inż. Pawła Stasiewicza (WPIPB ETOB w Warszawie)

— wykonawca sprzężenia — realizacja „pod klucz", interface + emulator — spółka z o.o. SYKOMAT w Warszawie

— koszty (grudzień 1987 r.):

— interface — 140 tys. zł dla każdego mikrokomputera

— emulator — 150 tys. zł gdy praca na jednym stanowisku (mikrokomputerze AMSTRAD) lub 300 tys. zł przy dowolnej liczbie terminali.

W ramach tych kosztów realizowane jest wykonanie sprzężenia oraz zawarta jest roczna gwarancja.

— sprzężenie praktycznie zrealizowano z wykorzystaniem multipleksera i SO GEORGE-3. Można je wykonać przez inne urządzenia (np. skaner), pod kontrolą innego oprogramowania ODRY (MINIMOP) lub dowolnego, własnego oprogramowania np. multipleksera.

Wnioski

Sprzężenie komputera ODRA-1305 z mikrokomputerem AMSTRAD CPC-6128 zrealizowano w końcu 1987 roku w Ośrodku Obliczeniowym Wojskowej Akademii Politycznej. AMSTRAD nie jest droższym terminalem ODRY od dotychczas wykorzystywanych DZM lub monitorów MERA 7952 i jest dostępny na rynku.

AMSTRAD wykorzystywany jest w pracy pod kontrolą SO GEORGE-3 w każdej sesji — nawet chętniej niż inne terminale — przede wszystkim ze względu na klawiaturę. Edycja wprowadzanych informacji nie pozostawia na ekranie aktualizowanych (zmienianych, kasowanych) danych.

Należy nadmienić, że sprzężenie to rozbudowuje również możliwości mikrokomputera dzięki dostępowi do:

— pamięci masowych ODRY

— szybkiej (1000 wierszy/min) drukarki wierszowej.

oprac.
Krzysztof ŚWIĘTOŃ



— Na ciebie też reaguje negatywnie?

W szponach Atari (13)

Pamięć taśmowa

Tomasz MROWIEC, Ludwik Piela

Po wyłączeniu zasilania zawartość pamięci wewnętrznej komputera, zwykle półprzewodnikowej, ginie. Trudno sobie wyobrazić, aby za każdym razem, po włączeniu zasilania, użytkownik wprowadzał program za pomocą klawiatury. Podobna sytuacja jest ze zbiorami danych, o znacznej nieraz objętości. Dlatego niezbędna jest dodatkowa pamięć, której zawartość może być zachowana nawet po wyłączeniu zasilania i szybko wprowadzona do pamięci operacyjnej, po ponownym włączeniu komputera

Najprostszą i najtańszą możliwość przechowywania programów i danych stwarza standardowa kaseta magnetofonowa. Składa się ona z małego, plastikowego pojemnika o wymiarach 57x100 mm, zawierającego dwie szpule z nawiniętą taśmą o długości około 90 m i szerokości 4 mm. Otwór w bocznej krawędzi kasety umożliwia przesuwanie taśmy. Otwory w szpulach pozwalają na prowadzenie osi napędowych. Powszechnym rozwiązaniem mechanizmu przesuwu taśmy jest zastosowanie wałka napędowego i rolki dociskowej, które zapewniają stałą prędkość ruchu taśmy

Zapis i odczyt informacji odbywa się podobnie jak w magnetofonie. Część producentów umożliwia współpracę komputera domowego ze zwykłym magnetofonem kasetowym. Dzięki temu użytkownik uzyskuje pamięć zewnętrzną minimalnym kosztem. Magnetofon może być najczęściej bezpośrednio podłączony do komputera, bez konieczności stosowania dodatkowych urządzeń i umożliwia zarówno nagrywanie programów na kasecie, jak i wprowadzanie ich do pamięci wewnętrznej komputera

Niektóre komputery domowe, wśród nich również Atari, mogą współpracować tylko ze specjalnie do tego przeznaczonymi magnetofonami. Zapewnia to znacznie większy komfort obsługi (programowe włączanie i zatrzymywanie silnika) i większą niezawodność przesyłania programów i danych między komputerem i magnetofonem

Podstawowymi wadami tej pamięci są: szeregowa rejestracja informacji oraz bardzo mała szybkość przesyłania. Dlatego niezwykle długi, jak na wymagania komputera, jest czas wyszukiwania potrzebnej informacji na taśmie. Komputer potrafi znaleźć program lub dane utrwalone na nośniku magnetycznym, ale szukanie polega na sprawdzaniu zawartości w takiej samej kolejności i z taką samą szybkością, w jakiej realizowany był zapis. Dlatego, między innymi, specjalne kasety przeznaczone do rejestracji programów zawierają krótszą taśmę magnetyczną niż kasety do nagrań muzycznych. Znacznie mniej czasu zajmuje wymiana kaset niż wyszukanie potrzebnego programu na długiej taśmie. Mniejsza jest także szansa nierówności przesuwu na skutek oporów kaset.

Z tych względów nie polecamy nikomu używania taśmowej pamięci magnetycznej do poważniejszych prac, zarówno amatorskich, jak i profesjonalnych. Jednak rzeczywistość jest okrutna, cena stacji dysków (pamięci zewnętrznej o dostępie bezpośrednim) znacznie przewyższa cenę popularnego komputera ośmiobitowego. Dlatego większość krajowych posiadaczy komputerów domowych wykorzystuje pamięć taśmową, której cena nie przekracza połowy ceny komputera.

Jak już wspomnieliśmy, ośmiobitowe komputery Atari w zasadzie współpracują ze specjalnie skonstruowanym magnetofonem firmowym. Piszemy w zasadzie, ponieważ naszym rodakom, chyba jako jedynym na świecie, udało się dostosować do współpracy zwykłe magnetofony, zarówno stereo — (zalecane), jak i monofoniczne. Za ułamek ceny oryginalnego urządzenia, o nazwie Program Recorder, można nabyć specjalny układ pośredniczący, który umożliwia wykorzystanie zwykłego magnetofonu jako taniej pamięci zewnętrznej

Podobnie do różnorodnych sposobów organizacji wewnętrznej komputerów, zbudowanych często na podstawie tego samego mikroprocesora, istnieją różnorodne techniki zapisywania informacji na taśmie magnetycznej. Jest to rezultat fantazji projektantów w zakresie przyjętych rozwiązań układowych i programowych oraz braku jakichkolwiek standardów

Atari współpracuje z pamięcią taśmową za pośrednictwem układu **POKEY.** Zapis i odczyt informacji realizowany jest przez odpowiednie procedury systemu operacyjnego. W odróżnieniu od innych rodzajów pamięci, obsługa magnetofonu, pomimo programowego sterowania włączaniem i wyłączaniem silnika, wymaga znacznego udziału użytkownika. Niezbędne jest odpowiednie ustawienie taśmy oraz włączenie jednego lub dwóch klawiszy, w zależności od rodzaju wykonywanej operacji.

System operacyjny zapisuje informacje blokami o stałej długości z szybkością 600 bodów (fizycznych bitów/s). Do przesyłania danych między komputerem i magnetofonem używana jest szeregowa transmisja asynchroniczna. Układ POKEY rozpoznaje każdy przesyłany bajt w następującej kolejności: 1 bit startu („0" logiczne), 8 bitów danych oraz bit stopu („1" logiczna). Bajt wysyłany jest i odbierany od najmniej znaczącego bitu. Do reprezentowania „1" logicznej używana jest częstotliwość 5327 Hz, do reprezentowania „0" logicznego — 3995 Hz. Postać bajtu danych jest następująca.

s Ø 1 Ø 1 1 Ø Ø 1 s

Każdy blok składa się ze 132 bajtów. Wyróżnia się w nim: 2 znaki wskaźników, bajt kontrolny, 128 bajtów danych i bajt sumy kontrolnej.

Znaki wskaźników mają kod szesnastkowy 55. Wraz z bitami startu i stopu każdy wskaźnik ma długość 10 bitów. Służą do pomiaru szybkości transmisji. Zakłada się, że transmisja wejściowa odbywa się z nominalną szybkością 600 bodów. Do takiej szybkości dostrojone są układy odbiorcze. Jednak szybkość faktyczna nie jest stała, zależy od odchyleń w szybkości silnika, naciągu taśmy, itp. Jest ona zatem mierzona przez odpowiednią procedurę SIO. Po określeniu rzeczywistej szybkości transmisji wejściowej odpowiednio dostrajane są układy odbiorcze. W ten sposób może być obsługiwana transmisja wejściowa o szybkości zmieniającej się od 318 do 14Ø7 bodów. Pomiar szybkości wykonywany jest dla każdego bloku odczytywanego z taśmy.

Bajt kontrolowany przyjmuje jedną z trzech wartości

—$FC wskazuje, że blok jest zawiera dokładnie 128 bajtów danych.

—$FA wskazuje, że blok jest częściowo zapełniony; zawiera mniej niż 128 bajtów danych. Taki przypadek może wystąpić tylko w bloku przed końcem zbioru. Aktualna liczba bajtów danych, od 1 do 127, zapamiętana jest w ostatnim bajcie przed sumą kontrolną, tj. w 128 bajcie danych.

—$FE wskazuje, że jest to ostatni blok w zbiorze i występuje w nim 128 bajtów zerowych

Bajt sumy kontrolnej zawiera sumę wszystkich innych bajtów w bloku, włączając oba znaczniki. Suma obliczana jest z krążącym przeniesieniem, to znaczy, w miarę dodawania wartości nowych bajtów, dodawany jest także bit przeniesienia

Jest to przyczyną potencjalnych problemów. Oprogramowanie, które zapisuje na taśmę, musi zostawić wystarczająco długie przerwy, aby początki bloków nie były gubione podczas wprowadzania.

W celu odróżnienia jednego bloku od drugiego procedura obsługi magnetofonu dodaje przerwy międzyblokowe (IRG). Długość IRG zależy od trybu otwarcia zbioru do zapisu. Istnieją dwa typy IRG: normalna i skrócona. Normalna przerwa międzyblokowa używana jest do odczytu przemieszanego z przetwarzaniem, tzn. taśma zawsze zatrzymuje się po odczytaniu każdego bloku. Jeśli komputer zatrzyma taśmę i wykona swoje obliczenia wystarczająco szybko, może się zdarzyć, że następny rozkaz odczytu pojawi się przed zatrzymaniem taśmy. W trybie skróconego IRG taśma nie jest zatrzymywana między blokami, ani podczas zapisu, ani podczas odczytu. Powszechnym zastosowaniem tego trybu jest przechowywanie programów w języku BASIC za pomocą instrukcji CSAVE

Pamięć zewnętrzna może być używana do przechowywania programów i zbiorów danych. Zbiory danych dzielą się na rekordy. Liczba rekordów ograniczona jest długością taśmy. Rekord może mieć dowolną liczbę pól, chociaż wszystkie rekordy w tym samym zbiorze powinny mieć taką samą liczbę pól. Jeśli pole nie jest używane, może mieć wartość zerową lub być puste. Dla oznaczenia końca zbioru komputer zapisuje specjalny rekord.

Zbiory danych zapisane na taśmie magnetycznej mają jedno poważne ograniczenie: dane w nich zawarte mogą być osiąga-

ne tylko sekwencyjnie. Oznacza to, że musimy zaczynać czytanie od początku zbioru i czytać do końca. Nie możemy również dodawać, ani kasować rekordów.

Komputer nie wysyła danych do i z magnetofonu znak po znaku, lecz wykonuje to w 128 bajtowych blokach. Rekord może zajmować więcej niż jeden blok, dokładnie jeden blok lub mniej niż jeden blok, w zależności od jego długości. Ogólnie mówiąc, program nie martwi się o to jak blokowane są rekordy danych. Komputer wykonuje to automatycznie.

Atari komunikuje się z wszystkimi urządzeniami pośrednio, za pomocą kanałów wejścia/wyjścia. Istnieje osiem kanałów, numerowanych od 0 do 7. W celu komunikowania się z określonym urządzeniem, program najpierw podłącza urządzenie do jednego z kanałów. Jeśli program żąda wprowadzenia danych z tego kanału, nadchodzą one z żądanego urządzenia. Podobnie, program kieruje wyprowadzane dane do kanału, który przesyła je do wybranego wcześniej urządzenia wyjściowego.

BASIC rezerwuje kanały o numerach 0, 6 i 7 do określonych działań. Kanał nr 0 jest stale zarezerwowany dla edytora. Instrukcje wprowadzania z klawiatury, w trybie bezpośrednim, używają kanału nr 0. Wszystkie proste instrukcje INPUT i PRINT używają kanału 0. Kanał 7 używany jest do pewnych operacji drukarki oraz wprowadzania i przechowywania programów. Specjalne instrukcje graficzne używają kanału nr 6. Całkowicie dostępne dla programów są kanały o numerach od 1 do 5, kanały 6 i 7 — w ograniczonym zakresie.

Instrukcja OPEN podłącza kanał do wybranego urządzenia. Kolejne instrukcje wprowadzania lub wyprowadzania mogą mieć wtedy dostęp do tego urządzenia poprzez numer kanału. Postać instrukcji OPEN jest następująca

OPEN #k, a, t, nazwa

Pierwsza liczba po słowie OPEN jest numerem kanału (0<k<7).

Ostatni parametr jest nazwą urządzenia (np. C:, E:, S:, P:)

Druga liczba określa rodzaj działania (akcji), które będzie dozwolone w kanale. Może to być wprowadzenie (4), wyprowadzanie (8) lub oba jednocześnie (12). Jednak działanie musi mieć sens. Nie można, na przykład, otworzyć klawiatury dla wyprowadzania.

Jest jeszcze jeden parametr, tuż przed nazwą urządzenia. Jest on używany w zależności od urządzenia. Dla klawiatury (K:) oraz edytora (E:) jest ignorowany. Dla ekranu (S:) może wybierać tryb wyświetlania, tekstowy lub graficzny. W przypadku pamięci taśmowej parametr ten określa wielkość przerwy międzyblokowej (IRG). Zerowa wartość parametru wybiera normalną przerwę międzyblokową, a 128 — skróconą.

Kanał pozostaje otwarty do zakończenia działania programu lub wyraźnego zamknięcia go. Gdy program kończy działanie instrukcją END, zamykane są wszystkie kanały. Lecz jeśli zatrzyma się w wyniku realizacji instrukcji STOP, naciśnięcia klawisza BREAK lub błędu, wszystkie otwarte kanały pozostają otwarte. Instrukcja CLOSE zamyka tylko jeden kanał.

Informacja w zbiorze taśmowym staje się dostępna po otwarciu zbioru i pozostaje dostępna do czasu jego zamknięcia. Zbiór taśmowy nie może być jednocześnie otwarty dla wprowadzania i wyprowadzania.

Po napotkaniu instrukcji OPEN komputer sygnalizuje ten fakt dźwiękiem, jeden sygnał dla wprowadzania, dwa — dla wyprowadzania. Następnie czeka na ustawienie taśmy w odpowiedniej pozycji i naciśnięcie dowolnego klawisza. Dopiero wtedy rozpoczyna się ruch taśmy i komputer odczytuje lub zapisuje 20-sekundowy sygnał pilotujący. W tym czasie nie wykonuje on innych instrukcji!

Ważne jest zamknięcie nieaktywnego zbioru, ponieważ uwalnia kanał. Szczególnie ważne jest zamknięcie zbioru, który został otwarty do zapisu. Ominięcie tego może spowodować utratę danych. Bufor magnetofonu może być częściowo wypełniony danymi. Zamykanie zbioru powoduje wyprowadzenie tych danych z buforu do ostatniego bloku na taśmie. Jeśli zbiór nie zostanie zamknięty, zawartość częściowo zapełnionego buforu nigdy nie zostanie wyprowadzona.

Dane mogą być wyprowadzane do zbioru za pomocą instrukcji PRINT# lub PUT. Obie instrukcje wysyłają dane do otwartego kanału. Nie jest ważne do jakiego urządzenia podłączony jest kanał.

Instrukcja PRINT# wyprowadza wartości numeryczne i tekstowe w kodzie ATASCII. Dla rozdzielenia pozycji należy używać raczej średników niż przecinków. Przecinki są dozwolone, lecz powodują wyprowadzenie dodatkowych spacji.

Każda oddzielna wartość wysyłana do magnetofonu musi kończyć się znakiem EOL (end of line). Podczas wprowadzania znak EOL określa, gdzie ona się kończy. Jednym ze sposobów zapewnienia, że wystąpi znak EOL, jest wyprowadzenie każdej wartości oddzielną instrukcją PRINT#. Nigdy nie należy jej kończyć średnikiem lub przecinkiem. Innym sposobem wyprowadzenia znaku EOL jest funkcja CHR$ **(program nr 1).**

```
FK 1 REM ***************
FJ 2 REM *             *
XA 3 REM * Program nr 1 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
II 10 PRINT "Ustaw tasme i nacisn
   ij <RETURN>";
YU 20 OPEN #3,8,0,"C:"
QY 30 PRINT #3;A;CHR$(155);B;CHR$
   (155);C
MG 40 CLOSE #3
```

Każda instrukcja PUT wyprowadza pojedynczą wartość numeryczną od 0 do 255. Zajmuje ona tyle samo miejsca co jeden

```
FK 1 REM ***************
FJ 2 REM *             *
XU 3 REM * Program nr 2 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
II 10 PRINT "Ustaw tasme i nacisn
   ij <RETURN>";
YU 20 OPEN #3,8,0,"C:"
PV 29 REM Wyprowadzenie pustego r
   ekordu
SH 30 FOR J=1 TO 128
BS 40 PUT #3,0
JC 50 NEXT J
KA 60 PRINT #3;"Poczatek danych"
MJ 70 CLOSE #3
YW 80 END
```

znak ATASCII. **Program nr 2** wyprowadza pusty blok po otwarciu kanału wyjściowego.

Instrukcja INPUT# wprowadza wartości przechowane na taśmie instrukcją PRINT#. Określony przez nią kanał musi być otwarty dla wprowadzania z magnetofonu. Aby zobaczyć jak to działa, użyjmy najpierw **programu nr 3** do utworzenia zbioru danych.

```
FK 1 REM ***************
FJ 2 REM *             *
YO 3 REM * Program nr 3 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
PX 10 DIM A$(120)
WY 20 PRINT "Jaki komunikat chces
   z zapisac?"
YC 30 INPUT A$
XE 40 PRINT
IM 50 PRINT "Ustaw tasme i nacisn
   ij <RETURN>";
JK 60 OPEN #3,8,0,"C:"
SN 70 PRINT #3;A$
MK 80 CLOSE #3
YX 90 END
```

Następny program, nr 4, używa instrukcji INPUT# do odczytania komunikatu zapisanego

```
FK 1 REM ***************
FJ 2 REM *             *
ZI 3 REM * Program nr 4 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
PX 10 DIM A$(120)
IJ 20 PRINT "Ustaw tasme i nacisn
   ij <RETURN>";
WZ 30 OPEN #3,4,0,"C:"
UB 40 INPUT #3;A$
WN 50 PRINT A$
MI 60 CLOSE #3
YV 70 END
```

nego przez poprzedni program. Nie zapomnijmy o przewinięciu taśmy!

Instrukcja INPUT# interpretuje dane ze zbioru taśmowego jako kody ATASCII. Za każdym razem, gdy napotka znak EOL (kod ATASCII 155), podstawia znaki przeczytane od ostatniego EOL do następnej zmiennej w swoim wykazie. Jeśli zmienna jest numeryczna, INPUT# tłumaczy wczytane znaki na wartość numeryczną.

Każda instrukcja GET wprowadza jedną wartość numeryczną. Program musi zadecydować, jak tę wartość interpretować. Na przykład, może on interpretować wartość jako kod ATASCII i zamienić na znak za

```
FK 1 REM ***************
FJ 2 REM *             *
AC 3 REM * Program nr 5 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
II 10 PRINT "Ustaw tasme i nacisn
   ij <RETURN>";
WY 20 OPEN #3,4,0,"C:"
EK 30 GET #3,A
JM 40 IF A<>155 THEN PRINT CHR$(A
   );:GOTO 30
XF 50 PRINT
MI 60 CLOSE #3
YV 70 END
```

pomocą funkcji CHR$. **Program nr 5** używa funkcji GET do odczytania tego samego zbioru co program nr 4.

Program nr 6 jest odpowiedzią na częste pytania czytelników, dotyczące przechowywania na kasecie zawartości tablic numerycznych. W liniach 20—50 tego programu poszczególnym elementom tablicy A nadawane są wartości losowe z przedziału (0,100). W linii 60 następuje otwarcie zbioru

```
FK 1 REM ****************
FJ 2 REM *              *
AW 3 REM * Prosram nr 6 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
JL 10 DIM A(10,10),B(10,10)
GC 20 FOR I=1 TO 10
GL 30 FOR J=1 TO 10
MT 40 A(I,J)=RND(0)*100
OW 50 NEXT J:NEXT I
YE 60 OPEN #1,8,0,"C:"
GH 70 FOR I=1 TO 10
GQ 80 FOR J=1 TO 10
JW 90 PRINT #1;A(I,J)
EJ 100 NEXT J:NEXT I
KZ 110 CLOSE #1
CM 120 PRINT "Przewin tasme i nac
   isnij <RETURN>."
KD 130 OPEN #2,4,0,"C:"
EU 140 FOR I=1 TO 10:FOR J=1 TO 1
   0
AW 150 INPUT #2;C:B(I,J)=C
EV 160 NEXT J:NEXT I
LL 170 CLOSE #1
FC 180 FOR I=1 TO 10:FOR J=1 TO 1
   0
GL 190 PRINT A(I,J),B(I,J)
EK 200 NEXT J:NEXT I
NT 210 END
```

do zapisu, następnie zapisanie tablicy na taśmie (linie 70—110). Linie 130—170 wprowadzają zapisane wartości do tablicy B. Proszę zwrócić uwagę na kolejność wyprowadzania, a następnie wprowadzania danych. Tablica dwuwymiarowa może być zapisywana (i odczytywana) kolumnami lub wierszami. Jeśli nie będziemy wystarczająco ostrożni, może okazać się, że przypadkowo uzyskaliśmy macierz transponowaną.

Rozważmy praktyczne wykorzystanie zbiorów taśmowych: wykaz adresów bliskich i znajomych. Potrzebne będą dwa programy. Pierwszy będzie służył do wprowadzania nazwisk i adresów z klawiatury i przechowywania ich na taśmie. Drugi będzie wprowadzał nazwiska i adresy z taśmy i wyświetlał je na ekranie.

Każdy z rekordów zbioru adresowego ma pięć pól: imię, nazwisko, ulica, miasto i kod pocztowy. Oba programy używają tych samych nazw zmiennych dla pól: IM$, NAZ$, UL$, MI$, KOD$.

Program nr 7, po zadeklarowaniu zmiennych, otwiera kanał wyjściowy dla magnetofonu i wyprowadza pusty blok. Podczas

```
FK 1 REM ****************
FJ 2 REM *              *
BQ 3 REM * Prosram nr 7 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
EY 10 DIM PRMT$(40),MSK$(40),BL$(
   40),ERM$(13)
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$
   (2)=MSK$
GM 40 ERRHDL=8000
SF 50 DIM IM$(20),NAZ$(20),UL$(20
   ),MI$(20),KOD$(6),R$(20),EOL$(
   1)
HW 60 EOL$=CHR$(155)
ND 70 GOSUB 3000
IC 90 GOTO 1000
XA 97 REM *********************
XR 98 REM *  Kasowanie linii  *
XE 99 REM *********************
EI 100 FOR J=L1 TO L2
VX 110 POSITION 0,J
ZP 120 PRINT BL$(1,39);
GH 130 NEXT J
ZF 140 RETURN
VX 165 REM *********************
FY 166 REM *  Komunikat bledu  *
WD 167 REM *********************
NE 170 FOR J=1 TO 3
MT 180 POSITION 20,0
JV 190 ? ERM$;" ERROR";CHR$(253);
SM 200 FOR J1=1 TO 100:NEXT J1
MG 210 POSITION 20,0
QQ 220 ? BL$(1,19)
GI 230 NEXT J
ZG 240 RETURN
EG 250 L1=20:L2=23:GOSUB 100
OA 260 POSITION 2,20
ZM 270 RETURN
MI 596 REM ****************
BK 597 REM *  Podprosram  *
FK 598 REM * wprowadzania *
MR 599 REM ****************
IB 600 TRAP ERRHDL.
TO 620 GOSUB 250
AV 630 POSITION IC+1,IR
GB 640 ? MSK$(1,IL)
FE 650 POSITION IC,IR
HD 660 INPUT R$
ZE 665 FOR J1=1 TO IL:IF R$(J1,J1
   )<>MSK$(1,1) THEN NEXT J1
MI 666 R$=R$(1,J1-1)
TK 670 IF R$="?" THEN GOTO 630
MN 680 IF LO>HI THEN 750
GF 700 IF R$="" THEN R$="0"
UW 710 R=VAL(R$)
BS 720 IF R>=LO AND R<=HI THEN RE
   TURN
VY 730 ERM$="Zakres liczby"
GP 740 GOSUB 170:GOTO 620
PG 750 IF LEN(R$)<=IL THEN RETURN
US 760 ERM$="Dlusosc ciasu"
GV 770 GOSUB 170:GOTO 620
IK 796 REM ********************
TL 797 REM * Wprowadzanie z   *
ZF 798 REM *   propozycja     *
IT 799 REM ********************
CU 800 L1=1:L2=2:GOSUB 100
HW 810 POSITION 2,1
AY 820 ? PRMT$
XI 830 IC=2:IR=2:GOSUB 600
ZM 840 RETURN
ZV 1000 ? CHR$(125);"Podaj nastep
   ujace dane"
XN 1010 ? :? :?
MA 1020 ? "1)      Imie:"
PE 1030 ? "2) Nazwisko:"
DJ 1040 ? "3)     Ulica:"
JJ 1050 ? "4)    Miasto:"
VV 1060 ? "5)       Kod:"
WZ 1100 FOR F=1 TO 5
XA 1110 GOSUB 2000
DQ 1120 NEXT F
FI 1130 PRMT$="Czy wprowadzasz zm
   iany?"
IO 1140 LO=1:HI=0
VD 1150 IL=1:GOSUB 800
MO 1159 REM Analiza odpowiedzi
HW 1160 IF R$="N" OR R$="n" THEN
   1300
PB 1170 IF R$="T" OR R$="t" THEN
   1200
UL 1180 ERM$="PODAJ T LUB N":GOSU
   B 170
PQ 1190 GOTO 1130
HV 1200 PRMT$="Ktore pole"
KZ 1210 LO=1:HI=5
UW 1220 IL=1:GOSUB 800
WF 1230 F=R:GOSUB 2000:GOTO 1130
HI 1300 POSITION 8,21
YG 1310 ? "* * * PROSZE CZEKAC *
   * *";
MX 1319 REM Wyprowadzenie rekordu
SI 1320 ? #1;IM$;EOL$;NAZ$;EOL$;U
   L$;EOL$;MI$;EOL$;KOD$
IP 1330 ? CHR$(156)
VT 1340 POSITION 2,1
SO 1350 PRMT$="Wprowadzasz dalej?
   "
IY 1360 LO=1:HI=0
VN 1370 IL=1:GOSUB 800
MW 1380 IF R$="T" OR R$="t" THEN
   1000
SJ 1390 IF R$<>"N" AND R$<>"n" TH
   EN 1350
YT 1400 FOR J=1 TO 5
XO 1410 ? #1;CHR$(253);EOL$;
FO 1420 NEXT J
NA 1430 CLOSE #1
FF 1500 END
IB 2000 LO=1:HI=0
RZ 2010 ON F GOTO 2100,2200,2300,
   2400,2500
WX 2100 IC=13:IR=4:IL=19:GOSUB 60
   0
MT 2110 IM$=R$:RETURN
XO 2200 IC=13:IR=5:IL=19:GOSUB 60
   0
PK 2210 NAZ$=R$:RETURN
YF 2300 IC=13:IR=6:IL=19:GOSUB 60
   0
PK 2310 UL$=R$:RETURN
YW 2400 IC=13:IR=7:IL=19:GOSUB 60
   0
MV 2410 MI$=R$:RETURN
LI 2500 IC=13:IR=8:IL=6:GOSUB 600
LY 2510 KOD$=R$:RETURN
CX 2999 REM Otwarcie kanalu wyjsc
   iowego
MX 3000 ? CHR$(125);"Przygotuj ka
   sete i nacisnij RETURN"
BA 3010 OPEN #1,8,0,"C:"
DV 3020 FOR J1=1 TO 128:PUT #1,0:
   NEXT J1
AK 3030 RETURN
HN 8000 E=PEEK(195)
BT 8010 EL=PEEK(187)*256+PEEK(186
   )
BC 8020 IF E=3 OR E=8 THEN 8100
VP 8030 ? "ZNALEZIONO BLAD ";E
YF 8040 ? "W LINII NR.";EL
FS 8080 END
IF 8100 REM
LW 8110 ERM$="INPUT"
YT 8120 GOSUB 170
VQ 8130 TRAP ERRHDL
KX 8140 GOTO EL
```

wprowadzania rekordów użytkownik może go sprawdzić i poprawić w przypadku błędu.

Po zakończeniu wprowadzania danych program wyprowadza jeden dodatkowy rekord (zawierający specjalne wartości we wszystkich polach), oznaczający koniec zbioru, a następnie zamyka zbiór.

Program nr 8 otwiera kanał wejściowy dla magnetofonu, wprowadza z taśmy po-

```
FK 1 REM ****************
FJ 2 REM *              *
CK 3 REM * Prosram nr 8 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
JB 50 DIM IM$(20),NAZ$(20),UL$(20
   ),MI$(20),KOD$(6),R$(20)
NO 70 GOSUB 3100
FV 1000 ? CHR$(125)
TW 1010 ? "KSIAZKA ADRESOWA"
OV 1020 POSITION 2,21:? "Nacisnij
    CTRL-1 dla zatrzymania."
RI 1030 POSITION 0,22:? "Ponowne
   nacisniecie wznowi przesladani
   e."
HM 1050 POSITION 2,2:? "Nacisnij
   RETURN sdy bedziesz sotowy"
LB 1070 INPUT R$
OV 1080 OPEN #4,8,0,"S:"
LX 1090 INPUT #1,IM$,NAZ$,UL$,MI$
   ,KOD$
SS 1099 REM Sprawdzenie, czy koni
   ec zbioru
IS 1100 IF IM$=CHR$(253) THEN 120
   0
GZ 1110 ? IM$
OM 1120 ? NAZ$
```

```
KO 1130 ? UL$
HE 1140 ? MI$
KH 1150 ? KOD$
KG 1160 ? :?
RY 1170 GOTO 1090
VP 1200 CLOSE #1:CLOSE #4
FF 1900 END
LO 3099 REM Otwarcie kanalu dla w
   prowadzania
YV 3100 OPEN #1,4,0,"C:"
UL 3119 REM Wprowadzenie pustego
   rekordu
UB 3120 FOR J1=1 TO 128:GET #1,R:
   NEXT J1
AM 3130 RETURN
```

szczególne rekordy i wyświetla je na ekranie

Istnieją trzy instrukcje służące do przechowywania programów w języku BASIC na kasecie: CSAVE, LIST i SAVE. Każda z nich ma swój odpowiednik: CLOAD, ENTER i LOAD, które wprowadzają programy z powrotem do pamięci.

CSAVE jest specjalną instrukcją do przechowywania programów tylko na kasecie. LIST i SAVE wyprowadzają program na dowolne urządzenie, w szczególności na magnetofon. Dla instrukcji LIST i SAVE musimy określić urządzenie, np. SAVE „C:".

Zarówno CSAVE, jak i SAVE „C:" zawsze przechowują cały program z pamięci. Instrukcją LIST „C:" można przechować całość lub część programu. Podobnie jak dla innych postaci instrukcji LIST, możemy określić pierwszą i ostatnią linię programu, które mają być wyprowadzone (np. LIST „C:", 1000, 3000)

Instrukcja CLOAD wprowadza programy przechowane na kasecie instrukcją CSAVE. ENTER i LOAD wprowadzają programy z dowolnego urządzenia. ENTER „C:" może wprowadzić tylko programy przechowane za pomocą instrukcji LIST „C:". LOAD „C:" może wprowadzić tylko programy przechowane za pomocą SAVE „C:".

Jak widać z powyższego przeglądu, żadna z instrukcji przechowywania programów na kasecie nie wymaga podania nazwy programu. Każdy z programów tworzy na taśmie odrębny zbiór, który nie jest rozróżniany przez komputer. Wprowadzany jest program, na początku którego ustawiona jest taśma magnetyczna. Identyfikacja programu może być dokonana tylko przez użytkownika, albo za pomocą licznika taśmy, albo dopiero po wprowadzeniu programu do pamięci komputera.

Instrukcje CLOAD i LOAD kasują program rezydujący w pamięci przed wprowadzeniem nowego. Instrukcja ENTER łączy wprowadzany program z istniejącym w pamięci. Jeśli wprowadzane wiersze programu mają takie same numery jak istniejące, to te ostatnie są podmieniane przez nowe.

Zapisywanie programu jest zewnętrznie takie samo, bez względu na użytą instrukcję CSAVE, LIST „C:" lub SAVE „C:". Lecz każda z nich zapisuje programy w innym formacie.

Instrukcja LIST wyprowadza programy w takim samym formacie bez względu na urządzenie. Wysyła ona kod ATASCII każdego znaku programu.

Instrukcje CSAVE i SAVE podmieniają słowa kluczowe jednoznakowymi symbolami. Różnica między nimi polega na długości przerw między blokami na taśmie. Instrukcja CSAVE zapisuje programy gęściej niż SAVE „C:", z krótszymi przerwami międzyblokowymi. Zatem przechowywanie i wprowadzanie programów odbywa się trochę szybciej za pomocą CSAVE i CLOAD niż za pomocą SAVE „C:" i LOAD „C:".

Program może być wprowadzony z taśmy magnetycznej do pamięci również za pomocą instrukcji RUN „C:", która jest połączeniem instrukcji LOAD „C:" i RUN. Zatem programy muszą być przechowane na taśmie instrukcją SAVE „C:". Instrukcja RUN „C:" działa jednakowo w trybie programowym i bezpośrednim. Program, który zawiera instrukcję RUN „C:", może wprowadzić i uruchomić inny program. Poważną wadą tego sposobu jest kasowanie wartości wszystkich zmiennych przed wprowadzeniem następnego programu. Dlatego nie można używać wartości, które były wprowadzone lub obliczone przez wcześniejszy program. Dodatkowo zapis na taśmie ma długie przerwy międzyblokowe.

Istnieje sposób automatycznego wprowadzania i uruchamiania programów w języku BASIC zapisanych z krótkimi przerwami międzyblokowymi. Dodatkowo można to zrobić bez podawania jakiejkolwiek instrukcji. Niezbędne jest jednak umieszczenie na taśmie specjalnego programu wprowadzającego, który będzie wprowadzany do pamięci w chwili włączania zasilania, jako część procesu inicjowania systemu. Podczas włączania komputera musi być naciśnięty klawisz START. Po wprowadzeniu programu wprowadzającego przejmuje on sterowanie i wprowadza, a następnie uruchamia program przechowany bezpośrednio za nim.

Program, o którym mowa, można spotkać przed programami w języku BASIC wprowadzanymi poprzez włączenie komputera z naciśniętym klawiszem START (np. Konwerter 80k). Jeśli dysponujemy takim programem, wystarczy skopiować program wprowadzający. Dla czytelników, którzy nie mają skąd wziąć tego programu, przygotowaliśmy **program nr 9**, który po wprowa-

```
FK 1 REM ****************
FJ 2 REM *              *
DE 3 REM * Program nr 9 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
KP 8 GRAPHICS 0:POSITION 10,10:?
   "Prosze czekac..."
OD 10 DIM A$(270):I=0
GQ 20 READ A:IF A=-1 THEN 100
BJ 30 I=I+1:A$(I,I)=CHR$(A)
RU 40 GOTO 20
TR 100 OPEN #1,8,128,"C:"
PO 110 POKE 766,1:? #1;A$;:POKE 7
   66,0
PF 120 CLOSE #1:GRAPHICS 0
EB 130 ? "KONIEC"
TP 140 STOP
ND 2060 DATA 0,2,0,6,16,6,169,60,
   141,2,211,169,0,133,65,96,169,
   148,141,197
EY 2070 DATA 2,173,48,2,133,128,1
   73,49,2,133,129,160,4,177,128,
   133,130,200,177,128
ZO 2080 DATA 133,131,230,131,230,
   131,160,169,162,0,189,91,6,145
   ,130,232,136,208,247,169
WR 2090 DATA 13,141,74,3,169,0,13
   3,84,169,254,141,179,2,169,127
   ,141,180,2,76,0
IT 2100 DATA 160,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0
HB 2110 DATA 0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0
AE 2120 DATA 0,0,0,0,0,0,0,0,9,9,
   4,0,0,56,8,50,36,33,8,50
QP 2130 DATA 51,53,29,56,26,16,14
   ,50,39,26,18,17,12,20,22,23,0,
   37,43,47
NS 2140 DATA 48,26,18,17,12,18,20
   ,24,0,37,43,0,0,47,48,26,2,155
   ,68,44
IA 2150 DATA 255,137,155,150,0,68
   ,137,40,52,137,40,151,137,216,
   253,130,2,29,4,56
QQ 2160 DATA 26,9,16,20,8,4,56,0,
   45,41,36,0,0,0,0,0,0,0,0
HT 2170 DATA 0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0
HW 2180 DATA 0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0
EC 2190 DATA 0,0,-1
```

dzeniu i uruchomieniu, tworzy na taśmie magnetycznej właśnie taki program wprowadzający.

Oczywiście program ten może być bardziej rozbudowany i dostosowany do programu, który będzie przez niego wprowadzany. Chcieliśmy jednak zaproponować czytelnikom program uniwersalny, który umożliwia wprowadzenie do pamięci dowolnego programu w języku BASIC. Nie przeszkadza to w zindywidualizowaniu procesu wprowadzania programu. Zamiast dopasowywania programu wprowadzającego, możemy dopasować proces wprowadzania.

Opisywany problem nie został przez nas wydumany, lecz jest odpowiedzią na liczne listy czytelników, których interesowało automatyczne wprowadzenie i uruchomienie programu oraz wprowadzanie programu po uprzednim wyświetleniu charakterystycznej dla niego czołówki. Pierwsza z tych spraw została już, mamy nadzieję, wyjaśniona.

Druga sprawa jest bardziej skomplikowana. Chcemy uzyskać następujący efekt: po odpowiednim ustawieniu taśmy magnetycznej włączamy komputer z wciśniętym klawiszem START, automatycznie pojawia się czołówka, charakterystyczna dla danego programu, która pozostaje na ekranie do chwili załadowania i automatycznego uruchomienia właściwego programu.

Rozwiązanie powyższego zadania wymaga podjęcia decyzji o sposobie tworzenia czołówki programu. Czy przygotowana będzie przez jeden z ogólnie dostępnych programów graficznych (np. Koala Micro, Magic Painter, Design Master, itp.), czy utworzona przez własny program?

Rozważmy pierwszy wariant. Proponujemy tworzenie czołówki za pomocą programu Magic Painter. Wybraliśmy go dlatego, że prawdopodobnie jest najbardziej rozpowszechniony. Dwukrotnie emitowany był w audycji „Radiokomputer nocą", dostępny jest na giełdach i w wypożyczalniach. Program ten umożliwia przygotowanie dowolnej czołówki, a następnie przechowanie jej na taśmie.

Aby mieć wszystkie niezbędne elementy, musimy jeszcze wprowadzić i przechować na taśmie magnetycznej (za pomocą CSAVE) **program nr 10**. Jego zadaniem jest wprowadzenie i wyświetlenie czołówki przy-

```
QO 1 REM *****************
MP 2 REM * Program nr 10 *
QQ 3 REM *****************
BT 4 DIM ML$(28),X$(40):GRAPHICS
   7+16
QF 5 FOR I=1 TO 28:READ A:ML$(I,I
   )=CHR$(A):NEXT I
ZL 6 DATA 104,104,104,170,104,104
   ,157,66,3,104,157,69,3,104,157
   ,68,3,104,157,73,3,104,157,72,
   3,76,86,228
```

```
ID 7 DL=PEEK(560)+256*PEEK(561)
MI 8 ADRR=PEEK(DL+4)+256*PEEK(DL+
   5)
EW 9 POKE 764,12
AF 10 OPEN #1,4,128,"C:"
DY 30 FOR I=0 TO 4
LJ 40 GET #1,A:POKE 708+I,A
IT 50 NEXT I
EG 60 GET #1,B
TQ 70 X=USR(ADR(ML$),16,7,ADRR,43
   00)
LO 80 CLOSE #1
EX 90 REM LADOWACZ PROGRAMU
LF 100 DATA 162,253,154,169,183,7
   2,169,84,72,169,4,32,182,187,1
   69,255,76,4,187
JX 110 FOR I=1 TO 19:READ A
QI 120 X$(I,I)=CHR$(A)
FX 130 NEXT I
FN 140 POKE 764,12:X=USR(ADR(X$))
```

gotowanej programem Magic Painter (realizuje to procedura przechowywana w zmiennej ML$), a następnie wprowadzenie i uruchomienie właściwego programu (wykonuje to procedura przechowywana w zmiennej X$)

Możemy już przystąpić do przygotowania taśmy wynikowej

1. Zapisujemy program wprowadzający

2. Zapisujemy, instrukcją CSAVE, **program nr 10**

3. Zapisujemy na taśmie obraz czołówki, utworzony programem Magic Painter.

4. Przechowujemy, za pomocą CSAVE, właściwy program

Powyższe możemy wykonać dowolnym sposobem, ważne jest otrzymanie takiej kolejności poszczególnych zbiorów. Po przewinięciu taśmy możemy sprawdzić uzyskany efekt. Wprowadzanie programu trwa nieco dłużej (sama czołówka wprowadzana jest około 100 sekund), lecz jest bardziej efektowne

W drugim wariancie czołówka programu tworzona jest przez własną procedurę. Proponujemy włączyć ją do programu nr 10 w miejsce linii wprowadzających dotychczas obraz czołówki. Jej postać zależy od naszych umiejętności oraz wyobraźni. Może to być sam tytuł programu napisany dużymi literami (tryb graficzny 1 lub 2), rysunek wykonany w trybie 0, rysunek w dowolnym trybie graficznym z napisem w oknie tekstowym. Może to być również bardziej rozbudowany obraz, z tytułem napisanym dużymi literami i stosownym rysunkiem, uzyskany poprzez utworzenie własnego programu wyświetlania

W tym wariancie na taśmie magnetycznej musimy zapisać następujące zbiory

1. Program wprowadzający

2. Zmodyfikowany program nr 10, który tworzy obraz czołówki oraz wprowadza właściwy program

3. Właściwy program

Programy wymienione w punktach 2 i 3 powinny być zapisane instrukcją CSAVE

Zauważmy, że procedura przechowywana w zmiennej X$ programu nr 10, może być wykorzystana do łączenia programów zapisanych na taśmie za pomocą CSAVE. Podobnie jak w przypadku używania instrukcji RUN „C:", program nie może używać wartości, które były wprowadzone lub obliczone przez wcześniejszy program.

Podczas przygotowania niniejszego artykułu korzystaliśmy z następujących publikacji: De RE Atari oraz Your Atari Computer

Tomasz MROWIEC
Ludwik PIELA

PROGRAM 42

ZX SPECTRUM

TABELARYZYCJA

```
 100 REM program TABFUN

 110 BORDER 4: DIM v(256): DIM v
$(256): DIM x(256): DIM c$(32)
 120 INPUT PAPER 7;"funkcja y";
PAPER 4;" = "; LINE f$: IF f$=""
 THEN GO SUB 580: GO TO 120
 130 INPUT "x min = ";xmin,"x ma
x = ";xmax
 140 IF xmax<xmin OR xmax=xmin T
HEN GO SUB 560: GO TO 130
 150 CLS : LET st=(xmax-xmin)/25
4
 160 REM oblicz. tab. x() i v()
 170 LET m=1: LET ymin=10↑38: LE
T ymax=-ymin: GO SUB 620: PRINT
AT 0,tab;t$
 180 FOR z=1 TO 255: LET x=xmin+
st*(z-1): LET x(z)=x: LET v(z)=V
AL (f$)
 190 IF ymin>v(z) THEN LET ymin=
v(z)
 200 IF ymax<v(z) THEN LET ymax=
v(z)
 210 PLOT z-1,157: NEXT z
 220 BEEP .1,2: GO SUB 590
 230 REM proporcje wydruku
 240 LET z0=0: LET dy=ymax-ymin
 250 PRINT AT 0,tab;t$
 260 PRINT #0;AT 0,tab1; BRIGHT
1; PAPER 7;ymin;"<y<";ymax
 270 FOR n=1 TO 255: LET x=x(n):
 LET v$(n)=CHR$ INT ((v(n)-ymin)
*166/dy): PLOT n-1,166: NEXT n
 280 CLS : PRINT AT 0,tab;t$: PL
OT 0,CODE v$(1)
 290 PLOT 128,81: DRAW 0,4: PLOT
 126,83: DRAW 4,0
 300 IF xmin<0 AND xmax>0 THEN L
ET z0=INT (256*(xmin/(xmin-xmax)
)): PLOT z0,0: DRAW 0,166
 310 IF ymax>0 AND ymin<0 THEN P
LOT 0,ymin*166/dy: DRAW 255,0
 320 PRINT #0;AT 0,tab1; BRIGHT
1; PAPER 7;ymin;"<y<";ymax
 330 REM pocz. petli PLOT/DRAW
 340 FOR n=2 TO 254
 350 IF v(n-1)<0 AND v(n)>0 OR v
(n-1)>0 AND v(n)<0 THEN GO SUB
480
 360 IF v(n)=0 THEN GO SUB 530
 370 IF (v(n-1)<v(n)) AND (v(n)>
v(n+1)) THEN GO SUB 510
 380 IF (v(n-1)>v(n)) AND (v(n)<
v(n+1)) THEN GO SUB 540
 390 IF n=z0 THEN GO SUB 650
 410 PLOT n-1,CODE (v$(n-1)): DR
AW 1,CODE (v$(n))-CODE (v$(n-1))
 420 NEXT n
 430 REM nowa funkcja ?
 440 GO SUB 560:
     PRINT #0;AT 1,0; PAPER 7;"
nowe Wartosci,nowa Funkcja,COPY?
": PAUSE 0: IF INKEY$="f" THEN G
O TO 120
 450 IF INKEY$="c" THEN PLOT 0,0
: DRAW 255,0: DRAW 0,167: DRAW -
255,0: DRAW 0,-167: LPRINT : COP
Y
 460 GO TO 130
 470 REM podprogramy
 480 BEEP .1,14: FOR q=166 TO 1
STEP -3: PLOT n-1,q: DRAW 0,-1:
NEXT q
 490 LET yp=v(n-1): LET yk=v(n):
 LET xp=x(n-1): LET xk=x(n): LET
 xz=xp+(xk-xp)*(ABS yp/(ABS yk+A
BS yp)): LET xz=.001*INT (1000*x
z+.0005): PRINT "x0  =";: IF xz>
0 THEN PRINT "+";
```

ZAGADKA

```
   1 REM    PROGRAM - ZAGADKA
  10 LET a$="": FOR n=1 TO 10:
READ data: LET a$=a$+CHR$ data:
NEXT n
  15 SAVE a$ LINE 10
  20 DATA 22,11,15,19,1,137,8,21
,1,79
```

NAGRAJ PROGRAM NA TAŚMĘ,WYŁĄCZ KOMPUTER I WPROWADŹ PROGRAM Z TAŚMY. SKĄD WZIĄŁ SIĘ ZNAK PO ŚRODKU EKRANU ?
CZEKAMY NA LISTY CZYTELNIKÓW !

ZNAKI x 24

```
  10 BORDER 4: PAPER 4: PAUSE 0:
 LET a=CODE INKEY$: CLS : PRINT
AT 21,0; INK 4;CHR$ a: LET a$=CH
R$ a+CHR$ a+CHR$ a
  20 BEEP .1,2: FOR n=7 TO 0 STE
P -1: FOR m=0 TO 7
  30 IF POINT (m,n)=1 THEN PRINT
 BRIGHT 1; PAPER 7;AT 18-3*n,4+3
*m;a$;AT 19-3*n,4+3*m;a$;AT 20-3
*n,4+3*m;a$
  40 NEXT m: NEXT n: GO TO 10
  50 SAVE "24 x" LINE 10
```

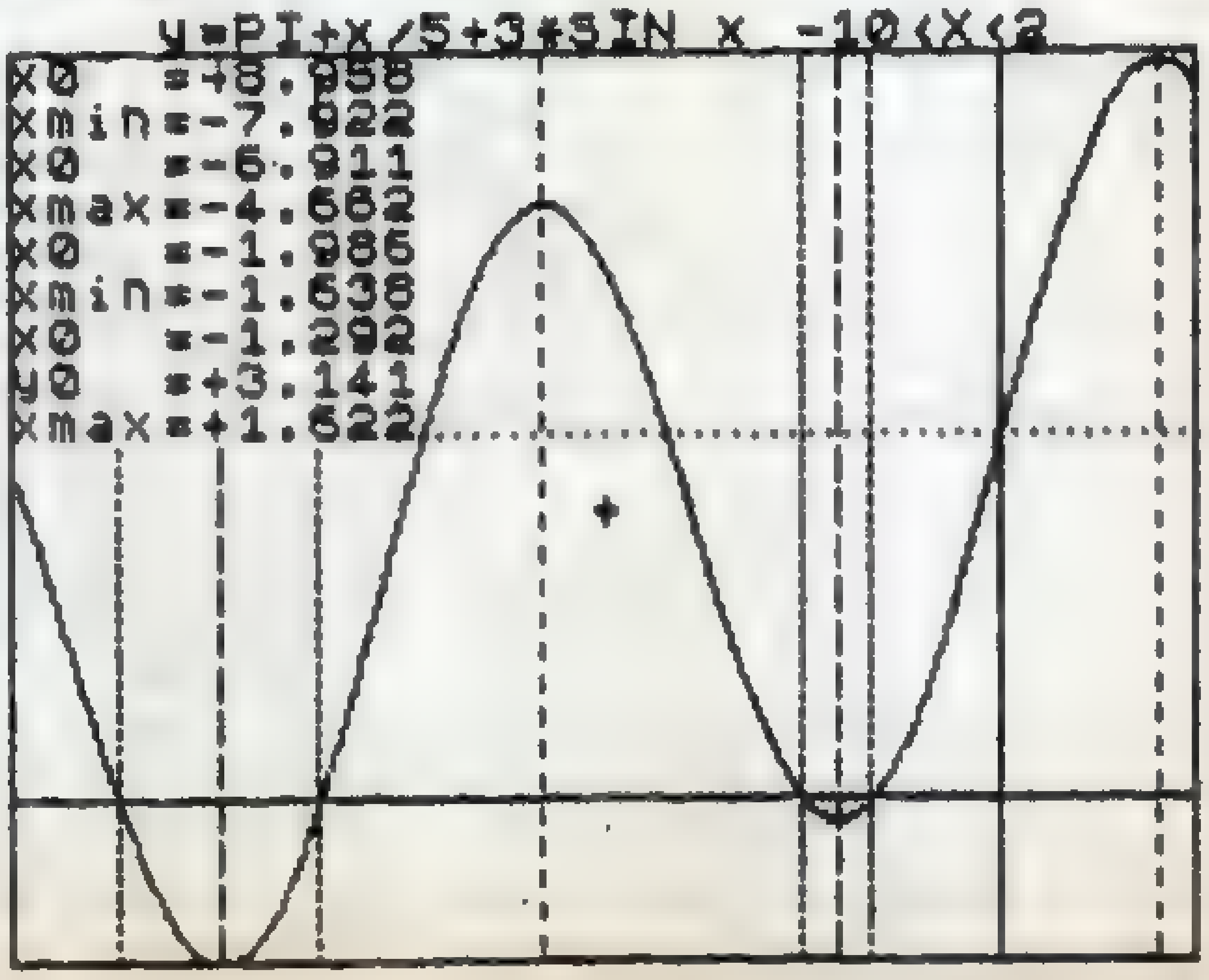
Program wyświetla znaki w 24-krotnym powiększeniu

J. RUBINOWICZ

```
 495 IF xz>0 AND xz<.1 THEN PRIN
T "0";
 500 PRINT xz: RETURN
 510 BEEP .1,26: FOR q=166 TO 1
STEP -8: PLOT n,q: DRAW 0,-2: NE
XT q
 520 PRINT "xmax=";: GO SUB 0560
: RETURN
 530 BEEP .1,38: FOR q=166 TO 1
STEP -2: PLOT n,q: NEXT q: PRINT
 "x0  =";: GO SUB 0560: RETURN
 540 BEEP .1,2: FOR q=166 TO 1 S
TEP -8: PLOT n,q: DRAW 0,-5: NEX
T q
 550 PRINT "xmin=";: GO SUB 560:
 RETURN
 560 LET yp=v(n-1): LET yk=v(n+1
): LET xp=x(n-1): LET xk=x(n+1):
 LET xz=xp+(xk-xp)*(ABS yp/(ABS
yk+ABS yp)): LET xz=.001*INT (10
00*xz+.0005): IF xz>0 THEN PRINT
 "+";
 565 IF xz>0 AND xz<.1 THEN PRIN
T "0";
 567 IF xz=0 THEN PRINT " ";
 570 PRINT xz: RETURN
 580 FOR e=0 TO 7: BORDER e: BEE
P .1,e: NEXT e: BORDER 4: RETURN

 590 PRINT AT 21,0;c$;AT 21,0; I
NK 7;ymin;">y>";ymax
 600 FOR n=0 TO 31: IF SCREEN$ (
21,31-n)<>" " THEN PRINT AT 21,0
;c$: LET tab1=n/2: RETURN
 610 NEXT n: RETURN
 620 LET i$=STR$ xmin: LET j$=ST
R$ xmax: LET t$=CHR$ 19+CHR$ 1+"
y="+f$+CHR$ 19+CHR$ 0+CHR$ 32+CH
R$ 19+CHR$ 1+i$+"<X<"+j$: PRINT
INK 7;AT 0,0;t$
 630 FOR n=0 TO 31: IF SCREEN$ (
0,31-n)<>" " THEN LET tab=n/2: P
RINT AT 0,0;c$: RETURN
 640 NEXT n: RETURN
 650 PRINT "y0  =";: BEEP .1,-10
: LET x=0: LET y0=.001*INT (1000
*VAL (f$)): IF y0>0 THEN PRINT "
+";
 660 IF y0=0 THEN PRINT " ";
 670 PRINT y0: FOR e=0 TO 255 ST
EP 4: PLOT e,CODE v$(n): NEXT e:
 RETURN
```

```
 700 REM  s t a r t
 710 BORDER 4: DIM v(256): DIM x
(256): DIM v$(256): DIM c$(32):
LET f$="PI+x/5+3*SIN x": LET xmi
n=-10: LET xmax=2:
 720 CLS : PRINT '' PAPER 6;c$;"
 Program rysuje wykresy funkcji
              y=f(x)
 dla dowolnych wartosci x z wy-
 jatkiem  funkcji  typu  y=1/x
 gdzie  y  dazy  do nieskonczo-
 nosci.Program wyznacza maxima,
 minima i punkty zerowe dla x i
 y.  Skale  pozioma  i  pionowa
 wyznaczane sa  przy  pierwszym
 przebiegu funkcji, drugi prze-
 bieg oblicza metoda interpola-
 cji wartosci punktow charakte-
 rystycznych  funkcji i  kresli
 jej wykres.
";c$: PAUSE 0: GO TO 150
9998 REM zapis na tasme
9999 CLEAR : SAVE CHR$ 19+CHR$ 1
+CHR$ 22+CHR$ 8+CHR$ 13+CHR$ 173
+CHR$ 8+CHR$ 69+CHR$ 76+CHR$ 65
LINE 700: BEEP .5,5: BEEP .1,17:
 VERIFY "": GO TO 110
```



ATARI

Interfejs magnetofonu

W artykule opisano konstrukcję interfejsu magnetofonu przeznaczonego dla komputerów Atari 800 XL, 65 XE i 130 XE. Jest to układ wzorowany na rozwiązaniu zaczerpniętym z czasopisma „Funkschau" [1]. Dzięki wykorzystaniu cyfrowych układów scalonych interfejs charakteryzuje się dużą pewnością działania oraz umożliwia współpracę komputera z dowolnym magnetofonem kasetowym. Układ wymaga jednego napięcia+5V i może być zasilany z komputera ze względu na mały pobór prądu. W urządzeniu zastosowano w większości elementy produkcji krajowej poza jednym wyjątkiem — komparatorem napięcia LM 339

Zasada działania

Zasada działania interfejsu oparta jest na pomiarze okresu sygnału wejściowego z magnetofonu i przyporządkowaniu wynikowi pomiaru odpowiedniego stanu logicznego. Jak pamiętamy, [2] częstotliwości 5,1 kHz odpowiada stan wysoki, a częstotliwości 3,9 kHz stan niski. Częstotliwościom tym odpowiadają okresy odpowiednio równe 196 i 25. Średni okres wynosi (196÷256):2, co odpowiada częstotliwości 4,42 kHz. Układ porównuje okres sygnału wejściowego z okresem średnim. Jeżeli okres sygnału wejściowego jest mniejszy od 226 μs (f>4,42 kHz), to przyporządkowuje mu się stan wysoki. W przypadku okresu równego lub większego od 226μs (f≤4,42 kHz) przyporządkowuje się jemu stan niski. Możemy w skrócie zapisać to tak:

f>4,42 kHz to stan wysoki
f≤4,42 kHz to stan niski

Konkretne rozwiązanie układowe interfejsu pracującego wg opisanej zasady przedstawiono na rys. 1

Opis układu

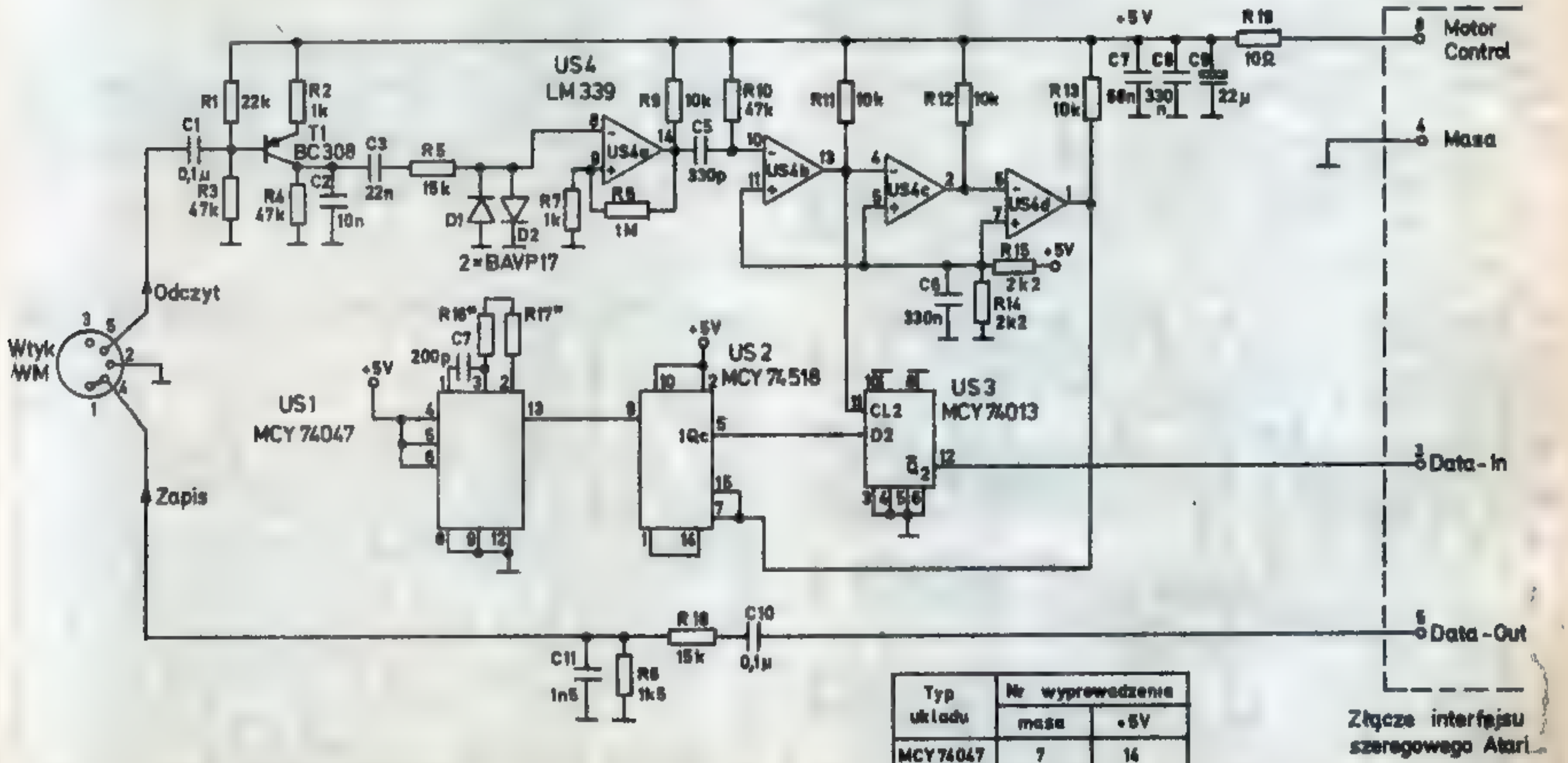
W konstrukcji możemy wyróżnić kilka bloków funkcjonalnych. Układ z tranzystorem T1 jest przeciwwzmacniaczem sterującym pracą wzmacniacza, którego funkcję pełni jeden z komparatorów układu scalonego US4. Pozostałe trzy komparatory tego układu sterują właściwą pracą pozostałej „cyfrowej" części przystawki. Układ US1 jest generatorem częstotliwości wzorcowej 186 kHz (T = 5,376 μs). Wraz z licznikiem US2 pełnią one funkcje układu porównującego okresy sygnału wejściowego ze średnim. Układem, który wynikom porównania przyporządkowuje odpowiednie stany logiczne jest przerzutnik D US3. Bezpośrednio z jego wyjścia Q sterowane jest wejście Data-In złącza szeregowego Atari.

Przed dalszym opisem pracy interfejsu wprowadzimy pojęcie cyklu pracy. Przez pojęcie to rozumieć będziemy pomiar jednego okresu sygnału wejściowego i przyporządkowanie wynikowi pomiaru odpowiedniego stanu logicznego.

Cykl pracy

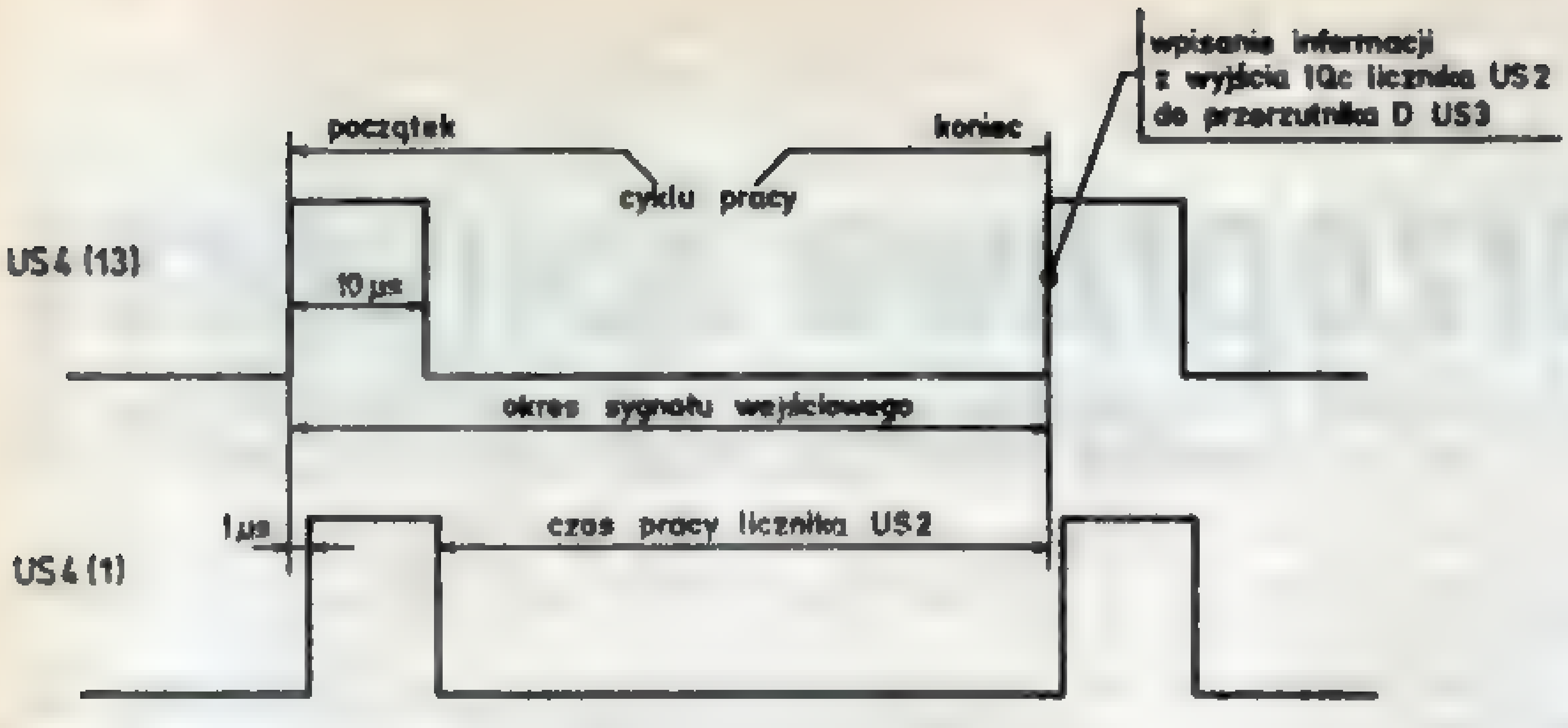
Sygnał z magnetofonu wzmacniany jest przez tranzystor T1 i układ US4a. Po wzmocnieniu jest różniczkowany (C5, R10) i formowany w układzie US4b.

Komparator US4b wraz z elementami C5, R10 tworzy przerzutnik monostabilny, generujący dodatni impuls w odpowiedzi na opadające zbocze na wejściu (punkt C5, wypr. 14 US4). Szerokość impulsu, ustalona przez stałą czasową C5, R10, wynosi ok. 10 μs. Zauważmy, że przerzutnik generuje te impulsy z częstotliwością sygnału wejściowego. Impuls steruje wejściem zegarowym przerzutnika D US3, a po przejściu przez komparatory US4, cd (z opóźnieniem ok. 1 μs) licznik US2. Układ scalony US2 zawiera dwa niezależne liczniki BCD. W układzie połączono je w jeden licznik liczący do 40. Wyjście licznika 1Qc (wypr. 5 US2) ustawia się w stan wysoki, gdy licznik zliczy 40 i więcej impulsów generatora wzorcowego US1. W przeliczeniu na czas wyniesie to 40 Okres wzorcowy, czyli 405,376μs =215 μs. Po uwzględnieniu czasu trwania impulsu zegarowego oraz jego opóźnienia (rys. 2) łączny czas tego cyklu pracy wynosi 215 μs + 10 μs + 1 μs = 226 μs. Jest to okres sygnałów o częstotliwości 4,42 kHz, który przyjęto jako wzorzec do porównań. W czasie normalnej pracy układu czas pracy licznika



Typ układu	Nr wyprowadzenia	
	masa	+5V
MCY 74047	7	14
MCY 74013	7	14
MCY 74518	8	16
LM 339	12	3

Rys. 1 Schemat ideowy interfejsu.



Rys. 2 Przebiegi czasowe cyklu.

jest rózny i ograniczony okresem sygnału wejściowego. Zawartość licznika mozemy obliczyć z zależności

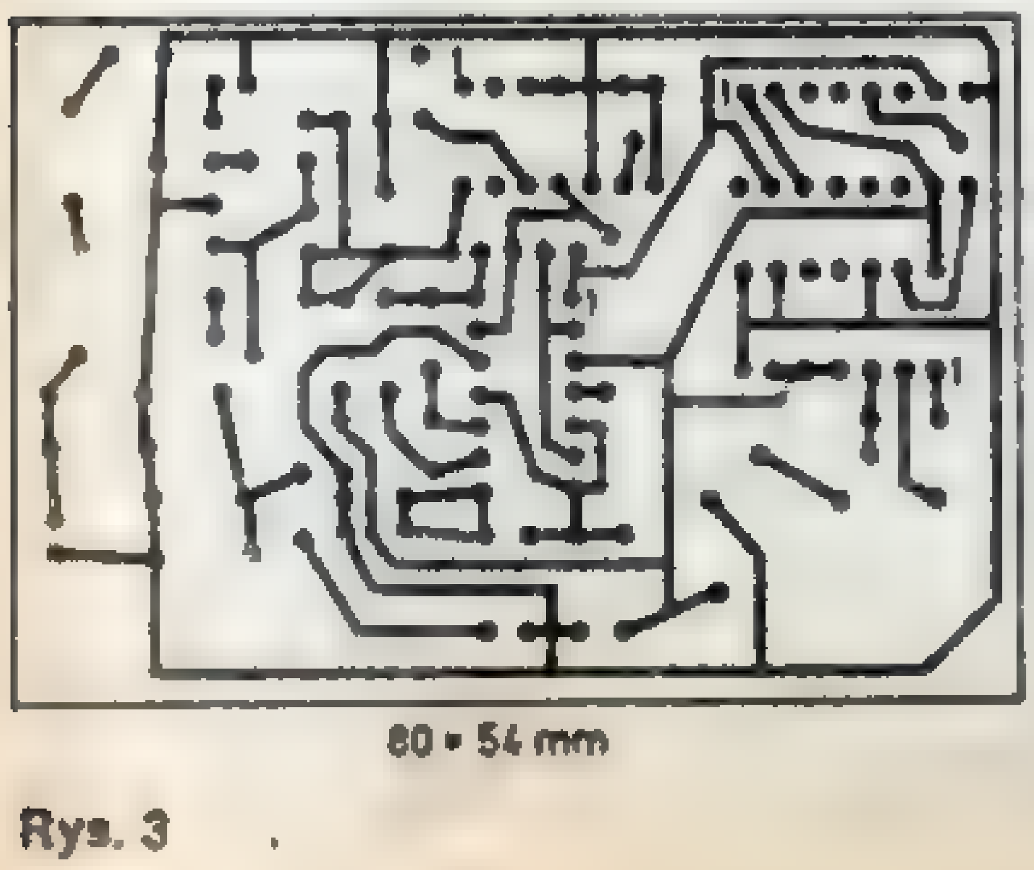
zawartość licznika = (okres sygn. wej 1 1µs) : 5,376µs

Dla sygnału o częstotliwości 3,9 kHz licznik zdąży policzyć do 45, a dla częstotliwości 5,1 kHz tylko do 34 . W pierwszym przypadku wyjście 1Qc (US2) będzie w stanie wysokim, a w drugim w stanie niskim. Informacja z tego wyjścia wpisywana jest do przerzutnika D US3. Po jej wpisaniu w takim samym stanie będzie wyjście Q przerzutnika D. Natomiast na wyjście Q będzie przeciwny stan logiczny. Dla częstotliwości 3,9 kHz na wyjściu Q przerzutnika D będzie niski stan, a dla 5,1 kHz będzie stan wysoki. Informacja o kazdym bicie przekazywana jest do komputera przez 1,666 ms (szybkość transmisji 600 bd). W związku z tym na informację o kazdym bicie składa się kilka cykli pracy. Dla informacji o „1" logicznej stan wyjścia Q przerzutnika D nie będzie zmieniał się przez 8 cykli pracy, a dla informacji o „0" logicznym przez 6 cykli. Z wyjścia Q przerzutnika D informacja przekazywana jest bezpośrednio do komputera.

Podczas nagrywania programów sygnał z komputera (wyjście Data-Out), poprzez dzielnik napięcia R18/R6, podawany jest na wejście magnetofonu i zapisywany na kasecie magnetofonowej. Zadaniem dzielnika jest obnizenie amplitudy sygnału do poziomu wymaganego przez magnetofon.

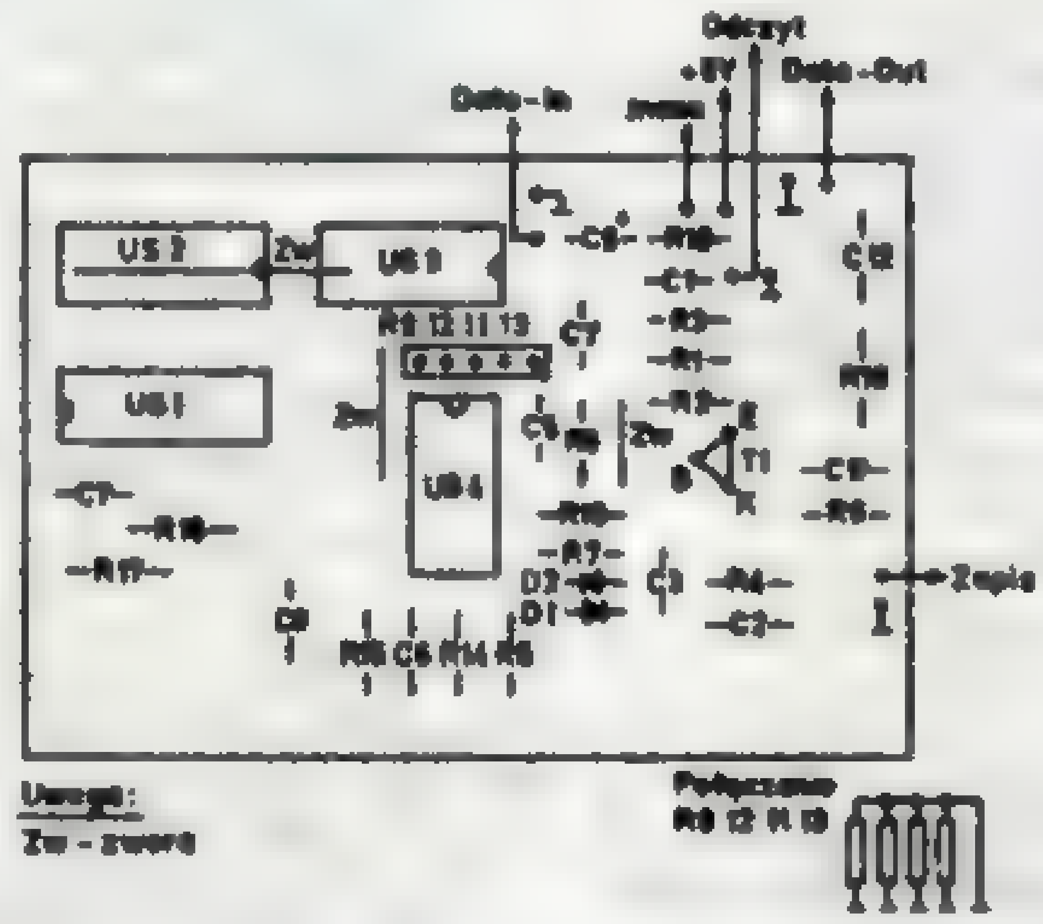
Konstrukcja i uruchomienie

Układ zmontowano na jednostronnie drukowanej płytce wykonanej wg rys. 3. Rozmieszczenie elementów zamieszczono na rys. 4. W pierwszej kolejności lutujemy elementy dyskretne, a następnie układy scalone. Szczególną ostrozność należy zachować przy montazu cyfrowych układów scalonych, ponieważ są wykonane technologią CMOS (US1, US2, US3).



80 × 54 mm

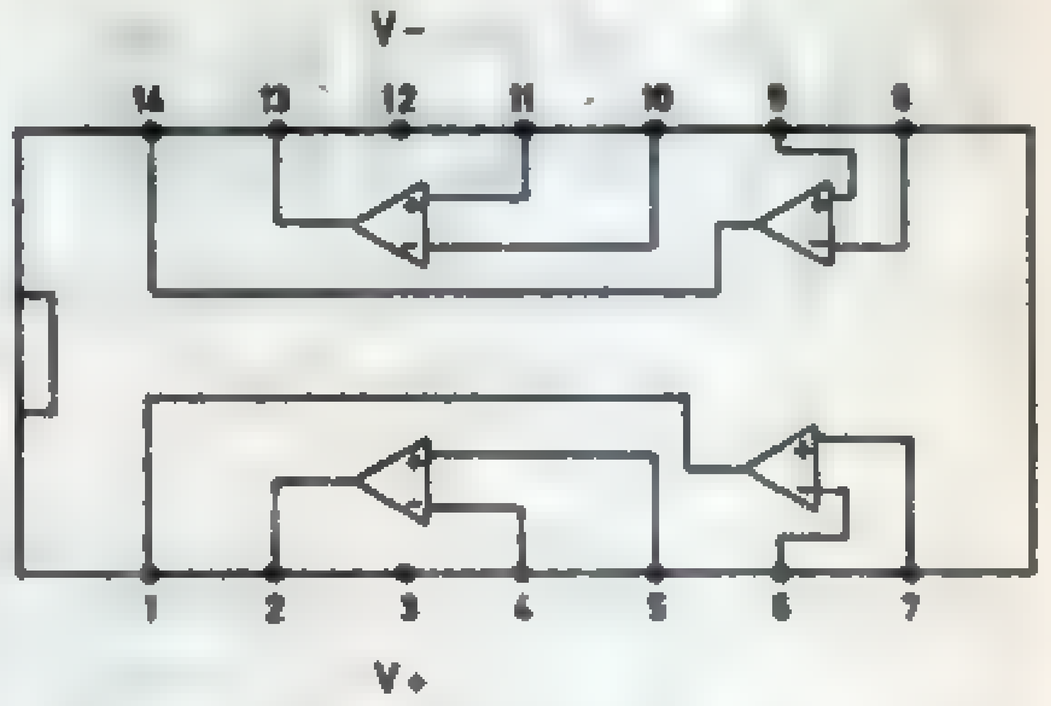
Rys. 3



Rys. 4 Schemat montażowy

Po kontroli wzrokowej montażu przystępujemy do strojenia interfejsu, wykorzystując miernik częstotliwości lub oscyloskop. Interfejs zasilany napięciem + 5V i do wyprowadzenia 13 układu US1 dołączamy miernik częstotliwości. Dobierając wartości rezystorów, R16 zgrubnie i R17 dokładnie staramy się uzyskać przebieg o częstotliwości 186 kHz. Po uzyskaniu tej częstotliwości strojenie interfejsu jest zakończone. W rozwiązaniu modelowym wartość rezystora R16 ustalono na 9,1 kΩ. Dobrane w ten sposób rezystory dolutowujemy do układu i jeszcze raz sprawdzamy częstotliwość generatora wzorcowego.

Przystawkę z magnetofonem łączymy dwużyłowym przewodem ekranowym zakończonym typowym pięciostykowym wtykiem typu WM. Interfejs łączymy z komputerem trzyżyłowym przewodem ekranowym zakończonym wtyczką wykonaną wg opisu zamieszczonego w „IKS-ie" [2]. Interfejs zasilany jest ze styku MOTOR CONTROL złącza Atari. Ma to znaczenie przy jednoczesnej współpracy komputera ze stacją dysków i magnetofonem. Na zakończenie podamy kilka informacji o układzie LM 339. Jest to poczwórny komparator monolityczny. Jedną z głównych jego zalet jest mozliwość pracy z jednym napięciem zasilającym (już od + 2V). Komparator zawiera na wyjściu układ z otwartym kolektorem, dzięki czemu istnieje mozliwość doboru poziomów logicznych napięcia wyjściowego przez dołączenie obwodów zewnętrznych. Rozmieszczenie wyprowadzeń układu przedstawiono na rys. 5.



Rys. 5 Rozmieszczenie wyprowadzeń układu scalonego LM339

Opisy pozostałych układów scalonych możemy znaleźć w [3] i [4].

Wykaz elementów

MCY 74013N — US3
MCY 74047N — US1
MCY 74518N — US2
LM 339 — US4

Diody i tranzystory

BC 308A — T1
BAVP17 — D1, D2

Kondensatory

200 pF — C17 (typu KSF)
330 pF — C5
1,5 nF — C11
10 nF — C2
22 nF — C3
68 nF — C7
100 nF — C1, C10
330 nF — C6, C8
22 F — 6,3V — C9

Rezystory 0,125 W

10 R — R19
1 k — R2, R7
1,5 k — R6
2,2 k — R14, R15
4,7 k — R4
10 k — R9, R11, R12, R13
15 k — R5, R18
22 k — R1
47 k — R3, R10
1 M — R8
R16, R17 — dobieramy w czasie strojenia — opis w tekście

Inne

Wtyk magnetofonowy pięciostykowy typu WM

Bogusław LIPCZYŃSKI

LITERATURA

1. H. Reelsen: „Recoederinterface", „Funkschau" nr 17/1984
2. Interfejs magnetofonu dla Atari, „IKS" 1987
3. M. Kręciejewski: „Układy cyfrowe CMOS", „Radioelektronik" nr 10/11/1985
4. M. Łakomy, J. Zabrodzki: „Cyfrowe układy scalone" PWN 1980

LILAVATI (cz. 3)

Mozolna przeprawa żołnierzy

Danuta KWASIŻUR, Mieczysław SKONIECZNY

W trzecim odcinku cyklu poświęconego rozwiązywaniu anegdot matematycznych dzięki wykorzystaniu mikrokomputera omówimy problem przeprawy żołnierzy przez rzekę (Lilavati — anegdota nr 50). Treść historyjki jest następująca:

Oddział żołnierzy doszedł do rzeki, przez którą musi się przeprawić. Most po niedawnej powodzi jest jeszcze w ruinie, rzeka zaś zbyt głęboka, by próbować przebrnąć ją w bród. W małej łódce przy brzegu bawią się dwaj chłopcy. Łódka jest tak maleńka, że zaledwie jeden żołnierz mógłby się w niej zmieścić. Mimo to, ta właśnie łódka przy czynnym udziale chłopców przewiozła na drugą stronę cały oddział żołnierzy. Jak to się stało?!

Segment sterujący programu na mikrokomputer AMSTRAD CPC 6128 ma następującą postać (instrukcje 10—290):

W tym fragmencie użytkownik zostaje wprowadzony w temat zagadnienia. Następnie pojawia się plansza gry oraz następuje zapoznanie grającego z możliwościami sterowania przebiegiem przeprawy. Możliwości i odpowiadające klawisze sterujące są następujące:

1. Chłopiec wysiada z łódki.
2. Chłopiec wsiada do łódki.
3. Żołnierz wysiada z łódki.
4. Żołnierz wsiada do łódki.
5. Łódka płynie.

Cała gra rozgrywana jest na tej samej planszy, a kończy się sukcesem w momencie, gdy cały oddział (9 żołnierzy) znajdzie się na drugim brzegu. Aby uzyskać większą przejrzystość wersji źródłowej programu, do segmentu zasadniczego dołączono następujące podprogramy:

1. Rysowanie żołnierza (instrukcje 300—400)

```
10 DIM zol(9,2):DIM chl(2,2):DIM lodka$(2):GOSUB 1070
20 MODE 1:INK 0,26:INK 1,0:INK 2,9:INK 3,14:PAPER 2:BORDER 9:CLS:PEN 1
30 LODKA$(1)="C":LODKA$(2)="C"
40 FOR k=1 TO 9:zol(k,1)=1:NEXT k:s=1
50 FOR k=1 TO 9:zol(k,2)=0:NEXT k:chl(1,2)=0:chl(2,2)=0:chl(1,1)=0:chl(2,1)=0
60 PRINT"9 zolnierzy doszlo do rzeki, przez ktorakoniecznie musza sie przeprawic
. Most poniedawnej powodzi jest jeszcze w ruinie,rze
ka zas zbyt gleboka, by probowac     przebrnac ja w brod. W malej lodce u    brz
egu rzeki bawia sie dwaj chlopcy."
70 PRINT"Lodka jest tak malenka, ze zaledwie je- den zolnierz moglby sie w niej
zmiescic.Mimo to ta wlasnie lodka przy czynnym   udz
iale chlopcow przewiozla na druga    strone rzeki wszystkich zolnierzy."
80 PRINT "Jak sie to stalo?"
90 PEN 0:LOCATE 1,20:PRINT "NACISNIJ DOWOLNY KLAWISZ"
100 IF INKEY$="" THEN GOTO 100
110 MODE 1:PAPER 0:BORDER 9:CLS:WINDOW £1,1,10,20,25:PAPER £1,2:WINDOW £2,31,40,
20,25:PAPER £2,2:WINDOW £3,11,30,21,25:PAPER £3,3:CL
S £1:CLS£2:CLS £3:PEN 1
120 WINDOW £4,1,40,1,1:PAPER £4,2:PEN £4,1:CLS £4
130 GRAPHICS PEN 1:FOR A=160 TO 440 STEP 56:PLOT A,16:DRAWR 16,0:PLOT A+22,40:DR
AWR 16,0:PLOT A,64:DRAWR 16,0:NEXT
140 FOR y=3 TO 15 STEP 6:FOR x=3 TO 11 STEP 3:GOSUB 300:GOSUB 370:NEXT:NEXT
150 PEN 1:LOCATE 11,20:PRINT CHR$(213);STRING$(5,CHR$(143));CHR$(212)
160 PAPER 0:y=18:FOR x=11 TO 14 STEP 3:GOSUB 410:NEXT
170 WINDOW £5,11,30,2,13:PAPER £5,3:CLS £5:PEN 0:PAPER 2
180 LOCATE 12,2:PRINT "MOZLIWOSCI:"
190 LOCATE 12,4:PRINT "1.Chlopiec wysiada":LOCATE 14,5:PRINT "z lodki"
200 LOCATE 12,6:PRINT "2.Chlopiec wsiada":LOCATE 14,7:PRINT "do lodki"
210 LOCATE 12,8:PRINT "3.Zolnierz wysiada":LOCATE 14,9:PRINT"z lodki"
220 LOCATE 12,10:PRINT "4.Zolnierz wsiada":LOCATE 14,11:PRINT"do lodki"
230 LOCATE 12,12:PRINT "5.Lodka plynie"
240 a$=INKEY$
250 IF a$="" THEN GOTO 240
260 P=VAL(A$)
270 IF P<1 OR P>5 THEN GOTO 240
280 ON P GOSUB 480,560,680,810,950
290 LOCATE 1,1:GOTO 240
```

```
300 PAPER 0:PEN 2:LOCATE X,y:PRINT CHR$(190)
310 LOCATE X-1,y+1:PRINT CHR$(191);CHR$(143);CHR$(192)
320 PLOT (x-1)*16+3,(26-Y)*16-10:DRAWR 0,-4:DRAWR 2,-2:DRAWR 6,0:DRAWR 2,2:DRAWR
 0,4
330 LOCATE X-1,y+2:PRINT CHR$(193);CHR$(143);CHR$(194)
340 GRAPHICS PEN 1:PLOT (x-1)*16+3,(26-Y)*16-10:DRAWR 0,-4:DRAWR 2,-2:DRAWR 6,0:
DRAWR 2,2:DRAWR 0,4
350 PLOT (x-1)*16+1,(26-y)*16-40:DRAWR 14,0:DRAWR 0,-2:DRAWR -14,0
360 RETURN
370 LOCATE X-1,y+3:PRINT CHR$(195);CHR$(196);CHR$(197)
380 LOCATE X-1,y+4:PRINT CHR$(198);CHR$(199);CHR$(200)
390 PLOT (x-2)*16+1,(26-y)*16-34:DRAWR 0,-36:DRAWR 2,0:DRAWR 0,10:DRAWR 0,-20:DR
AWR 2,0:DRAWR 0,12:PLOTR 2,-6:DRAWR 0,-6
400 RETURN
```

Podprogram w zależności od sytuacji rysuje sylwetkę żołnierza:

— na brzegu (cała postać);

— w łódce (górna część sylwetki).

2. Rysowanie chłopca (instrukcje 410—470).

```
410 PEN 3:LOCATE x,y:PRINT CHR$(201);CHR$(210);CHR$(202)
420 GRAPHICS PEN 1:PLOT x*16+4,(25-y)*16+6:DRAWR 6,0:DRAWR 2,2:DRAWR 0,6:DRAWR -
2,2:DRAWR -6,0:DRAWR -2,-2:DRAWR 0,-6:PLOT x*16+4,(2
5-y)*16+4:PLOT x*16+10,(25-y)*16+4:PLOT x*16+4,(25-y)*16+14:DRAWR 6,0:MOVE x*16+
8,(25-y)*16+10:FILL 0
430 LOCATE x,y+1:PRINT CHR$(203);CHR$(204);CHR$(205):RETURN
440 LOCATE x,y+2:PRINT CHR$(206);" ";CHR$(208):PEN 1:LOCATE x+1,y+2:PRINT CHR$(2
07):LOCATE x+1,y+3:PRINT CHR$(209)
450 GRAPHICS PEN 3:PLOT x*16+4,(25-y)*16-46:DRAWR -6,0:PLOT x*16+4,(25-y)*16-48:
DRAWR -6,0
460 PLOT x*16+10,(25-y)*16-46:DRAWR 6,0:PLOT x*16+10,(25-y)*16-48:DRAWR 6,0
470 RETURN
```

Podprogram w zależności od sytuacji rysuje sylwetkę chłopca:

— na brzegu (cała postać);

— w łódce (górna część sylwetki).

3. Chłopiec wysiada z łódki (instrukcje 480—550).

```
480 FOR k=1 TO 2
490 IF lodka$(k)="c" OR lodka$(k)="C" THEN GOTO 520
500 NEXT
510 SOUND 1,458:PRINT £4,"NIE MA CHLOPCA W LODZI":GOTO 670
520 LODKA$(K)=" ":PAPER 0:FOR A=1 TO 3:LOCATE 11+(K-1)*3+13*(S-1),16+A:PRINT "
 ":NEXT
530 IF CHL(1,S)=1 THEN  X=2+30*(S-1):Y=21:PAPER 2:GOSUB 410:GOSUB 440:CHL(2,S)=1
:GOTO 550
540 CHL(1,S)=1:PAPER 2:X=6+30*(S-1):Y=21:GOSUB 410:GOSUB 440
550 RETURN
```

4. Chłopiec wsiada do łódki (instrukcje 560—670).

```
560 FOR k=1 TO 2
570 IF chl(k,s)=1 THEN GOTO 600
580 NEXT k
590 PRINT £4,"NIE MA CHLOPCA NA TYM BRZEGU":GOTO 670
600 IF NOT(LODKA$(1)="Z") THEN GOTO 620
610 PRINT £4,"NIE MA MIEJSCA W LODCE":GOTO 670
620 FOR A=1 TO 2:IF LODKA$(A)=" " THEN GOTO 640
630 NEXT A
640 CHL(k,S)=0:LODKA$(A)="C":X=11+(A-1)*3+13*(s-1):Y=18:PAPER 0:GOSUB 410
650 PAPER 2:FOR t=1 TO 5:LOCATE 6-(k-1)*4+30*(s-1),19+t:PRINT "   ":NEXT t
660 RETURN
670 SOUND 1,458:SOUND 1,348:FOR T=1 TO 3000:NEXT:CLS£4:RETURN
```

5. Chłopiec wysiada z łódki (instrukcje 680—800).

```
680 IF lodka$(1)="Z" THEN GOTO 700
690 PRINT £4,"NIE MA ZOLNIERZA W LODCE":GOTO 670
700 LODKA$(1)=" "
710 FOR K=1 TO 9
720 IF ZOL(K,S)=0 THEN GOTO 740
730 NEXT K
740 ZOL(K,S)=1
750 FOR T=1 TO 3:LOCATE 12+(S-1)*13,16+T:PRINT "   ":NEXT T
760 X=9+INT((K-1)/3)*9-(K-1)*3+30*(S-1):Y=15-INT((K-1)/3)*6:GOSUB 300:GOSUB 370
770 IF S=1 THEN RETURN
780 FOR K=1 TO 9
790 IF ZOL(K,S)=0 THEN RETURN
800 NEXT K:GOTO 1020
```

6. Żołnierz wsiada do łódki (instrukcje 810—940).

```
810 IF ZOL(1,S)=1 THEN GOTO 830
820 PRINT £4,"NIE MA ZOLNIERZY NA TYM BRZEGU":GOTO 670
830 FOR A=9 TO 1 STEP -1
840 IF ZOL(A,S)=1 THEN GOTO 860
850 NEXT A
860 IF LODKA$(1)="C" OR LODKA$(1)="Z" OR LODKA$(2)="C" THEN GOTO 610
870 ZOL(A,S)=0:LODKA$(1)="Z"
880 IF s=2 THEN GOTO 990
890 PAPER 0:FOR k=15 TO 19:LOCATE 8+30*(S-1),K:PRINT "   ":NEXT K
900 PAPER 0:X=13+13*(S-1):Y=17:GOSUB 300
910 IF A=1 THEN RETURN
920 X=8+INT((A-1)/3)*9-(A-1)*3+30*(S-1):Y=15-INT((A-1)/3)*[illegible]
930 FOR T=0 TO 4:LOCATE X,Y+T:PRINT "   ":NEXT T
940 X=9:Y=15:GOSUB 300:GOSUB 370:RETURN
```

7. Łódka płynie na drugi brzeg (instrukcje 950—1060).

```
950 IF lodka$(1)="C" OR lodka$(1)="Z" OR LODKA$(2)="C" THEN GOTO 970
960 PRINT £4,"NIE MA NIKOGO W LODCE":GOTO 670
970 IF S=1 THEN GOSUB 1250:S=2:RETURN
980 GOSUB 1380:S=1:RETURN
990 X=8+INT((A-1)/3)*9-(A-1)*3+30*(S-1):Y=15-INT((A-1)/3)*6
1000 FOR T=0 TO 4:LOCATE X,Y+T:PRINT "    ":NEXT T
1010 PAPER 0:X=13+13*(S-1):Y=17:GOSUB 300:RETURN
1020 CLS£5:PEN 0:PAPER 2:LOCATE 12,3:PRINT"BRAWO'":LOCATE 12,5:PRINT"KONIEC PRZE
PRAWY":FOR A=1 TO 5:SOUND 1,358:SOUND 1,458:NEXT
1030 PEN 1:LOCATE 12,8:INPUT"JESZCZE RAZ(T/N)";K$
1040 IF K$="T" OR K$="t" THEN GOTO 20
1050 IF K$="N" OR K$="n" THEN LOCATE 1,1:END
1060 GOTO 120
```

8. Definicja własnych symboli graficznych (instrukcje 1070—1240).

```
1070 SYMBOL AFTER 190
1080 SYMBOL 190,60,126,126,255
1090 SYMBOL 191,0,3,7,15,15,14,14,14
1100 SYMBOL 192,0,192,224,240,240,112,112,112
1110 SYMBOL 193,14,14,14,30,62,60,120,112
1120 SYMBOL 194,112,112,112,112,112,112,112,112
1130 SYMBOL 195,112,96:SYMBOL 196,255,255,231,231,231,231,231,231
1140 SYMBOL 197,112,112,112:SYMBOL 198,0,0,0,0,0,1,7,7
1150 SYMBOL 199,231,231,231,231,231,231,231,231:SYMBOL 200,0,0,0,0,0,128,224,224
1160 SYMBOL 201,0,0,0,0,0,0,0,1:SYMBOL 202,0,0,0,0,0,0,0,128
1170 SYMBOL 203,3,3,3,3,3,3,3,3
1180 SYMBOL 204,255,255,126,126,126,126,126,126
1190 SYMBOL 205,192,192,192,192,192,192,192,192
1200 SYMBOL 206,3,3,3
1210 SYMBOL 207,126,126,126,126,102,102,102,102
1220 SYMBOL 208,192,192,192
1230 SYMBOL 209,102,102,102,102,102,102
1240 RETURN
```

Symbole 190—209 są wykorzystywane do rysowania sylwetek żołnierzy i chłopców.

9. Przesuwanie łódki w prawo i w lewo (instrukcje 1250—1530).

```
1250 REM ' przesuwanie w prawo'
1260 RESTORE 1280:FOR i=&7F76 TO &7FA1
1270 READ v:POKE i,v:NEXT
1280 DATA 1,8,4,33,39,192,84
1290 DATA 93,19,197,1,14,0,237,184,193,35,54,0
1300 DATA 13,40,6,17,13,8,25,24,234,5,40,12,124
1310 DATA 214,56,103,17,93,0,25,14,8,24,219,201
1320 x=17:y=17:GOSUB 1490
1330 POKE &7F7B,w1
1340 FOR z=w2+1 TO w2+26
1350 POKE &7F7A,z:CALL &7F76
1360 FRAME:FOR t=1 TO 60:NEXT:NEXT:RETURN
1370 REM 'przesuwanie w lewo'
1380 RESTORE 1400:FOR i=&7F76 TO &7FA1
1390 READ v:POKE i,v:NEXT
1400 DATA 1,8,4,33,36,192,84,93
1410 DATA 27,197,1,14,0,237,176,193,43,54,0
1420 DATA 13,40,6,17,243,7,25,24,234,5,40,12,124
1430 DATA 214,56,103,17,67,0,25,14,8,24,219,201
1440 x=24:y=17:GOSUB 1490
1450 POKE &7F7B,w1
1460 FOR z=w2 TO w2-26+1 STEP -1
1470 POKE &7F7A,z:CALL &7F76
1480 FRAME:FOR t=1 TO 60:NEXT:NEXT:RETURN
1490 A=(Y-1)*80+(x-1)*2
1500 W1=INT(a/256)
1510 W2=a-W1*256
1520 w1=192+w1
1530 RETURN
```

W trakcie sterowania przeprawą (klawisze numeryczne 1—5) program przeprowadza kontrolę logiczną zaistniałej sytuacji. W razie błędów logicznych są wyświetlane odpowiednie komunikaty i program czeka na poprawną dyrektywę. Całkowity zestaw komunikatów o błędach jest następujący:

NIE MA CHŁOPCA W ŁODZI.
NIE MA CHŁOPCA NA TYM BRZEGU.
NIE MA MIEJSCA W ŁÓDCE.
NIE MA ŻOŁNIERZA W ŁÓDCE.
NIE MA ŻOŁNIERZA NA TYM BRZEGU.
NIE MA NIKOGO W ŁÓDCE.

Życzymy wytrwałym Czytelnikom sukcesów w sterowaniu przeprawą żołnierzy przy użyciu mikrokomputera.

Kostka w tekście

Compact Disc

Włodzimierz KALETA

Płyta kompaktowa pojawiła się w 1982 r w Japonii. W marcu 1983 była już sprzedawana w Europie Zachodniej. Od początku sprzedaż płyt, zwanych najczęściej CD (od ang. Compact Disc) a także odtwarzaczy laserowych osiągnęła ogromne rozmiary. Do dziś nie ma zgody, jak po polsku te urządzenia nazwać — jedni lansują nazwę laserofon, inni — dyskofon.

Niezwykłe powodzenie płyta laserowa zawdzięcza przede wszystkim doskonałej jakości odtwarzanego dźwięku, wygodzie korzystania z niej i jednakowemu w całym świecie standardowi.

Doskonałą jakość osiąga się dzięki nowej technologii nagrywania i odtwarzania dźwięku. Najistotniejszym elementem tej technologii jest laser, który pełni tu rolę igły gramofonowej. Drugim, nie mniej ważnym, jest mikrokomputer. Jest on czymś w rodzaju fabryki, w której cyfrowy „surowiec" jest wszechstronnie obrabiany.

W sygnale zakodowane są nie tylko dźwięki, lecz także informacje kontrolno-sterujące, bez których ta fabryka nie byłaby w stanie funkcjonować. Dzięki nim komputer wykrywa i natychmiast poprawia błędy wynikające np. z zadrapania płyty. Nawet plamka o średnicy ok. 2 mm nie jest odczuwalna podczas odtwarzania płyty.

Współczesne gramofony CD potrafią samoczynnie korygować nawet 60 najczęściej spotykanych błędów i przekłamań odczytu. Jest więc on skomplikowanym urządzeniem laserowo-komputerowym. Natomiast płyta CD odtwarzana jest przeciwnie, niż longplay tradycyjny — od środka.

Ponieważ strumień odtwarzanych informacji powinien być stały, płyta na początku wiruje 500, a pod koniec odtwarzania 215 razy na minutę. W ciągu każdej sekundy mikroprocesor przetwarza 4 321 800 bitów informacji zawartych w sygnale cyfrowym.

Śpiewający laser, dziewczyny i złoto

Z czym się nam najczęściej kojarzy obecnie postęp? Z nowoczesnością przede wszystkim. A dziś nowoczesność, to w głównej mierze — elektronika. Ona wyznacza swoisty próg rozwojowy, dzieli na „wielki świat" i okolice, wyznacza miejsce w czołówce państw najbardziej technologicznie rozwiniętych lub spycha je do cywilizacyjnego ogona.

Ten „wielki świat" zawitał przed kilku laty do.... Bydgoszczy. Stało się to głównie za sprawą Zakładów Radiowych „Unitra — Eltra". Japonia, Francja, RFN. Nazwy firm znanych na całym świecie takich jak: ITT Canon, Thomson, Brandt, Nordmende, Continental Edison Sabaj — nie wzbudzają tutaj żadnych emocji.

Radioodbiorniki oraz radiomagnetofony produkowane w „Eltrze" kupują klienci na całym świecie. I chwalą ich jakość oraz estetykę wykonania, parametry. Co warte podkreślenia, ten eksport nie jest wcale mały. Dlaczegóż więc mielibyśmy mieć jakieś kompleksy?

— W bieżącym roku — mówi główny specjalista zakładów, inż. Piotr Terlecki wyprodukujemy 900 tysięcy różnorodnego sprzętu radiowo-elektronicznego. Z czego 40—50 procent radiomagnetofonów i ponad połowę odbiorników przenośnych wyślemy na eksport. Głównie do krajów Europy Zachodniej — Francji, Republiki Federalnej Niemiec, Szwecji, Jugosławii oraz Japonii. Produkcja eksportowa obejmuje około 24 proc. naszej globalnej produkcji.

Co my z tego mamy?

Pięknie prezentują się wyroby „Eltry". Większość radioodbiorników nosi żeńskie imiona: „Ania", „Zosia", „Klaudia", „Jowita", „Hania", „Liza"... Lubią tutaj dziewczyny?

— To może dlatego — tłumaczy kierownik sekcji technologii, inż. Eugeniusz Bazak — że w naszym zakładzie pracuje wiele kobiet. Produkcja tu wytwarzana wymaga ogromnej cierpliwości i dokładności. Wiadomo, że kobiety do tego rodzaju pracy nadają się najlepiej. Są cierpliwe z natury. No, może z wyjątkiem kilku naszych żon.

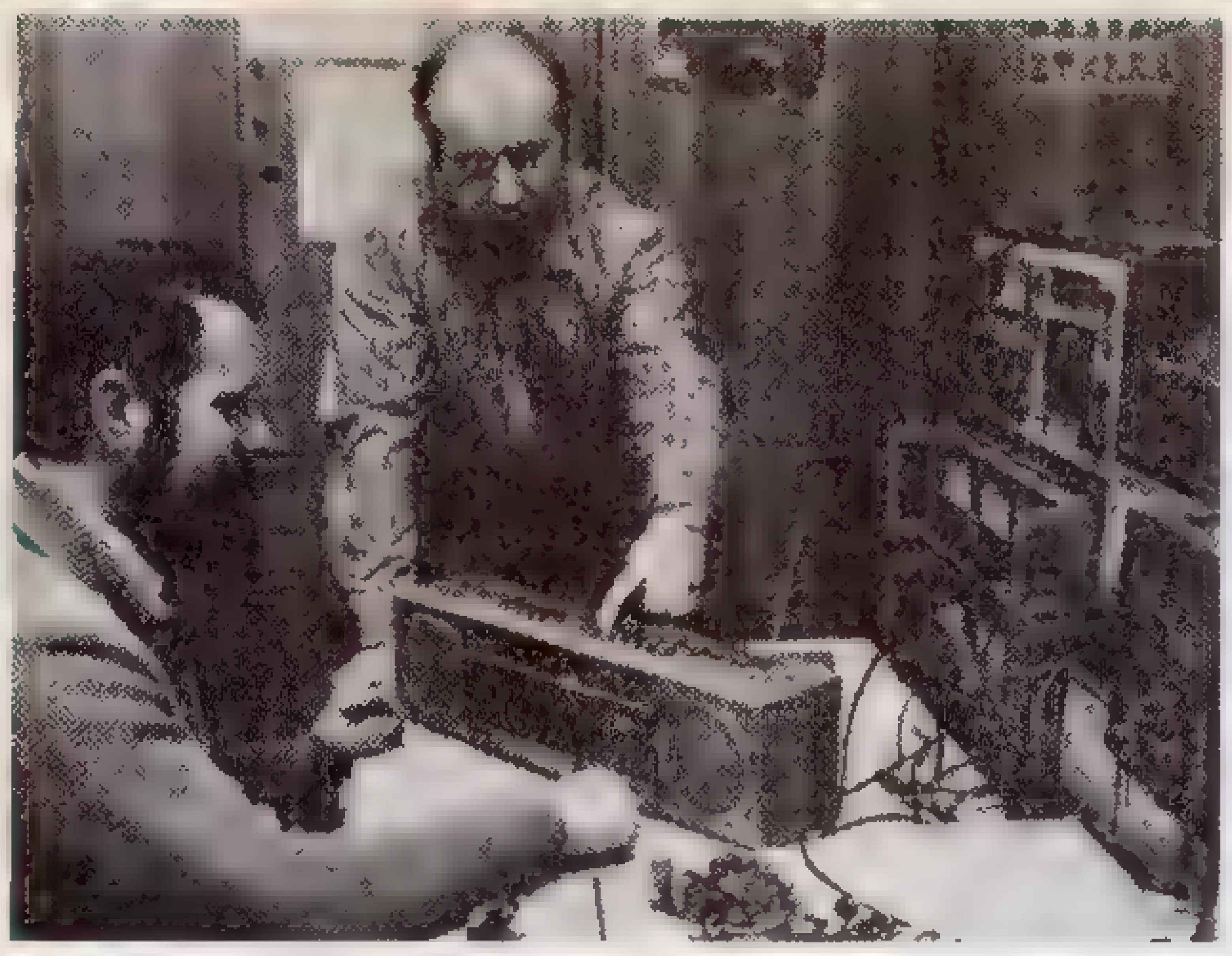
Patrząc na wytwarzane tutaj cacuszka, aż łza się kręci w oku, że radioodbiorniki o wdzięcznych dziewczęcych imionach opuszczają kraj i cieszyć będą wzrok i serca innych. Cóż z tego eksportu mamy my, polscy melomani i zwykli zjadacze chleba, którzy lubią posłuchać miłej piosenki przy goleniu?

— Nasza oferta na rynek krajowy jest bardzo bogata — wyjaśnia kierownik montażu odbiorników, inż. Andrzej Kazubowski. — Trafia nań przecież 46 proc. naszej produkcji. Nie rozróżniamy wśród niej tej na eksport i na rynek krajowy. Radioodbiorniki i radiomagnetofony do nas i za granicę wytwarzają ci sami ludzie, przestrzegając tych samych — zawsze wysokich — norm jakości. Gdybyśmy postępowali inaczej, demoralizowalibyśmy ludzi.

Przykład pierwszy z brzegu: „Ania", „Liza", i „Emilia"... Te same odbiorniki — pod względem koloru, jakości wykonania — można kupić w sklepach Francji, RFN i naszych. Wyjątek stanowią radioodbiorniki, sprzedawane szwedzkiej firmie „Luxor". Nie ma ich na naszym rynku. Ale tylko dlatego,



Z elektroniką mają w zakładzie do czynienia na co dzień. Na zdjęciu: monter układów elektrycznych Jarosław Sierant przeprowadza korektę zakresów radioodbiornika „Ania". (Foto: Marek Chełminiak)



Jan Meier (z prawej) jest jednym z głównych konstruktorów najnowszej generacji radiomagnetofonów produkowanych przez „ELTRĘ"



„Manuela" prezentowana przez inż. Piotra Terleckiego, to najnowsze dziecko zakładów. Ambicją firmy jest, by wszystkie modele eksportowe znalazły się na krajowym rynku nie później niż po roku.

iz nie ma na nie zbytu. To radio tylko z zakresem fal UKF i możliwością ich programowania, za którym przepadają Szwedzi, nam nie przypadło do gustu

W dyrekcji zapewniają, że najnowsze dziecko zakładu, „Manuela", również trafi do naszych sklepów i to jeszcze w drugim kwartale. Jest to radiomagnetofon dwukasetowy, stereofoniczny, o mocy 2x4 W, z dwiema prędkościami przesuwu taśmy, czterozakresowy, możliwością współpracy z każdym innym sprzętem

Radiomagnetofon „palce lizać". Mogliby na eksport wysłać każdą ilość. Ale przecież nie o to chodzi. W bieżącym roku sprzedaż na rynek krajowy wyniesie 117 proc. ubiegłorocznej produkcji i wzrośnie o około 1 mld zł

— Eksportujemy przede wszystkim dlatego — zapewniał inż. Terlecki — aby móc produkować dla kraju sprzęt najnowszej generacji. Bez tego eksportu nie mielibyśmy dewiz niezbędnych na unowocześnianie naszej produkcji, nie znalibyśmy nowinek technicznych w tym zakresie

Okazjonalne słowa? W „Eltrze" mają wiele przekonujących argumentów, że nie. Tu potrafią nie tylko korzystać z obcych licencji, ale także udoskonalać je. I to tak, że już obecnie firmy zachodnie mają w nich groźnego konkurenta

Przepchać wąskie gardło

— Co jest piętą achillesową krajowej elektroniki? — Inż. Terlecki wie, że to pytanie retoryczne, dlatego nie czekając na odpowiedź stwierdza: — Brak podzespołów. Aby je wytworzyć potrzebne są odpowiednie narzędzia, precyzja ich wykonania. Mozolne linie do ich wytwarzania, wymagają żmudnych badań oraz wyjątkowego wręcz przestrzegania wymogów jakościowych

Do niedawna nie było problemów — kupowało się potrzebne podzespoły na Zachodzie. A dziś?

W latach osiemdziesiątych zakupiono w „Eltrze" holenderską licencję linii do selektywnego pokrywania styków. W latach 1982 i 1983, kiedy rozpoczynali pracę „pełną parą" kontakt z krajami zachodnimi był — nie z naszej zresztą winy — ograniczony. Dlatego w zakładzie wzięto się poważnie za bary z najnowszymi technologiami dotyczącymi wytwarzania podzespołów stykowych

Efekty tej pracy nie dały długo na siebie czekać. Wkrótce mieli swoją własną linię. Równie dobrą, a może nawet lepszą. W każdym razie na pewno oszczędniejszą. Styki pokrywało się bowiem dotychczas... złotem. Rocznie zużywano go około 300 kg. Metoda selektywnej galwanizacji styków, którą opracowali, pozwala zaoszczędzić w roku blisko 100 kg tego cennego kruszcu. Bez obniżenia jakości elementów

Już obecnie część tych podzespołów wysyła się na eksport. I nikt z kupujących — a konkurencja duża i wymagania jakościowe nie mniejsze — nie narzeka. Przeciwnie, chcieliby kupować więcej

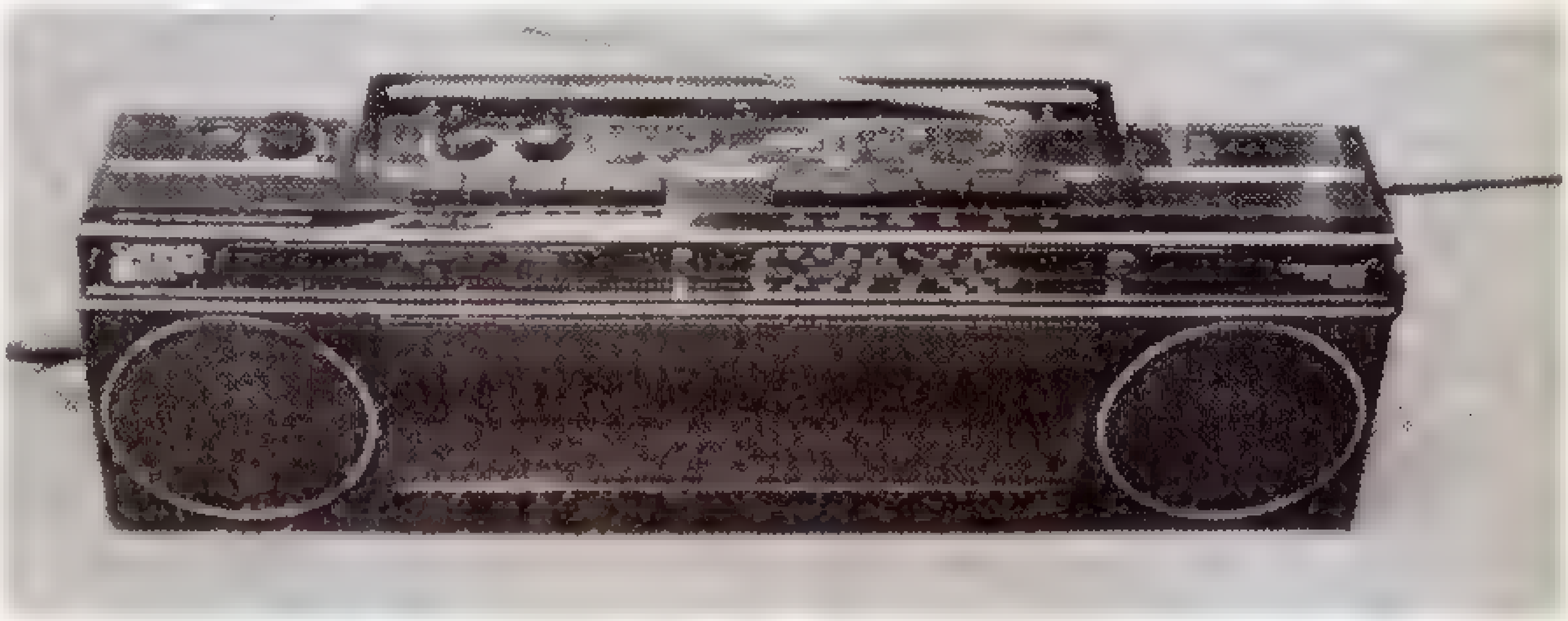
— Naszym głównym zadaniem, jakie stawiamy sobie na najbliższe lata — mówi inż. Terlecki — jest wzrost produkcji podzespołów. Przy tym samym np. zużyciu złota i nie zwiększając zatrudnienia chcemy, w ciągu najbliższych pięciu lat, zwiększyć ich produkcję do 6 mln sztuk, czyli dwukrotnie

Obecnie produkcja podzespołów stanowi 35 proc. produkcji globalnej. Warto wiedzieć, że jesteśmy największym w Polsce producentem podzespołów dla wszystkich przedsiębiorstw, które mają coś wspólnego z elektroniką. Zaopatrujemy w nie przemysł komputerowy, górnictwo, przemysł maszynowy. Wszystkich, którzy przy produkcji potrzebują łączyć obwody drukowane, scalone, panele

Stale rozwija się także zaplecze naukowo-badawcze. Dziś w służbie głównego konstruktora, w sześciu działach, pracuje kilkaset ludzi. Właściwie byłoby powiedzieć — myśli. I efekty tego myślenia przynoszą zakładowi coraz większe zyski i... oszczędności. Kierownik działu konstrukcyjnego, inż Janusz Pazderski, nie chce rozmawiać o tym, nad czym obecnie pracują. Co prawda konkurencja pomiędzy naszymi zakładami nie taka, jak w innych krajach na zachód od Łaby, ale... ostrożność nie zawadzi

Ogólnie można powiedzieć tak: myślą w zakładzie o stykach przyszłości. Jeszcze lepszych, nowocześniejszych, bardziej zminiaturyzowanych. Ale nie tylko. O nowych modelach odbiorników z zastosowaniem ostatnich osiągnięć w zakresie przełączników i kondensatorów. Opracowano już wiele nowych rozwiązań. M.in. przełączniki suwakowo-obrotowe, uchylne, włączniki przyciskowe i impulsowe. Zamiast kondensatorów powietrznych stosować się będzie obrotowe

Słowem, dalsza faza elektronicznej rozbudowy i wprzęgnięcia jej do sterowania układami cyfrowymi oraz do innych techno



Radiomagnetofon produkowany wspólnie z Japończykami SANKEI. Najnowocześniejszy z odtwarzaczem laserowym.

logii specjalnych. Chcą bowiem opracować własne linie produkcyjne włączników, myślą o możliwości produkowania własnych mechanizmów napędowych, gdyż te, które dostarczał kooperujący z „Eltrą" zakład były zbyt niskiej jakości



Przy taśmie montażowej wiele dziewcząt. One najbardziej nadają się do prac, które wymagają cierpliwości i precyzji.

A przecież i tutaj nie wszystko przebiega bez zgrzytów i trudności. W przyzakładowej szkole zawodowej kształci się radiomechaników i ślusarzy. Właśnie brak ślusarzy narzędziowych hamuje tempo wzrostu produkcji. Nie zawsze też wywiązują się ze swych zobowiązań kooperanci. Nikt tu jednak nie szuka usprawiedliwień, co zrobić, żeby nic nie robić, tylko szuka sposobów rozwiązywania kłopotów własnymi siłami

Podpatrywanie Japończyków

Krążyła kiedyś anegdota o tym, jak to pewien wynalazca pobiegł zgłosić do odpowiedniej komórki swój najnowszy wynalazek. Niestety — wrócił z niczym. Gdyby to było dobre, usłyszał od urzędnika zza biurka, to Japończycy już dawno by to wymyślili

Tymczasem Japończycy do „Eltry" przybyli sami. Uczyć się? No, bez przesady. Ale powierzyli zakładowi produkcję najnowszej generacji radiomagnetofonów. Prawdziwych miniaturowych „kombajnów" muzycznych

Właśnie tu, produkuje się od kilku lat serię radiomagnetofonów „Sankej", z najnowszym jej modelem „Sankej" CDR-99. Cóż to takiego? Przenośna, tranzystorowa miniwieża zasilana bateriami lub z sieci, złożona z magnetofonu dwukasetowego, ekalizera, tunera z syntezą częstotliwości, programowaniem elektronicznym i automatycznym wyborem stacji oraz odtwarzaczem CD — również programowanego elektronicznie, umożliwiającego automatyczny wybór określonego utworu z płyty kompaktowej

Cóż się kryje za tymi dwiema literami „CD". Tak, tak, to najnowszy laserowy odtwarzacz komputerowy. Przebój ostatnich lat produkowany w „Eltrze"? A dlaczegóż by nie? Chociaż, co tu kryć, bez pomocy Japończyków nie mogłoby być mowy o podobnej produkcji. Jeszcze nie

„Sankej" CD-99 — wyjaśnia inż. Terlecki — to model, w którym główne podzespoły otrzymujemy od Japończyków. To skomplikowane urządzenie komputerowe, do produkcji którego trzeba specjalistycznych, i wyjątkowo nowoczesnych narzędzi

Aby uzmysłowić sobie doskonałość jego funkcjonowania, a co za tym idzie, precyzję wykonania, najlepiej wyobrazić sobie płytę CD w tysiąckrotnym powiększeniu. Miałaby ona wtedy 120 m średnicy, a plamki tworzące spiralną ścieżkę zapisu, byłyby około 0,5 mm szerokości, 1 mm długości, i 0,1 głębokości. Mimo tak ogromnego powiększenia

Odległość natomiast między dwiema sąsiednimi ścieżkami wynosiłaby 1,6 mm, a czytnik laserowy musiałby przemieszczać się nad taką spiralną ścieżką z dokładnością do 0,1 mm. Nie jesteśmy jeszcze po prostu technologicznie przygotowani do samodzielnego wyprodukowania takiego odtwarzacza

Oczywiście nie oznacza to, że w najbliższej przyszłości nie będziemy umieli tego robić. Na razie podpatrujemy Japończyków, uczymy się najnowszych technologii, zdobywamy doświadczenie

Inżynier twierdzi, że to, co robią obecnie, jest przymiarką do rodzimych rozwiązań. Przekroczyli już wiele technologicznych progów i nie potknęli się. Dlaczego w tym przypadku miałoby być inaczej? Dziś cały przemysł elektroniczny w Polsce przygląda się japońskim osiągnięciom. A jutro?

W „Eltrze" są przekonani, że za kilka lat odtwarzacze kompaktowe naszej produkcji znajdą się w sklepach. Przymierza się do ich produkcji łódzka „Fonika". Być może i oni włączą się do tej produkcji, kto wie?

W zakładzie najwyraźniej widać, jak w elektronicznym pędzie mkną w przyszłość najbardziej rozwinięte kraje świata. Tu też najlepiej rozumieją, że nie zabrać się do tego elektronicznego pociągu, oznacza ograniczyć tempo rozwoju, trafić w cywilizacyjną dziurę. Stąd ta zapobiegliwość. Chęć trzymania się za wszelką cenę, jeśli nie w czołówce, to przynajmniej w peletonie

I może dlatego to, co robią w „Eltrze", napawa optymizmem. Mimo wszystko

Włodzimierz KALETA

CPC 6128

EKSPLOZJA

```
10 ' Eksplozja
20 INPUT"ILE RAZY POWTORZYC GLOS EKSPLOZJI";I
30 FOR L=1 TO I
40 ENT -1,15,1,5
50 SOUND 1,70,75,5,0,1
60 ENV 1,1,15,1,15,-1,10
70 SOUND 1,140,0,0,1,0,12
80 NEXT
```

KARABIN MASZYNOWY

```
10 ' Karabin maszynowy
20 INPUT "ILE RAZY POWTORZYC GLOS KARABINU";I
30 FOR L=1 TO I
40 FOR k = 1 TO 3
50 ENV 1,5,[illegible],1,1,-5,[illegible],1,1,-15,1
60 ENT -1,1,5,[illegible],1,-5,1
70 SOUND 1,162,-7,5,[illegible],1,7
80 SOUND 1,0,10
90 NEXT
100 NEXT
```

SYRENA

```
10 ' Syrena
20 INPUT"ILE RAZY POWTORZYC GLOS SYRENY";I
30 FOR L=1 TO I
40 ENV 2,10,1,1,2,1,1
50 ENT -2,12,2,0.5
60 p=40:g=1
70 FOR k=1 TO 50
80 IF k<25 AND p<45 THEN p=p+0.5
90 IF k>25 AND g>1 THEN g=g-1
100 IF k>20 AND g<12 THEN g=g+1
110 SOUND 2,p,-2,g,2,2
120 NEXT
130 NEXT
```

Standard MSX, a mikrokomputer SVI-738 X'PRESS

Józef RAJZER

Pod koniec 1986 roku w Centralnej Składnicy Harcerskiej oprócz mikrokomputerów TC—2048 pojawiły się również mikrokomputery SVI—738 X'PRESS firmy SPECTRAVIDEO z Hongkongu wraz z pewną liczbą urządzeń zewnętrznych i oprogramowaniem podstawowym

SVI—738 należy do tzw. standard MSX, który jest próbą ujednolicenia produkcji mikrokomputerów domowych na wzór standardu IBM, jaki wytworzył się dla profesjonalnych mikrokomputerów personalnych. Wraz z SVI—738 omówiony zatem zostanie jednocześnie standard MSX.

Prekursorem tego standardu jest tokijska filia amerykańskiej firmy MICROSOFT, a głównie jej projektant, Mr. Mishi, który w 1983 r. zaprojektował model mikrokomputera MSX dla firmy SPECTRAVIDEO. W standardzie MSX przyjęto szereg ujednoliceń. W mikrokomputerach zastosowano mikroprocesor Z80 z taktem częstotliwości 3.58 MHz. Pamięć mikrokomputera składa się z kilku bloków: bloku pamięci ROM o pojemności 32 kB, w której umieszczony jest interpreter rozszerzonego języka BASIC, bloku ROM 16 kB dla potrzeb mikrodysku, 8kB ROM dla potrzeb łącza szeregowego RS—232C, bloku 16 kB RAM przeznaczonego do pamięci obrazu i bloku 64 kB jako pamięci operacyjnej. Dodatkowo można dołączyć zewnętrzną kartę pamięci RAM o pojemności 64 kB. Mikrokomputer ma wbudowany generator akustyczny o trzech niezależnych kanałach akustycznych i kanale szumowym. Generator wytwarza dźwięki w zakresie 8 oktaw. Amplituda dźwięku i kształt obwiedni mogą być tworzone programowo.

Klawiatura mikrokomputerów MSX nie jest ściśle znowelizowana, występują jednak te same grupy klawiszy w ten sam sposób rozmieszczone. Mamy więc:

— cztery duże klawisze do przesuwania kursora w 4 kierunkach,
— klawiaturę alfanumeryczną, która może być dostosowana do alfabetu danego kraju. Pierwsza partia sprowadzonych SVI ma klawiaturę angielską, następne mają mieć już klawiaturę polską. Klawisze alfanumeryczne po użyciu klawiszy sterujących CODE, SHIFT — CODE, GRAPM i SHIFT—GRAPM dają różne znaki graficzne, symbole arytmetyczne, litery charakterystyczne dla alfabetów greckiego, niemieckiego, szwedzkiego itp. Łącznie dzięki klawiaturze alfanumerycznej można uzyskać 255 znaków,
— klawisze sterujące np. ENTER, TAB, CTRL, STOP, CLS/HM itp
— klawisze funkcyjne F1 — F10, które mogą być zaprogramowane zgodnie z potrzebami użytkownika.

We wspólnej obudowie z mikrokomputerem znajduje się stacja mikrodysku o mało znanej u nas wersji dyskietek 3,5 cala. Dyskietka zapisywana jest jednostronnie z podwójną gęstością. Powierzchnia robocza dyskietki podzielona jest na 80 ścieżek po 9 sektorów. Pojemność dyskietki wynosi 360 kB, a szybkość transmisji 250 kb/s

Teraz o tym, co można dołączyć do mikrokomputera, a można dość dużo. Mikrokomputer wyposażony jest w szereg gniazd, przez które można do mikrokomputera dołączyć:

— monitor kolorowy lub odbiornik telewizyjny. Obraz na ekranie tworzony jest za pomocą kontrolera graficznego TMS—9918A. Rastrowy wymiar ekranu w trybie grafiki wykreślnej wynosi 256x192 punkty. W trybie grafiki tekstowej możemy uzyskiwać wydruk w 24 wierszach po 40 znaków lub w 24 wierszach po 80 znaków. Obraz może być wyświetlany w 16 kolorach,
— dla potrzeb gier można podłączyć dwa manipulatory (joysticki) typu QUICSHOT MSX,
— magnetofon kasetowy SVI—767 jako pamięć zewnętrzna, w której znajduje się oprogramowanie firmowe,
— dodatkową stację mikrodysków 3,5 cala lub 5.25 cala, co znacznie zwiększa możliwości przetwarzania dość dużych zbiorów danych,
— drukarkę — przez równoległe złącze typu Centronics. Drukarka może drukować różnymi wzorami druku. Do SVI—738 można dołączać drukarki wyposażone w trakcie łącza

Mikrokomputer wyposażony jest ponadto w gniazdo łącza RS—232—C, które umożliwia przekazywanie informacji między dwoma mikrokomputerami. Można łączyć mikrokomputery liniami telefonicznymi. W tym wypadku mikrokomputery są dołączone do linii telefonicznych przez modemy

Ciekawostką jest ponadto możliwość zastosowania karty sprzęgu SVI—709 która umożliwia podłączenie mikrokomputera do lokalnej sieci mikrokomputerowej SVI—609 jako inteligentnej końcówki i korzystanie z zasobów informacji, znajdujących się na dysku stałym komputera głównego tej siec



Grafika na ekranie (cz. 4)

Andrzej CETERA

Poprzednio rozpatrywany był obrót zobrazowania na ekranie wokół początku układu współrzędnych.
Znajduje on również zastosowanie przy obrocie względem dowolnego punktu, nie będącego początkiem układu współrzędnych. W ogólnym wypadku następuje to przez przesunięcie początku układu współrzędnych do punktu obrotu, w obrocie wokół początku układu i przesunięciu układu współrzędnych do pierwotnego położenia.
Obrót wektora położenia [xy1] wokół punktu (m, n) o dowolny kąt θ może być wykonany za pomocą przekształcenia

$$[xy1]\begin{bmatrix}1&0&0\\0&1&0\\-m&-n&1\end{bmatrix}\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\m&n&1\end{bmatrix}=[XYH]$$

Po przekształceniu otrzymamy

$$[XYH]=[xy1]\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\-m(\cos\theta-1)+&-m(\sin\theta)-&\\+n(\sin\theta)&-n(\cos\theta-1)&1\end{bmatrix}$$

Rozpatrzmy następujący przykład.

Przyjmijmy, że chcemy wykonać obrót zobrazowania o kąt 90°, w kierunku przeciwnym do obrotu wskazówek zegara, wokół osi prostopadłej do płaszczyzn XY i przechodzącej przez punkt (4,3).
Obrót względem początku układu współrzędnych możemy wykonać za pomocą macierzy transformacji

$$\begin{bmatrix}0&1&0\\-1&0&0\\0&0&1\end{bmatrix}$$

Przesunięcie układu współrzędnych realizujemy za pomocą macierzy transformacji

$$\begin{bmatrix}1&0&0\\0&1&0\\-4&3&1\end{bmatrix}$$

Po wykonaniu przesunięcia i obrotu otrzymany rezultat należy sprowadzić do pierwotnego układu współrzędnych, co wykonamy wykorzystując wsteczną macierz transformacji

$$\begin{bmatrix}1&0&0\\0&1&0\\4&3&1\end{bmatrix}$$

Całą zatem operację zapiszemy

$$[XYH]=[xy1]\begin{bmatrix}1&0&0\\0&1&0\\-4&-3&1\end{bmatrix}\begin{bmatrix}0&1&0\\-1&0&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\4&3&1\end{bmatrix}$$

Po wymnożeniu macierzy transformacji otrzymamy

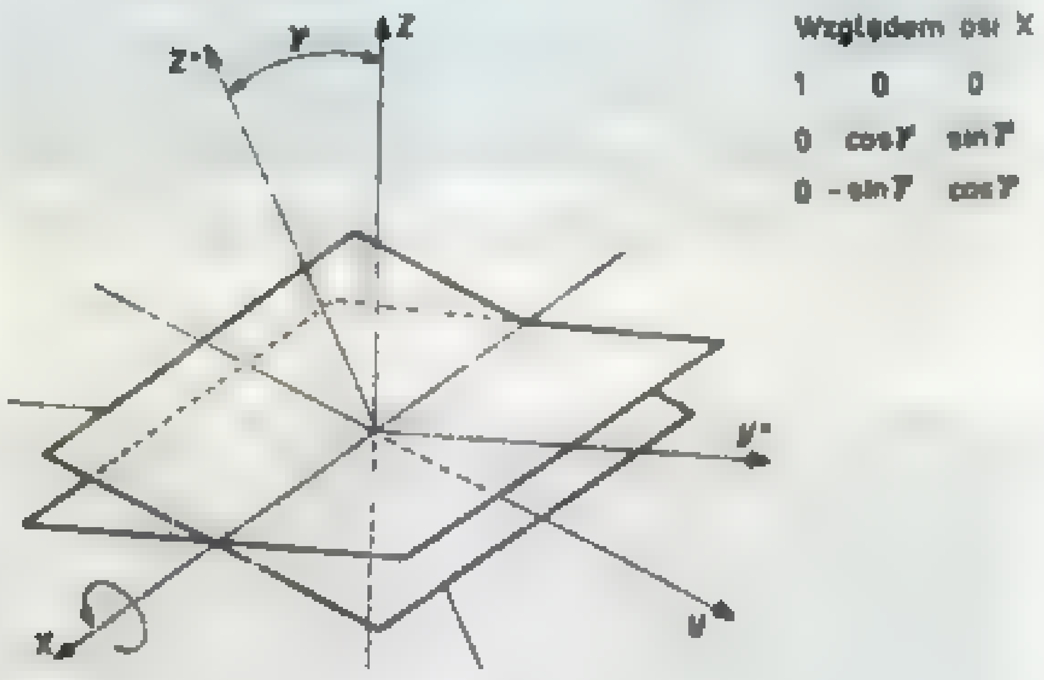
$$[XYH]=[xy1]\begin{bmatrix}0&1&0\\-1&0&0\\7&-1&1\end{bmatrix}$$

Rozważymy teraz przykłady, w których osią obrotu jest jedna z osi współrzędnych. Mamy więc trzy warianty do rozpatrzenia, w których jedna z osi będzie prostopadła do ekranu (oś połdodatnia).

a) oś x biegnie do ekranu
b) oś y biegnie do ekranu
c) oś z biegnie do ekranu

Zadanie zatem jest zredukowane do obrotu w przestrzeni dwuwymiarowej i punkt (x, y, z) w pierwotnym układzie współrzędnych staje się w nowym układzie punktem (x', y', z'), gdzie

dla obrotu wokół osi x

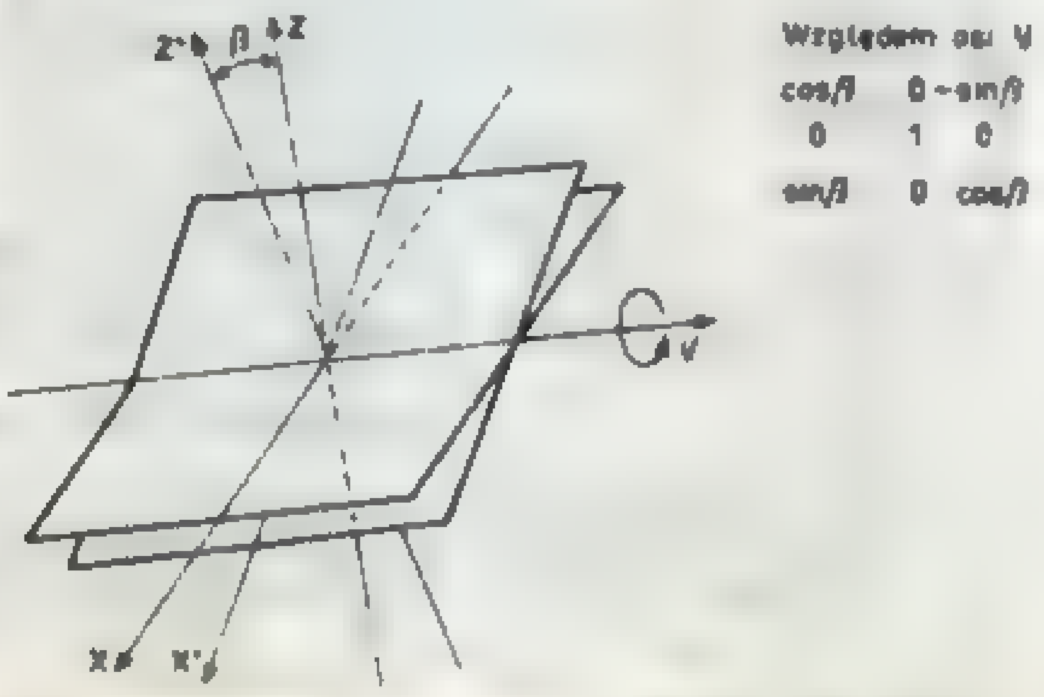


$$x' = x$$
$$y' = y\cos\gamma + z\sin\gamma$$
$$z' = -y\sin\gamma + z\cos\gamma$$

a macierz transformacji ma postać

$$\begin{bmatrix}1&0&0&0\\0&\cos\gamma&\sin\gamma&0\\0&-\sin\gamma&\cos\gamma&0\\0&0&0&1\end{bmatrix}$$

dla obrotu wokół osi y



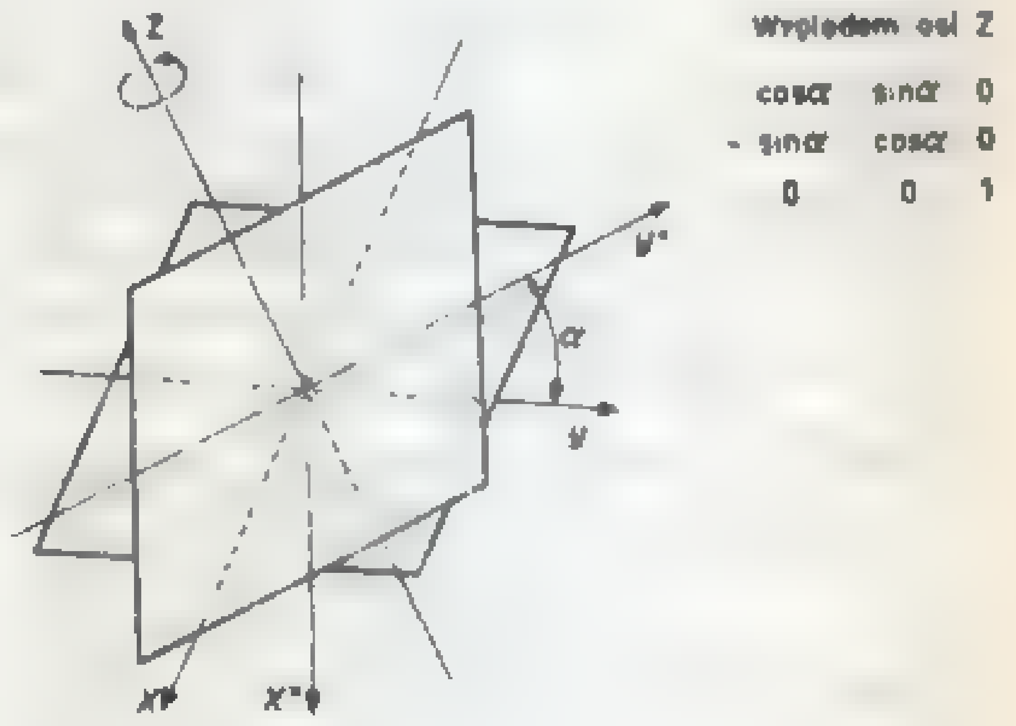
$$x' = z\sin\beta + x\cos\beta$$
$$y' = y$$
$$z' = z\cos\beta + x - \sin\beta$$

a macierz transformacji ma postać

$$\begin{bmatrix}\cos\beta&0&-\sin\beta&0\\0&1&0&0\\\sin\beta&0&\cos\beta&0\\0&0&0&1\end{bmatrix}$$

dla obrotu wokół osi z

$$x' = x\cos\alpha + y\sin\alpha$$
$$y' = -x\sin\alpha + y\cos\alpha$$
$$z' = z$$



a macierz transformacji ma postać

$$\begin{bmatrix}\cos\alpha&\sin\alpha&0&0\\-\sin\alpha&\cos\alpha&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

W sytuacji, gdy mamy do czynienia z figurami płaskimi zdefiniowanymi na płaszczyźnie, macierze transformacji przyjmą następującą postać

R1, R2, R3

Omówiona problematyka została praktycznie wykorzystana w prezentowanym poniżej programie.
Program realizuje obroty zdefiniowanych figur względem dowolnie wybranego punktu lub osi układu współrzędnych. Możliwe jest także modyfikowanie figur przez dorysowywanie kolejnych elementów.
Współrzędne punktów definiujących figurę wprowadza się w jednostkach umownych, które są odmierzone na osiach układu współrzędnych. Mogą nimi być liczby całkowite i zmienno-przecinkowe. Gdy cała zdefiniowana figura ma być widoczna na ekranie monitora, współrzędne muszą się zawierać w przedziałach

x (—16, 16) i y (—9.5, 9,5)

W celu łatwiejszej orientacji figur na ekranie, należy wyświetlić siatkę punktów z dyskretnością jednostek umownych.
Wartości kątów obrotu wprowadzamy w stopniach. Wprowadzenie punktu o współrzędnych x 998 i dowolnej wartości współrzędnej y powoduje zakończenie definiowania figury.

Poniżej przedstawiamy przebieg i obsługę operatorską programu

KOMUNIKAT	OPERACJA
1. Całkowita widoczność rysowanej figury, gdy współrzędne zawierają się w przedziale: x:(—16, 16), y:(—9,5, 9,5) Siatka współrzędnych (t/n)	Wprowadzamy literę „t", gdy chcemy mieć na ekranie wyświetloną siatkę punktów, a gdy nie chcemy — literę „n".
2. Podaj wsp. punktu x, y	Wprowadzamy wsp. (x, y) punktu definiującego figurę lub punkt kończący definiowanie
3. Podaj kąt obrotu	Wprowadzamy w stopniach kąt obrotu
4. Obrót względem punktu (t/n)	Wprowadzenie litery „t" spowoduje wybranie funkcji, obrót względem punktu i przejście do komunikatu 10. Wprowadzenie litery „n", to wybranie funkcji obrót względem osi
5. Obrót względem osi (x/y/z)	Wprowadzamy literę odpowiadającą jednej z osi x; y; z. W wyniku otrzymujemy zobrazowanie figury po obrocie.

```
10 REM ****************************************
20 REM
30 REM Obroty figury wzgledem wybranego
40 REM        punktu lub osi
50 REM
60 REM ****************************************
70 DIM wsp(101,3),mat(3,3),nwsp(101,3)
80 MODE 1:INK 0,13:INK 1,3:INK 2,12:INK 3,0
90 BORDER 0:PEN 1:PAPER 2
100 CLS
110 PRINT "Calkowita widocznosc rysowanej"
120 PRINT "figury gdy wspolrzedne zawieraja"
130 PRINT "sie w przedziale x:(-16,16),y:(-9.5,9.5)"
140 PRINT
150 PRINT "Siatka wspolrzednych (t/n)"
160 z$=INKEY$
170 IF z$=" " THEN 160
180 IF z$="t" THEN fl=1:GOTO 210
190 IF z$="n" THEN fl=0:GOTO 210
200 GOTO 160
210 GRAPHICS PAPER 1:REM Tlo grafiki
220 ORIGIN 320,208,0,640,16,400
230 CLEAR INPUT
240 WINDOW 1,40,25,25
250 GOSUB 1570:REM Rys. ukladu wspol.
260 i=0 :REM Licznik punktow
270 INPUT "Podaj wsp. punktu x,y";x,y
280 IF x>998 THEN 350
290 i=i+1
300 wsp(i,1)=x*20
310 wsp(i,2)=y*20
320 wsp(i,3)=1
330 GOSUB 1050:REM Wykresl. punkt lub odcinek
340 GOTO 270
350 INPUT"Podaj kat obrotu";ak
360 ak=ak*PI/180
370 ac=COS(ak):as=SIN(ak)
380 PRINT"Obrot wzgledem punktu (t/n)"
390 nk$=INKEY$
400 IF nk$=" "THEN 390
410 IF nk$="t"THEN 440
420 IF nk$="n"THEN 480
430 GOTO 390
440 INPUT"Podaj wsp.puktu obrotu";p,q
450 p=-p*20:q=-q*20
460 GOSUB 1120:REM Ustaw macierz przesuniecia
470 GOTO 620
480 PRINT"Obrot wzgledem osi (x/y/z)"
490 nk$=INKEY$
500 IF nk$=" "THEN 490
510 IF nk$<>"x" THEN 540
520 GOSUB 1960:REM Ustawienie macierzy
530 GOTO 590
540 IF nk$<>"y" THEN 570
550 GOSUB 2040:REM Ustawienie macierzy
560 GOTO 590
570 IF nk$<>"z" THEN 490
580 GOSUB 1410:REM Ustawienie macierzy
590 GOSUB 1240:REM Mnozenie macierzy
600 GOSUB 1480:REM Zobrazowanie
610 GOTO 690
620 GOSUB 1240 :REM Mnozenie
630 GOSUB 1410:REM Ustaw macierz obrotu
640 GOSUB 1240:REM Mnozenie macierzy
650 p=-p:q=-q
660 GOSUB 1120:REM Ustaw macierz przesuniecia
670 GOSUB 1240:REM Mnozenie macierzy
680 GOSUB 1480 :REM Zobrazowanie
690 PRINT"Czy koniec (t/n)
700 nk$=INKEY$
710 IF nk$=" "THEN 700
720 IF nk$="t"THEN 2120
730 IF nk$="n"THEN 750
740 GOTO 700
750 PRINT"Czy od poczatku (t/n)"
760 nk$=INKEY$
770 IF nk$=" "THEN 760

780 IF nk$="t"THEN 80
790 IF nk$="n"THEN 810
800 GOTO 760
810 PRINT"Czy obrot (t/n)"
820 nk$=INKEY$
830 IF nk$=" "THEN 820
840 IF nk$="t"THEN 350
850 IF nk$="n"THEN 870
860 GOTO 820
870 PRINT"Czy dorysowywanie (t/n)"
880 nk$=INKEY$
890 IF nk$=" "THEN 880
900 IF nk$="t"THEN 930
910 IF nk$="n"THEN 690
920 GOTO 880
930 CLS
940 GRAPHICS PAPER 1:REM Tlo grafiki
950 ORIGIN 320,208,0,640,16,400
960 CLEAR INPUT
970 WINDOW 1,40,25,25
980 GOSUB 1570:REM Rys.ukladu wspolrzednych
990 GOSUB 1480:REM Rysowanie figury
1000 GOTO 270
1010 REM ***************************************
1020 REM       PODPROGRAMY
1030 REM ***************************************
1040 REM
1050 REM Rysowanie punktu lub odcinka
1060 REM
1070 GRAPHICS PEN 3:REM stare wspolrzedne
1080 IF i=1 THEN PLOT wsp(i;1),wsp(i,2)
1090 IF i>1 THEN DRAW wsp(i,1),wsp(i,2)
1100 RETURN
1110 REM
1120 REM Ustawianie macierzy przesuniecia
1130 REM
1140 mat(1,1)=1:mat(1,2)=0:mat(1,3)=0
1150 mat(2,1)=0:mat(2,2)=1:mat(2,3)=0
1160 mat(3,1)=p:mat(3,2)=q:mat(3,3)=1
1170 FOR k=1  TO i:REM Zerowanie nwsp
1180 FOR l=1 TO 3
1190 nwsp(k,l)=0
1200 NEXT l
1210 NEXT k
1220 RETURN
1230 REM
1240 REM Mnozenie macierzy
1250 REM
1260 FOR n=1 TO i
1270 FOR k=1 TO 3
1280 FOR m=1 TO 3
1290 nwsp(n,k)=nwsp(n,k)+wsp(n,m)*mat(m,k)
1300 NEXT m
1310 NEXT k
1320 NEXT n
1330 FOR n=1 TO i
1340 FOR m=1 TO 3
1350 wsp(n,m)=nwsp(n,m)
1360 nwsp(n,m)=0
1370 NEXT m
1380 NEXT n
1390 RETURN
1400 REM
1410 REM Ustawianie macierzy obrotu
1420 REM
1430 mat(1,1)=ac:mat(1,2)=as:mat(1,3)=0
1440 mat(2,1)=-as:mat(2,2)=ac:mat(2,3)=0
1450 mat(3,1)=0:mat(3,2)=0:mat(3,3)=1
1460 RETURN
1470 REM
1480 REM Rysowanie nowej figury
1490 REM
1500 GRAPHICS PEN 0:REM PIORO nowych wspol.
1510 PLOT wsp(1,1),wsp(1,2)
1520 FOR n=2 TO i
1530 DRAW wsp(n,1),wsp(n,2)
1540 NEXT
1550 RETURN
1560 REM
1570 REM Rysowanie ukladu wspolrzednych
1580 REM
1590 CLG 1
1600 GRAPHICS PEN 2:REM Pioro ukladu wspol.
1610 FOR a=-200 TO 200 STEP 20
1620 PLOT -2,a
1630 DRAWR 4,0
1640 NEXT a
1650 PLOT 0,-208
1660 DRAW 0,208
1670 TAG
1680 MOVER -16,-20
1690 PRINT "Y";
1700 MOVER -310,0
1710 PRINT "20";
1720 TAGOFF
1730 PLOT -307,170 : REM Kreslenie podzialki
1740 DRAWR 0,-4
1750 DRAWR 0,2
1760 DRAWR 20,0
1770 DRAWR 0,2
1780 DRAWR 0,-4
1790 IF fl=1 THEN 1800 ELSE 1850
1800 FOR b=-200 TO 200 STEP 20
1810 FOR a=-320 TO 320 STEP 20
1820 PLOT a,b
1830 NEXT a
1840 NEXT b
1850 FOR a=-320 TO 320 STEP 20
1860 PLOT a,-2
1870 DRAWR 0,4
1880 NEXT a
1890 PLOT -320,0
1900 DRAW 320,0
1910 TAG
1920 MOVER -16,-4
1930 PRINT "X";
1940 TAGOFF
1950 RETURN
1960 REM
1970 REM ustawianie macierzy obrotu
1980 REM        wzgledem osi X
1990 REM
2000 mat(1,1)=1:mat(1,2)=0:mat(1,3)=0
2010 mat(2,1)=0:mat(2,2)=ac:mat(2,3)=as
2020 mat(3,1)=0:mat(3,2)=-as:mat(3,3)=ac
2030 RETURN
2040 REM
2050 REM ustawianie macierzy obrotu
2060 REM        wzgledem osi Y
2070 REM
2080 mat(1,1)=ac:mat(1,2)=0:mat(1,3)=-as
2090 mat(2,1)=0:mat(2,2)=1:mat(2,3)=0
2100 mat(3,1)=as:mat(3,2)=0:mat(3,3)=ac
2110 RETURN
2120 REM
2130 REM Koniec
2140 REM
2150 WINDOW 1,40,1,25
2160 CLS
2170 CLG 1
2180 END
```

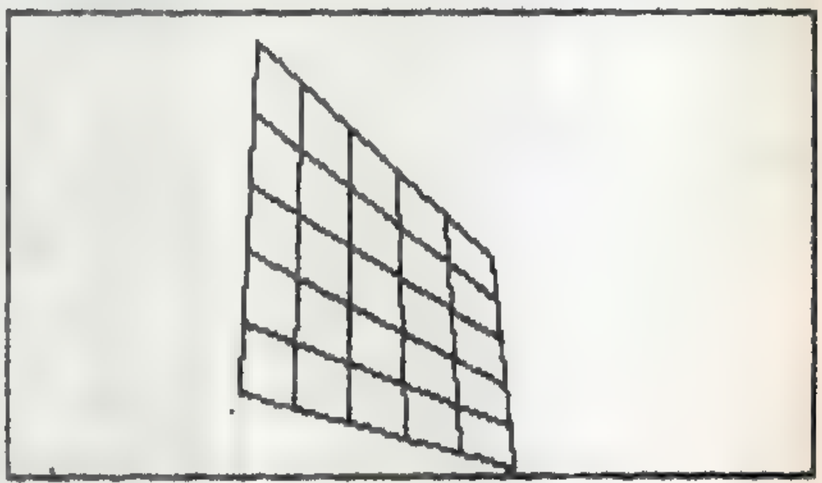
KOMUNIKAT	OPERACJA
6. Czy koniec (t/n)	Wybranie litery „t" kończy działanie programu. Kontynuacja programu, gdy wybierzemy literę „n"
7. Czy od początku (t/n)	Wybranie litery „t" powoduje przejście do komunikatu 1. Wybranie „n" kontynuacja programu
8. Czy obrót (t/n)	Wybranie „t" przejście do komunikatu 3, litera „n" dalszy ciąg programu.
9. Czy dorysowywanie (t/n)	Wybranie litery „t" to wybranie funkcji dorysowywania i przejście do komunikatu 2. Wybranie „n" powoduje przejście do komunikatu 6.
10. Podaj wsp punktu obrotu	Wprowadzamy współrzędne punktu obrotu. Otrzymujemy nowe zobrazowanie i przejście do komunikatu 6.

Animacja kwadratu

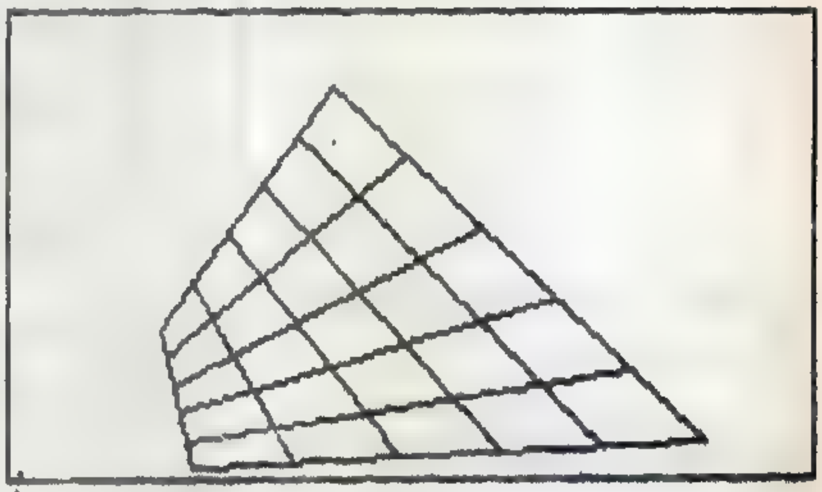
```
 100 REM   ANIMACJA KWADRATU

 110 CLEAR 32755: FOR n=32756 TO
32767: READ b: POKE n,b: NEXT n
: REM przeniesienie blokowe
 120 DATA 33,0,128,17,0,72,1,0,1
6,237,176,201
 125 CLS : BEEP .02,26: BEEP .01
,14
 130 PLOT 1,49: DRAW 253,0: DRAW
0,125: DRAW -253,0: DRAW 0,-125
 140 DIM a$(512): DIM x(8,4): DI
M y(8,4)
 149 REM oblicz. wspolrzednych
 150 FOR m=1 TO 4:
LET x(1,m)=2+INT (251*RND):
LET x(8,m)=2+INT (251*RND):
LET y(1,m)=50+INT (124*RND):
LET y(8,m)=50+INT (124*RND)
 160 NEXT m
 170 FOR m=1 TO 4
 180 LET skokx=(x(8,m)-x(1,m))/8
:    LET skoky=(y(8,m)-y(1,m))/8
 190 FOR n=1 TO 8:
LET x(n,m)=x(1,m)+(n-1)*skokx:
LET y(n,m)=y(1,m)+(n-1)*skoky:
NEXT n: NEXT m
 200 FOR m=1 TO 8:
LET x1=(x(m,2)-x(m,1))/5:
LET x2=(x(m,4)-x(m,3))/5:
LET x3=(x(m,4)-x(m,1))/5:
LET x4=(x(m,2)-x(m,3))/5:
LET y1=(y(m,2)-y(m,1))/5:
LET y2=(y(m,4)-y(m,3))/5:
LET y3=(y(m,4)-y(m,1))/5:
LET y4=(y(m,2)-y(m,3))/5
 209 REM zapis kolejnych faz
 210 PLOT 1,49: DRAW 253,0: DRAW
0,125: DRAW -253,0: DRAW 0,-125
: PLOT 1+2*m,51
 220 FOR z=0 TO 5: PLOT x(m,1)+z
*x1,y(m,1)+z*y1: DRAW x(m,4)-z*x
2-PEEK 23677,y(m,4)-z*y2-PEEK 23
678

 230 IF INKEY$<>"" THEN GO TO 12
0
 240 NEXT z
 250 FOR z=0 TO 5: PLOT x(m,1)+z
*x3,y(m,1)+z*y3: DRAW x(m,2)-z*x
4-PEEK 23677,y(m,2)-z*y4-PEEK 23
678
 260 IF INKEY$<>"" THEN GO TO 12
0
 270 NEXT z
 280 POKE 32758,64: POKE 32761,1
28+16*(m-1): RANDOMIZE USR 32756
 290 BEEP .02,m+26: CLS : NEXT m
 300 PRINT BRIGHT 1;a$
 309 REM wyswietlanie "tam"
 310 FOR r=1 TO 12: PRINT AT 16,
0;r: FOR n=1 TO 8: BEEP .01,n: P
AUSE r/2
 320 POKE 32761,64: POKE 32758,1
28+16*(n-1): RANDOMIZE USR 32756
 330 IF INKEY$<>"" THEN GO TO 12
0
 340 NEXT n
 349 REM wyswietl."z powrotem"
 350 FOR n=7 TO 2 STEP -1: BEEP
.01,n: PAUSE r/2
 360 POKE 32761,64: POKE 32758,1
28+16*(n-1): RANDOMIZE USR 32756
 370 IF INKEY$<>"" THEN GO TO 12
0
 380 NEXT n: NEXT r: GO TO 120
8999 REM zapis na tasme
9000 CLEAR : RESTORE : SAVE "ANI
MACJA" LINE 100: VERIFY "": GO T
O 100
```



6



4

ZX SPECTRUM

Przeliczanie

```
   1 CLS : PRINT BRIGHT 1;
   AT 8,12;"ENTER=CLS";
   AT 14,12;" 0 LUB 1 "
   2 LET a$=""
   3 PAUSE 0: LET b$=INKEY$:
IF b$=CHR$ 13 THEN GO TO 1
   4 IF (b$="0" OR b$="1") AND L
EN a$<=15 THEN BEEP .01,14: LET
a$=a$+b$: PRINT AT 11,2;a$;" = B
IN ";VAL ("BIN "+a$)
   5 GO TO 3
   6 SAVE "BIN" LINE 1
```

Budżet domowy

S. WASILEWSKI

Każdy z nas prowadzi działalność finansową na mniejszą lub większą skalę. Jeśli chcesz ocenić i usprawnić gospodarowanie własnymi zasobami pieniężnymi, skorzystaj z programu „BUDŻET DOMOWY", opracowanego na mikrokomputer Commodore 64 (lub C-128 w trybie pracy C-64).

„BUDŻET DOMOWY" ma na tyle uniwersalną konstrukcję, że może być wykorzystywany zarówno przez pojedyncze osoby, jak i grupy do rejestrowania dowolnej formy działalności finansowej. Nie wymaga tworzenia i korzystania z dodatkowych zbiorów danych. Wszystkie niezbędne informacje gromadzone są na bieżąco w nim samym. Z tego względu zamieszczona postać programu jest wyjściową dla wszystkich jego zastosowań.

Obszar działań finansowych rejestrowanych przez program określa użytkownik. Następuje to podczas wprowadzania elementów struktury tego obszaru. Dla przykładu weźmy pod uwagę obszar działań finansowych rodziny. Można go podzielić na grupy artykułów: spożywcze, mięso, wędliny i inne. W obrębie grupy można określić pojedyncze artykuły (mleko, sery, pieczywo i inne w grupie artykułów spożywczych). Tworząc taką sieć powiązań elementów struktury za pomocą programu, użytkownik sam ustala obszar pracy programu i sposób odwoływania się do poszczególnych artykułów w trakcie prowadzenia ewidencji finansowej. Dla poszczególnych artykułów, grup artykułów, a także dla całej istniejącej struktury (globalnie) prowadzona jest ewidencja finansowa w zakresie:

— wartości środków finansowych planowanych na zakupy (pozycja plan);
— wartość wydatków (pozycja wydatki).

Wyświetlając aktualny stan poszczególnych pozycji „BUDŻET DOMOWY" wyprowadza dodatkowo pozycję bilans, która jest różnicą między planem i wydatkami.

Z punktu widzenia użytkownika w programie można wyróżnić trzy poziomy pracy. Na każdym poziomie istnieje zestaw funkcji możliwych do wykonania. Są one każdorazowo prezentowane na ekranie za pomocą jednoznakowego symbolu wywołującego funkcję oraz wyrazu opisującego jej znaczenie. Przed rozpoczęciem eksploatacji programu „BUDŻET DOMOWY" należy zapoznać się z jego obsługą.

Poziom pierwszy — występuje po uruchomieniu programu komendą RUN, a także po zakończeniu pracy na poziomie drugim. Umożliwia globalne operacje na istniejącej strukturze finansowej. Zawiera funkcje:

KONIEC — kończy pracę programu. Po pojawieniu się pytania DATA? należy podać datę (MM.DD) lub inny dowolny łańcuch pięcioznakowy identyfikujący aktualnie utworzoną wersję „BUDŻETU DOMOWEGO". Program kończy się po wyprowadzeniu wiersza postaci:
SAVE BUDŻET xxxxx gdzie xxxxx jest podanym identyfikatorem wersji. Użytkownik może teraz skopiować aktualną wersję programu na taśmę (jeśli ją akceptuje).

BUDŻET — zobrazowuje globalny stan gospodarki finansowej, jaką obejmuje program. Zawiera pozycje: plan, wydatki i bilans.

PLAN — pozwala uaktualnić globalną pozycję plan. Wartość początkowa zero, a kolejne podawane wartości są sumowane.

LISTA — powoduje przejście do poziomu drugiego. Jeżeli była utworzona struktura finansowa, to jest wyprowadzana na ekranie jej pierwsza część.

ZEROWANIE — automatyczne wyzerowanie wszystkich pozycji w utworzonej strukturze finansowej bez naruszenia tej struktury.

Poziom drugi — występuje po zainicjowaniu funkcji LISTA z poziomu pierwszego lub funkcji KONIEC z poziomu trzeciego. Umożliwia operacje w obrębie grup artykułów lub artykułów nie tworzących grupy. Zawiera funkcje:

KONIEC — powrót na poziom pierwszy.

DOPISANIE — umożliwia dopisanie na listę nowego elementu struktury. Wymagane jest podanie nazwy. Dostęp do dopisanych na tym poziomie elementów uzyskasz po ponownym wejściu z poziomu pierwszego (wykonanie kolejno funkcji: KONIEC, LISTA).

GRUPA — powoduje przejście na poziom drugi. Jeśli wskazany element ma tworzyć grupę, to na poziomie trzecim udostępniane są operacje pozwalające utworzyć listę artykułów wchodzących w skład tej grupy. Jeśli wskazany element tworzy już grupę, to na ekranie jest wyprowadzana pierwsza część listy artykułów wchodzących w jej skład.

USUNIĘCIE — umożliwia usunięcie z istniejącej struktury wskazanego elementu. Jeśli jest to grupa, to usuwana jest wraz ze wszystkimi wchodzącymi w jej skład artykułami. Globalna suma wydatków zarejestrowanych w programie nie jest pomniejszana o wartość wydatków usuniętych struktur.

LISTA — wyprowadza (jeśli istnieje) kolejną część listy z tego poziomu.

PLAN — pozwala zmienić wartość planu wskazanego elementu struktury (grupy artykułów lub pojedynczego artykułu). Wartość początkowa zero, a kolejne podawane wartości są sumowane.

WYDATKI — uaktualnia wartość wydatków związanych ze wskazanym elementem. Wartość początkowa zero, a kolejne podawane wartości są sumowane.

ZEROWANIE — wykonuje automatyczne wyzerowanie wszystkich pozycji we wskazanym elemencie struktury bez naruszania tej struktury.

Poziom trzeci — występuje po zainicjowaniu funkcji **GRUPA** z poziomu drugiego. Umożliwia operacje w obrębie grupy artykułów wskazanej na poziomie drugim, a udostępnianych na poziomie trzecim. Zawiera funkcje podobne jak na poziomie drugim. Niedostępna jest tylko funkcja **GRUPA**.

Na poziomach drugim i trzecim można poruszać się w obrębie wyświetlonej na ekranie listy za pomocą klawiszy przesuwających kursor

w dół i w górę. Takie funkcje jak: **PLAN, WYDATKI, ZEROWANIE** i **GRUPA** dotyczą zawsze tych elementów struktury, na które aktualnie wskazuje kursor.

```
10 REM ****** BUDZET DOMOWY ******
11 GOSUB30:POKE53280,0:POKE53281,0:POKET
,192:PRINTC$:POKET,4
12 FORI=HTO739:POKEI,0:NEXT:POKEL,2:POKE
D,9:POKEG,0
13 RESTORE:READL$,A:B=718:GOSUB40:B=700:
A=2001:GOSUB40:GOSUB20:GOTO100
20 RESTORE:READL$:PRINTC$:POKEE,19:PRINT
:PRINTA$D$L$"      BUDZET DOMOWY      C S.
W.";
21 PRINT:PRINT"K"B$" - KONIEC    "A$"L"B$
" - LISTA      ";
22 IFPEEK(L)=0THENPRINTA$"G" - GRUPA"
23 IFPEEK(L)=1THENPRINT
24 IFPEEK(L)=2THENPRINTA$"B"B$" - BUDZET
";GOTO26
25 PRINTA$"W"B$" - WYDATKI   "A$"D"B$" -
DOPISANIE "A$"U"B$" - USUNIECIE";
26 PRINTA$"P"B$" - PLAN";
27 PRINTSPC(19)A$"Z"B$" - ZEROWANIE";:RE
TURN
30 A$=CHR$(18):B$=CHR$(146):C$=CHR$(147)
:H$=CHR$(19):R$=A$:FORI=1TO40:R$=R$+" "
31 S$=S$+"    ":NEXT:R$=R$+B$:E=214:D=68
2:G=684:H=680:L=679:T=648:D$="DATA"
32 DEFFNB(X)=700+PEEK(L)*20+X*2:DEFFNP(Y
)=PEEK(Y+PEEK(L)):S=58732:X$="RETURN:"
33 DEFFNA(Q)=PEEK(Q)+PEEK(Q+1)*256:Y$=CH
R$(17):Z$=CHR$(145):RETURN
40 POKEB,A-INT(A/256)*256:POKEB+1,INT(A/
256):RETURN
50 PRINT"* "L$
55 PRINTLEFT$(S$,40)Z$TAB(8)P,W,P-W:RETU
RN
60 POKEE,FNP(H)*2:SYSS:PRINT"*":IFK=0THE
N66
65 POKEE,K*2:SYSS:PRINTA$"*"B$
66 POKEH+PEEK(L),K:RETURN
70 POKET,192:PRINTC$:POKEE,2:PRINT:PRINT
"GOSUB"A":POKE65,PEEK(61):POKE66,";
71 PRINT"PEEK(62):GOTO"Y":PRINTC$";H$:PO
KE631,13:POKE198,1:END
75 W$=STR$(A)+D$+STR$(W+WR)
80 POKET,192:IFPEEK(688)=0THENPRINTC$
81 POKEE,2+PEEK(688):PRINT:PRINTW$:POKET
,4:POKE688,PEEK(688)+1:RETURN
85 GOSUB80
90 POKET,192:POKEE,2+PEEK(688):PRINT:PRI
NT"GOSUB30:GOTO"Y;H$:FORI=0TOPEEK(688)+1
91 POKE631+I,13:NEXT:POKE198,I:POKE688,0
:END
95 POKEE,16:PRINT:PRINTA$Y$WR$" WARTOSC"
;:INPUTWR$:PRINTB$:WR=VAL(WR$):RETURN
96 FORI=1TO2:W$=STR$(A+I)+D$+"0":GOSUB80
:NEXT:GOTO90
100 POKET,4:POKEE,16:PRINT:PRINTY$;LEFT$
(S$,80);
101 GETL$:IFL$=""THEN101
102 ONPEEK(L)+1GOTO106,107,103
103 IFL$="B"THEN800
104 IFL$="L"THENPOKEL,0:GOSUB20:GOTO250
105 GOTO113
106 IFL$="G"THEN200
107 IFL$=Z$THENZ=-1:GOTO300
108 IFL$=Y$THENZ=1:GOTO300
109 IFL$="L"THEN250
110 IFL$="W"THEN550
111 IFL$="D"THEN500
112 IFL$="U"THEN350
113 IFL$="K"THEN150
114 IFL$="Z"THEN650
115 IFL$="P"THEN600
116 GOTO101

149 REM**********************************
149 REM          *** KONIEC ***
150 ONPEEK(L)+1GOTO12,170,155
155 PRINTC$:INPUT"DATA";L$:I=5-LEN(L$):I
FI<0THENI=0
160 L$=LEFT$(L$,5)+LEFT$(S$,I):W$="2000"
+D$+CHR$(34)+L$+CHR$(34):Y=165:GOTO85
165 POKET,4:READL$:PRINTC$"SAVE"CHR$(34)
"BUDZET "L$:END
170 POKEL,0:A=FNA(FNB(FNP(D))):POKEE,20:
PRINT:PRINTLEFT$(S$,40);:GOSUB20:GOTO260

198 REM**********************************
199 REM          *** GRUPA ***
200 IFPEEK(H)=0THEN100
205 A=FNA(FNB(PEEK(H))):B=720:A=A+4:GOSU
B40:A=A-4:Y=210:GOTO70
210 POKET,4:READL$,L$,L$,L$,A
215 POKEL,1:FORI=722TO737:POKEI,0:NEXT
220 B=738:GOSUB40:POKE681,0:POKE683,9:PO
KE685,0:GOSUB20

248 REM**********************************
249 REM          *** LISTA ***
250 A=FNA(FNB(9)):IFA=0THEN100
255 FORI=0TO1:POKEFNB(0)+I,PEEK(FNB(FNP(
G)))
260 PRINTH$A$"*   NAZWA ARTYKULU / GRUPY
 ARTYKULOW    ";
261 PRINT"          PLAN       WYDATKI
 BILANS   "B$;
265 PRINTS$S$S$S$:POKED+PEEK(L),1:K=0:Y=
270:GOTO70
270 POKET,4:B=FNB(K+1):GOSUB40:IFK=8THEN
285
275 READA,W,P,L$:POKEE,2+K*2:SYSS:GOSUB5
0:K=K+1:IFATHEN70
280 B=FNB(9):GOSUB40
285 POKEG+PEEK(L),K:K=1:GOSUB65:GOTO100

298 REM**********************************
299 REM *** POZYCJONOWANIE KURSORA ***
300 IFFNP(D)>FNP(G)THEN100
305 K=FNP(H):IFK=9THENPOKEH+PEEK(L),0
310 IFK=0THEN335
315 IFK+Z<FNP(D)THEN340
320 IFK+Z>FNP(G)THEN100
325 K=K+Z:IFFNA(FNB(K))=0THEN325
330 GOSUB60:GOTO100
335 IFZ<>1THEN100
340 K=FNP(D):GOSUB65:GOTO100

348 REM**********************************
349 REM          *** USUNIECIE ***
350 K=FNP(H):IFK=0ORK=9THEN100
355 POKEE,17:PRINT:PRINTA$Y$"USUNAC T/N"
;:INPUTL$:IFL$="T"THEN370
360 IFL$="N"THEN100
365 POKEE,17:PRINT:PRINTY$;LEFT$(S$,40);
:GOTO355
370 IFPEEK(L)THEN405
375 A=FNA(FNB(PEEK(H))):Y=380:GOTO70
380 READL$,L$,L$,L$,P
385 IFP=0THEN405
```

```
390 A=P:Y=395:GOTO70
395 READA:B=686:GOSUB40:FORI=0TO3:W$=STR
$(P+I):GOSUB80:NEXT:Y=400:GOTO90
400 P=FNA(686):GOTO385
405 A=FNA(FNB(FNP(H))):FORI=0TO4-PEEK(L)
:W$=STR$(A+I):GOSUB80:NEXT:Y=410:GOTO90
410 K=FNP(H):B=FNB(K):POKEB,0:POKEB+1,0:
I=K+1
415 P=FNA(FNB(I)):IFP=0ANDI<>9THENI=I+1:
GOTO415
420 I=K-1
425 A=FNA(FNB(I)):IFA=0ANDI<>0THENI=I-1:
GOTO425
430 W$=STR$(A)+X$+D$+STR$(P):Y=435:GOTO8
5
435 K=FNP(H):IFK<>FNP(G)THEN450
440 POKEG+PEEK(L),FNP(G)-1:IFFNP(G)<FNP(
D)THENPOKEG+PEEK(L),0:GOTO450
445 IFFNA(FNB(FNP(G)))=0THEN440
450 IFK<>FNP(D)THEN465
455 POKED+PEEK(L),FNP(D)+1:IFFNP(G)<FNP(
D)THENPOKED+PEEK(L),9:GOTO465
460 IFFNA(FNB(FNP(D)))=0THEN455
465 POKET,4:POKEE,FNP(H)*2-1:PRINT:PRINT
LEFT$(S$,80);:POKEH+PEEK(L),0:GOTO100

498 REM********************************

499 REM          *** DOPISANIE ***
500 POKEE,18:SYSS:PRINTA$"NAZWA";:INPUTW
R$:PRINTB$
505 RESTORE:READL$,P,L$,L$,W:IFPEEK(L)TH
EN515
510 W$="2001"+X$+D$+STR$(W):GOSUB80:W$=S
TR$(W+4)+X$+D$+"0":GOSUB80:GOTO525
515 A=FNA(FNB(FNP(H))):Y=520:GOTO70
520 READP:W$=STR$(A)+X$+D$+STR$(W):GOSUB
80
525 W$="2004"+D$+STR$(W+5-PEEK(L)):GOSUB
80:W$=STR$(W)+X$+D$+STR$(P):GOSUB80
530 FORI=1TO2:W$=STR$(W+I)+D$+"0":GOSUB8
0:NEXT:W$=STR$(W+3)+D$+WR$:GOSUB80
535 Y=100:GOTO90

548 REM********************************
549 REM          *** WYDATKI ***
550 K=FNP(H):IFK=0ORK=9THEN100
555 WR$="WYDATKI":GOSUB95:IFPEEK(L)=0THE
N570
560 A=FNA(700+PEEK(H)*2):Y=565:GOTO70
565 READL$,W1:A1=A+1
570 A=FNA(FNB(K)):Y=575:GOTO70
575 POKET,4:READL$,W,P:POKEE,K*2:PRINT:P
RINT;:W=W+WR:GOSUB55:A=A+1:W=W-WR
580 GOSUB75:IFPEEK(L)THENW=W1:A=A1:GOSUB
75
585 RESTORE:READL$,L$,W:A=2002:GOSUB75:Y
=100:GOTO90
```


```
499 REM          *** DOPISANIE ***
500 POKEE,18:SYSS:PRINTA$"NAZWA";:INPUTW
R$:PRINTB$
505 RESTORE:READL$,P,L$,L$,W:IFPEEK(L)TH
EN515
510 W$="2001"+X$+D$+STR$(W):GOSUB80:W$=S
TR$(W+4)+X$+D$+"0":GOSUB80:GOTO525
515 A=FNA(FNB(FNP(H))):Y=520:GOTO70
520 READP:W$=STR$(A)+X$+D$+STR$(W):GOSUB
80
525 W$="2004"+D$+STR$(W+5-PEEK(L)):GOSUB
80:W$=STR$(W)+X$+D$+STR$(P):GOSUB80
530 FORI=1TO2:W$=STR$(W+I)+D$+"0":GOSUB8
0:NEXT:W$=STR$(W+3)+D$+WR$:GOSUB80
535 Y=100:GOTO90
```



```
548 REM********************************
549 REM          *** WYDATKI ***
550 K=FNP(H):IFK=0ORK=9THEN100
555 WR$="WYDATKI":GOSUB95:IFPEEK(L)=0THE
N570
560 A=FNA(700+PEEK(H)*2):Y=565:GOTO70
565 READL$,W1:A1=A+1
570 A=FNA(FNB(K)):Y=575:GOTO70
575 POKET,4:READL$,W,P:POKEE,K*2:PRINT:P
RINT;:W=W+WR:GOSUB55:A=A+1:W=W-WR
580 GOSUB75:IFPEEK(L)THENW=W1:A=A1:GOSUB
75
585 RESTORE:READL$,L$,W:A=2002:GOSUB75:Y
=100:GOTO90
```


```
598 REM********************************
599 REM           *** PLAN ***
600 K=FNP(H):IF(K=0ORK=9)ANDPEEK(L)<>2TH
EN100
605 WR$="PLAN":GOSUB95:IFPEEK(L)<>2THEN6
15
610 RESTORE:READL$,L$,L$,W:A=2003:GOSUB7
5:Y=100:GOTO90
615 A=FNA(FNB(K)):Y=620:GOTO70
620 READL$,I,W:A=A+2:GOSUB75:P=W+WR:W=I:
POKEE,K*2:PRINT:PRINT;:GOSUB55
625 Y=100:GOTO90

648 REM********************************
649 REM          *** ZEROWANIE ***
650 K=FNP(H):IF(K=0ORK=9)ANDPEEK(L)<>2TH
EN100
655 POKEE,17:PRINT:PRINTA$Y$"ZEROWAC T/N
";:INPUTL$:IFL$="T"THEN670
660 IFL$="N"THEN100
665 POKEE,17:PRINT:PRINTY$;LEFT$(S$,40);
:GOTO655
670 ONPEEK(L)+1GOTO735,740,675
675 A=2001:Y=680:GOTO96
680 RESTORE:READL$,A
685 IFA=0ORZTHEN745
690 B=740:GOSUB40:Y=695:GOTO96
695 A=FNA(740):Y=700:GOTO70
700 READA,L$,L$,L$,P:IFP=0THEN730
705 A=P:B=742:GOSUB40:Y=710:GOTO96
710 A=FNA(742):Y=715:GOTO70
715 READP:IFPTHEN705
720 A=FNA(740):Y=725:GOTO70
725 READA
730 IFPEEK(L)=0THENZ=1:GOTO685
731 GOTO685
735 Z=0:A=FNA(FNB(PEEK(H))):GOTO685
740 A=FNA(FNB(PEEK(681))):Y=750:GOTO96
745 IFPEEK(L)=2THEN100
750 POKET,4:POKEE,FNP(H)*2:PRINT:PRINTLE
FT$(S$,40)Z$Z$:N=0:P=0:GOSUB55:GOTO100

798 REM********************************
799 REM          *** BUDZET ***
800 RESTORE:READL$,L$,W,P:POKEE,4:PRINT:
PRINTS$S$
805 POKEE,4:PRINT:PRINT:PRINTTAB(10)"PLA
N......"A$P
810 PRINTTAB(10)"WYDATKI..."A$W:PRINTTAB
(10)"BILANS...."A$P-W:GOTO100

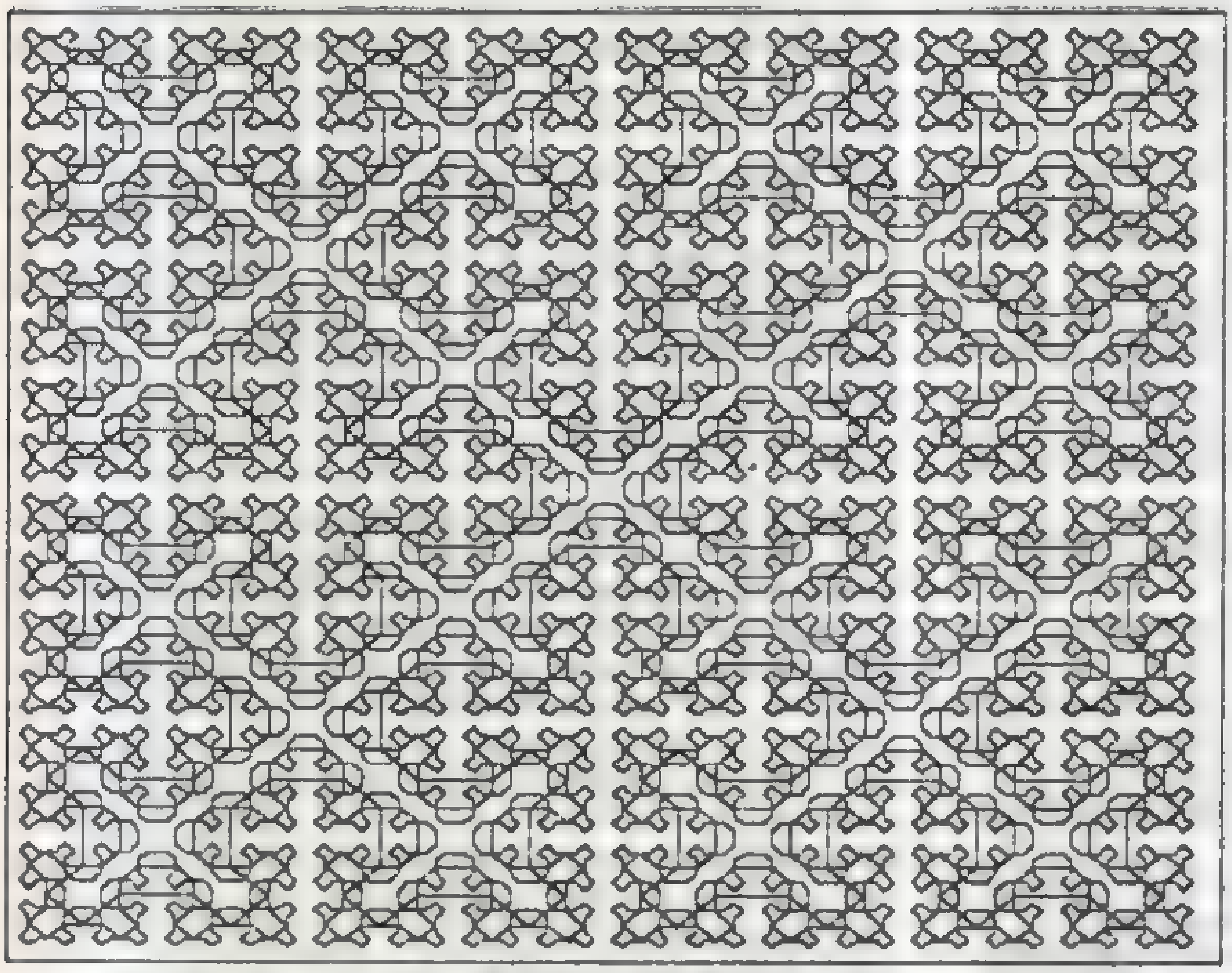
1998 REM*******************************
1999 REM        *** BAZA DANYCH ***
2000 DATA"MM.DD"
2001 RETURN:DATA0
2002 DATA0
2003 DATA0
2004 DATA2005
```

S. WASIELEWSKI

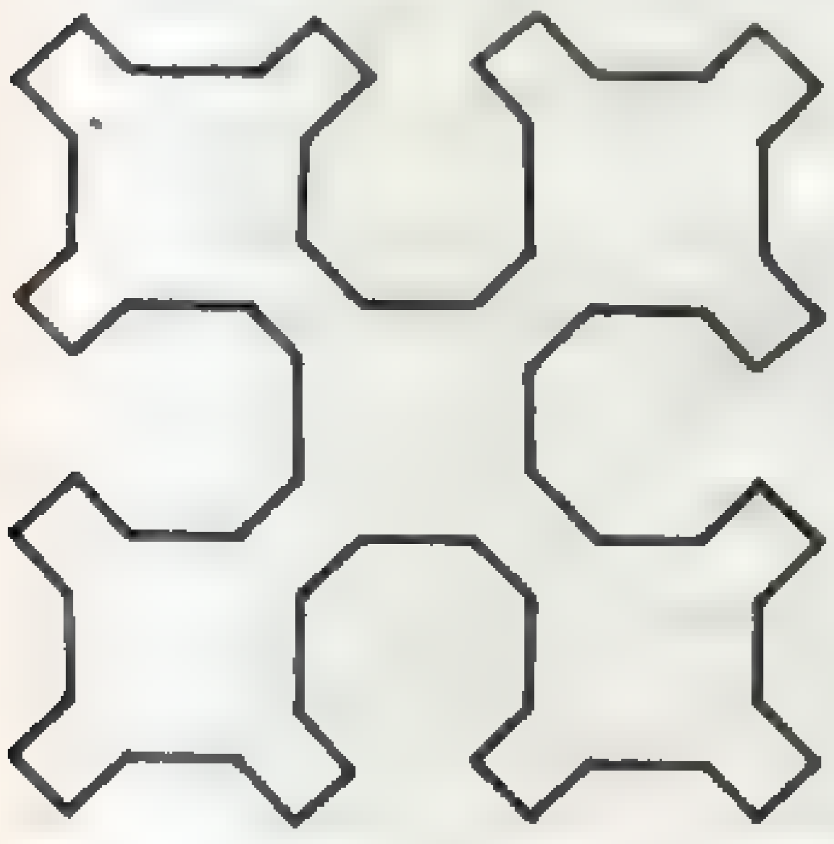
AMSTRAD CPC 6128

Krzywe Sierpińskiego

Efektowne układy graficzne można uzyskać, nakładając na siebie krzywe o regularnych kształtach.

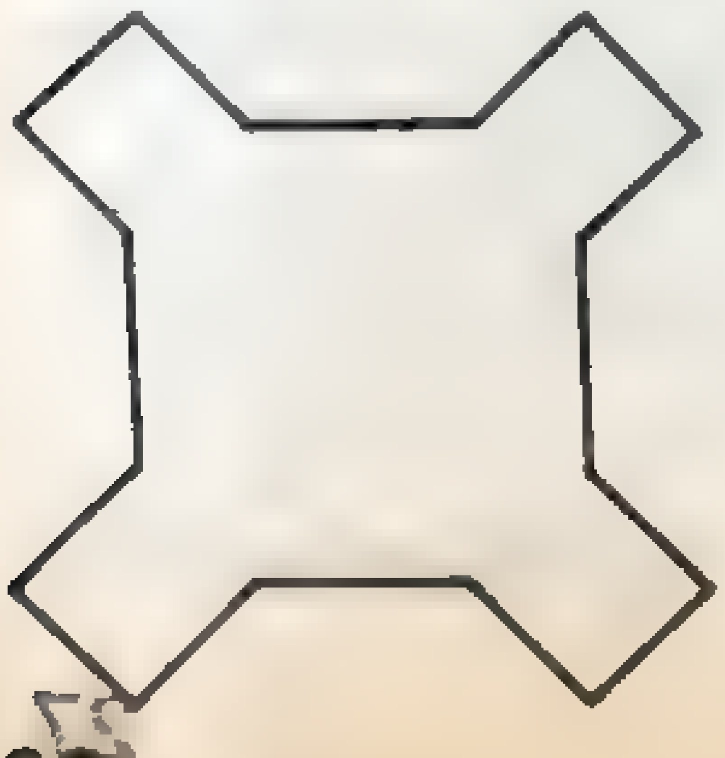


Interesująca jest sytuacja, gdy kształt określonej krzywej K_i (*krzywa rzędu i*) można otrzymać ze złożenia odpowiedniej ilości krzywych K_{i-1} (o rząd niższych).
Na przykład krzywą S_2



S_2

można uzyskać z nałożenia na siebie krzywych S_1.



S_1

Krzywe S_2 i S_1 nazywane są krzywymi Sierpińskiego rzędu S_i. Obraz krzywej, której częścią jest ta sama krzywa, nazywa się obrazem rekurencyjnym. Wykorzystując algorytm rekurencji, zamieszczony w książce Niklaus Wirth „ALGORYTMY +STRUKTURY DANYCH = PROGRAMY", zapisany został w języku PASCAL program KRZYWE SIERPIŃSKIEGO.

```
   {Program kresli krzywe           }
   {Sierpinskiego rzedu 1 - 5       }
{$a-}
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{    STALE I ZMIENNE PROGRAMU       }
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
CONST getmode:byte =2;
      tagflag:boolean =false;
      tagaddr:integer =0;
      h0=256;
 VAR
   i,h,x,y,x0,y0,n:integer;
   z,rz:char;
   o:boolean;
   w:array [1..5] of
            record
             px,py,dl:integer;
            end;

{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{        PROCEDURY GRAFICZNE        }
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}

PROCEDURE initgrafik;
BEGIN
   write(#$1b,'0',#$1b,'y');
END;

PROCEDURE leavegrafik;
BEGIN
   write(#$1b,'1',#$1b,'x');
END;

PROCEDURE gotoxy(x,y:integer);
BEGIN
 write(#$1b,'Y',chr(y+31),chr(x+31));
END;

PROCEDURE sound(g:byte;h,i:integer;j,
                k,l,m:byte);
VAR sd:array[0..8] of byte;
BEGIN
  sd[0] := g;
  move(h,sd[3],2);
  move(i,sd[7],2);
  sd[1] := k;
  sd[2] := l;
  sd[5] := m;
  sd[6] := j;
  inline($21/sd/$cd/$5a/$fc/$aa/$bc);
END;

PROCEDURE drawr(x,y:integer);
BEGIN
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
        $F9/$BB);
END;

PROCEDURE draw(x,y:integer);
BEGIN
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
        $F6/$BB);
END;

PROCEDURE plot(x,y:integer);
BEGIN
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
        $EA/$BB);

END;

PROCEDURE grafwindow
          (x1,y1,x2,y2:integer);
BEGIN
  inline($2A/x2/$EB/$2A/x1/$cd/$5a/
         $fc/$CF/$BB/
         $2A/y2/$EB/$2A/y1/$cd/
         $5a/$fc/$D2/$BB);
END;

PROCEDURE grafpen(fs:byte);
BEGIN
  inline($3A/fs/$cd/$5a/$fc/$DE/$BB);
END;

PROCEDURE grafpaper(fs:byte);
BEGIN
  inline($3A/fs/$cd/$5a/$fc/$E4/$BB);
END;

PROCEDURE clg(fs:byte);
BEGIN
  inline($cd/$5a/$fc/$DB/$BB);
  grafpaper(fs);
END;

PROCEDURE ink(fs,f1,f2:byte);
BEGIN
 inline($3a/f1/$47/$3a/f2/$4f/$3a/fs/
        $cd/$5a/$fc/$32/$bc);
END;

PROCEDURE paper(fs:byte);
BEGIN
  inline($3A/fs/$cd/$5a/$fc/$96/$BB);
END;
```

```
PROCEDURE pen(fs:byte);
BEGIN
  inline($3A/fs/$cd/$5a/$fc/$90/$BB);
END;

PROCEDURE mode(md:byte);
BEGIN
  if md in [0..2] then
  BEGIN
   getmode := md;
   inline($3a/md/$cd/$5a/$fc/$0e/$bc);
  END;
END;

PROCEDURE border(v1,v2:byte);
BEGIN
  inline($3a/v1/$47/$3a/v2/$4f/$cd/
         $5a/$fc/$38/$bc);
END;

PROCEDURE grafout(ch:char);
BEGIN
  inline($3A/ch/$cd/$5a/$fc/$FC/$BB);
END;

PROCEDURE tag;
BEGIN
  if not tagflag then
  BEGIN
    tagaddr := conoutptr;
    tagflag := true;
    conoutptr := addr(grafout);
  END;
END;

PROCEDURE tagoff;
BEGIN
  if tagflag then
  BEGIN
    conoutptr := tagaddr;
    tagflag := false;
  END;
END;


{************************************}
{    PROCEDURY WYPRZEDZAJACE         }
{************************************}
 PROCEDURE b(i:integer); FORWARD;
 PROCEDURE c(i:integer); FORWARD;
 PROCEDURE d(i:integer); FORWARD;


{************************************}
{    PROCEDURY POMOCNICZE            }
{************************************}
PROCEDURE napis;
 VAR nap:string[20];
     j:integer;
 BEGIN
  nap:='KRZYWE SIERPINSKIEGO';
  grafpen(1);
  tag; plot(10,200);
  FOR j:=1 TO 20 DO
   grafout(nap[j]);
  tagoff;
 END;

PROCEDURE ramka;
BEGIN
 plot(w[n].px,w[n].py);
 drawr(w[n].dl,0); drawr(0,w[n].dl);
 drawr(-w[n].dl,0); drawr(0,-w[n].dl);
END;

PROCEDURE rzad;
BEGIN
clg(6); napis;
REPEAT
 clg(6); napis;
 n:=0;
 gotoxy(1,25);
 WRITE('RZAD(1-5)/ESC KONIEC');
 REPEAT
  sound(1,random(250),10,10,5,0,0);
  sound(2,random(200),10,10,10,0,0);
 UNTIL keypressed;
 READ(kbd,rz);
 n:=ORD(rz)-48;
UNTIL ((n>0) and (n<6)) or (rz=#252);
END;


{************************************}
{ PROCEDURY WYZNACZANIA ELEMENTOW    }
{ KRZYWEJ RZEDU i                    }
{************************************}
PROCEDURE a(i:integer);
  BEGIN
   IF i>0 THEN
    BEGIN
     a(i-1);
     x:=x+h; y:=y-h; draw(x,y);
     sound(1,x*5,13,10,0,0,0);
     sound(2,y*10,7,10,10,0,0);
     b(i-1);
     x:=x+2*h; draw(x,y);
     d(i-1);
     x:=x+h; y:=y+h; draw(x,y);
     a(i-1)
    END
  END;

PROCEDURE b;
 BEGIN
  IF i>0 THEN
    BEGIN
     b(i-1);
     x:=x-h; y:=y-h; draw(x,y);
     sound(1,x+20,13,10,0,0,0);
     sound(2,y+35,7,10,0,0,0);
     c(i-1);
     y:=y-2*h; draw(x,y);
     a(i-1);
     x:=x+h; y:=y-h; draw(x,y);
     b(i-1)
    END
  END;

PROCEDURE c;
 BEGIN
   IF i>0 THEN
    BEGIN
     c(i-1);
     x:=x-h; y:=y+h; draw(x,y);
     sound(1,x+50,13,10,0,0,0);
     sound(2,y+75,7,10,0,0,0);
     d(i-1);
     x:=x-2*h; draw(x,y);
     b(i-1);
     x:=x-h; y:=y-h; draw(x,y);
     c(i-1)
    END
  END;

PROCEDURE d;
 BEGIN
   IF i>0 THEN
    BEGIN
     d(i-1);
     x:=x+h; y:=y+h; draw(x,y);
     sound(1,x+60,13,10,0,0,0);
     sound(2,y+90,7,10,0,0,0);
     a(i-1);
     y:=y+2*h; draw(x,y);
     c(i-1);
     x:=x-h; y:=y+h; draw(x,y);
     d(i-1)
    END
  END;

{************************************}
{       PROCEDURY INICJACJI          }
{************************************}
PROCEDURE inic1;
 BEGIN
  mode(0);
  ink(0,1,1);
  ink(1,26,26);
  ink(2,18,18);
  ink(3,6,6);
  ink(4,14,14);
  ink(5,10,10);
  ink(6,0,0);
  grafwindow(0,16,640,400);
  paper(1); pen(3);
  clg(6);
 END;

PROCEDURE inicdanych;
 BEGIN
  inic1;
  w[1].px:=225;
  w[1].py:=125;
  w[1].dl:=210;
  w[2].px:=210;
  w[2].py:=110;
  w[2].dl:=240;
  w[3].px:=202;
  w[3].py:=102;
  w[3].dl:=254;
  w[4].px:=199;
  w[4].py:=98;
  w[4].dl:=262;
  w[5].px:=196;
  w[5].py:=96;
  w[5].dl:=266;
 END;

PROCEDURE pocz;
VAR op:char;
BEGIN
mode(1);
ink(1,26,26);
ink(2,6,6);
paper(1); pen(2);
border(9,9);
REPEAT
 gotoxy(1,10);
 WRITE('      PROGRAM KRZYWE ');
 WRITELN('SIERPINSKIEGO        ');
 WRITELN;
 WRITE('CZY CHCESZ AKCEPTOWAC');
 WRITELN(' -WCISKAJAC DOWOLNY');
 WRITE(' KLAWISZ- POSTAC KRZYWEJ ');
 WRITELN('KOLEJNEGO RZEDU');
 WRITE('(t/n)? ');
 REPEAT
 sound(1,20+random(150),15,15,10,0,0);
 sound(2,30+random(200),12,12,20,0,0);
 UNTIL keypressed;
 READ(kbd,op);
UNTIL (op='t') or (op='T') or
      (op='n') or (op='N');
IF (op='t') or (op='T')
  THEN  o:=true
  ELSE  o:=false;
END;

{************************************}
{        MODUL  STERUJACY            }
{************************************}
BEGIN
 initgrafik;
 pocz;
 inicdanych;
```

```
REPEAT
 rzad; clg(6);
 IF rz<>#252 THEN
  BEGIN
   IF n=5 THEN
    BEGIN
     mode(2);
     grafwindow(0,16,640,400);
     grafpen(1); pen(0);
    END
   ELSE
    grafpen(3);
   ramka;
   i:=0; h:=h0 div 4;
   x0:=2*h+200;
   y0:=3*h+100;
   REPEAT
     i:=i+1; x0:=x0-h; h:=h DIV 2;
     y0:=y0+h;
     x:=x0; y:=y0;
     plot(x,y);
     IF n>4 THEN
      grafpen(1)
   ELSE
    grafpen(i);
    a(i); x:=x+h;
    y:=y-h; draw(x,y);
    sound(1,x*i,13,20,0,0,0);
    sound(2,y*i,7,20,0,0,0);
    b(i); x:=x-h;
    y:=y-h; draw(x,y);
    sound(1,x+5*i,13,20,0,0,0);
    sound(2,y+5*i,7,20,0,0,0);
    c(i); x:=x-h;
    y:=y+h; draw(x,y);
    sound(1,x+10*i,13,20,0,0,0);
    sound(2,y+10*i,7,20,0,0,0);
    d(i); x:=x+h;
    y:=y+h; draw(x,y);
    sound(1,x+15*i,13,20,0,0,0);
    sound(2,y+15*i,7,20,0,0,0);
   IF o THEN
   BEGIN
   gotoxy(1,25);
   WRITE('RZAD',i:2,' WCISNIJ KL.');
    REPEAT
     sound(2,15+i*20,20,15,10,0,0);
     sound(2,100+i*20,15,12,20,0,0);
    UNTIL keypressed;
    READ(kbd,z);

       END;
       UNTIL i=n;
       gotoxy(1,25);
       WRITE('WCISNIJ KLAWISZ     ');
       REPEAT
            sound(1,55,5,30,20,0,0);
            sound(1,200,5,30,20,0,0);

       UNTIL keypressed;
       READ(kbd,z);
      END;
      IF n=5 THEN inicl;
    UNTIL rz=#252;
    mode(2); grafwindow(1,1,1,1);
    paper(0); pen(1);
    gotoxy(1,1);
    leavegrafik
   END.
```

AMSTRAD

Mieczysław SKONIECZNY
Robert ZUGEHOR

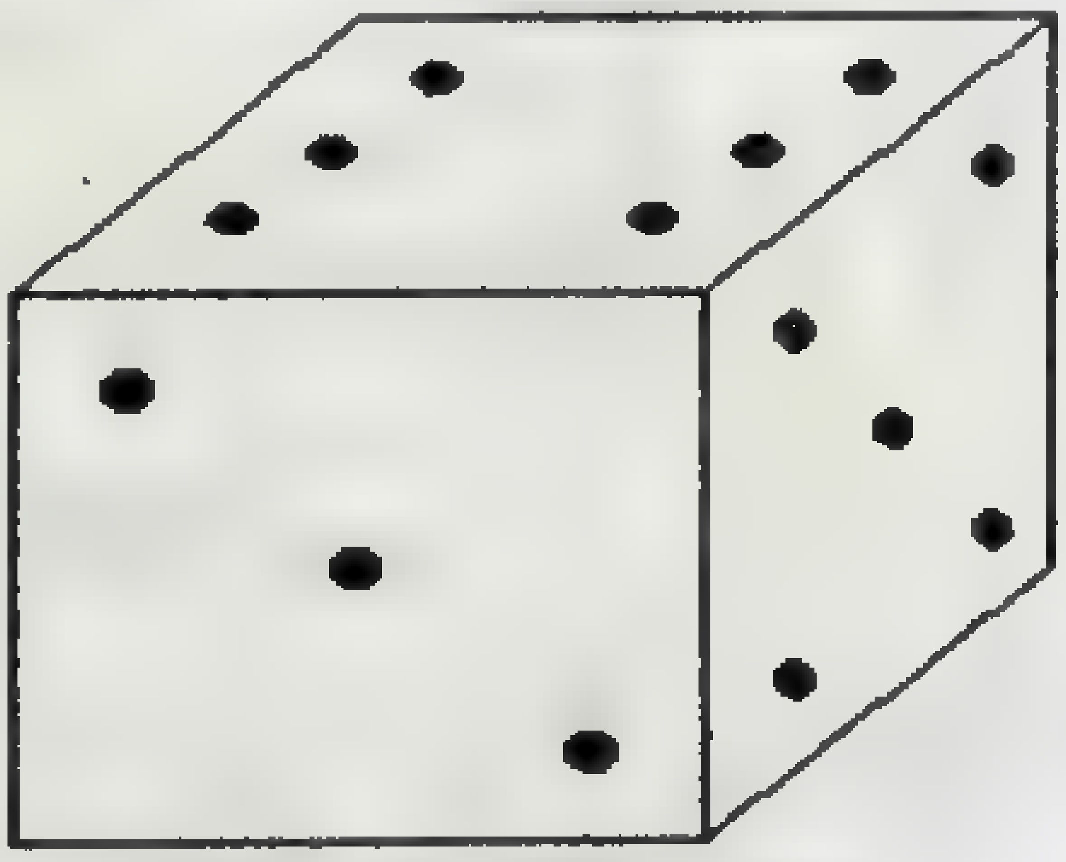
Kostka do gry

Ten program jest przykładem generowania liczb pseudolosowych przy pomocy funkcji RND. Ta instrukcja daje jako wartość wywoływanej funkcji liczbę rzeczywistą z przedziału <0,1). Aby otrzymać liczby całkowite z ustalonego przedziału należy dodatkowo zastosować inną funkcję: INT. Wtedy zmienna całkowita y o wartości.

y=a+INT(RND*b)

należy do przedziału <a,b+a—1>.

W prezentowanym programie wykorzystano w/w własności generując liczbę całkowitą z przedziału <1,6>. Odpowiada ona liczbie wyrzuconych oczek przy jednokrotnym rzucie kostką do gry. Obraz kostki widoczny na ekranie mikrokomputera pokazuje poniższy rysunek



WYLOSOWANO 3

NACISNIJ DOWOLNY KLAWISZ

Po zainicjowaniu pracy programu naciśnięcie dowolnego klawisza powoduje szybką zmianę zawartości 3 widocznych ścianek kostki (odpowiednik toczenia się kostki po rzucie) i ustalenie wylosowanej ilości oczek na ściance czołowej. Użytkownik ma nieograniczoną ilość rzutów

Warto zauważyc, że funkcję RND wykorzystano także do ustalenia ilości prezentowanych sekwencji w czasie „toczenia" — ich liczba wynosi

x=10+INT(RND*20)

a więc jest liczbą całkowitą z przedziału <10,29>. Losowo ustalona wartość zmiennej OCZKO wpływa także na losowe generowanie dźwięku w instrukcji SOUND.

Szczególnie polecamy stosowanie tego programu do wszelkich gier i zabaw, w których niezbędne jest używanie kostki, jak również do wspomagania zajęć z matematyki dotyczących przeprowadzenia doświadczeń losowych z kostką

Niezbędne do poprawnego działania tego programu na mikrokomputerze AMSTRAD CPC 6128 jest wcześniejsze wprowadzenie programu BANKMAN z dyskietki systemowej

```
10 MODE 1:INK 0,2:INK 1,6:INK 2,26:INK 3,24:BORDER 26:PAPER 2:CLS
20 GOSUB 10000
30 CLS:a=150:b=64
35 GRAPHICS PEN 3
40 PLOT a,b
50 DRAWR 224,0:DRAWR 0,214:DRAWR -224,0:DRAWR 0,-214
60 PLOT a,b+214:DRAWR 112,107:DRAWR 224,0:DRAWR 0,-214:DRAWR -112,-107:PLOT a+22
4,b+214:DRAWR 112,107
65 MOVE a+10,b+10:FILL 0:MOVE a+10,b+220:FILL 0:MOVE a+230,b+100:FILL 0
70 |SCREENCOPY,2,1
80 LOCATE 1,25:PEN 0:PRINT"NACISNIJ DOWOLNY KLAWISZ";
90 A$=INKEY$
100 IF A$="" GOTO 90
110 x=10+INT(RND*20)
120 FOR C=1 TO x
130 oczko=1+INT(RND*6)
140 |SCREENCOPY,1,2
155 GRAPHICS PEN 3:PAPER 2
156 TAG
160 ON oczko GOSUB 1000,2000,3000,4000,5000,6000
170 SOUND 1,OCZKO*20,10,15
180 TAGOFF
190 NEXT
199 PEN 1:PAPER 2
200 LOCATE 13,23:PRINT"WYLOSOWANO ";OCZKO;
210 GOTO 80
1000 GOSUB 9000
1010 GOSUB 9150:GOSUB 9170:GOSUB 9190
1020 GOSUB 9320:GOSUB 9340
1999 RETURN
2000 GOSUB 9020
2010 GOSUB 9170:GOSUB 9190
2020 GOSUB 9320:GOSUB 9340:GOSUB 9360
2999 RETURN
3000 GOSUB 9000:GOSUB 9020
3010 GOSUB 9170:GOSUB 9190:GOSUB 9210
3020 GOSUB 9300:GOSUB 9320:GOSUB 9340
3999 RETURN
4000 GOSUB 9020:GOSUB 9060
4010 GOSUB 9170
4020 GOSUB 9300
4999 RETURN
5000 GOSUB 9000:GOSUB 9020:GOSUB 9060
5010 GOSUB 9150
5020 GOSUB 9300:GOSUB 9320
5999 RETURN
6000 GOSUB 9020:GOSUB 9060:GOSUB 9120
6010 GOSUB 9150:GOSUB 9170
6020 GOSUB 9320
6999 RETURN
9000 MOVE a+104,b+115:PRINT CHR$(241);
9010 RETURN
9020 MOVE a+30,b+185:PRINT CHR$(241);:MOVE a+180,b+43:PRINT CHR$(241);
9030 RETURN
9060 MOVE a+30,b+43:PRINT CHR$(241);:MOVE a+180,b+185:PRINT CHR$(241);
9070 RETURN
9120 MOVE a+30,b+115:PRINT CHR$(241);:MOVE a+180,b+115:PRINT CHR$(241);
9140 RETURN
9150 MOVE a+160,b+278:PRINT CHR$(243);
9160 RETURN
9170 MOVE a+130,b+305:PRINT CHR$(243);:MOVE a+200,b+250:PRINT CHR$(243);
9180 RETURN
9190 MOVE a+64,b+250:PRINT CHR$(243);:MOVE a+270,b+305:PRINT CHR$(243);
9200 RETURN
9210 MOVE a+97,b+276:PRINT CHR$(243);:MOVE a+234,b+276:PRINT CHR$(243);
9220 RETURN
9300 MOVE a+278,b+168:PRINT CHR$(242);
9310 RETURN
9320 MOVE a+246,b+206:PRINT CHR$(242);:MOVE a+310,b+128:PRINT CHR$(242);
9330 RETURN
9340 MOVE a+246,b+70:PRINT CHR$(242);:MOVE a+310,b+270:PRINT CHR$(242);
9350 RETURN
9360 MOVE a+246,b+140:PRINT CHR$(242);:MOVE a+310,b+200:PRINT CHR$(242);
9370 RETURN
10000 SYMBOL AFTER 240
10010 SYMBOL 241,60,126,255,255,255,255,126,60
10020 SYMBOL 242,24,60,126,126,126,126,60,24
10030 SYMBOL 243,0,60,126,255,255,126,60
10040 RETURN
```

DATA BANK - bank danych, baza danych,
DATA BASE - baza danych,
DATABASE - patrz: DATA BASE,
DATA BASE ADMINISTRATOR - administrator bazy danych,
DATA BASE MANAGEMENT SYSTEM - system zarządzania bazą danych,
DATA BIT - bit informacyjny,
DATA BLOCK - blok danych,
DATA BUS - szyna danych,
DATA BUS ENABLE - blokada szyny danych,
DATA CAPTURE - porcja danych,
DATA CARRIER - nośnik danych,
DATA CHAINING - wiązania łańcuchowe danych,
DATA CHANNEL - kanał transmisji danych, kanał informacyjny,
DATA-COLLECTING UNIT - urządzenie gromadzenia danych,
DATA COLLECTION - gromadzenie danych, zbieranie danych,
DATA COMPATIBILITY - wymienność danych,
DATA COMPRESSION - zagęszczanie danych,
DATA CONSISTENCY - zgodność (jednolitość) danych,
DATA DENSITY - gęstość zapisu danych,
DATA DESCRIPTION LANGUAGE (DDL) - język opisu danych,
DATA DISPLAY FORMAT - postać danych na ekranie monitora,
DATA DIVISION - część danych,
DATA DRIVEN - także: wnioskowanie uprzedzające (w systemach ekspertowych),
DATA ENCRYPTION STANDARD - system utajniania danych,
DATA EXCHANGE CONTROL UNIT - jednostka sterująca wymianą danych, jednostka wymiany danych,
DATA EXCHANGE CONTROLLER - jednostka sterująca wymianą danych, jednostka wymiany danych,
DATA FLOW - przepływ danych,
DATA FLOW CHART - schemat blokowy, schemat działania systemu,
DATA FLOW DIAGRAM - patrz: DATA FLOW CHART,
DATA FORMAT - układ danych, struktura danych,
DATA GATHERING - rejestracja danych,
DATA-GATHERING UNIT - urządzenie gromadzenia danych,
DATA GROUP - grupa danych,
DATA HANDLING - obróbka danych,
DATA HOLE - dziurka informacyjna (taśmy, karty),
DATA HOLE TRACK - ścieżka informacyjna (taśmy dziurkowanej),
DATA-IN - dane wejściowe,
DATA INPUT - wprowadzanie danych,
DATA INTERFACE UNIT - złącze transmisji danych, sprzęg transmisji danych, interfejs transmisji danych,
DATA INTERLOCK - blokada dostępu do danych,
DATA ITEMS - elementy danych,
DATA KEYS - klawisze danych,
DATA LINE - przewód danych,
DATA LINK - kanał przesyłający dane, kanał informacyjny,
DATA LINK CONTROL - sterowanie łączem na poziomie danych,
DATA LOGGER - rejestrator danych,
DATA MANAGEMENT - kierowaniem przepływem danych, zarządzanie danymi,
DATA MANIPULATION - operowanie danymi,
DATA MANIPULATION LANGUAGE (DML) - język manipulowania danymi, DML,
DATA MEDIUM - nośnik danych,
DATA NAME - nazwa danych,
DATA ORDERING - porządkowanie danych,
DATA ORGANIZATION - organizacja danych,
DATA-OUT - dane wyjściowe,
DATA OUTPUT - wyprowadzanie danych,
DATA PATH - ścieżka danych,
DATA PLATE - tabliczka znamionowa,
DATA POINTER (DP) - wskaźnik danych,
DATA PREPARATION - przygotowywanie danych,
DATA PROCESSING - przetwarzanie danych,
DATA PROCESSING CENTRE - ośrodek przetwarzania danych,
DATA PROCESSING COMPUTER - komputer do przetwarzania danych,
DATA PROCESSING ENVIRONMENT - środowisko przetwarzania danych,
DATA PROCESSING SYSTEM - system do przetwarzania danych,
DATA PROCESSING TERMINAL - urządzenie końcowe, terminal, końcówka,
DATA PROCESSOR - procesor, jednostka centralna,
DATA PROTECTION - zabezpieczanie danych,
DATA PURFICATION - usuwanie błędnych elementów ze zbioru danych,
DATA RATE - szybkość transmisji (przesyłania) danych,
DATA RECORD - zapis (rekord) danych,
DATA REDUCTION - kumulowanie danych,
DATA RETRIEVAL - wyszukiwanie danych,
DATA SEARCH - przeszukiwanie danych,
DATA SEARCH DEVICE - urządzenie wyszukiwania danych,
DATA SEGMENT (DS) - segment danych,
DATA SET - zbiór danych, także: urządzenie do transmisji danych,
DATA SINK - ujście danych,
DATA STATION - stacja abonencka, stacja teledacyjna,
DATA STORAGE - pamięć danych,
DATA SYSTEMS ENGINEERING - środki techniczne przetwarzania danych,
DATA TERMINAL EQUIPMENT (DTE) - urządzenie końcowe transmisji danych,
DATA TRANSFER - przenoszenie danych,
DATA TRANSMISSION - transmisja danych,
DATA TRANSMISSION DEVICE - urządzenie transmisji danych (UTD),
DATA TRANSMISSION FACILITIES - urządzenia transmisji danych,
DATA TRANSMISSION LINE - łącze transmisji danych, łącze przesyłowe,
DATA TRANSMISSION PROGRAM - program transmisji danych,
DATA TRANSMISSION RATE - szybkość transmisji, szybkość przesyłania (danych),
DATA TRANSMISSION RELIABILITY - niezawodność transmisji, niezawodność przesyłania,
DATA TRANSMISSION SPEED - patrz: DATA TRANSMISSION RATE,
DATA TRANSMITTING CHANNEL - kanał transmisji danych,
DATA UPDATING - aktualizacja danych,
DATA WORD - słowo danych,
DATE - data,
DATEL SERVICE - usługi transmisji danych, usługi teledacyjne,
DATER - datownik,
DATUM - punkt wyjściowy (początkowy), także: dana wielkość, wielkość założona, element danych,
DATUM LEVEL - poziom odniesienia,
DATUM POINT - punkt odniesienia,
DBMS - patrz: DATA BASE MANAGEMENT SYSTEM,
DDL - patrz: DATA DESCRIPTION LANGUAGE,
DE-ALLOCATION - wymazywanie (zbioru, obszaru zbioru),
DEAD END - "ślepa uliczka" (na schemacie blokowym),
DEADLINE DATE - termin prekluzyjny, nieprzekraczalny,
DEADLOCK - blokada systemu,
DEALLOCATION MEMORY - zwolnienie miejsca pamięci,
DEBLOCKING - odblokowanie,
DEBUG - usuwanie błędów z programu, uruchamianie,
DEBUG DECK - pakiet testowy,
DEBUGGING - "odpluskwianie", proces usuwania błędów z programu,
DEBUG TIME - czas związany z wyszukiwaniem i usuwaniem błędów w programie,
DECAY - rozpad, zanik,
DECELERATE - zwalniać, zmniejszać np. prędkość ruchu taśmy magnetycznej,
DECIDE - decydować,
DECIMAL - cyfra dziesiętna, także: ułamek dziesiętny,
DECIMAL CARRY - przeniesienie dziesiętne,
DECIMAL CLASSIFICATION - klasyfikacja dziesiętna,
DECIMAL-CODED DIGIT - cyfra dziesiętna,
DECIMAL COMPUTER - komputer dziesiętny, maszyna dziesiętna,
DECIMAL COUNTER - licznik dziesiętny,
DECIMAL DIGIT - dit (dziesiątkowa jednostka informacji), cyfra dziesiętna,
DECIMAL FRACTION - ułamek dziesiętny,
DECIMAL NOTATION - zapis dziesiętny,
DECIMAL NUMBER - liczba dziesiętna,
DECIMAL NUMERATION SYSTEM - dziesiętny system liczbowy,
DECIMAL PLACE - miejsce dziesiętne, przecinek dziesiętny,
DECIMAL POINT - patrz: DECIMAL PLACE,
DECIMAL SYSTEM - system dziesiętny, układ dziesiętny,
DECIMAL-TO-BINARY CONVERSION - konwersja dziesiętno-dwójkowa,
DECIMALIZATION - przejście na system dziesiętny,
DECIPHER - rozszyfrowywać,
DECISION - decyzja,
DECISION BOX - figura na schemacie blokowym symbolizująca warunek,
DECISION ELEMENT - element decyzyjny,
DECISION FUNCTION - funkcja decyzyjna,
DECISION INSTRUCTION - rozkaz decyzji,
DECISION-MAKING - decydowanie,
DECISION TABLE - tablica decyzyjna, tablica proceduralna,
DECISION THEORY - teoria podejmowania decyzji, teoria decydowania,
DECISION TREE - dendryt decyzyjny,
DECIT - dit (dziesiątkowa jednostka informacji),
DECK OF CARDS - plik kart dziurkowanych,
DECLARATION - zgłoszenie, deklaracja,
DECLARATIVE STATEMENT - deklaracja procedury,

DECODE – dekodować,
DECODER – dekoder,
DECODER MATRIX – tablica dekodowania,
DECODER NETWORK – patrz: DECODER MATRIX,
DECODIFICATION – patrz: DECODING,
DECODING – dekodowanie,
DECODING CIRCUIT – układ dekodujący,
DECOLLATOR – urządzenie do rozkładania (rozdzielania) druków wielowarstwowych,
DECOMPOSE – rozkładać (np. rozkaz na jego elementy składowe), dekomponować,
DECREASING FUNCTION – funkcja malejąca,
DECREMENT – zmniejszać, także: wielkość o jaką maleje zmienna,
DEDICATED SYSTEM – system (informatyczny) dedykowany,
DEDUCE – dedukować, wnioskować, wyprowadzać,
DEEPEN – pogłębiać,
DEFAULT – wartość standardowa, wartość zastępcza, wartość domyślna,
DEFAULT DEVICE – urządzenie standardowe, urządzenie domyślne,
DEFAULT VALUE – wartość domyślna,
DEFAUT – zaniedbanie,
DEFECT – błąd (np. kodowania), defekt, wada,
DEFERRED ENTRY – wejście asynchroniczne,
DEFINE – definiować, określać,
DEFINED NOTION – pojęcie zdefiniowane,
DEFINITION – definicja,
DEFLECTION – odchylenie, wychylenie (np. strzałki przyrządu),
DEFORMITY – zniekształcenie (np. karty),
DEGREE – stopień,
DEGREE OF ACCURACY – stopień dokładności,
DEGREE OF CONVERGENCE – stopień zbieżności,
DEGREE OF CORELATION – stopień korelacji,
DEL – patrz: DELETE CHARACTER,
DELAY – opóźnienie, zwłoka,
DELAY LINE – element opóźniający,
DELETE – anulować, kasować, usuwać, wymazywać,
DELETE CHARACTER – znak wymazujący, kasujący,
DELETING A RECORD – wymazywanie (kasowanie) zapisu (rekordu)
DELIBERATELY – świadomie,
DELIMIT – oddzielać, separować, rozgraniczać,
DELIMITER – separator, ogranicznik,
DELIMITER CARD – karta rozgraniczająca,
DELINEATE – narysować, naszkicować,
DELVE – pogłębiać, zgłębiać (np. problem, zagadnienie),
DEMAND – żądanie, wymaganie, także: popyt,
DEMAND (DATA) PROCESSING – przetwarzanie danych swobodne,
DEMAND FETCHING – sprowadzanie na żądanie (sposób przemiennego wykorzystania pamięci),
DEMODULATION – demodulacja,
DEMODULATOR – demodulator,
DEMOUNTABLE – wymienny, możliwy do demontażu,
DENOMINATE NUMBER – liczba mianowana,
DENOMINATOR – mianownik,
DENOTATION – oznaczenie,
DENOTE – określać, oznaczać (np. wymiary urządzenia),
DENSITY – gęstość, zagęszczenie, upakowanie,
DENUMERABLE SET – zbiór przeliczalny,
DEPARTMENT – dział, wydział, departament,
DEPENDABILITY – niezawodność, pewność (działania),
DEPENDABLE – niezawodny,
DEPENDENCE – zależność,
DEPENDING ON ... – w zależności od ...,
DEPICT – opisywać coś, przedstawiać na rysunku,
DEPLOY – rozmieszczać (np. dane w pamięci maszyny),
DEPOSIT – odkładać, wkładać, także: przenosić (np. informacje z pamięci operacyjnej do zewnętrznej),
DEPRESS – naciskać, uruchamiać (np. klawisz-przycisk maszyny),
DEPTH QUEUING – obrazowanie głębi (tworzenie perspektywy),
DERIVE – wyprowadzać (np. wynik, wzór, zależność),
DES – patrz: DATA ENCRYPTION STANDARD,
DESCENDING – malejący (porządek), malejąco,
DESCENDING ORDER – porządek malejący,
DESCRAMBLER – deszyfrator,
DESCRIBE – opisywać,
DESCRIPTION – opis, deskrypt,
DESCRIPTIVE – opisowy,
DESCRIPTOR – deskryptor,
DESCRIPTOR PRIVILEGE LEVEL – deskrypcyjny poziom uprzywilejowania,
DESIGN – koncepcja, zamiar, projekt, schemat, także: konstruować, projektować, zamierzać,
DESIGN AUTOMATION – automatyzacja projektowania,
DESIGNATE – oznaczać, przedstawiać, wyznaczać,
DESIGNATOR – oznacznik,
DESK COMPUTER – komputer biurkowy, komputer przenośny,
DESKTOP COMPUTER – komputer stołowy,
DESTINATION INDEX REGISTER – rejestr indeksowy odbiorcy,
DESTINATION VARIABLE – zmienna przeznaczenia,
DESTROY – anulować, zniszczyć,
DESTRUCTIVE READ – odczyt niszczący, odczyt wymazujący,
DESTRUCTIVE READ-OUT STORE – pamięć o odczycie niszczącym, pamięć destrukcyjna,
DETACH – odejmować, odłączać,
DETACHABLE – odejmowany, odłączany,
DETAIL – drobiazg, szczegół, także: szczegółowy, wyszczególniać,
DETAIL BIT – bit szczegółowy,
DETAILED – szczegółowy,
DETECT – wykrywać,
DETECTABILITY – wybieralność (cecha elementu obrazu), wykrywalność,
DETECTABLE – wykrywalny,
DETECTABLE ELEMENT – element wykrywalny (np. przez pióro świetlne),
DETECTION – wykrywanie, wynajdywanie,
DETERIORATION – uszkodzenie,
DETERMINANT – wyznacznik,
DETERMINE – określać, ustalać, wyznaczać,
DETERMINE A RELATIONSHIP – określić zależność,
DETERMINE CONFORMITY – ustalać zgodność,
DEVELOP – budować, konstruować, także: opracowywać, przekształcać,
DEVELOP A METHOD – opracować metodę,
DEVELOPMENT – rozwój, rozbudowa, rozwinięcie, także: przekształcenie, adaptacja,
DEVELOPMENT SYSTEM – system wspomagania, także: system uruchomieniowy,
DEVELOPMENT TIME – czas usunięcia błędów (np. w programie lub sprzęcie),
DEVIATE FROM – odbiegać od, odchylać się,
DEVIATION – odchylenie, także: błąd,
DEVICE – przyrząd, urządzenie,
DEVICE ALLOCATION – przydzielanie urządzeń, przyporządkowywanie urządzeń,
DEVICE CONTROL CHARACTER – znak sterujący urządzeniem,
DEVICE COORDINATES – układ współrzędnych urządzenia,
DEVICE DEPENDENCE – uzależnienie od jednostki peryferyjnej,
DEVICE DRIVER – program obsługi urządzenia,
DEVICE HANDLER – (standardowy) program obsługi urządzeń,
DEVICE SPACE – przestrzeń urządzenia graficznego, (obszar ograniczony pojemnością rejestrów pozycji X, Y),
DEVICE SPECIFIER – specyfikator urządzenia,
DEVICE STATUS – stan urządzenia,
DFA – patrz: DIGITAL FAULT ANALYSIS,
DI – patrz: DIGITAL INPUT,
DI (DESTINATION INDEX) – rejestr indeksowy odbiorcy,
DIAGNOSTIC – diagnostyczny, także: samoczynnie wysyłany przez komputer komunikat w celu zasygnalizowania i identyfikacji błędu,
DIAGNOSTIC CHECK – sprawdzenie sprzętu lub programu,
DIAGNOSTIC PROGRAM – program diagnostyczny,
DIAGNOSTICS ON-CHIP – układy diagnostyczne zawarte w układach scalonych,
DIAGONNALLY – ukośnie,
DIAGRAM – rysunek, schemat, wykres,
DIAGRAM CHECK – kontrola schematu,
DIAGRAMMATIC – wykreślny, schematyczny,
DIAGRAMMATICALLY – schematycznie,
DIAGRAM PAPER – papier milimetrowy,
DIAGRAM SYMBOL – symbol schematu,
DIAL COMMUNICATION LINE – łącze komutowane, łącze abonenckie,
DIC – patrz: DIGITAL INTEGRATED CIRCUIT,
DICHOTOMIZING SEARCH – wyszukiwanie dwudzielne, wyszukiwanie dychotomiczne,
DICTIONARY – słownik,
DID – patrz: DIGITAL INFORMATION DISPLAY,
DIE SORTING – sortowanie struktur półprzewodnikowych,
DIFFERENCE – różnica (np. przy odejmowaniu),
DIFFERENT LEVELS OF SOPHISTICATION – różne stopnie utrudnienia,
DIFFICULTY – trudność,
DIGIT – cyfra,
DIGITAL – cyfrowy,
DIGITAL ADDER – sumator,
DIGITAL CAPACITY – długość liczby,
DIGITAL-CIRCUIT ENGINEERING – patrz: DIGITAL ELECTRONICS,
DIGITAL COMPARATOR – komparator cyfrowy,
DIGITAL COMPUTER – komputer cyfrowy,
DIGITAL CONTROL – sterowanie cyfrowe,
DIGITAL CONVERTER – konwerter cyfrowy,
DIGITAL DATA – dane cyfrowe,
DIGITAL DESIGN – projektowanie układów logicznych,
DIGITAL DISPLAY – przedstawienie cyfrowe,
DIGITAL ELECTRONICS – technika cyfrowa, cyfronika,
DIGITAL FAULT ANALYSIS – cyfrowa analiza uszkodzeń,
DIGITAL INDICATOR – wskaźnik cyfrowy,

GIEŁDA POMYSŁÓW

Kredka

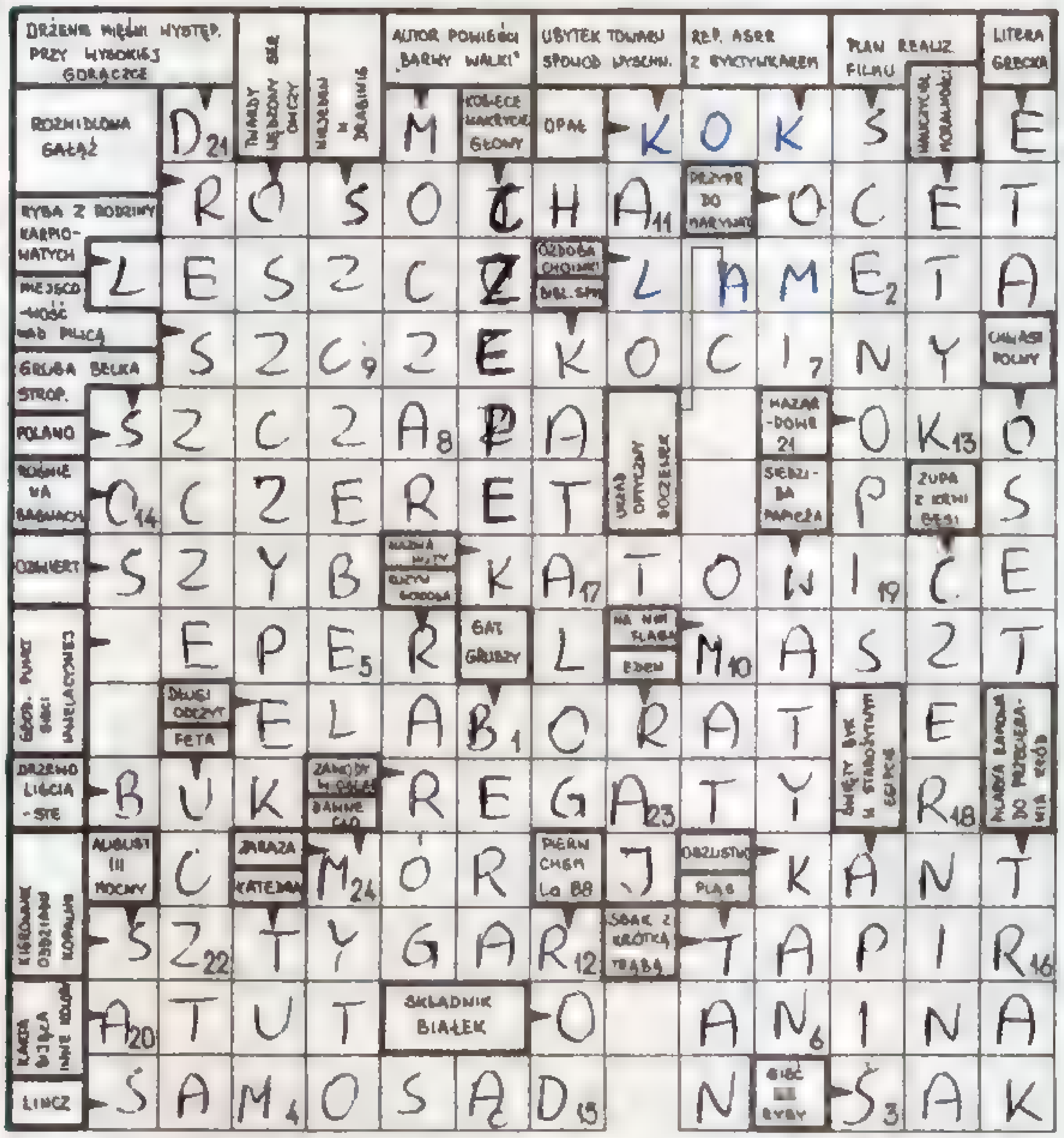
Jest to program graficzny, który może służyć do robienia rysunków, wykresów w zależności od umiejętności i cierpliwości użytkownika. Cierpliwości, bo poza jego powolnością, ma on i tę wadę, że jest czarno-biały. Poza tym sądzę, że ma resztę zalet, do których należy prostota w obsłudze. Rysowanie instrukcją PLOT odbywa się za pomocą kursorów (5,6,7,8) przy opuszczonej kredce lub odpowiednio R, T, U i Y przy kredce podniesionej (która jest w tym przypadku gumką). Ciągły odczyt współrzędnych x i y. Literą „S" można dokonać nagrania na taśmie, po zakończeniu którego obraz jest wczytywany w inny obszar pamięci komputera w celu zabezpieczenia go przed uszkodzeniem w przypadku wadliwego nagrania. W celu ponownego nagrania można go wywołać przez RANDOMIZE USR 30000. Program ma możliwość rysowania okręgów w wariancie „0 — normal", lub usuwania w wariancie „I"—inverse", albo rysowania białego okręgu na czarnym tle.

Promień podany jest w punktach tzw. pikselach. Po uruchomieniu programu mamy możliwość wgrania niedokończonego rysunku lub rozpoczęcia rysowania od początku. W obu przypadkach przed rysowaniem należy tylko podać współrzędne x i y, w których punkt ma się ukazać.

Zygmunt WIŚNIEWSKI

```
  1 CLEAR 29999: GO TO 12
  2 FOR f=16384 TO 23296: POKE 13829+f,PEEK f: NEXT f
  3 FOR n=30000 TO 30011
  4 READ d: POKE n,d
  5 NEXT n
  6 DATA 17,0,64,33,60,117,1,0,27,237,176,201
  7 CLS
  8 PRINT PAPER 6;AT 0,0;"Rysunek zostal nagrany": PRINT AT 2,0;"1-wykonaj "NEW"";AT 3,0;"2-wczytaj nagranie przez ";AT 4,2;"LOAD ""SCREEN$ "
  9 PRINT AT 8,0;"  W przypadku zle dokonanego ";AT 9,0;"nagrania nalezy wywolac rysunek";AT 10,0;"przez "RANDOMIZE USR 30000"";AT 11,0;"wczytac na tasme przez ";AT 12,0;"SAVE "nazwa"SCREEN$ ";AT 13,0;"i dalej jak w punkcie 1 i 2 "
 10 PRINT AT 16,0;"  Wywolywac rysunek mozna wie-";AT 17,0;"lokrotnie"
 11 STOP
 12 PRINT AT 8,14;"L-Load";AT 12,14;"D-Draw"; BRIGHT 1;AT 8,14;"L";AT 12,14;"D"
 15 PRINT PAPER 6;AT 21,2;"Nacisnij odpowiedni klawisz"
 16 IF INKEY$="l" THEN CLS : LOAD ""SCREEN$ : GO TO 35
 17 IF INKEY$="d" THEN CLS : GO TO 20
 18 GO TO 15
 20 PRINT BRIGHT 1; INVERSE 1;AT 0,5;"Klawisze do sterowania"
 21 PRINT AT 2,3;"kredka";AT 2,23;"kredka"
 22 PRINT AT 3,1;"opuszczona";AT 3,20;"podniesiona"
 23 PRINT BRIGHT 1;AT 6,5;"5";AT 8,5;"6";AT 10,5;"7";AT 12,5;"8";AT 6,25;"R";AT 8,25;"T";AT 10,25;"U";AT 12,25;"I"
 24 PRINT PAPER 6;AT 21,3;"Nacisnij dowolny klawisz"
 25 PAUSE 0: CLS
 35 INPUT "x=";x
 40 IF x<0 OR x>255 THEN GO TO 35
 45 INPUT "y=";y
 50 IF y<0 OR y>175 THEN GO TO 45
 55 PRINT #0;AT 1,0;"x=";x;AT 1,7;"y=";y; BRIGHT 1;AT 1,15;"S";AT 1,23;"C"; BRIGHT 0;AT 1,16;"-Save";AT 1,24;"-Circle"
 60 PLOT x,y
 65 PAUSE 0
 70 PRINT #0;AT 1,2;"    ";AT 1,9;"   "
 80 IF INKEY$="5" THEN LET x=x-1
 90 IF INKEY$="8" THEN LET x=x+1
100 IF INKEY$="6" THEN LET y=y-1
110 IF INKEY$="7" THEN LET y=y+1
120 IF x<0 THEN LET x=0
130 IF x>255 THEN LET x=255
140 IF y<0 THEN LET y=0
150 IF y>175 THEN LET y=175
160 IF INKEY$="r" THEN GO SUB 250
170 IF INKEY$="t" THEN GO SUB 350
180 IF INKEY$="u" THEN GO SUB 450
190 IF INKEY$="i" THEN GO SUB 550
200 IF INKEY$="s" THEN GO TO 600
210 IF INKEY$="c" THEN GO SUB 700
220 GO TO 55
250 LET x=x-1
255 IF x<0 THEN LET x=0
260 PLOT x,y: PLOT INVERSE 1;x+1,y
265 RETURN
350 LET y=y-1
355 IF y<0 THEN LET y=0
360 PLOT x,y: PLOT INVERSE 1;x,y+1
365 RETURN
450 LET y=y+1
455 IF y>175 THEN LET y=175
460 PLOT x,y: PLOT INVERSE 1;x,y-1
465 RETURN
550 LET x=x+1
555 IF x>255 THEN LET x=255
560 PLOT x,y: PLOT INVERSE 1;x-1,y
565 RETURN
600 INPUT "nazwa nagrania ";" ";a$
610 SAVE a$SCREEN$
620 PRINT #0; FLASH 1;AT 1,0;"WPISYWANIE RYSUNKU ZA RAMTOP"
630 GO TO 2
700 INPUT "circle : 1-inverse 0-normal ";c
701 IF c>1 THEN GO TO 700
702 INPUT "promien = ";p
710 IF p>175-y OR y-p<0 THEN GO TO 702
720 IF p>255-x OR x-p<0 THEN GO TO 702
740 CIRCLE INVERSE c;x,y,p
750 RETURN
```

KRZYŻÓWKA NR 3



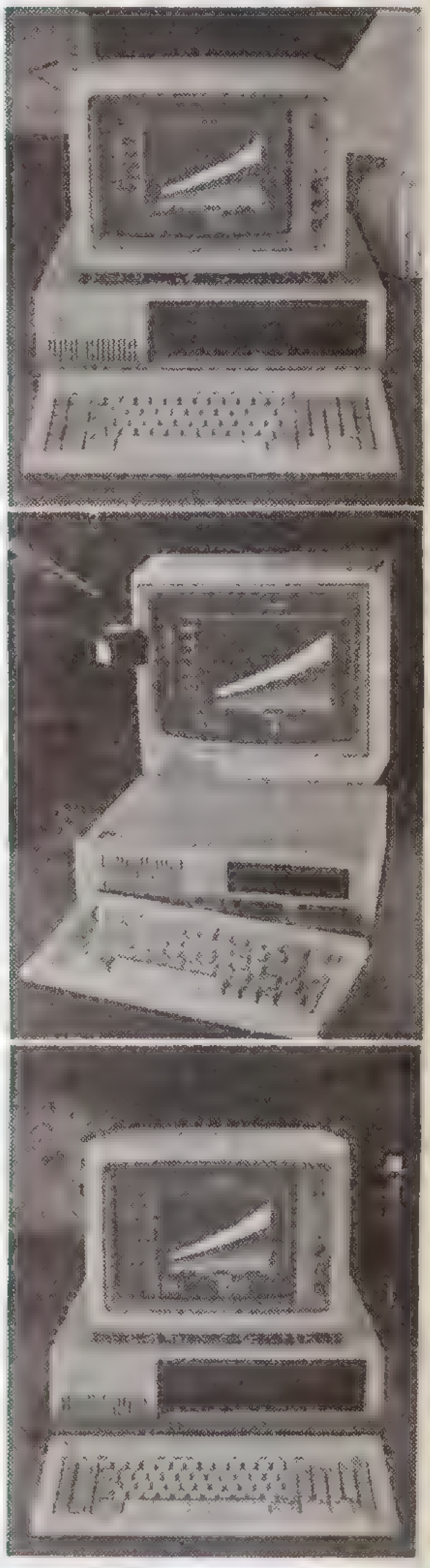
Litery z ponumerowanych kratek od 1 do 24 utworzą nazwy 6 komputerów, które wystarczy podać jako rozwiązanie zadania, przesyłając pod adresem redakcji na kartach pocztowych do końca kwietnia, naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

KUPON ▶



MIKROCIEKAWOSTKI

IBM Personal System 2



„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1144. Nr. ind. 361682.

U-41

IKS
INFORMATYKA KOMPUTERY SYSTEMY



CENA 100 zł

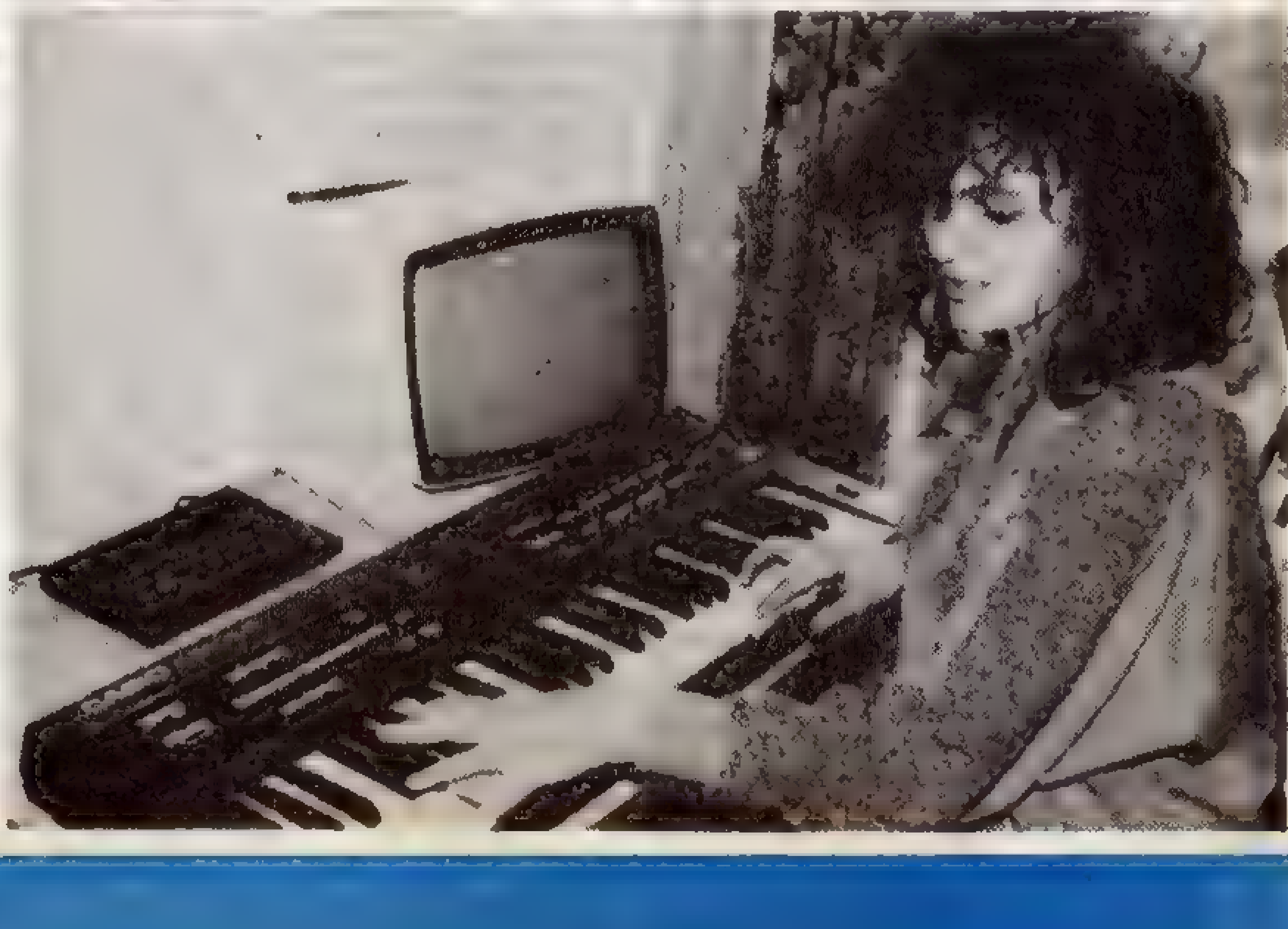
DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 4/1988 ISSN 0860-2794

Specjalne programy graficzne umożliwiają tworzenie dźwięków na ekranie monitora, wygodną edycję parametrów itp. Komputer przetwarza pomysły muzyka na odpowiedni ciąg danych i przesyła do syntezatora. W tym miejscu powstaje bardzo ważny problem: w jaki sposób komputer ma nawiązać kontakt z syntezatorem? Odpowiedź w numerze: Nuty z komputera — str. 16

Liczby i tabele na wykresach — czternasty odcinek „W szponach Atari" tym razem poświęcony grafice, a w nim także 11 krótkich programów — str. 5

UWAGA! Kolejna edycja „Ligi Myślących" w tym numerze wyjątkowo na stronie 25.





Informatyczny krajobraz naszego kraju mieni się wszystkimi barwami. Ta dziedzina, jak żadna inna tętni dziś życiem, a to głównie dlatego, że szybko powstające, na ogół prężnie działające w warunkach rzeczywistej konkurencji firmy prześcigają się w oferowaniu usług i technicznych nowości. Komputery stały się interesem, i to zarówno dla tych, którzy nimi handlują, jak i tych, co postanowili zrobić z nich użytek. Stworzył się zdrowy rynek, na którym decydujący głos należy wreszcie do klienta.

Dziś o nabywcę trzeba zabiegać. Jesteśmy jedynym krajem socjalistycznym, w którym organizuje się tak wiele imprez handlowych. W Warszawie mieliśmy trzecią już wystawę komputerów. Początkowo „Agpol" zorganizował ją w salach hotelu Victoria, ale już po paru dniach jej trwania („Komputer '86") stało się oczywiste, że miejsca jest zbyt mało i to zarówno dla wystawców, jak i dla zainteresowanych. Kolejne wystawy odbywały się już w Pałacu Kultury i Nauki. Zainteresowanie nie słabnie, natomiast już bez głębszych analiz można zauważyć, że z roku na rok przybywa na niej profesjonalistów. Ci, którzy chcą sprzedać przyznają, że miły uśmiech i elegancka powierzchowność już nie wystarczają, choć korzystne kontrakty pomagają często załatwić zawiłości naszego prawa. Nietrudno się domyśleć, kto na tym traci. Na szczęście firmy funkcjonujące tylko na zasadzie pośrednictwa, łaskawie przystawiające pieczęci pozwalające załatwić sprawę w majestacie prawa, spotkać można coraz rzadziej. Zdecydowana większość, to firmy pragnące rzetelnie załatwić klienta — bo przecież z niego żyją.

Tłok panował również na „Infosystemie '88" — międzynarodowych targach elektroniki, telekomunikacji i techniki komputerowej, które po benefisie we wrocławskiej Hali Ludowej przeniosły się na tereny Międzynarodowych Targów Poznańskich.

Niestety amatorzy i miłośnicy informatyki niewiele z tych imprez skorzystają. Ceny nowoczesnego sprzętu wielokrotnie przerastają możliwości normalnej kieszeni. Jedynym pocieszeniem niech będzie fakt, że Spectrum czy Atari nie zdrożało (!) — co praktycznie równa się jego relatywnie niższej dziś cenie. Naszym Czytelnikom pozostaje Centralna Składnica Harcerska, giełda, sklepy Pewexu i nadzieja, że może kiedyś doczekamy się polskiego mikrokomputera. Czego i sobie życzymy.

SPIS TREŚCI

Temperatura, głos i komputery — str. 3
Dyskowy system operacyjny dla ZX Spektrum — str. 4
Wszponach Atari (14) — str. 5
Program — Słownik — str. 9
Lilavati — cz. 4 — str. 10
Gra w ośmiu hetmanów — str. 12
Przygotowanie drukarki GEMINI—70XI — str. 15
Nuty z komputera (1) — str. 16
Programowanie mikroprocesora 6502 — str. 18
Kto pomoże? — str. 25
Liga Myślących — str. 25
Ocalony — str. 26
Program — Dźwięki. Klawiatura — str. 27
Spis najciekawszych programów — str. 28
Enigma — str. 31

TYLKO O 20 zł

Uprzejmie informujemy Czytelników, że rosnące ceny papieru i usług poligraficznych zmusiły nas do zmiany ceny.
Od czwartego numeru IKS kosztować będzie 100 zł. Cena w prenumeracie pozostanie bez zmian. Przepraszamy.

NAUKA • TECHNIKA • NAUKA

Już w 1990 roku w pierwszy rejs morski ma wyruszyć eksperymentalny japoński statek o nazwie YAMATO i wyposażony w nowego typu agregat, w którym wykorzystane zostanie zjawisko nadprzewodnictwa. Będzie on wyposażony w dwa silniki umieszczone na dnie statku, gdzie pobierać one będą wodę morską, która następnie zostanie wyrzucona na zewnątrz przy pomocy silnego pola magnetycznego. Wytwarzane ono będzie przez dwa silne elektromagnesy zbudowane z materiałów nadprzewodzących. Dzięki temu silnik zużywa nie tylko mniej energii elektrycznej, ale też zwiększa się znacznie jego moc. Zdaniem autorów projektu, w przyszłości tego typu statki będą w stanie rozwijać prędkości nawet do 100 węzłów, czyli ponad 180 km/godz.

■ ■ ■

W klinice uniwersyteckiej w Hamburgu testowane jest nowe urządzenie diagnostyczne tzw. "cyfrowa radiografia". Aparatura , podobnie jak inne obecnie stosowane urządzenia do badania układu kostnego, wykorzystuje również promienie rentgenowskie. Badanie sterowane jest jednak przez komputer, co umożliwiło zmniejszenie dawki promieniowania od 50 do 90 procent. W celu powiększenia niektórych badanych elementów układu kostnego nie trzeba przeprowadzać dodatkowych zdjęć rentgenowskich, lecz wystarczy wywołać z pamięci komputera potrzebny obraz detalu. Cyfrowa radiografia - zdaniem specjalistów - jest szczególnie przydatna w badaniach diagnostycznych dzieci oraz tych osób, u których ze względu na leczenie musi ono być często powtarzane.

■ ■ ■

Pracownicy Politechniki w LULEA (północna Szwecja) zerwali ze stereotypem, według którego paliwo do silników samochodowych powinno się spalać w wysokich temperaturach. Oryginalny pojazd szwedzkich inżynierów rozpoczyna jazdę dopiero wówczas, gdy paliwo w jego zbiorniku zamarznie. Prototyp pojazdu napędzanego lodem zaopatrzono w 25-litrowy zbiornik na wodę. W momencie zamarzania zwiększa ona swoją objętość, co powoduje wzrost ciśnienia w stalowej butli. Hydrauliczny mechanizm wykorzystuje podwyższone ciśnienie do uruchomienia napędu samochodu. Podczas prób niewielki samochodzik rozwijał prędkość do 50 km/godz., ale jego twórcy utrzymują, że przy drobnych zmianach konstrukcyjnych ten sam model powinien przekroczyć granicę 100 km/godz.

■ ■ ■

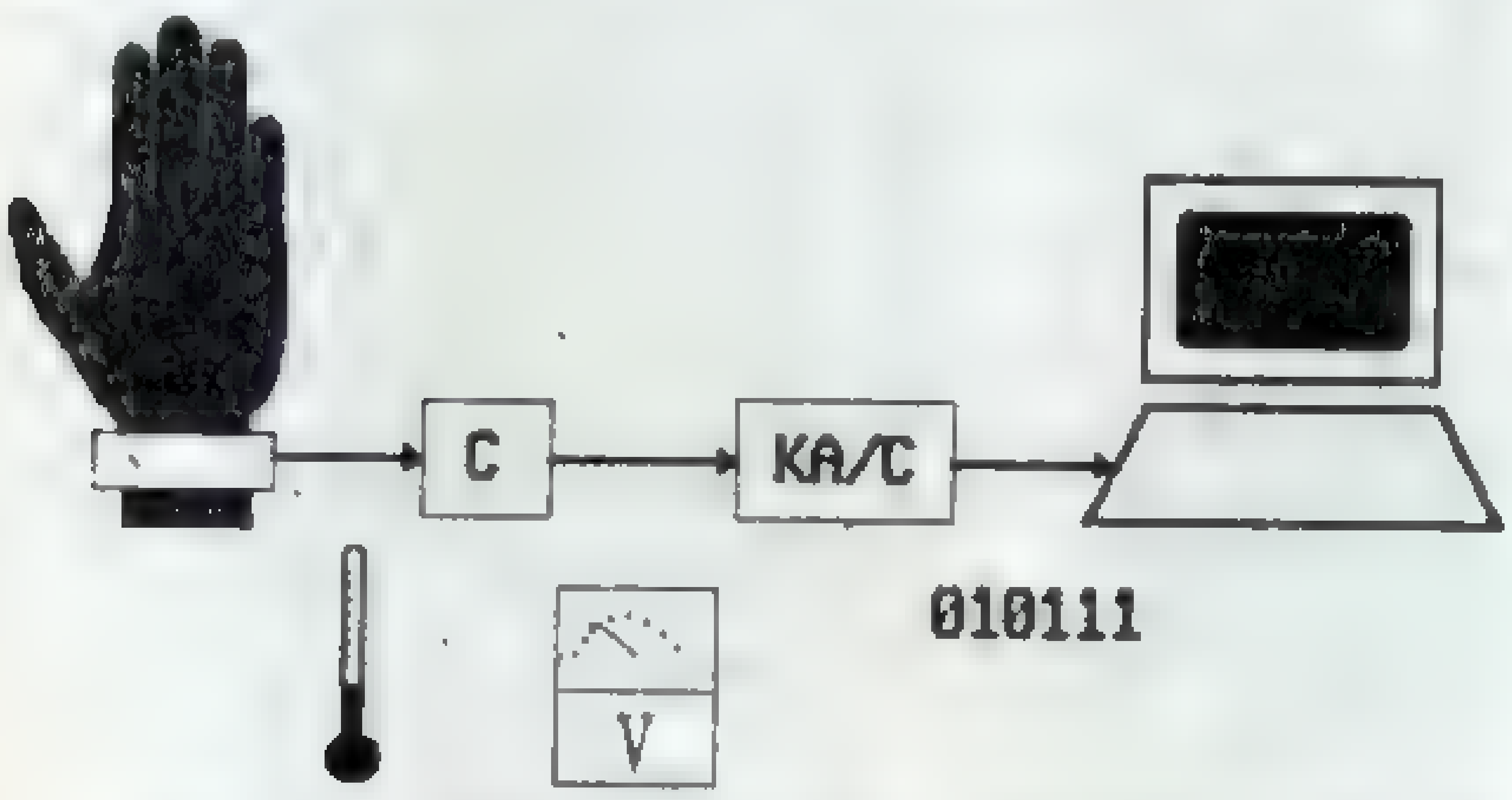
W zachodnioniemieckiej firmie SIEMENS opracowano nowy kompilator języka ADA. ADA jest językiem programowania powstałym na zamówienie amerykańskiego ministerstwa obrony między innymi w celu ograniczenia kosztów budowy i konserwacji oprogramowania wojskowych systemów informatycznych. Nowy kompilator charakteryzuje się budową modułową i może być wykorzystywany zarówno pod specjalizowanym systemem operacyjnym BS 2000, jak też przenoszony na inne systemy operacyjne i komputery.

TEMPERATURA, GŁOS I KOMPUTERY

Dziewięć, a może dziesięć lat temu w każdym razie na długo przed mikrokomputerami, w firmie mojej powstały warunki do realizacji „ambitnych planów młodego inżyniera informatyka". Wiązały się one z budową urządzeń, które umożliwiają wprowadzenie danych bezpośrednio do komputera, bez stosowania klawiatury lub innych papierowych nośników informacji. Innymi słowy chciałem, by informacje wprowadzane były do maszyny prosto ze źródła informacji, w moim przypadku od pacjenta, a ściślej z jego skóry. Przedmiotem pomiaru była bowiem temperatura osoby poddanej działaniu określonych czynników zewnętrznych, także farmakologicznych.

Dysponowałem wówczas odpowiednimi czujnikami temperatury. Zmieniały one jej wielkość na proporcjonalną wartość napięcia: od 0V do 5V. Gdy temperatura wynosiła 36°C czujnik „dawał" napięcie +0,5, gdy 43°C — 4,5V. Problem polegał zatem na tym, by uzyskane z czujników pomiary przekazać do komputera. Tradycyjnie odbywało się to w ten sposób, że odczytywane wielkości były notowane na papierze, by potem wprowadzić je do komputera za pośrednictwem klawiatury. Procedura ta była oczywiście bardzo pracochłonna i trudno było wykorzystać komputer do śledzenia temperatury w czasie rzeczywistym, na przykład w celu alarmowania, gdy wynik pomiaru przekroczył wartość 38.

W celu realizacji tego sprzężenia, czujnik — komputer, niezbędny okazał się konwerter A/C (analog/cyfra) — urządzenie elektroniczne, które „zmienia" napięcie na odpowiadającą jego wartości liczbę. Jest ona w takiej postaci (binarnej) jaką „akceptuje" każdy komputer.

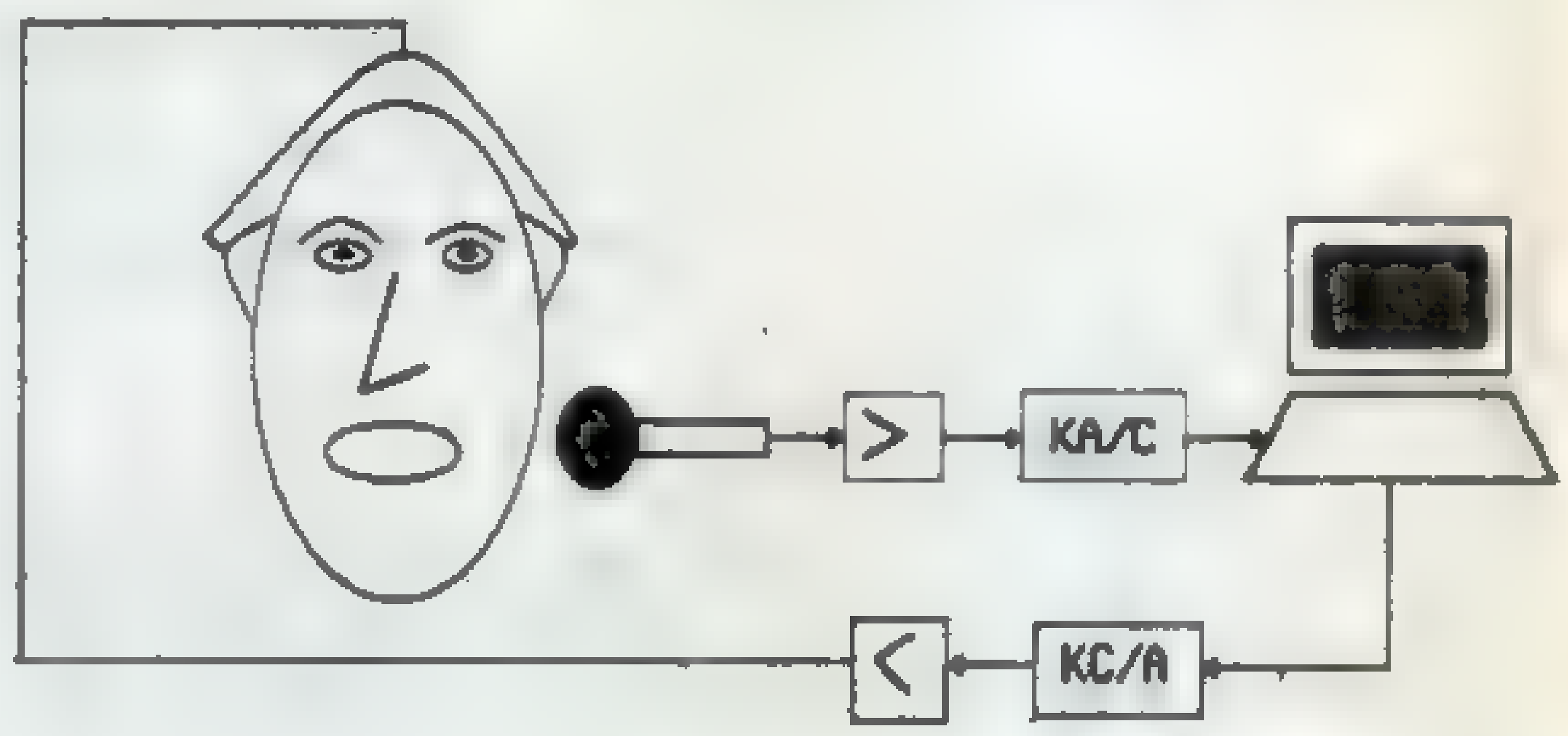


W praktycznej realizacji tego sprzężenia maszyna co pewien czas wysyła impuls „prośbę" do konwertera A/C, by „zechciał zobaczyć", jaka jest wartość temperatury. W odpowiedzi konwerter A/C zamienia napięcie (odpowiednie temperaturze) na liczbę i wysyła ją do komputera (za pośrednictwem interfejsu). Pomiar został dokonany — bez udziału człowieka, maszyna odczytała temperaturę i może ją dalej analizować.

Po podłączeniu czujnika do konwertera z jednej strony, komputera z drugiej strony, okazało się, że zestaw ten pracuje nadzwyczajnie — cicho, poprawnie i bardzo szybko. Poszedłem do pacjenta, który siedział w pokoju znacznie oddalonym od komputera, uśmiechnęliśmy się do siebie i... zwątpiłem czy na pewno komputer mierzy i rejestruje temperaturę. Szybko pobiegłem na salę maszyny, uruchomiłem drukarkę. Okazało się, że sprzężenie działa bez zarzutu — temperatura wolno się zmieniała. Wróciłem do pacjenta. Po chwili znowu byłem pełen wątpliwości, czy maszyna nadal odbiera informacje o temperaturze?

Oczywiście najprościej było by zainstalować obok pacjenta monitor sprzężony z komputerem i po prostu od czasu do czasu spojrzeć na ekran. Niestety kredyty się skończyły i o kupnie monitora nie było mowy. Należało wymyśleć coś tańszego, ale co?

Jak zwykle w takich sytuacjach pomógł przypadek. Koledze popsuł się magnetofon, po prostu przestał nagrywać. Mechanizm działał poprawnie, odtwarzanie także, a wypowiadane do mikrofonu słowa ledwo były słyszane podczas ich odtwarzania. Postanowiłem sprawdzić całą drogę sygnału — od mikrofonu do głowicy nagrywającej. Uruchomiłem oscyloskop, podłączyłem go do głowicy magnetofonu i w takt wypowiadanych do mikrofonu liczebników: raz, dwa, trzy... na ekranie oscyloskopu linia zaczęła podskakiwać. Im głośniejsze głoski słowa, tym wyżej na ekranie tańcowała linia. Było to oczywiste — mikrofon, podobnie jak czujnik temperatury, zamienia głos na sygnał elektryczny, który właśnie ukazywał oscyloskop. Stąd już tylko krok, by do komputera wprowadzać głos! Oczywiście pośredniczył w tym nasz znajomy konwerter A/C — podobnie jak miało to miejsce w przypadku temperatury. Innymi słowy konwerter odbiera z mikrofonu prąd, zamienia go na cyfry i wysyła do maszyny. Oczywiście proces zamiany i wysyłania przebiega bardzo szybko (chcąc uzyskać jakość dźwięku porównywalną do rozmowy telefonicznej, „zamiany" te powinny odbywać się z szybkością około 10 tys. w czasie 1 sek.).



To był pierwszy krok w konstruowaniu taniego urządzenia wyjściowego komputera, które pozwoliłoby mi na bieżące śledzenie, z dużej odległości, co komputer w danej chwili robi. Dysponując zatem sprzężeniem: mikrofon — konwerter A/C — komputer, z łatwością mogłem przekazać do pamięci maszyny wyrazy: jeden, dwa, trzy, ... dziewięć oraz trzydzieści i czterdzieści. W ten sposób maszyna „poznała" wzorce słownej interpretacji niektórych liczb. Wykorzystując odpowiedni program, komputer z łatwością mógł łączyć poszczególne słowa i generować cyfrowe odwzorowania słów: od trzydzieści do czterdzieści dziewięć.

Komputer dysponując wzorcami liczebników mógł je oczywiście odtwarzać — „dawać" na drutach interfejsu w postaci liczb binarnych. Ale przecież tego nie słychać! To są tylko pewne poziomy napięć — dla jedynek +5V, a dla zer 0V. Należało zatem wykonać zabieg odwrotny, niż to uczynił konwerter A/C. Cyfry podawane przez interfejs trzeba było z powrotem zamienić na sygnał elektryczny — taki jaki dawał wcześniej mikrofon. Okazało się to niezwykle proste — korzystając z konwertera C/A. Układ ten zamienia liczby binarne (odbierane z interfejsu komputera) na przebieg analogowy — ciągły sygnał elektryczny, taki jaki wcześniej uzyskiwałem z mikrofonu. Pozostało zatem tylko dodanie do konwertera głośnika i... komputer zaczął gadać i to jak! Szybko podłączyłem wyjście konwertera C/A do wzmacniacza, pociągnąłem przewód z sali komputerowej do pacjenta (ten ciągle na szczęście czekał), założyłem słuchawki. Słyszałem: trzydzieści sześć, trzydzieści sześć siedem, trzydzieści sześć sześć siedem, ciąg dalszy nastąpi...

PRAKTYK

Dyskowy system operacyjny dla ZX Spectrum

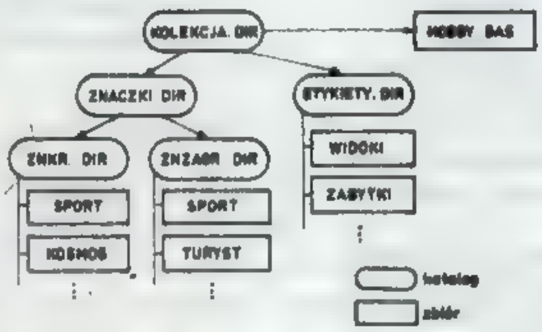
Do mikrokomputerów ZX SPECTRUM, UNIPOLBRIT 2086 i TIMEX TC 2048 może być dołączona tego samego typu pamięć mikrodyskowa (pmd) firmy TIMEX. Współpraca pmd z mikrokomputerem odbywa się na podstawie dyskowego systemu operacyjnego DOS (disk operating system). W skład pmd wchodzą: sterownik, napędy dyskietek, zasilacz i sprzęg (interface) łączący mikrokomputer z pmd. Całość może być w jednej obudowie lub stanowić oddzielne moduły łączone między sobą przewodami. Z jednym mikrokomputerem może współpracować od 1 do 4 napędów dyskietek. Poszczególne napędy oznaczone są literami A, B, C, D. W takim wypadku jeden zasilacz obsługuje dwa napędy dyskietek. Pracą napędów zarządza sterownik (controber). Jest to samodzielny mikrokomputer zawierający mikroprocesor Z80A i 16kB pamięci RAM. W pamięci tej rezyduje system operacyjny DOS. Sterownik ma dodatkowo wyprowadzone na zewnątrz dwa łącza typu RS-232C, umożliwiające współpracę z innymi komputerami oraz urządzeniami zewnętrznymi takimi jak: modemy, drukarki, plotery itp.

Nośnikiem informacji jest dyskietka o średnicy 3 cali. Jest to elastyczny krążek z tworzywa sztucznego, pokryty materiałem magnetycznym i zamknięty w sztywnej plastikowej kasecie. W kasecie znajdują się dwa okienka, przez które dyskietka ma styczność z głowicami odczytu — zapisu odpowiedniego napędu. Gdy dyskietka znajduje się na zewnątrz napędu, to okienka są zamknięte zasłonami i chronią powierzchnię roboczą dyskietki przed zanieczyszczeniami. W czasie wsuwania dyskietki do napędu, zasłony automatycznie otwierają się i umożliwiają styk głowic z dyskietką.

Informacja na dyskietce jest zapisana na dwóch powierzchniach oznaczonych literami A i B. Napęd dyskietki umożliwia jednocześnie dostęp tylko do jednej powierzchni A lub B. Pojemność jednej powierzchni wynosi 160 kB, a całej dyskietki 320 kB. Powierzchnia robocza dyskietki podzielona jest przez DOS w trakcie jej formatowania na 40 ścieżek po 16 sektorów i po 256 bajtów w sektorze. W czasie odczytu lub zapisu dyskietka wiruje ze stałą prędkością. Dyskietka ma mechaniczne zabezpieczenia przed możliwością niepożądanego zapisu w postaci małych plastikowych przycisków. Wciśnięcie przycisku uniemożliwia zapis na dyskietce.

Producenci dyskietek zalecają, aby dyskietek nie poddawać działaniu pól magnetycznych, wysokiej temperaturze, nie dotykać żadnymi przedmiotami powierzchni roboczych i chronić przed kurzem i wilgocią. Przestrzeganie tych zaleceń zapewni niezawodność użytkowania dyskietki.

Informacje zapisywane na dyskietkach (programy, zbiory danych) są organizowane przez DOS w odpowiednie struktury danych. Dane można grupować w katalogach, te katalogi zagnieżdżać w innych katalogach itp. Tworzy się w ten sposób „drzewo struktury danych". Zaletą takiej organizacji jest możliwość dogodnego podziału istniejących grup informacji i programów wg nazw katalogów i zbiorów. Schemat takiej struktury przedstawia poniższy rysunek. Na rysunku przedstawiono przykładową strukturę katalogów i zbiorów, utworzoną dla potrzeb np. kolekcjonera znaczków pocztowych i etykietek zapałczanych. Katalogiem głównym jest tu katalog o nazwie KOLEKCJA. DIR, który zawiera dwa katalogi niższego poziomu o nazwach ZNACZKI. DIR i ETYKIETY. DIR oraz program obsługujący te zbiory o nazwie HOBBY. BAS. Z kolei katalog ZNACZKI.



DIR znów zawiera dwa katalogi ZNKR. DIR I ZNZAGR. DIR, a katalog ETYKIETY. DIR zawiera już nazwy zbiorów etykiet. Nazwy zbiorów znaczków krajowych i zagranicznych zawierają katalogi ZNKR. DIR i ZNZAGR. DIR. Nazwa katalogu lub zbioru może zawierać maksymalnie 8 liter i cyfr i zaczyna się od litery. Po kropce podaje się tzw. rozszerzenie. Rozszerzenie DIR oznacza, że nazwa jest nazwą katalogu, a np. BAS oznacza nazwę zbioru będącego programem w języku BASIC. Katalogi muszą być obowiązkowo oznaczone rozszerzeniem DIR, natomiast zbiory mogą mieć rozszerzenie dowolne lub nie posiadać ich wcale.

Dojście do zbioru znajdującego się na najniższym poziomie odbywa się poprzez określenie tzw. drogi do zbioru lub ciągu nazw katalogów i zbioru. Przykładowo, aby dojść do zbioru znaczków KOSMOS, należy określić ciąg nazw w sposób następujący: KOLEKCJA. DIR: ZNACZKI. DIR: ZNKR. DIR: KOSMOS

Tworzenie katalogów i zbiorów odbywa się za pomocą dyrektywy DIM i podania ciągu nazw. Na przykład, aby utworzyć katalog ZNKR. DIR, trzeba przedtem utworzyć katalogi KOLEKCJA. DIR I ZNACZKI. DIR. Będzie to wyglądało następująco:
DIM „KOLEKCJA.DIR" — utworzenie katalogu KOLEKCJA.DIR
DIM „KOLEKCJA.DIR: ZNACZKI.DIR" — utworzenie katalogu ZNACZKI.DIR
DIM „KOLEKCJA.DIR: ZNACZKI.DIR: ZNKR.DIR" — utworzenie katalogu ZNKR.DIR

Informację o zawartości katalogów uzyskujemy za pomocą dyrektywy CAT i odpowiedniego ciągu nazw do katalogu. Chcąc uzyskać informację o zawartości katalogu ZNZAGR.DIR polecenie należy sformułować następująco:
CAT „KOLEKCJA.DIR: ZNACZKI.-DIR: ZNZGR.DIR"

W wyniku wykonywania tej dyrektywy na ekranie wyświetli się wykaz zbiorów znajdujących się w katalogu ZNZAGR.DIR z informacją o wielkości zbioru w bajtach i o wielkości pamięci zajętej przez ten zbiór. Występuje tu pewna rozrzutność w gospodarowaniu pojemnością dyskietki przez DOS, który na zbiór rezerwuje obszar w pełnych kB i np. zbiór o wielkości 50 B będzie miał zarezerwowane 1 kB, a zbiór o 1040B — 2 kB.

Zbiory mogą być programowo zabezpieczone przed zapisem lub odczytem. Ochrona przed odczytem polega na tym, że nazwy zbiorów nie wpisują się w trakcie działania dyrektywy CAT, są niewidoczne. Ochrony zbiorów przed zapisem lub odczytem i zdjęcie tej ochrony dokonuje się za pomocą dyrektywy ATTR.

Zabezpieczenie programowe nie chroni przed zniszczeniem informacji dyrektywą FORMAT. Dyrektywa ta przygotowuje dyskietkę do pracy z DOS. Wyznacza ona ścieżki i sektory dyskietki oraz zeruje dyskietkę. Czynność ta nazywa się formatowaniem dyskietki. Ponieważ dyrektywa ta niszczy całą zawartość dyskietki, należy się dobrze zastanowić przed jej formatowaniem, aby nie zniszczyć przypadkowo ważnych informacji.

Zapisu do zbioru, odczytu ze zbioru, łączenia dwóch zbiorów dokonuje się za pomocą dyrektyw podobnych jak przy współpracy z magnetofonem kasetowym. I tak SAVE — powoduje zapis do zbioru, LOAD — powoduje przepisanie zbioru do pamięci LOAD musi być obowiązkowo podana nazwa zbioru lub ciąg nazw. MERGE — pozwala na połączenie dwóch zbiorów w RAM. W programie można używać dyrektyw zarówno do współpracy z pmd, jak i magnetofonem. DOS rozróżnia dyrektywę ŁOAD od LOAD. Dyrektyw LOAD i SAVE można używać do odczytu i zapisu obrazów monitora, tablic, programów w języku BASIC i kodzie maszynowym.

Oprócz tego istnieją dyrektywy pozwalające na:

— skasowanie wybranego zbioru — ERASE

— zmiany nazwy zbioru — LET

— kopiowanie zbiorów z jednego katalogu do innego — MOWE

Dla ułatwienia pracy można tworzyć program startu, który automatycznie uruchamia się po wczytaniu i naciśnięciu przycisku RESET na sprzęgu pmd. Program ten może ściągać z dyskietki do RAM potrzebne do dalszej pracy zbiory i programy, wyświetlać listę czynności, itp.

Jak więc widać, pmd ma wiele zalet, zapisywanie i wczytywanie programów z dyskietki jest wielokrotnie szybsze w porównaniu z magnetofonem i bardziej niezawodne. Programy mogą się w każdej chwili komunikować z dyskiem i pobierać potrzebne dane, a także zapisywać wyniki obliczeń. Wadą zaś jest zbyt wysoka cena. Pamięć mikrodyskowa jest kilkakrotnie droższa od samego mikrokomputera.

J.R.

W szponach Atari (14)

Graficzna prezentacja danych liczbowych

Tomasz MROWIEC, Ludwik PIELA

Dane, przedstawione w postaci kolumn liczb lub tablicy, trudniej jest zwykle analizować niż ich graficzną prezentację w postaci wykresów. Te ostatnie pozwalają szybciej ocenić informację zawartą w zbiorze liczb. Umożliwiają poglądowe przedstawienie różnorodnych zależności między danymi, które często trudno wykryć w kolumnie wydrukowanych liczb.

Programy przeznaczone do graficznej prezentacji danych liczbowych wchodzą w skład oprogramowania większości komputerów. O ich przydatności najlepiej świadczy fakt, że są elementem składowym większości pakietów zintegrowanych „poważnych" mikrokomputerów. Również komputery domowe, tej klasy co Atari serii XL lub XE, posiadają zwykle kilka programów tego typu. Dla Atari są to, na przykład: GRAPH IT, SYNGRAPH oraz B/GRAPH. Zwykle są to programy uniwersalne, przeznaczone albo wyłącznie do graficznej prezentacji danych (GRAPH IT), albo do obliczeń statystycznych z mozliwością wyprowadzania wyników w postaci graficznej (B/GRAPH).

Często zdarza się jednak, że chcemy przedstawić w postaci graficznej wyniki obliczeń własnego programu. Wtedy musimy włączyć do niego moduł tworzący odpowiednie wykresy, albo dysponować programem, który przyjmie wyniki w charakterze danych wejściowych i przedstawi je w postaci graficznej. Dlatego spróbujemy zaprezentować kilka metod i przykładowych programów umozliwiających uzyskanie kilku podstawowych typów wykresów i diagramów, od najprostszych do dosyć wyrafinowanych. Mamy nadzieję, że czytelnicy zaadaptują je do własnych potrzeb oraz uzupełnią o dodatkowe mozliwości.

Do tworzenia wykresów mozna używać dowolnego trybu graficznego komputera Atari oraz odpowiednich instrukcji, bądź PRINT do trybów tekstowych, bądź PLOT i DRAWTO dla graficznych. Chociaż w pewnych przypadkach wygodnie jest posługiwać się instrukcją PRINT, to środki graficzne pozwalają tworzyć wykresy bardziej szczegółowe.

Do najprostszych należą wykresy prezentujące ogólne tendencje lub charakter zmian danych, zwykle bez żadnego lub z minimalnym opisem. Mogą one przedstawiać, na przykład, obraz zmian jakiejś wielkości w zadanym okresie. Można go tak umieścić, aby wielkości odkładały się albo w poziomie, albo w pionie. W pierwszym przypadku okresy czasu odliczane są od góry do dołu, a odpowiadające im wielkości — w kierunku poziomym od lewej do prawej.

Przybliżony wykres wartości mozna utworzyć wyświetlając wybrane symbole w miejscach ekranu odpowiadających ich względnym wartościom. Załóżmy, że postanowiliśmy umieszczać wielkości poziomo na ekranie, który ma 24 wiersze po 40 znaków. Pozycje symboli w linii numerujemy od lewej do prawej od 0 do 39, a wiersze — z góry na dół od 0 do 23. Będziemy wykreślać tylko w wierszach nieparzystych, a w nich dla przedstawienia wartości — pozycje od 8 do 38. W tym przypadku maksymalna wielkość odpowiadać będzie 38 pozycji, a minimalna — 8 pozycji.

W celu zobrazowania wielkości w zakresie od 8 do 38 pozycji musimy wykonać skalowanie. Możemy to zrobić według następującego wzoru:

N=(W—Wm)*Zp/Zw+Nm

gdzie:

N — numer pozycji w wierszu,
W — kreślona wielkość,
Wm — wielkość minimalna,
Zp — zakres numerów pozycji,
Zw — zakres wielkości,
Nm — pozycja minimalna

W naszym przykładzie zakres wartości przedstawianych danych równy jest WM — Wm, zakres numerów pozycji 38—8=30 i numer pozycji minimalnej — 8. A zatem numer pozycji wydruku symbolu gwiazdki („*") w wierszu, dla dowolnej wartości przedstawianych danych, mozna wyznaczyć według wzoru:

Np=(W—Wm)·30/(WM—Wm)+8

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Prosram nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
LS 40 GRAPHICS 0
MA 50 R=(38-8)/(651-99)
GW 60 FOR K=1 TO 10
XQ 65 PRINT
DH 70 READ S
MX 80 P=INT((S-99)*R+8+0.5)
EM 90 FOR I=1 TO P-1:PRINT " ";:N
   EXT I
EE 100 PRINT "*"
GP 120 NEXT K
WU 130 DATA 210,150,99,250,183,35
   2,410,390,300,651
NG 140 GOTO 140
```

Program nr 1 tworzy przybliżony wykres wykorzystując powyższy wzór. Dla każdej wartości wyznaczana jest pozycja wydruku symbolu gwiazdki, która zaokrąglona jest do najbliższej wartości całkowitej za pomocą funkcji INT. Po pewnych modyfikacjach program może pracować z dowolnym zbiorem danych, przy dowolnych wymiarach ekranu. W tym celu w podanym wzorze należy wykorzystać wartości minimalną i maksymalną oraz numery pozycji wprowadzone instrukcją INPUT.

Aby otrzymać wykres, w którym wartości odkładają się w pionie, wprowadzimy do powyższego programu dwie zmiany. Po pierwsze, dla każdego elementu danych określimy odpowiedni numer wiersza. Po drugie, w tej linii wybierzemy numer pozycji, odpowiadający zadanemu okresowi czasu. W ten sposób kazdy element danych będzie przedstawiony na ekranie symbolem gwiazdki, w pozycji o obliczonych numerach wiersza i kolumny.

Do wskazania odpowiedniej pozycji gwiazdki na ekranie można wykorzystać instrukcję POSITION, która określa położenie symbolu na ekranie dla następnej instrukcji PRINT.

Dla danych, używanych w programie nr 1 i dla ekranu zawierającego 24 wiersze po 40 znaków, wybierzemy dla miesięcy kazdą trzecią kolumnę, zaczynając od 3, a kończąc na 36. Na wykres wykorzystamy 20 górnych linii ekranu. W tym przypadku wartości maksymalnej (724) odpowiadać będzie pozycja 33 najwyższego wiersza ekranu (o numerze 1). Numery pozycji wzdłuz każdego wiersza określa się jako 3·M, gdzie M — numer miesiąca. Dla skalowania wartości w zakresie pierwszych 20 wierszy ekranu wykorzystamy następujący wzór:

Nw=(WM—W)·Zw/Z+Sm

gdzie

Nw — numer wiersza,
WM — wartość maksymalna
W — wartość bieżąca,
Zw — zakres wierszy,
Z — zakres wartości,
Sm — minimalny numer wiersza.

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Prosram nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
CL 50 GRAPHICS 0:POKE 752,2
HP 60 T=(20-1)/(724-99)
IT 70 FOR M=1 TO 12
DI 80 READ S
MT 90 WI=INT((724-S)*T+1+0.5)
SZ 100 KO=M*3
PP 110 POSITION KO,WI
KC 120 PRINT "*";
HL 130 NEXT M
LY 140 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NT 150 GOTO 150
```

W tym przykładzie zakres numerów wierszy wynosi 19 (czyli 20—1), minimalny numer wiersza równy jest 1. Opisane zadanie realizuje program nr 2. Można go tak zmodyfikować, aby zapewnić dowolne położenie wykresu. W takim przypadku żądane liczby wierszy i kolumn wydruku wprowadza się za pomocą instrukcji INPUT. Zakres wartości danych i wartość maksymalna mogą być określone przez program podczas wprowadzania danych.

Powyższe przykłady byłyby bardzo przydatne, gdybyśmy chcieli pracować wyłącznie w trybie tekstowym. Mozemy jednak wybrać jeden z wielu trybów graficznych i wykorzystać specjalne instrukcje graficzne w celu działania na współrzędnych punktów elementarnych, zamiast na numerach wierszy i kolumn. Zyskamy wtedy na dokładności wykresu. Spróbujmy zatem utworzyć wykres, podobny do powyższych, wykorzystując instrukcję PLOT. Załóżmy, ze rozdzielczość ekranu równa jest 160 punktów w poziomie i 80 punktów w pionie (GRAPHICS 7). Wykres rozmieścimy tak,

aby wykorzystać linie od 9 do 79 i kolumny od 10 do 120. Dla oznaczenia miesięcy użyjemy co 10 punktu zaczynając od 10. W celu wyskalowania wartości danych w zakresie od 0 do 79 wykorzystamy następujący wzór:

$$Y=(WM-W)*Zw/Z+Ym$$

Omawiany, bardzo uproszczony, wykres pionowy kreślony jest przez program nr 3. Aby otrzymać wykres, w którym wartości danych odkładają się w poziomie, należy zamienić miejscami współrzędne X i Y, biorąc pod uwagę rozmiary ekranu.

```
FK 1 REM ***************
FJ 2 REM *             *
YO 3 REM * Prosram nr 3 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
YG 50 GRAPHICS 7:COLOR 1
HL 60 R=(40-0)/(724-99)
HA 70 X=10
IE 80 FOR K=1 TO 12
DJ 90 READ S
VA 100 Y=INT((724-S)*R+0.5)
SU 110 PLOT X,Y
PO 120 X=X+10
GR 130 NEXT K
LY 140 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
OA 150 END
```

Punkty wykresu można połączyć między sobą odcinkami prostych. Taki wykres, zwany liniowym, tworzony jest przez program nr 4.

```
FK 1 REM ***************
FJ 2 REM *             *
ZI 3 REM * Prosram nr 4 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
YG 50 GRAPHICS 7:COLOR 1
GL 60 DIM X(12),Y(12)
HM 70 R=(40-0)/(724-99)
WV 80 X1=20
IV 90 FOR M=1 TO 12
ZE 100 READ S
DQ 110 Y(M)=INT((724-S)*R+0.5)
VK 120 X(M)=X1
IK 130 X1=X1+10
HN 140 NEXT M
XN 150 PLOT X(1),Y(1)
TI 160 FOR K=2 TO 12
VV 170 DRAWTO X(K),Y(K)
HB 180 NEXT K
MI 190 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NR 200 END
```

Najprostsze metody, opisane powyżej, przydatne są do szybkiego tworzenia prostych wykresów i przedstawiania charakteru zmian danych. Zawierają one jednak zbyt mało informacji ilościowej. Zwykle interesuje nas uzyskanie z wykresu informacji bardziej szczegółowych. Wyskalowanie i opisanie osi współrzędnych pozwala dokładniej określić zależności między wielkościami, a także dokonać interpolacji danych.

Opisywanie wykresów wymaga pewnych zmian w równaniach skalowania. Wybierając zakres wartości należy uwzględnić opis osi współrzędnych. Jeśli, na przykład, przetwarzamy zbiór liczb o zakresie wartości (—96, 89), a osie współrzędnych opisane są w przedziale (—100, 100), to w równaniach należy podstawić zakres wartości równy 200. Analogicznie, wartość minimalna będzie równa —100, a maksymalna — 100.

Osie współrzędnych możemy wykreślić za pomocą ciągu symboli „+", lub „—", tak jak w programie nr 5. Kreśli on wykres punktowy w trybie tekstowym, pomimo tego osie są wyskalowane i opisane.

```
FK 1 REM ***************
FJ 2 REM *             *
AC 3 REM * Prosram nr 5 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
FR 40 DIM M$(3),CLS$(1):CLS$=CHR$
   (125):POKE 752,1
VO 45 ? CLS$
YJ 50 POSITION 11,0:? "ROCZNY WYK
   RES CEN"
CD 70 POSITION 6,2:? "  1   2
   3   4   5   6   7   8"
VR 80 POSITION 6,3:? "    0   0
   0   0   0   0   0   0"
IP 90 POSITION 6,4:? "0   0   0
   0   0   0   0   0   0"
NM 100 POSITION 6,5:? "+---+---+-
   --+---+---+---+---+---+"
WY 110 R=(38-6)/(800-0)
HN 120 MIN=10000
BE 130 MAX=0
HN 140 T=0
SV 150 FOR K=1 TO 12
WJ 160 READ M$,S
PB 170 IF S<MIN THEN MIN=S
SZ 180 IF S>MAX THEN MAX=S
HO 190 T=T+S
HC 200 P=INT((S-0)*R+6+0.5)
DD 210 POSITION 0,K+5:? M$;"    :"
   ;:POSITION P,K+5:? "*";:POSITI
   ON 38,K+5:? ":";
GQ 220 NEXT K
IV 230 POSITION 6,18:? "+---+---+
   ---+---+---+---+---+---+"
NT 250 POSITION 11,20:? "CENA MIN
   IMALNA ";:POSITION 29+(3-LEN(S
   TR$(MIN))),20:? MIN;
UH 260 POSITION 11,21:? "CENA MAK
   SYMALNA ";:POSITION 29+(3-LEN(
   STR$(MAX))),21:? MAX;
KX 270 POSITION 11,23:? "CENA SRE
   DNIA ";:POSITION 29+(3-LEN(STR
   $(INT(T/12)))),23:? INT(10*T/1
   2)/10;
PR 280 GOTO 280
FB 290 DATA STY,210,LUT,155,MAR,7
   9,KWI,150,MAJ,175,CZE,252,LIP,
   510,SIE,280,WRZ,300,PAZ,751,LI
   S,724,GRU,726
```

Wygodniejszym sposobem kreślenia osi współrzędnych jest użycie trybu graficznego o dużej rozdzielczości. Pojawia się jednak pewien problem z opisywaniem wykresów. W komputerach Atari za pomocą instrukcji języka BASIC nie możemy, w sposób bezpośredni, umieścić znaków alfabetycznych w dowolnym miejscu ekranu (za wyjątkiem okna tekstowego). Musimy do tego wykorzystać specjalny podprogram „kreślący" te znaki. W programie nr 6, i dalszych, zaczyna się on od linii 10000. Przed wywołaniem go należy określić współrzędne położenia początku tekstu (X i Y w liniach ekranu), a drukowany tekst umieścić w zmiennej tekstowej T. Niestety, takie rozwiązanie ogranicza możliwości używania różnych trybów graficznych. Wykresy z opisami (czytelnymi) możemy uzyskać tylko w trybie graficznym 8.

```
FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Prosram nr 6 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
AD 60 DIM X(12),Y(12),T$(40),TX$(
   21)
NT 80 GRAPHICS 8+16:COLOR 1
CW 100 T$="WYKRES PRODUKCJI":X=15
   :Y=1:GOSUB 10000
EH 110 PLOT 80,27:DRAWTO 80,155
WE 120 PLOT 304,155:DRAWTO 304,27
QF 140 FOR Y=27 TO 155 STEP 16
WF 150 PLOT 79,Y:DRAWTO 304,Y
MH 160 NEXT Y
RA 170 REM
AP 180 Y=19
WL 190 FOR S=0 TO 800 STEP 100
XE 200 T$="   ":T$(4-LEN(STR$(S))
   )=STR$(S)
AP 210 X=6:GOSUB 10000
YO 220 Y=Y-2
JU 230 NEXT S
QV 240 REM
OA 250 FOR CO=12 TO 34 STEP 2
BZ 260 X=CO:Y=19:T$="+"
SP 270 GOSUB 10000
LL 280 NEXT CO
NJ 290 T$="S L M K M C L S W P L
   G":X=12:Y=20:GOSUB 10000
GM 300 T$="T U A W A Z I I R A I
   R":X=12:Y=21:GOSUB 10000
YY 310 T$="Y T R I J E P E Z Z S
   U":X=12:Y=22:GOSUB 10000
IY 320 TX$="WARTOSC PRODUKCJI"
SI 330 FOR K=1 TO 17:T$=TX$(K,K)
BM 340 X=2:Y=K+2:GOSUB 10000
GX 350 NEXT K
ZX 370 T=(155-27)/(800-0)
TQ 380 X1=96
TF 390 FOR K=1 TO 12
UV 400 X(K)=X1
LD 410 X1=X1+16
ZL 420 READ S
JN 430 Y(K)=INT((800-S)*T+27+0.5)
GW 440 NEXT K
SH 450 FOR K=1 TO 11
BY 460 PLOT X(K),Y(K):DRAWTO X(K+
   1),Y(K+1)
MG 480 PLOT X(K),Y(K)-1:DRAWTO X(
   K+1),Y(K+1)-1
HG 490 NEXT K
LU 500 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NL 510 GOTO 510
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
```

Podczas tworzenia wykresów z opisem osi należy przestrzegać pewnych reguł. Opis powinien być prosty i krótki. Większa ilość objaśnień komplikuje wykres i utrudnia jego analizę. Tekst wyjaśniający należy umieszczać, w miarę możliwości, w tym wierszu lub obszarze co obrazowane dane, a nie wynosić go w dodatkowe tablice lub opisy. Krok opisu osi współrzędnych należy wybrać dogodnym do analizy wykresu, na przykład, krok opisu równy 10 będzie wygodniejszy, niż krok opisu równy 7. Analizę ułatwia zaznaczenie początku układu współrzędnych. Przydatne jest również, szczególnie dla ułatwienia interpolacji danych, naniesienie podziałów na osiach współrzędnych. Linie wykresu powinny być grubsze lub jaśniejsze niż osie lub linie siatki współrzędnych.

Powyższe zalecenia zostały uwzględnione podczas tworzenia wykresu programem nr 6.

Wygodnym sposobem, ułatwiającym analizę, jest przedstawienie danych w postaci zbioru kolumn, zamiast oddzielnych punktów. Sposób ten ilustruje program nr 7, wykorzystujący instrukcję PRINT do tworzenia wykresu.

Przykładem tworzenia histogramu (wykresu słupkowego) w trybie graficznym jest program nr 8. W otrzymanym wykresie szerokość kolumn równa jest szerokości odstępu między nimi. Należy trzymać się tej zasady, ponieważ wąskie słupki są mniej poglądowe.

Wygląd lub poglądowość kreślonych histogramów można poprawić za pomocą kolorów. Wyboru kombinacji kolorów należy dokonać bardzo ostrożnie, gdyż zbyt duża ich liczba lub ich niezgodność mogą pogorszyć wygląd.

```
FK 1 REM ***************
FJ 2 REM *             *
BQ 3 REM * Prosram nr 7 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
FR 40 DIM M$(3),CLS$(1):CLS$=CHR$
   (125):POKE 752,1
VO 45 ? CLS$
YJ 50 POSITION 11,0:? "ROCZNY WYK
   RES CEN"
CD 70 POSITION 6,2:? "    1    2
    3    4    5    6    7    8"
VR 80 POSITION 6,3:? "    0    0
    0    0    0    0    0    0"
IP 90 POSITION 6,4:? "0    0    0
    0    0    0    0    0    0"
NM 100 POSITION 6,5:? "+----+----+
   ----+----+----+----+----+----+"
WY 110 R=(38-6)/(800-0)
HN 120 MIN=10000
BE 130 MAX=0
HN 140 T=0
SV 150 FOR K=1 TO 12
WJ 160 READ M$,S
PB 170 IF S<MIN THEN MIN=S
SZ 180 IF S>MAX THEN MAX=S
HO 190 T=T+S
HC 200 P=INT((S-0)*R+6+0.5)
UB 210 POSITION 0,K+5:? M$;"    :"
   ;:FOR C=7 TO P:? "*";:NEXT C:P
   OSITION 38,K+5:? ":";
YW 215 REM W miejsce "*" mozna ws
   tawic
DI 216 REM znak "⊔" (CHR$(21))
GQ 220 NEXT K
IV 230 POSITION 6,18:? "+----+----+
   ----+----+----+----+----+----+"
NT 250 POSITION 11,20:? "CENA MIN
   IMALNA ";:POSITION 29+(3-LEN(S
   TR$(MIN))),20:? MIN;
UH 260 POSITION 11,21:? "CENA MAK
   SYMALNA ";:POSITION 29+(3-LEN(
   STR$(MAX))),21:? MAX;
KX 270 POSITION 11,23:? "CENA SRE
   DNIA ";:POSITION 29+(3-LEN(STR
   $(INT(T/12)))),23:? INT(10*T/1
   2)/10;
PR 280 GOTO 280
AX 290 DATA STY,210,LUT,155,MAR,9
   9,KWI,150,MAJ,175,CZE,252,LIP,
   510,SIE,590,WRZ,600,PAZ,751,LI
   S,724,GRU,726
```

```
FK 1 REM ***************
FJ 2 REM *             *
CK 3 REM * Prosram nr 8 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
AD 60 DIM X(12),Y(12),T$(40),TX$(
   21)
NT 80 GRAPHICS 8+16:COLOR 1
CW 100 T$="WYKRES PRODUKCJI":X=15
   :Y=1:GOSUB 10000
ZK 110 PLOT 80,20:DRAWTO 80,163
NL 120 PLOT 304,163:DRAWTO 304,20
GE 125 PLOT 80,163:DRAWTO 304,163
XT 130 PLOT 304,20:DRAWTO 80,20
TD 140 FOR Y=27 TO 155 STEP 8
AR 150 PLOT 78,Y:DRAWTO 82,Y
MH 160 NEXT Y
RA 170 REM
AP 180 Y=19
WL 190 FOR S=0 TO 800 STEP 100
XE 200 T$="   ":T$(4-LEN(STR$(S))
   )=STR$(S)
AP 210 X=6:GOSUB 10000
YO 220 Y=Y-2
JU 230 NEXT S
QV 240 REM
OA 250 FOR CO=12 TO 34 STEP 2
XY 260 X=CO:Y=20:T$="+"
SP 270 GOSUB 10000
LL 280 NEXT CO
OZ 290 T$="S L M K M C L S W P L
   G":X=12:Y=21:GOSUB 10000
IC 300 T$="T U A W A Z I I R A I
   R":X=12:Y=22:GOSUB 10000
AO 310 T$="Y T R I J E P E Z Z S
   U":X=12:Y=23:GOSUB 10000
IY 320 TX$="WARTOSC PRODUKCJI"
SI 330 FOR K=1 TO 17:T$=TX$(K,K)
BM 340 X=2:Y=K+2:GOSUB 10000
GX 350 NEXT K
IG 360 REM WYKRESLANIE SLUPKOW
ZX 370 T=(155-27)/(800-0)
TQ 380 X1=96
TF 390 FOR K=1 TO 12
ZL 420 READ S
KU 430 Y=INT((800-S)*T+27+0.5)
SI 440 FOR X=X1 TO X1+8
XB 450 PLOT X,Y:DRAWTO X,155
MA 460 NEXT X
LP 470 X1=X1+16
HG 490 NEXT K
LU 500 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NL 510 GOTO 510
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
   TURN
```

```
FK 1 REM ***************
FJ 2 REM *             *
DE 3 REM * Prosram nr 9 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
CY 10 REM ******************
CV 20 REM *KRESKOWANE WYKRESY*
AZ 30 REM *   W GRAFICE 8    *
YM 40 REM *Z OPISEM TEKSTOWYM*
DC 50 REM ******************
VP 60 DIM T$(40),TX$(21)
NT 80 GRAPHICS 8+16:COLOR 1
VD 100 T$="PRODUKCJA GLOBALNA":X=
   15:Y=1:GOSUB 10000
AM 110 PLOT 80,24:DRAWTO 80,162
RL 120 PLOT 304,162:DRAWTO 304,24
OL 125 PLOT 80,162:DRAWTO 304,162
EJ 130 PLOT 304,24:DRAWTO 80,24
TV 140 FOR Y=36 TO 156 STEP 8
AR 150 PLOT 78,Y:DRAWTO 82,Y
MH 160 NEXT Y
RA 170 REM
AP 180 Y=19
VY 190 FOR S=0 TO 30 STEP 10
XE 200 T$="   ":T$(4-LEN(STR$(S))
   )=STR$(S)
AP 210 X=6:GOSUB 10000
ZP 220 Y=Y-5
JU 230 NEXT S
GK 290 X=0:Y=21:T$="KWARTAL":GOSU
   B 10000
VL 300 X=13:Y=21:T$="1      2
   3      4":GOSUB 10000
IW 310 TX$="WARTOSC PRODUKCJI"
PU 320 FOR K=1 TO 17:T$=TX$(K,K):
   X=1:Y=K+2:GOSUB 10000:NEXT K
IM 330 TX$="W MILIONACH ZL."
UE 340 FOR K=1 TO 15:T$=TX$(K,K):
   X=3:Y=K+4:GOSUB 10000:NEXT K
DA 370 X=35:Y=8:T$="W1":GOSUB 100
   00
SR 380 X=35:Y=13:T$="W2":GOSUB 10
   000
WA 390 X=35:Y=18:T$="W3":GOSUB 10
   000
II 410 T=(156-36)/(30-0)
ZL 420 XL=96
BG 430 XR=XL+32
QC 440 FOR Q=1 TO 4
PN 450 YB=156
DS 500 A=0
KU 510 FOR D=1 TO 3
ZM 520 READ S
BM 540 S=S/10^6
UL 550 YT=INT((30-S)*T+36+0.5)
PA 580 YT=YT-A
KX 590 PLOT XL,YT:DRAWTO XL,YB
SI 600 PLOT XR,YT:DRAWTO XR;YB
IO 610 PLOT XL,YT:DRAWTO XR,YT
UQ 630 FOR X1=XR TO XL STEP -D*3
II 640 X=X1
TQ 650 Y=YT
TJ 660 PLOT X,Y
YD 670 Y=Y+1
YJ 680 X=X-1
QT 690 IF Y<=YB AND X>XL THEN 660
GK 700 NEXT X1
QP 720 FOR Y1=YT TO YB STEP D*3
IT 730 Y=Y1
SN 740 X=XR
TI 750 PLOT X,Y
YC 760 Y=Y+1
YI 770 X=X-1
QR 780 IF Y<=YB AND X>XL THEN 750
HM 790 NEXT Y1
CX 810 A=156-YT
DT 820 YB=YT-1
EG 830 NEXT D
DH 840 XL=XL+48
BO 850 XR=XL+32
JO 870 NEXT Q
MZ 880 DATA 7000000,11000000,4000
   000
XB 890 DATA 8800000,10500000,7000
   000
VN 900 DATA 4000000,12000000,8500
   000
GZ 910 DATA 7000000,11333000,1050
   0000
PQ 920 GOTO 920
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
GP 10025 IF E3=0 THEN 10060
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
   TURN
```

W wykresach kolorowych i czarno-białych można stosować różnorodne warianty kreskowania. Podobnie, jak przy wyborze kolorów, sposób kreskowania należy wybrać w taki sposób aby nie pogorszyć wyglądu rysunku. Jeśli kreskowane są sąsiednie obszary, najbardziej efektywne jest stopniowe zwiększanie kroku kreskowania. Zgodnie z tym zaleceniem tworzony jest histogram za pomocą programu nr 9.

Wykresy punktowe, liniowe lub słupkowe umożliwiają przedstawienie zależności między poszczególnymi wartościami. Niekiedy jednak chcemy pokazać, jakie są relacje wartości zarówno między sobą, jak i względem ich wartości sumarycznej, czyli zależności względne. Można to zrobić za pomocą wykresu słupkowego, podobnie jak w programie nr 9. Jednak znacznie wygodniejsze są diagramy kołowe, szczególnie do przedstawiania informacji o zależnościach procentowych. Poszczególne wartości przedstawiane są jako wycinki koła (sektory). Pole powierzchni takiego wycinka tak się ma do pola powierzchni koła, jak przedstawiana wartość do sumy prezentowanych wartości. Istnieje jednak pewne ograniczenie. Aby diagram był poglądowy, nie powinien mieć więcej niż 6–8 sektorów.

Podczas tworzenia diagramów kołowych podstawowym składnikiem programu jest podprogram kreślenia okręgu. Jego szybkość działania decyduje o szybkości kreś- ▶

lenia diagramu. W programie nr 10 wykorzystano jedno z wielu możliwych rozwiązań opisanych w „IKS-ie" nr 5 z 1987 roku.

Odrębnym zadaniem jest pozycjonowanie tekstu nazwy sektora na diagramie. Realizuje się to poprzez wyznaczenie dwusiecznej kąta każdego sektora. Jeśli diagram umieszczony jest w prawej części koła, początek nazwy będzie umieszczony na dwusiecznej, w odległości czterech kroków rastra od okręgu. Jeśli sektor leży w lewej części diagramu, jego nazwa kończy się na dwusiecznej. Początkową pozycję wyświetlania nazwy określa się za pomocą operacji przekształcania współrzędnej punktu w numer pozycji znaku. W tym celu współrzędne X i Y dzieli się przez liczbę elementów rastra przypadających odpowiednio w poziomie i pionie na 1 znak

```
QO 1 REM *****************
IB 2 REM *               *
MQ 3 REM * Prosram nr 10 *
ID 4 REM *               *
QS 5 REM *****************
NL 6 REM
LQ 20 GRAPHICS 0
QX 30 DIM N$(100),V(8),T$(40),X$(
   40),D(8)
BP 40 XM=320:YM=192
LH 100 C1=40
EE 110 PC=XM/C1
GK 150 PR=8
CQ 160 ?
SG 170 ? "PODAJ WSPOLRZEDNE SRODK
   A WYKRESU"
IF 180 ? "X=";:INPUT XC
JI 190 ? "Y=";:INPUT YC
VV 200 ? "PODAJ PROMIEN WYKRESU "
   ;:INPUT R
NE 210 IF XC+R>XM OR XC-R<0 OR YC
   +R>YM OR YC-R<0 THEN 830
CJ 220 ?
TG 230 ? "PODAJ ILOSC DANYCH (MAX
   8)"
ON 240 INPUT N
OL 250 ? "PODAJ NAZWE WYKRESU"
HV 260 INPUT T$
SV 270 ? "PODAJ NAZWE I WARTOSC "
HW 280 T=0
RQ 290 L=1:REM WPROWADZANIE NAZW
   I WARTOSCI DANYCH
DR 300 FOR K=1 TO N
HA 310 ? "NAZWA ";K;:INPUT X$:? "
   WARTOSC ";:INPUT V:N$(L)=X$:D(
   K)=LEN(X$):X$="":V(K)=V
FH 320 T=T+V(K):L=L+D(K)
GT 330 NEXT K
UP 350 GRAPHICS 8+16:COLOR 1
CL 360 X=INT(C1/2-0.5*LEN(T$)):Y=
   1:REM CENTROWANIE TYTULU
AC 370 GOSUB 10000:REM WYPISANIE
   TYTULU
GK 380 GOSUB 11000:REM KRESLENIE
   OKREGU
CL 390 YA=1
DW 400 B=0
HF 410 S=0
GD 420 RE=360*3.14159/180
CO 425 LL=1
DY 430 FOR K=1 TO N
WB 460 S=S+V(K)
JW 490 A=RE*S/T
SI 500 XP=XC+R*COS(A)
QL 510 YP=YC+R*SIN(A)*YA
PQ 520 PLOT XC,YC:DRAWTO XP,YP
MW 560 AC=B+(A-B)/2
ZP 570 XL=XC+(R+4)*COS(AC)
KE 580 YL=YC+(R+4)*SIN(AC)*YA
OY 590 REM (XL,YL) JEST PUNKTEM
XO 600 REM OKRESLAJACYM START NAZ
   WY JESLI JEST ONA Z PRAWEJ STR
   ONY OKREGU
OY 610 REM KONIEC NAZWY JESLI Z L
   EWEJ
WV 620 REM SRODEK NAZWY JESLI NA
   DOLE LUB GORZE OKREGU
HN 630 REM PUNKT POCZATKOWY
XX 640 IF XL>XC+10 THEN 740
BY 650 REM PUNKT KONCOWY
VQ 660 IF XL<XC-10 THEN 710
WN 670 REM PUNKT SRODKOWY
TA 690 XL=XL-D(K)/2*PC
PM 700 GOTO 740
QU 710 REM
YE 730 XL=XL-D(K)*PC
DZ 740 REM KONWERSJA PUNKTU GR.8
   NA POZYCJE ZNAKU
PS 760 RO=INT(YL/PR+0.5)
DF 770 CO=INT(XL/PC+0.5)
DE 780 X=CO:Y=RO:T$=N$(LL,LL-1+D(
   K)):LL=LL+D(K)
SY 790 GOSUB 10000
IP 800 B=A
GU 810 NEXT K
PF 820 GOTO 820
RE 830 GRAPHICS 0:? "ZLE WSPOLRZE
   DNE !!":FOR I=1 TO 1000:NEXT I
   :RUN
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
GP 10025 IF E3=0 THEN 10060
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
ST 10070 START=START+1:NEXT E1:RE
   TURN
UE 10999 REM PODPROGRAM KRESLENIA
    OKREGU
HO 11000 FI=0:Y1=0:X1=R
LQ 11010 FIY=FI+Y1+Y1+1
HO 11020 FIXY=FIY-X1-X1+1
BE 11030 PLOT XC+X1,YC+Y1:PLOT XC
   -X1,YC+Y1
FS 11040 PLOT XC+X1,YC-Y1:PLOT XC
   -X1,YC-Y1
BA 11050 PLOT XC+Y1,YC+X1:PLOT XC
   -Y1,YC+X1
FO 11060 PLOT XC+Y1,YC-X1:PLOT XC
   -Y1,YC-X1
HO 11070 FI=FIY:Y1=Y1+1
JY 11080 IF ABS(FIXY)<ABS(FIY) TH
   EN FI=FIXY:X1=X1-1
JS 11090 IF X1>=Y1 THEN 11010
DC 11100 RETURN
```



W celu wyróżnienia jednego lub kilku segmentów diagramu kołowego, przemieszcza się je na pewną odległość od środka, wzdłuż promienia. Tworzenie takiego diagramu, wykorzystane w programie nr 11, opiera się na metodzie pozycjonowania napisów na diagramach kołowych (opisanej powyżej). Najpierw określamy granice kątowe wydzielanego sektora. Następnie obliczamy kąt dla promienia dzielącego ten sektor na połowę. Określamy nowe położenie środka wydzielonego sektora przemieszczonego wzdłuż tego promienia względem środka diagramu. W końcu kreślimy wysunięty sektor. W programie nr 11 wydzielany wycinek koła przesuwany jest o 1/5 promienia okręgu.

```
QO 1 REM *****************
IB 2 REM *               *
NL 3 REM * Prosram nr 11 *
ID 4 REM *               *
QS 5 REM *****************
NL 6 REM
LR 30 GRAPHICS 0
SA 40 DIM N$(100),V(8),T$(40),D(8
   )
BS 70 XM=320:YM=192
CS 120 PC=8
GM 160 PR=8
CS 170 ?
QW 180 ? "PODAJ WSPOLRZEDNE SRODK
   A WYKRESU "
BX 190 ? "X =";:INPUT XN:? "Y =";
   :INPUT YN
MJ 200 ? "PODAJ PROMIEN OKREGU ";
   :INPUT R
VZ 220 IF XN+R+R/5>XM OR XN-R-R/5
   <0 OR YN+R+R/5>YM OR YN-R-R/5<
   0 THEN 1020
CL 230 ?
MZ 240 REM WPROWADZANIE DANYCH
BZ 250 ? "PODAJ ILOSC DANYCH (MAX
   =8)"
OR 260 INPUT N
SV 270 ? "PODAJ NAZWE I WARTOSC "
SF 280 T=0:L=1
DR 300 FOR K=1 TO N
PZ 310 ? "NAZWA ";K;:INPUT T$:? "
   WARTOSC ";:INPUT V:V(K)=V:D(K)
   =LEN(T$):N$(L)=T$:L=L+D(K):T$=
   ""
WE 320 T=T+V(K)
GT 330 NEXT K
XF 340 ? "KTORA WARTOSC WYROZNIC
   (1 -";N;")";
KV 350 INPUT E
HM 360 IF E<1 OR E>N THEN 340
OI 370 REM KONSTRUKCJA WYKRESU
AG 390 GRAPHICS 8+16:COLOR 1:LL=1
BU 400 YA=1
DY 410 B=0
HH 420 S=0
OE 430 RE=2*3.14159
ER 440 FOR K=1 TO N
```

```
PL 450 XC=XN
QR 460 YC=YN
WH 490 S=S+V(K)
JJ 520 A=RE*S/T
CE 530 IF K<>E THEN 660
MS 540 AC=B+(A-B)/2
TO 550 XE=XC+R/5*COS(AC)
OR 560 YE=YC+R/5*SIN(AC)*YA
UF 570 XP=XE+R*COS(B)
SI 580 YP=YE+R*SIN(B)*YA
OJ 590 TRAP 600:PLOT XE,YE:DRAWTO
    XP,YP
SJ 600 XP=XC+R*COS(A)
QM 610 YP=YC+R*SIN(A)*YA
LV 620 TRAP 630:PLOT XC,YC:DRAWTO
    XP,YP
MG 630 XC=XE
MV 640 YC=YE
MN 660 FOR A1=B TO A STEP 1/R
CK 670 XP=XC+R*COS(A1)
IB 680 YP=YC+R*SIN(A1)*YA
PX 690 TRAP 700:PLOT XP,YP
XO 700 NEXT A1
NA 710 TRAP 750:PLOT XC,YC:DRAWTO
    XP,YP
MW 750 AC=B+(A-B)/2
ZP 760 XL=XC+(R+4)*COS(AC)
KE 770 YL=YC+(R+4)*SIN(AC)*YA
Y8 830 IF XL>XC+10 THEN 930
WL 850 IF XL<XC-10 THEN 900
TA 880 XL=XL-D(K)/2*PC
QO 890 GOTO 930
QU 900 REM
YE 920 XL=XL-D(K)*PC
RA 930 REM
DR 950 RO=INT(YL/PR)
RE 960 CO=INT(XL/PC)
DE 970 X=CO:Y=RO:T$=N$(LL,LL-1+D(
    K)):LL=LL+D(K)
SY 980 GOSUB 10000
JI 990 B=A
FL 1000 NEXT K
NE 1010 GOTO 1010
XY 1020 GRAPHICS 0:? "ZLA WARTOSC
    WSPOLRZEDNYCH":FOR I=0 TO 100
    0:NEXT I:RUN
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
    88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
    SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
    31 AND E3<96)
GP 10025 IF E3=0 THEN 10060
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
    +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STHR
    T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
    TURN
```

Poszczególne segmenty diagramu kołowego mogą być zapełniane kolorem lub kreskowane, zgodnie z zasadami jak dla histogramów. Kreskowanie powinno być proste, a jego dokładność winna stopniowo wzrastać od sektora do sektora.

Proszę zwrócić uwagę, że w większości przedstawianych programów, wyświetlane dane umieszczone były na stałe w programie, w liniach DATA. Nic nie stoi na przeszkodzie, aby czytelnicy zmodyfikowali je tak, żeby dane były wprowadzane bądź z klawiatury, bądź ze zbioru przechowywanego w pamięci zewnętrznej.

W następnych odcinkach tego cyklu zaprezentujemy „prawdziwy" program do kreślenia różnorodnych wykresów, głównie dla tych czytelników, którzy nie mają ochoty na bawienie się w tworzenie własnych programów i wolą skorzystać z gotowych.

Tomasz MROWIEC
Ludwik PIELA

PROGRAM 46

C-64

Słownik

Po uruchomieniu programu i przedstawieniu się jesteśmy pytani o polski odpowiednik słowa angielskiego lub odwrotnie. Wystawiana przez komputer ocena jest surowa, ale zawsze obiektywna. Program może służyć do nauki języka angielskiego, ale nie tylko. Wszystko zależy od tego, jakie wyrazy wpiszemy w liniach danych. Zmniejszenie lub zwiększenie pojemności naszego słownika wymaga tylko nielicznych zmian w programie. Zaprezentowane w programie efekty dźwiękowe również mogą się podobać i skutecznie zachęcać do nauki języków obcych.

```
10 REM SLOWNIK
30 AT$=CHR$(17):FORT=1TO6:AT$=AT$+AT$:NE
XT T:AT$=CHR$(19)+AT$
35 POKE53281,1:POKE53280,14:GOSUB 10000
40 SC=0
50 ON=0
70 DIME$(20)
80 DIMF$(20)
100 FOR J=1TO20
110 READ E$(J)
120 READ F$(J)
130 NEXT J
135 PRINT CHR$(147)
140 INPUT"      JAK MASZ NA IMIE?';N$
142 N$=LEFT$(N$,8)
150 FOR DL=1TO1000:NEXT DL
154 PRINT CHR$(147);CHR$(144)
160 PRINT LEFT$(AT$,2)SPC(5);'WYNIK: ';S
C
170 PRINT LEFT$(AT$,2)SPC(23);'PYTANIE:
';ON
180 R=INT(RND(1)*20)+1

190 IF RND(1)).5 THEN GOTO 330
200 PRINT LEFT$(AT$,7)SPC(8);'JAK JEST P
O ANGIELSKU';
202 PRINT LEFT$(AT$,8)SPC(8);E$(R);'?
'
205 PRINT LEFT$(AT$,10)SPC(13);
210 INPUT A$
220 ON=ON+1
230 IF LEN(A$)(15 THENA$=A$+' ':GOTO 230

240 IF LEFT$(A$,LEN(F$(R)))=F$(R)THEN GO
TO 280
250 FOR T=80 TO 0 STEP-1:FOR TT=1TO3:POK
ES0+1,TT*T:NEXT TT,T
260 PRINT LEFT$(AT$,13)SPC(10);'NIE ';N$
;' POWINNO BYC'
262 PRINTLEFT$(AT$,15)SPC(14);CHR$(28);F
$(R)
270 GOTO 150
280 FOR T=10 TO 100:FOR TT=1TO2:POKE S0+
1,T*TT:NEXTTT,T:POKE S0+1,0
290 PRINTLEFT$(AT$,13)SPC(15);'DOBRZE!'
300 SC=SC+1
310 IF SC=20 THEN GOTO 410
320 GOTO 150
330 PRINT LEFT$(AT$,7)SPC(8);'JAK JEST P
O POLSKU'
332 PRINTLEFT$(AT$,8)SPC(8);F$(R);'?
'
335 PRINTLEFT$(AT$,10)SPC(13);
340 INPUT A$
350 ON=ON+1

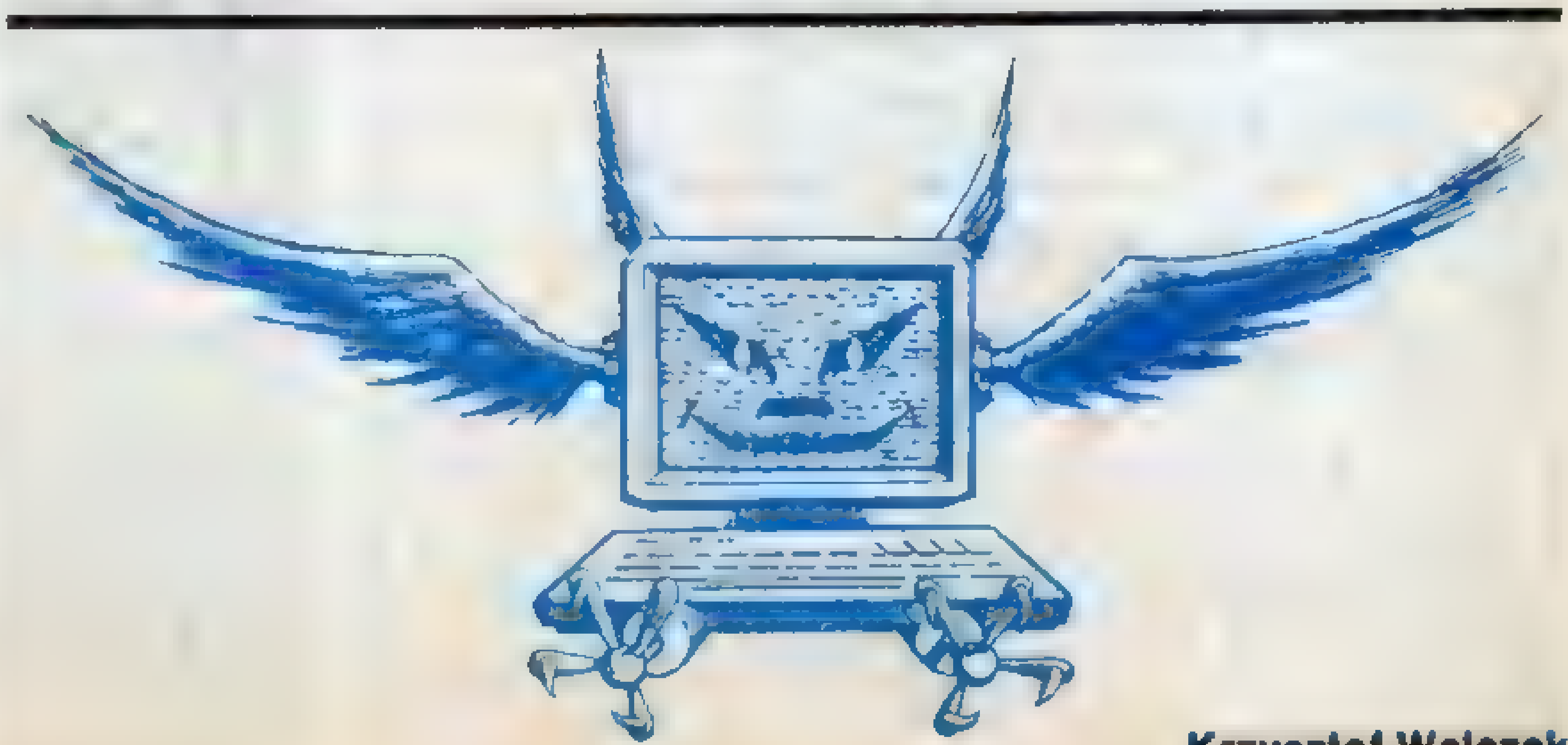
360 IF LEN(A$)(10 THEN A$=A$+' ':GOTO 36
0
370 IF LEFT$(A$,LEN(E$(R)))=E$(R) THEN G
OTO 280
380 FOR T=200 TO0 STEP -1:POKE S0+1,T:NE
XT T

390 PRINTLEFT$(AT$,13)SPC(10);'NIE ';N$;
' POWINNO BYC '
392 PRINTLEFT$(AT$,15)SPC(15);CHR$(28);E
$(R)
400 GOTO 150
410 POKE 53281,0
420 POKE 53280,0
430 PRINTCHR$(147);CHR$(158)
440 PRINTLEFT$(AT$,8)SPC(6);'GRATULACJE!
 ODPOWIEDZIALES'
450 PRINT LEFT$(AT$,12)SPC(6);' NA 20 PY
TAN'
460 ,PRINTLEFT$(AT$,16)SPC(6);'      DOBRZE
, '
470 :

480 :
490 :
500 FOR DL=1TO1000:NEXT DL
510 POKE 53281,7
520 POKE 53280,7
530 PRINT CHR$(147)
540 PRINT CHR$(144):END
600 DATA'DRZWI','DOOR'
610 DATA'LAMPA','LAMP'
620 DATA'DOM','HOUSE'
630 DATA'PIES','DOG'
640 DATA'OGROD','GARDEN'
650 DATA'KOT','CAT'
660 DATA'KURTKA','JACKET'
670 DATA'PROSZEK','POWDER'
680 DATA'KAPELUSZ','HAT'
690 DATA'ROWER','BICYCLE'
700 DATA'KAWA','COFFE'
710 DATA'STACJA','STATION'
720 DATA'CHLEB','BREAD'

730 DATA'MLEKO','MILK'
740 DATA'KUBEK','CUP'
750 DATA'KOBIETA','WOMAN'
760 DATA'DROGA','ROAD'
770 DATA'MAPA','MAP'
780 DATA'MORZE','SEA'
790 DATA'BIURKO','DESK'
10000 REM DZWIEK
10010 S0=54272
10020 FOR T=0TO28:POKES0+T,0:NEXT
10040 POKES0+5,9
10050 POKE S0+6,240
10060 POKE S0+24,15
10070 POKE S0+4,17
10080 RETURN
```

T. CISEK



Krzysztof Walczak

Lilavati — cz. 4

Droga do portu

W kolejnej części cyklu poświęconego rozwiązywaniu anegdot matematycznych chcieliśmy zaprezentować zmodyfikowaną historyjkę o nr. 54 w książce „Lilavati" S. Jeleńskiego
Oto treść zadania

```
Wyjscie na pelne morze z portu wojenne-
go jest bardzo waskie.
3 okrety wojenne (1,2,3) maja za zadanie
opuscic port.
Natomiast inne 3 okrety (4,5,6) w tym
samym czasie musza wplynac do portu po
zakonczonych cwiczeniach.
Wyjscie jest takiej szerokosci ,ze dwa
okrety nie moga sie na nim minac ,lecz
po jednej stronie znajduje sie mala
zatoka ,w ktorej moze sie zmiescic jeden
okret.
Czy okrety wykonaja postawione zadanie?


NACISNIJ 'SPACJE'
```

Tak wygląda sytuacja początkowa.

```
PODAJ NUMER OKRETU: 3
```



Zasady gry:

1. Podawany nr okrętu musi być o wartości 1—6
2. Ruchem okrętu o wprowadzonym numerze steruje się za pomocą klawiszy ←, →, ↑, ↓.
3. Jeżeli chcemy zmienić nr okrętu, to należy nacisnąć w dowolnej chwili klawisz ENTER
4. Okręty poruszają się tylko po obszarze niebieskim
5. Brak możliwości wykonania ruchu jest sygnalizowany dźwiękiem
6. Gra kończy się sukcesem po wykonaniu zadania.

Segment sterujący przebiegami gry ma następującą postać

```
10 MODE 1:INK 0.11:INK 1.0:INK 2.26:INK 3.1:PAPER 0:BORDER 11:CLS:PEN 1
20 LOCATE 1.5
30 PRINT "Wyjscie na pelne morze z portu wojenne- go jest bardzo waski
e.                    3 okrety wojenne (1.2.3) maja za zadaniem op
uscic port.                              Natomiast inne 3 okrety (4.5.6)
w tym   samym czasie musza wplynac do portu po"
40 PRINT "zakonczonych cwiczeniach.                   Wyjscie jest takiej
szerokosci ,ze dwa  okrety nie moga sie na nim minac ,lecz  po
 jednej stronie znajduje sie mala      zatoka ,w ktorej moze sie zmiesc
ic jedenokret."
50 PRINT "Czy okrety wykonaja postawione zadanie?"
60 PEN 2:LOCATE 1.23:PRINT "NACISNIJ 'SPACJE'"
70 IF INKEY$="" THEN GOTO 70
80 MODE 1:INK 0.11:INK 1.0:INK 2.26:INK 3.1:PAPER 0:BORDER 0:CLS
90 GOSUB 1230
100 GRAPHICS PEN 1:PLOT 0.128:DRAWR 640.0:PLOT 0.200:DRAWR 240.0:DRAWR
 32.48:DRAWR 48.0:DRAWR 32.-48:DRAWR 360.0
110 MOVE 10.10:FILL 1:MOVE 10.300:FILL 1
120 PEN 2:PAPER 1:LOCATE 1.10:PRINT  CHR$(242);" PORT"
130 LOCATE 34.10:PRINT  "MORZE ";CHR$(243):PAPER 0
140 DIM wsp(6.2)
150 FOR a=1 TO 3
160 wsp(a.1)=7+(a-1)*3
170 wsp(a.2)=13
180 NEXT a
190 FOR a=4 TO 6
200 wsp(a.1)=26+(a-4)*3
210 wsp(a.2)=13
220 NEXT a
230 FOR K=1 TO 6
240 GOSUB 1210
250 NEXT
260 WINDOW £1.1,40,1,6:PAPER £1,1
270 CLS £1:PEN £1,2:LOCATE £1,2,2:INPUT £1,"PODAJ NUMER OKRETU: ",N
280 IF N>0 AND N<7 THEN GOTO 300
290 SOUND 1,358:SOUND 1,458:CLS £1:PRINT £1,"BLEDNY NUMER OKRETU":FOR
T=1 TO 2000:NEXT:GOTO 270
300 CLS £1:PRINT £1,"STERUJ RUCHEM OKRETU  NR ";USING "£";N;:PRINT £1,
"ZA POMOCA KLAWISZY:";CHR$(240);",";CHR$(241);",";CHR$(242);",
";CHR$(243):LOCATE £1,1,4:PRINT £1,"JEZELI CHCESZ ZMIENIC NR OKRETU NA
CISNIJKLAWISZ 'ENTER'"
310 A$=INKEY$
320 IF A$="" THEN GOTO 310
330 IF INKEY(6)=0 THEN GOTO 270
340 P=ASC(A$)
350 IF p<240 OR p>243 THEN GOTO 310
360 ON P-239 GOSUB 510,720,950,1080
370 FOR G=1 TO 3
380 IF WSP(G,1)<=18 THEN GOTO 500
390 NEXT G
400 FOR G=4 TO 6
410 IF WSP(G,1)>=18 THEN GOTO 500
420 NEXT G
430 CLS £1:PRINT £1,"BRAWO! UDALO SIE!"
440 FOR A=0 TO 26
450 BORDER A
460 FOR T=1 TO 50:NEXT T
470 NEXT A
480 IF INKEY$="" THEN GOTO 440
490 END
500 GOTO 310
```

Oprócz segmentu sterującego inne czynności zorganizowano w postaci podprogramów, których rola jest następująca:

1. Kontrola, czy ruch w górę jest możliwy (instrukcje 510—710)

```
510 IF wsp(n,1)<>18 THEN SOUND 1,358:RETURN
520 IF wsp(n,2)=10 THEN SOUND 1,358:RETURN
530 FOR I=1 TO 6
540 IF WSP(N,2)-1=WSP(I,2)+2 THEN GOTO 660
550 NEXT I
560 IF r1=1 THEN GOTO 580
570 GOSUB 1330
580 X=WSP(N,1):Y=WSP(N,2)+1:GOSUB 1840
590 POKE &7F74,W2:POKE &7F75,W1
600 FOR b=1 TO 3
610 FOR C=1 TO 8
620 POKE &7F02,C:CALL &7F00
630 FOR T=1 TO 2:NEXT:FRAME:NEXT:NEXT
640 WSP(N,2)=WSP(N,2)-3
650 RETURN
660 FOR R=0 TO 2
670 FOR J=0 TO 2
680 IF WSP(N,1)+R=WSP(I,1)+J THEN SOUND 1,358:RETURN
690 NEXT J
700 NEXT R
710 GOTO 550
```

2. Kontrola, czy ruch w dół jest możliwy (instrukcje 720—940)

```
720 IF wsp(n,1)<>18 OR WSP(N,2)<>10 THEN SOUND 1,358:RETURN
730 FOR I=1 TO 6
740 IF WSP(N,2)-3=WSP(I,2) THEN GOTO 880
750 IF n=1 THEN GOTO 770
760 IF WSP(I,1)=18 THEN SOUND 1,358:RETURN
770 NEXT I
780 IF r1=4 THEN GOTO 800
790 GOSUB 1580
800 X=WSP(N,1):Y=WSP(N,2)+3:GOSUB 1840
810 POKE &7F74,W2:POKE &7F75,W1+56
820 FOR b=1 TO 3
830 FOR C=1 TO 8
840 POKE &7F02,C:CALL &7F00
850 FOR T=1 TO 2:NEXT:FRAME:NEXT:NEXT
860 WSP(N,2)=WSP(N,2)+3
870 RETURN
880 FOR R=0 TO 2
890 FOR j=0 TO 2
900 IF WSP(N,1)+R=WSP(I,1)+J THEN SOUND 1,358:RETURN
910 NEXT J
920 NEXT R
930 GOTO 770
940 RETURN
```

3. Kontrola, czy ruch w lewo jest możliwy (instrukcje 950—1070).

```
950 IF WSP(N,2)<13 THEN SOUND 1,358:RETURN
960 IF WSP(N,1)=1 THEN SOUND 1,358:RETURN
970 FOR I=1 TO 6
980 IF WSP(N,1)=WSP(I,1)+3 AND WSP(I,2)=13 THEN SOUND 1,358:RETURN
990 NEXT I
1000 IF r1=3 THEN GOTO 1020
1010 GOSUB 1750
1020 x=wsp(n,1):y=wsp(n,2)+1:GOSUB 1840
1030 POKE &7F7B,w1:FOR z=w2 TO w2-1 STEP -1
1040 FRAME:POKE &7F7A,z::CALL &7F76
1050 NEXT
1060 WSP(N,1)=WSP(N,1)-1
1070 RETURN
```

4. Kontrola, czy ruch w prawo jest możliwy (instrukcje 1080—1200).

```
1080 IF WSP(N.2)<13 THEN SOUND 1.358:RETURN
1090 IF WSP(N.1)=38 THEN SOUND 1.358:RETURN
1100 FOR I=1 TO 6
1110 IF WSP(N.1)+3=WSP(I.1) AND wsp(i,2)=13 THEN SOUND 1.358:RETURN
1120 NEXT I
1130 IF r1=2 THEN GOTO 1150
1140 GOSUB 1490
1150 x=wsp(n,1)+2:y=wsp(n,2)+1:GOSUB 1840
1160 POKE &7F7B,w1:FOR z=w2+1 TO w2+2
1170 FRAME:POKE &7F7A,z::CALL &7F76
1180 NEXT
1190 WSP(N,1)=WSP(N,1)+1
1200 RETURN
```

5. Rysowanie jednego okrętu (instrukcje 1210—1220).

```
1210 FOR a=1 TO 3:PEN 3:LOCATE wsp(k,1),wsp(k,2)+a:PRINT b$(a):PEN 1:L
OCATE wsp(k,1),wsp(k,2)+2:PRINT USING "C":k:PEN 3:NEXT
1220 RETURN
```

6. Definiowanie symboli graficznych okrętu (instrukcje 1230—1320).

```
1230 SYMBOL AFTER 125
1240 SYMBOL 125.0.0.0.1.1,1.1.1
1250 SYMBOL 126.1.1.1,1.1,1,1,1
1260 SYMBOL 127,0.64.91,255.219.255.127.127
1270 SYMBOL 128.13,15,235,255,109,255.255,255
1280 SYMBOL 129.96.228.164.252.172.248.248.240
1290 b$(1)=" "+CHR$(125)+" "
1300 b$(2)=" "+CHR$(126)+" "
1310 b$(3)=CHR$(127)+CHR$(128)+CHR$(129)
1320 RETURN
```

7. 4 podprogramy (instrukcje 1330—1830) wczytywania do pamięci podprogramów ruchu w określonym kierunku w języku wewnętrznym:

```
1330 REM 'przesuwanie w gore'
1340 r1=1
1350 RESTORE 1370:FOR i=&7F00 TO &7F73
1360 READ v:POKE i.v:NEXT
1370 DATA 1.24.1.42.116.127,16,93
1380 DATA 229.124.198.56.103.17.80.0.237,82,84
1390 DATA 93.225.197.229.213.205.95.127.209.98.107
1400 DATA 34,116,127.225,193,13,40,46,6.7.84,93.124
1410 DATA 198.8.103.197.229.213
1420 DATA 205,95,127,209.225,193.13.40.26,16,236
1430 DATA 84.93.124.214.56.103.213.17,80.0.237.90
1440 DATA 209.197.229.213.205,95.127.209,225.193
1450 DATA 24.207.54.0,84.93.19.1.5.0.237.176.201
1460 DATA 1.6,0.237.176.201.124,214.8.87
1470 DATA 93.229.98,107,34,116.127.225.24.187,201
1480 RETURN
1490 REM 'przesuwanie w prawo'
1500 r1=2
1510 RESTORE 1530:FOR i=&7F76 TO &7FA1
1520 READ v:POKE i.v:NEXT
1530 DATA 1.8,3,33,39,192,84
1540 DATA 93,19.197,1,6,0.237,184.193,35,54,0
1550 DATA 13.40,6,17,5,8,25.24,234.5,40,12,124
1560 DATA 214,56,103,17.85,0,25,14,8,24,219,201
1570 RETURN
1580 REM 'przesuwanie w dol'
1590 r1=4
1600 RESTORE 1620:FOR i=&7F00 TO &7F73
1610 READ v:POKE i.v:NEXT
1620 DATA 1.24.1.42,116.127.16.93.229
1630 DATA 124.214.56,103,17,80,0,237,90.84
1640 DATA 93.225.197,229,213,205.95.127.209
1650 DATA 98,107,34,116,127.225.193,13,40,46
1660 DATA 6.7,84.93,124,214,8,103,197,229.213
1670 DATA 205,95,127,209,225,193,13,40,26,16,236
1680 DATA 84.93,124.198.56,103,213,17,80,0,237,82
1690 DATA 209.197.229.213,205.95,127
1700 DATA 209,225.193,24.207
1710 DATA 54,0,84,93,19,1.6,0.237,176,201
1720 DATA 1.6,0,237,176,201,124,198,8,87
1730 DATA 93.229,98,107,34,116,127,225,24,187,201
1740 RETURN
1750 REM 'przesuwanie w lewo'
1760 r1=3
1770 RESTORE 1790:FOR i=&7F76 TO &7FA1
1780 READ v:POKE i.v:NEXT
1790 DATA 1.8,3,33.36.192.84.93
1800 DATA 27,197,1,6,0,237,176.193,43,54,0
1810 DATA 13.40,6.17.251.7.25.24.234.5.40,12,124
1820 DATA 214,56,103,17,75.0,25,14,8,24,219,201
1830 RETURN
```

8. Przeliczanie współrzędnych okrętu na adresy ekranu, potrzebne do wymienionych w pkt. 7 podprogramów (instrukcje 1840—1880).

```
1840 A=(Y-1)*80+(x-1)*2
1850 W1=INT(a/256)
1860 W2=a-W1*256
1870 w1=192+w1
1880 RETURN
```

Życzymy przyjemnej zabawy i sukcesów w rozwiązywaniu tego problemu.

Danuta KWASIŻUR
Mieczysław SKONIECZNY

PROGRAM 47

ZX SPECTRUM

Test joystck'a

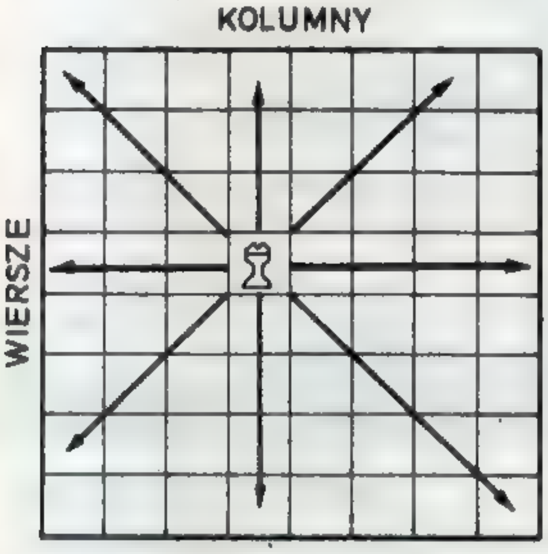
Po pewnym okresie używania joystick'a dają się zauważyć jego niedomagania. Niektóre z wychyleń przestają funkcjonować poprawnie. Czasem jesteśmy w rozterce: czy jest to wina źle załadowanego programu, czy też samej przystawki. Prezentowany test wyeliminuje te wątpliwości. Należy przyłączyć joystick, uruchomić program, a następnie poruszać dźwignią we wszystkich możliwych kierunkach. Jeśli uznacie, że test trwa zbyt długo, to wystarczy nacisnąć spację z klawiszem CONTROL i otrzymacie informacje o funkcjach, które nie działają poprawnie. Zwykle jest to wina styków, które po rozebraniu joystick'a należy odpowiednio przygiąć i przeczyścić.

Krzysztof MAMCARZ

```
10 ' ** testjoy '87
20 ' program do testowania
30 ' joystick'a
40 ' AMSTRAD CPC
50 '
60 MODE 1
70 DEFINT a-z
80 WINDOW #1,1,40,25,25
90 DIM f(8)
100 FOR a=5 TO 15 STEP 5
110 FOR b=5 TO 15 STEP 5
120 LOCATE b,a: PRINT ","
130 NEXT b,a
140 PRINT #1,"przygotuj JOYSTICK i nacis
nij cokolwiek.";
150 CLEAR INPUT: WHILE INKEY$="": WEND
160 PRINT #1, "naciskaj wszystkie poloze
nia joystick'a"
170 IF JOY(0)=1 THEN f(0)=1: LOCATE 10,5
: PRINT " "
180 IF JOY(0)=2 THEN f(1)=1: LOCATE 10,1
5: PRINT" "
190 IF JOY(0)=4 THEN f(2)=1: LOCATE 5,10
: PRINT " "
200 IF JOY(0)=5 THEN f(3)=1: LOCATE 5,5:
 PRINT " "
210 IF JOY(0)=6 THEN f(4)=1: LOCATE 5,15
: PRINT " "
220 IF JOY(0)=8 THEN f(5)=1: LOCATE 15,1
0: PRINT " "
230 IF JOY(0)=9 THEN f(6)=1: LOCATE 15,5
: PRINT " "
240 IF JOY(0)=10 THEN f(7)=1: LOCATE 15,
15: PRINT " "
250 IF JOY(0)=16 THEN f(8)=1· LOCATE 10,
10: PRINT " "
260 IF INKEY(47)=128 THEN 280
270 GOTO 170
280 ' CTRL-SPACE
290 CLS #1: CLEAR INPUT
300 f1=0: LOCATE 1,16
310 FOR a=1 TO 9
320 RESTORE 410
330 IF f(a-1)=0 THEN GOSUB 420
340 NEXT
350 IF f1=1 THEN LOCATE 1,1: PRINT "BRAK
 REAKCJI:" ELSE CLS: PRINT "JOYSTICK JES
T SPRAWNY"
360 LOCATE 20,12: PRINT "ponownie ? T/N"
370 WHILE INKEY$="": WEND
380 IF INKEY(51)>=0 THEN RUN
390 CLS :END
400 '
410 DATA "gora","dol","lewo","lewo-gora"
,"lewo-dol","prawo","prawo-gora","prawo-
dol","fire"
420 FOR b=1 TO a: READ h$:NEXT b
430 PRINT h$
440 f1=1
450 RETURN
```

Gra w ośmiu hetmanów

Zagadnienie ośmiu hetmanów polega na znalezieniu takiego ich rozmieszczenia na 64-polowej kwadratowej szachownicy, aby żaden hetman nie „bił" innego. Mówiąc dokładniej, żadne dwa hetmany nie mogą znajdować się w tej samej kolumnie, w tym samym wierszu, ani na tej samej przekątnej. Kierunki „bicia" hetmana prezentuje rys. 1.



Rys. 1. Kierunki „bicia" hetmana

Istnieje 92 rozwiązań rozmieszczenia ośmiu hetmanów.

W grze znalezienie poprawnego rozmieszczenia umożliwia przejście na kolejny z poziomów, określone jest ilością możliwych rozwiązań. Utworzenie jednego rozwiązania polega na ustawieniu w kolejnych kolumnach (początek w pierwszej) szachownicy hetmanów nie „bijących się" wzajemnie. Ustawienie zadawane jest poprzez numer wiersza. Numery kolumn ustawiane są automatycznie. Każde poprawne ustawienie hetmana jest rysowane na ekranie oraz zwiększa liczbę punktów o 25.

Próba błędnego ustawienia — hetman jest „bity" przez już ustawionego hetmana — powoduje utratę 25 punktów. Tak więc na jednym poziomie można uzyskać 200 punktów. Stracić można więcej, gdyż można doprowadzić do sytuacji, w której niemożliwe jest ustawienie ośmiu hetmanów. Wznowienie gry następuje wtedy na tym samym poziomie. Grę kończy ustawienie 92 różnych rozmieszczeń ośmiu hetmanów, lub zniechęcenie gracza, któremu po prawidłowym lub błędnym rozmieszczeniu dostarczana jest taka możliwość. Kolorowa grafika oraz efekty dźwiękowe powinny sprzyjać znajdowaniu prawidłowych rozmieszczeń hetmanów na szachownicy.

Gra ośmiu hetmanów napisana została w języku PASCAL oraz uruchomiona na komputerze AMSTRAD CPC 6128. Program źródłowy gry tworzą dwa zbiory. Zbiór główny zawiera procedury graficzne oraz moduł sterujący całego programu, z którego wywoływane są dwie procedury umieszczone w drugim zbiorze źródłowym gry. Jedna z tych procedur obrazuje czołówkę gry. Druga zawiera demonstrację gry oraz samą grę. Z uwagi na obszerność tych procedur oraz ich wzajemne wykluczanie się, zostały one zaimplementowane w postaci procedur nakładkowych.

URUCHOMIENIE PROGRAMU

Po przepisaniu programu źródłowego do dwóch zbiorów źródłowych o nazwach OSIEMHET. PAS i PROCPOM. PAS należy wywołać opcję kompilatora języka TURBO PASCAL — komenda O. Następnie użyć komendy C (kompilacja do zbioru) oraz Q (koniec opcji). Kolejnym krokiem jest wywołanie komendy M oraz wpisanie nazwy zbioru OSIEMHET. PAS. Kompilację uruchamia komenda C. W wyniku jej realizacji utworzone zostaną na dyskietce dwa zbiory o nazwie OSIEMHET. COM i OSIEMHET. 000. Od tej chwili uruchomienie gry jest możliwe pod systemem CP/M PLUS.

Życzymy satysfakcji w osiąganiu kolejnych rozwiązań, z którymi kłopoty miał nawet C. F. GAUSS.

Andrzej CETERA

```
PROGRAM osiemhetmanow;
{          PROGRAM GLOWNY              }
{$a-}
{ ################################## }
{       GRAFIKA TURBO PASCALA          }
{ ################################## }

const CPM3 = true;

const tagflag:boolean = false;
      tagaddr:integer = 0;
      indscr :integer = 0;
      fct             = 0.017453292520;
      xerror :real    = 0.0;
      yerror :real    = 0.0;
      getmode:byte    = 2;
      getgrafmode:byte= 0;

var   itg:integer;
      btg:byte;
      rd:array[1..13] of byte;
      re:real;

procedure initgrafik;
begin
   write(#$1b,'0',#$1b,'y');
end;

procedure leavegrafik;
begin
   write(#$1b,'1',#$1b,'x');
end;

procedure gotoxy(x,y:integer);
begin
   write(#$1b,'Y',chr(y+31),chr(x+31));
end;

procedure
sound(g:byte;h,i:integer;j,k,l,m:byte);
var sd:array[0..8] of byte;
begin
   sd[0] := g;
   move(h,sd[3],2);
   move(i,sd[7],2);
   sd[1] := k;
   sd[2] := l;
   sd[5] := m;
   sd[6] := j;
   inline($21/sd/$cd/$5a/$fc/$aa/$bc);
end;

procedure soundreset;
begin
  inline($cd/$5a/$fc/$a7/$bc);
end;
procedure draw(x,y:integer);
begin
inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/$F6/
       $BB);
end;
procedure drawr(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $F9/$BB);
end;

procedure grafmove(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $C0/$BB);
end;

procedure grafmover(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $c3/$BB);
end;

procedure plot(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $EA/$BB);
end;

procedure
grafwindow(x1,y1,x2,y2:integer);
begin
  inline($2A/x2/$EB/$2A/x1/$cd/$5a/$fc/
         $CF/$BB/
         $2A/y2/$EB/$2A/y1/$cd/$5a/$fc/
         $D2/$BB);
end;

procedure grafpen(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$DE/$BB);
end;

procedure grafpaper(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$E4/$BB);
end;

procedure clg(fs:byte);
begin
  inline($cd/$5a/$fc/$DB/$BB);
  grafpaper(fs);
end;

procedure ink(fs,f1,f2:byte);
begin
  inline($3a/f1/$47/$3a/f2/$4f/$3a/fs/
         $cd/$5a/$fc/
         $32/$bc);
end;

procedure paper(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$96/$BB);
end;

procedure pen(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$90/$BB);
end;

procedure mode(md:byte);
begin
  if md in [0..2] then
  begin
    getmode := md;
    inline($3a/md/$cd/$5a/$fc/$0e/$bc);
  end;
end;

procedure border(v1,v2:byte);
begin
  inline($3a/v1/$47/$3a/v2/$4f/$cd/$5a/
         $fc/$38/$bc);
end;

procedure grafout(ch:char);
begin
  inline($3A/ch/$cd/$5a/$fc/$FC/$BB);
end;

procedure tag;
begin
  if not tagflag then
  begin
    tagaddr := conoutptr;
    tagflag := true;
    conoutptr := addr(grafout);
  end;
end;

procedure tagoff;
begin
  if tagflag then
  begin
    conoutptr := tagaddr;
    tagflag := false;
  end;
end;
{$iprocpom.pas}
{######################################}
{             MODUL STERUJACY          }
{######################################}
begin
crtinit;
initgrafik;
czolowka;
g8h;
mode(2);
grafwindow(1,1,1,1);
gotoxy(1,1);
ink(0,26,26);
ink(1,1,1);
paper(1); pen(0);
border(1,1);
leavegrafik;
crtexit;
end.
```

WYDRUK ZBIORU PROCPOM.PAS

```
{****************************************}
{  PROCEDURA CZOLOWKA                    }
{****************************************}
overlay
procedure czolowka;

var wsp:array [1..16,1..3] of real;
    wsp1:array [1..16,1..3] of real;
    nwsp:array [1..16,1..3] of real;
     mp:array [1..3,1..3] of real;
     ms:array [1..3,1..3] of real;
     pw:array [1..16,1..2] of real;
     fla,licz,k,nh:integer;
    sx,sy,rx,ry,xp,yp,krok:real;

procedure tytul;
var t:string[18]; j:integer;
begin
t:=' B H E T M A N O W';
grafpen(9); tag;
plot(40,375);
for j:=1 to 18 do
grafout(t[j]);
tagoff;
end;

procedure podp;
var t:string[15]; j:integer;
begin
t:='wcisnij klawisz';
grafpen(10); tag;
plot(100,20);
for j:=1 to 15 do
grafout(t[j]);
tagoff;
end;
{****************************************}
{  PROCEDURY PRZESUWANIA I WYPELNIANIA   }
{****************************************}

procedure  mnoz(i:integer);
var n,k,m:integer;
begin
for n:=1 to i do
for k:=1 to 3 do
for m:=1 to 3 do
  nwsp[n,k]:=nwsp[n,k]+wsp[n,m]*ms[m,k];
end;

procedure  mnoz1(i:integer);
var n,k,m:integer;
begin
for n:=1 to i do
for k:=1 to 3 do
for m:=1 to 3 do
  nwsp[n,k]:=nwsp[n,k]+wsp[n,m]*mp[m,k];
end;
procedure  ustms(i:integer);
var n,k:integer;
begin
ms[1,2]:=0.0;
ms[1,3]:=0.0;
ms[1,1]:=sx;
ms[2,1]:=0.0;
ms[2,2]:=sy;
ms[2,3]:=0.0;
ms[3,1]:=-rx*sx+rx;
ms[3,2]:=-ry*sy+ry;
ms[3,3]:=1.0;
for k:=1 to i do
for n:=1 to 3 do
 nwsp[k,n]:=0.0;
end;
procedure  ustmp(i:integer; p,q:real);
var n,k:integer;
begin
mp[1,2]:=0.0;
mp[1,3]:=0.0;
mp[1,1]:=1.0;
mp[2,1]:=0.0;
mp[2,2]:=1.0;
mp[2,3]:=0.0;
mp[3,1]:=p;
mp[3,2]:=q;
mp[3,3]:=1.0;

for k:=1 to i do
for n:=1 to 3 do
 nwsp[k,n]:=0.0;
end;

procedure rysmo(i:integer);
var n:integer;
begin
plot(round(nwsp[1,1]),round(nwsp[1,2]));
for n:=2 to i do
begin
draw(round(nwsp[n,1]),round(nwsp[n,2]));
sound(1,round(nwsp[n,1])-100,2,2,0,0,0);
sound(2,round(nwsp[n,2]),2,2,0,0,0);
end;
end;

{****************************************}
{       INICJACJA DANYCH                 }
{****************************************}

procedure inicda;
begin
fla:=0;
pw[1,1]:=115.0;
pw[1,2]:=0.0;
pw[2,1]:=115.0;
pw[2,2]:=5.0;
pw[3,1]:=120.0;
pw[3,2]:=5.0;
pw[4,1]:=125.0;
pw[4,2]:=20.0;
pw[5,1]:=120.0;
pw[5,2]:=30.0;
pw[6,1]:=120.0;
pw[6,2]:=35.0;
pw[7,1]:=125.0;
pw[7,2]:=40.0;
pw[8,1]:=130.0;
pw[8,2]:=35.0;
pw[9,1]:=135.0;
pw[9,2]:=40.0;
pw[10,1]:=140.0;
pw[10,2]:=35.0;
pw[11,1]:=140.0;
pw[11,2]:=30.0;
pw[12,1]:=135.0;
pw[12,2]:=20.0;
pw[13,1]:=140.0;
pw[13,2]:=5.0;
pw[14,1]:=145.0;
pw[14,2]:=5.0;
pw[15,1]:=145.0;
pw[15,2]:=0.0;
pw[16,1]:=115.0;
pw[16,2]:=0.0;
mode(0);
grafwindow(0,0,640,400);
ink(0,0,0);
ink(1,6,6);
ink(2,9,9);
ink(3,26,26);
ink(4,7,7);
ink(5,16,16);
ink(6,11,11);
ink(7,15,15);
ink(8,23,23);
ink(9,12,12);
ink(10,25,25);
grafpaper(0); border(0,0);
clg(0);
end;

procedure pdan(nr:integer);
var i:integer;
begin
for i:=1 to 16 do
begin
wsp[i,1]:=pw[i,1];
wsp[i,2]:=pw[i,2] +nr*40;
wsp[i,3]:=1;
end;
xp:=130.0;
yp:=20.0+nr*40.0;
krok:=40.0;
rx:=xp;
ry:=yp;
licz:=16;
end;

procedure pdan1(i:integer);
var k,n:integer;
begin
for k:=1 to i do
for n:=1 to 3 do
  wsp1[k,n]:=wsp[k,n];
end;
{****************************************}
{      PROCEDURY POMOCNICZE              }
{****************************************}

procedure przep1(i:integer);
var k,n:integer;
begin
for k:=1 to i do
begin
for n:=1 to 2 do
  wsp[k,n]:=wsp1[k,n];
wsp[k,3]:=1;
end;
end;

procedure przep2(i:integer);
var k,n:integer;
begin
for k:=1 to i do
begin
for n:=1 to 2 do
  wsp[k,n]:=nwsp[k,n];
wsp[k,3]:=1;
end;
end;

procedure rysfig(i,nr:integer);
var x:integer;
begin
 grafpen(nr);
 sx:=1.0;
 sy:=1.0;
for x:=1 to 12 do
 begin
   ustms(i);
   mnoz(i);
   rysmo(i);
   przep2(i);
   sy:=sy-0.02;
   sx:=sx-0.02;
 end;
 end;
{****************************************}
{  MODUL STERUJACY PROCEDURY CZOLOWKA    }
{****************************************}
begin
inicda;
repeat
 if (fla=0) then
 nh:=1 else
 begin nh:=random(7)+1; tytul; podp;
 end;
repeat
 pdan(nh); pdan1(licz);
 rysfig(licz,nh);
 for k:=1 to 8-random(7) do
    begin
     przep1(licz); {odtworzenie}
     rx:=rx+ krok;
     ustmp(licz,krok,0);
     mnoz1(licz);
     przep2(licz); pdan1(licz);
     grafwindow(0,nh*40,640,40+nh*40);
     clg(0);
     grafwindow(0,0,640,400);
     rysfig(licz,nh);
    end;
 nh:=nh+1;
until (nh=9) or (fla=1);
  fla:=1;
until keypressed;
border(0,3);
end;

{****************************************}
{      PROCEDURA GRA                     }
{****************************************}
overlay
procedure g0h;
var i,poz,ilpun:integer; w:boolean;
    nr,m,n:integer;
    tuw:array [1..92,1..8] of integer;
    a:array [1..8] of boolean;
    b:array [2..16] of boolean;
    c:array [-7..7] of boolean;
    x:array [1..8] of integer;
    co,y,zc:char;
const
    poczgcy=390;
    poczgcx=100;
    poczbcy=363;
    poczbcx=68;
    poczl=86;
    dlink=288;
    dlh=36;
    pxh=50;
    pyh=374;

{****************************************}
{   PROCEDURY GRAFICZNE GRY              }
{****************************************}
procedure cyfra(ko:integer);
 begin
  case ko of
  1:zc:='1';
  2:zc:='2';
  3:zc:='3';
  4:zc:='4';
  5:zc:='5';
  6:zc:='6';
  7:zc:='7';
  8:zc:='8';
  end;
 end;

procedure ryszach;
var j:integer;
 begin
  grafpen(2);
  grafpaper(0);
  for j:=0 to 8 do
   begin
    plot(poczl,j*36+poczl);
    drawr(dlink,0);
   end;
  for j:=0 to 8 do
   begin
    plot(j*36+poczl,poczl);
    drawr(0,dlink);
   end;
  grafpen(3);
  tag;
  plot(poczgcx,poczgcy);
  for j:=1 to 8 do
   begin
      cyfra(j);
      grafout(zc);
      grafmove(poczgcx+j*36,poczgcy);
   end;
  plot(poczbcx,poczbcy);
  for j:=1 to 8 do
   begin
      cyfra(j);
      grafout(zc);
      grafmove(poczbcx,poczbcy-j*36);
   end;
  tagoff;
end;

procedure ryshet(k:integer);
var nw,j:integer;
begin
 nw:=x[k];
 grafpen(3);
 for j:=0 to 35 do
  begin
   plot(pxh+k*36,pyh-nw*36+j);
   drawr(dlh,0);
   sound(1,100-j*3,5,5,0,0,0);
   sound(2,150-j*4,5,5,0,0,0);
  end;
 grafpen(2);
 plot(pxh+k*36,pyh-nw*36);
 grafmover(5,4);
 drawr(6,18); drawr(-6,9);
 drawr(9,-3); drawr(4,7);
 drawr(4,-7); drawr(9,3);
 drawr(-6,-9); drawr(6,-18);
 drawr(-26,0);
end;

procedure napis;
var nap:string[12]; j:integer;
begin
 nap:='DEMONSTRACJA';
 grafpen(2); tag;
 for j:=1 to 12 do
   begin
     plot(544,390-j*20);
     grafout(nap[j]);
   end;
 tagoff;
end;
procedure ryspunkty;
var lan:string[6]; j:integer;
begin
 grafpen(1);
 str(ilpun:6,lan);
 plot(520,256);
 tag;
 for j:=1 to 6 do
  grafout(lan[j]);
 tagoff;
end;
procedure ryspoziom;
var lan:string[2]; j:integer;
begin
 grafpen(1);
 str(poz:2,lan);
 plot(544,336);
 tag;
 for j:=1 to 2 do
  grafout(lan[j]);
 tagoff;
end;
procedure ryspps;
var nap:string[7]; j:integer;
begin
  nap:='POZIOM ';
  grafpen(2); tag;
  plot(528,368);
  for j:=1 to 6 do
   grafout(nap[j]);
  nap:='ILOSC  ';
  plot(528,288);
  for j:=1 to 5 do
   grafout(nap[j]);
  nap:='PUNKTOW';
  plot(528,272);
  for j:=1 to 7 do
   grafout(nap[j]);
  tagoff;
end;

{****************************************}
{   PROCEDURA DEMONSTRACJI               }
{****************************************}

procedure
 proboj(i:integer; var w:boolean);
  var j:integer; ilp:integer;
begin
  j:=nr-1; ilp:=0;
  repeat
   j:=j+1; w:=false;
   if j>8 then j:=1;
   ilp:=ilp+1;
   if a[j] and b[i+j] and c[i-j]
     then
        begin
         x[i]:=j;
         a[j]:=false; b[i+j]:=false;
         c[i-j]:=false;
         if i<8
           then
             begin
                proboj(i+1,w);
                if w=false
                  then
                    begin
                      a[j]:=true;
                      b[i+j]:=true;
                      c[i-j]:=true;
                    end;
             end
           else
             w:=true;
        end;
  until w or (ilp=8);
end;
```

```
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{   PROCEDURY KOMUNIKATOW               }
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
procedure kom1;
var c:char;
begin
repeat
nr:=0;
gotoxy(1,23);
WRITELN
(' PODAJ NUMER WIERSZA               ');
sound(135,120,80,50,0,0,0);
repeat until keypressed;
READ(kbd,c);
nr:=ord(c)-48;
until (nr>0) and (nr<9);
gotoxy(1,23);
WRITE
('                                      ');
end;
procedure kom2;
var r:char;
begin
gotoxy(1,23);
WRITELN
(' BLEDNY RUCH - TRACISZ 25 PUNKTOW     ');
WRITE
(' WCISNIJ DOWOLNY KLAWISZ              ');
repeat
          sound(2,100,13,20,0,0,0);
          sound(1,150,20,10,0,0,0);
until  keypressed;
READ(kbd,r);
gotoxy(1,23);
WRITELN
('                                      ');
WRITE
('                                      ');
end;
procedure kom3;
var r:char;
begin
gotoxy(1,23);
WRITELN
('NIEMOZLIWE USTAWIENIE-TRACISZ 200 PUNKT.');
WRITE
(' WCISNIJ DOWOLNY KLAWISZ              ');
repeat
 sound(1,random(150)+20,10,10,0,0,0);
 sound(1,random(100)+50,10,10,0,0,0);
until keypressed;
READ(kbd,r);
gotoxy(1,23);
WRITELN
('                                      ');
WRITE
('                                      ');
end;
procedure kom6;
var r:char;
begin
gotoxy(1,23);
WRITELN
(' TO JUZ BYLO  - TRACISZ 200 PUNKTOW  ');
WRITE
(' WCISNIJ DOWOLNY KLAWISZ            ');
repeat
 sound(1,random(40)+60,5,5,0,0,0);
 sound(2,random(10)+80,5,5,0,0,0);
until keypressed;
READ(kbd,r);
gotoxy(1,23);
WRITELN
('                                      ');
WRITE
('                                      ');
end;
procedure pytaj(var y:char);
begin
repeat
  gotoxy(1,24);
  WRITE(' GRASZ DALEJ (t/n) ');
  repeat
   sound(1,50,15,5,0,0,0);
   sound(2,random(10)+80,15,5,0,0,0);
  until keypressed;
  READ(kbd,y);
until (y='t') or (y='T') or (y='n')
      or (y='N');
gotoxy(1,23);
WRITELN
('                                      ');
WRITE('                 ');
end;
procedure koms;
var j:integer;
begin
for j:=1 to 5 do
begin
gotoxy(1,23);
WRITE
(' DLACZEGO SIE POWTARZASZ              ');
sound(1,20,5,5,0,0,0);
sound(2,40,5,5,0,0,0);
end;
end;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{  PROCEDURY INICJACJI DANYCH           }
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
procedure warp;
begin
 for i:=1 to 8 do
  begin
  a[i]:=true;
  x[i]:=0;
  end;
 for i:=2 to 16 do
  b[i]:=true;
 for i:=-7 to 7 do
  c[i]:=true;
end;
procedure inicdanych;
begin
 grafwindow(0,60,640,400);
 poz:=1; ilpun:=0;
 for m:=1 to 92 do
 for n:=1 to 8 do
  tuw[m,n]:=0;
 warp;
 ink(0,26,26);
 ink(1,9,9);
 ink(2,0,0);
 ink(3,6,6);
 paper(2); pen(0);
 grafpaper(0); border(9,9);
 co:=' ';
 y:=' ';
 w:=false;
end;
procedure czyszcz;
begin
clg(0);
for i:=23 downto 22 do
begin
gotoxy(1,i);
WRITE
('                                      ');
end;
end;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{ PROCEDURA SPRAWDZAJACA USTAWIENIE     }
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
procedure
probojsam(ko:integer; var w:boolean);
var j:integer; ilp:integer;
    tnum:array [1..8] of boolean;
begin
 ilp:=ko-1; w:=false;
 for j:=1 to 8 do
  tnum[j]:=false;
 for j:=1 to 8 do
  if x[j]>0 then
   tnum[x[j]]:=true;
 while ((ilp<8) and (w=false)) do
  begin
   repeat
   kom1;
   if tnum[nr]=false then
    ilp:=ilp+1
   else koms;
   until (tnum[nr]=false);
   tnum[nr]:=true;
   j:=nr;
   if a[j] and b[ko+j] and c[ko-j]
    then
      begin
       x[ko]:=j;
       a[j]:=false; b[ko+j]:=false;
       c[ko-j]:=false;
       rysket(ko);
       ilpun:=ilpun+25;
       ryspunkty;
       w:=true;
      end
    else
      begin
       ilpun:=ilpun-25;
       ryspunkty;
       kom2;
      end;
  end;
  if (w=false)
   then
     begin
      ilpun:=ilpun-200;
      ryspunkty;  kom3;
     end
end;

{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{    MODUL STERUJACY PROCEDURY GRA      }
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
begin
mode(1);
inicdanych;
czyszcz;
repeat
repeat
gotoxy(1,23);
WRITE
('DEMONSTRACJA-"d",GRA-"g",KONIEC-"k" ');
repeat
sound(1,20+random(150),15,15,10,0,0);
sound(2,30+random(200),12,12,20,0,0);
until keypressed;
READ(kbd,co);
until (co='d') or  (co='g') or (co='k')
   or (co='D') or  (co='G') or (co='K');
gotoxy(1,23);
WRITE
('                                      ');
if   (co='d') or  (co='g') or
     (co='D') or  (co='G')
then
begin
if (co='d') or (co='D')
then
begin
nr:=RANDOM(7)+1;
warp;
proboj(1,w);
clg(0);
napis;
rysszach;
for i:=1 to 8 do
rysket(i);
gotoxy(2,23);
WRITE('WCISNIJ DOWOLNY KLAWISZ');
repeat
sound(1,90,15,15,10,0,0);
sound(2,30,12,12,20,0,0);
until keypressed;
READ(kbd,y);
gotoxy(2,23);
WRITE('                          ');
end
else
begin
czyszcz;     clg(0);     ryspps;
rysszach;
ryspunkty;
ryspoziom;
repeat
y:=' '; warp;
i:=1;
repeat
probojsam(i,w);
if (w=false)
then
begin
pytaj(y);
if (y='t') or (y='T')
then
begin
clg(0); ryspps; ryspunkty; ryspoziom;
rysszach; warp;
i:=1;
end;
end
else
i:=i+1;
until (y='n') or (i=9) or (y='N');
if i=9  then
begin
m:=1;
repeat
n:=1;
w:=false;
repeat
if tuw[m,n]<>x[n]
then
w:=true
else
n:=n+1;
until w or (n=9);
if w  then m:=m+1 else m:=93;
until (m=93);
if not  w
then
begin
ilpun:=ilpun-200;
ryspunkty;
kom6;
pytaj(y);
if (y='t') or(y='T') then
begin
clg(0); ryspps; ryspunkty;
ryspoziom; rysszach;
end;
end
else
begin
for n:=1 to 8 do
tuw[poz,n]:=x[n];
gotoxy(2,23);
WRITE
('BRAWO-MOZESZ PRZEJSC NA KOLEJNY POZIOM');
sound(1,10,1,5,10,0,0);
sound(2,40,2,2,20,0,0);
pytaj(y);
if (y='t') or (y='T') then
begin
clg(0); ryspps; ryspunkty;
rysszach;
poz:=poz+1;
ryspoziom;
end;
end;
end;
until (poz>93) or (y='n') or (y='N');
if (y='n') or (y='N')   then co:='k'
else
begin
gotoxy(1,23);
WRITE('    JESTES MISTRZEM  $$$');
sound(1,120,90,5,10,0,0);
sound(2,40,90,20,20,0,0);
co:='k';
end;
end;
end;
until (co='k') or (co='K');
end;
```

Przygotowanie drukarki GEMINI-10Xi do drukowania wersji źródłowych programów

Zestaw mikrokomputerowy AMSTRAD CPC 6128 z drukarką GEMINI—10Xi może być wykorzystywany do drukowania wersji źródłowych programów. Dokonuje się tego za pomocą komendy LIST#;8. Jednak uzyskany w ten sposób wydruk ma następujące mankamenty:

— brak stronicowania;
— pisanie długich instrukcji na perforacji i poza papierem;
— brak możliwości uzyskania innych formatów niż A4.

Problem ten można rozwiązać stosując kody sterujące pracą drukarki.

Prezentowany program umożliwia sformatowanie wydruków poprzez wstawienie odpowiednich kodów, zwalniając użytkownika od konieczności ich znajomości. Rola użytkownika ogranicza się do wyboru następujących opcji (w nawiasie wartości standardowe):

— długość strony (12 cali);
— rodzaj czcionki (10 znaków/cal);
— margines górny (2 linie);
— margines dolny (2 linie);
— margines lewy (2 znaki);
— szerokość tekstu (w zależności od rodzaju czcionki: 68, 83, 118 znaków);
— odstęp między wierszami (9/72 cala);
— druk wytłuszczony ? (T/N)

U dołu ekranu jest wyświetlana informacja, że naciśnięcie klawisza ENTER powoduje przyjęcie wartości podanej w nawiasach. Podawane przez użytkownika wartości poddawane są wnikliwej kontroli formalnej i merytorycznej.

Kontrola formalna polega na zbadaniu, czy wprowadzana wartość jest typu numerycznego (podprogram 2000—2030). Kontrola merytoryczna polega na sprawdzeniu każdego z podanych parametrów z wartościami granicznymi charakterystycznymi dla odpowiedniego kodu sterującego. W przypadku podania błędnego parametru na ekranie jest sygnalizowany błąd i należy powtórnie podać żądaną wartość.

Ustawienie drukarki (podanie odpowiednich kodów sterujących) odbywa się w instrukcji 500.

Po podaniu wszystkich parametrów należy załadować do pamięci program, który ma być drukowany oraz wprowadzić instrukcję LIST#8

Poszukiwanie różnorodnych form „listingów” satysfakcjonujących Czytelników pozostawiamy posiadaczom drukarek GEMINI—10Xi.

Mieczysław SKONIECZNY
Robert ZUGCHOR

```
10 MODE 2:INK 0,1:INK 1,26:PAPER 0:PEN 1:BORDER 1
20 PRINT"PRZYGOTOWANIE DRUKARKI  gemini-10Xi  DO DRUKOWANIA WERSJI ZRODLOWYCH PR
OGRAMOW"
25 WINDOW £1,1,80,22,25:PAPER £1,1:PEN £1,0:CLS £1:LOCATE £1,1,2:PRINT £1," U W
A G A :  Nacisniecie klawisza ENTER powoduje przyjec
ie wartosci podanych                    w nawiasach."
30 LOCATE 1,3:INPUT "Dlugosc strony (12 cali):
                 ",a$
40 IF a$="" THEN a$="12"
45 a=3:GOSUB 2000
46 IF blad=1 THEN GOTO 30
50 d.str    a$)
60 IF d.str  1 OR d.str>32 THEN GOSUB 1000:GOTO 30
70 LOCATE 1.5:PRINT"  Inne warianty : 12 znakow/cal  lub 17 znakow/cal"
80 LOCATE 1,4:INPUT "Rodzaj czcionki (10 znakow/cal):
                 ",a$
90 IF a$="" THEN a$="10"
95 a=4:GOSUB 2000
96 IF blad=1 THEN GOTO 80
100 rp=VAL(a$)
110 IF rp<>10 AND rp<>12 AND rp<>17 THEN GOSUB 1000:GOTO 80
120 IF rp=10 THEN rp=80
130 IF rp=12 THEN rp=77
140 IF rp=17 THEN rp=15
200 LOCATE 1,6:PRINT"  Maksymalny margines - polowa strony"
210 LOCATE 1,5:INPUT "Margines gorny (2 linie):
                 ",a$
220 IF a$="" THEN a$="2"
225 a=5:GOSUB 2000
226 IF blad=1 THEN GOTO 210
227 mg=VAL(a$)
230 IF mg<1 OR mg>INT(dlstr/2*6) THEN GOSUB 1000:GOTO 210
250 LOCATE 1,7:PRINT"  Maksymalny margines - polowa strony"
260 LOCATE 1,6:INPUT "Margines dolny (2 linie):
                 ",a$
270 IF a$="" THEN a$="2"
275 a=6:GOSUB 2000
276 IF blad=1 THEN GOTO 260
277 md=VAL(a$)
280 IF md<1 OR md>INT(dlstr/2*6) THEN GOSUB 1000:GOTO 260
300 LOCATE 1,8:PRINT "  Maksymalny margines - 30 znakow"
310 LOCATE 1,7:INPUT "Margines lewy (2 znaki):
                 ",a$
320 IF a$="" THEN a$="2"
325 a=7:GOSUB 2000
326 IF blad =1 THEN GOTO 310
327 ml=VAL(a$)
330 IF ml<1 OR ml>30 THEN GOSUB 1000:GOTO 310
350 x=70-ml
360 IF rp=77 THEN x=85-ml
370 IF rp=15 THEN x=120-ml
380 LOCATE 1,8:PRINT "Szerokosc drukowanego tekstu (";x;" znakow):":LOCATE 76,8:
INPUT "",a$
390 IF a$="" THEN a$=MID$(STR$(x),2,10)
395 a=9:GOSUB 2000
396 IF blad=1 THEN GOTO 380
397 sd=VAL(a$)
400 IF sd<1 OR sd>x THEN GOSUB 1000:GOTO 380
405 LOCATE 1,10:PRINT"  Podaj licznik ulamka  n/72";
410 LOCATE 1,9:INPUT "Odstep miedzy wierszami (9/72 cala):
                 ",a$
415 IF a$="" THEN a$="9"
416 a=9:GOSUB 2000
417 IF blad=1 THEN GOTO 410
418 od=VAL(a$)
420 IF od<9 OR od>255 THEN GOSUB 1000:GOTO 410
440 LOCATE 1,10:INPUT "Druk wytluszczony ? ( T / N ):
                 ",a$
450 IF a$="" THEN GOTO 440
460 IF UPPER$(a$)<>"T"AND UPPER$(a$)<>"N" THEN a=10:GOSUB 1000:GOTO 440
470 tl=0
480 IF UPPER$(a$)="T" THEN tl=1
500 PRINT£8,CHR$(27) CHR$(67) CHR$(0) CHR$(dlstr);CHR$(27) CHR$(rp);CHR$(27) CHR
$(114) CHR$(mg+1);CHR$(27) CHR$(78) CHR$(md);CHR$(27
) CHR$(108) CHR$(ml);CHR$(27) CHR$(81) CHR$(ml+sd);CHR$(27) CHR$(65) CHR$(od);CH
R$(27) CHR$(50);CHR$(27) CHR$(72-tl);
510 PRINT£8,CHR$(12);
520 CLS £1:PRINT£1,"":PRINT£1,"  1. Zaladuj program, ktory chcesz drukowac komen
da LOAD                        2. Wprowadz komende
  LIST £8"
999 END
1000 LOCATE 1,a:PRINT"Niepoprawny parametr
                 ";
1010 FOR t=1 TO 1500:NEXT t
1020 RETURN
2000 blad=0
2005 FOR i=1 TO LEN(a$)
2010 IF ASC(MID$(a$,i,1))<48 OR ASC(MID$(a$,i,1))>57 THEN blad =1
2015 NEXT i
2020 IF blad=1 THEN LOCATE 1,a:PRINT"  Wartosc nienumeryczna ";a$;"
             ";FOR t=1 TO 1500:NEXT
2030 RETURN
```

NUTY Z KOMPUTERA

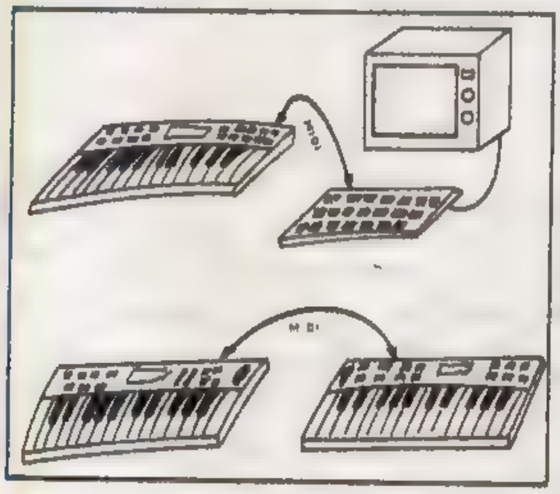
(1)

Krzysztof POŹNIAK, Jarosław ZIEMBICKI



Syntezatory muzyczne mają już długą historię. Od najprostszych monofonicznych urządzeń analogowych doszliśmy do skomputeryzowanych układów mogących naśladować jednocześnie całą gamę instrumentów i wytwarzać niezwykłe, nigdzie dotąd nie uzyskiwane dźwięki. Bogata paleta możliwości została okupiona koniecznością programowania licznych parametrów dźwięku. Obecnie produkowane syntezatory są to wyspecjalizowane programowane układy cyfrowe, gdyż tylko takie urządzenia zapewniają wysoką jakość brzmienia instrumentu. Niestety każdy kij ma dwa końce. Programowanie brzmień jest trudną i pracochłonną czynnością. Aby oszczędzić muzykom wiele godzin ciężkiej pracy producenci dołączają do swoich produktów tzw. *cartidge*, **czyli pakiety już zaprogramowanych brzmień. Innym, bardziej ambitnym sposobem jest programowanie za pomocą komputera. Specjalne programy umożliwiają graficzne tworzenie dźwięków na ekranie monitora, wygodną edycję parametrów itp. Komputer przetwarza pomysły muzyka na odpowiedni ciąg danych i przesyła je do syntezatora. W tym miejscu powstaje bardzo ważny problem: w jaki sposób komputer ma nawiązać kontakt z syntezatorem?**

Aby zapewnić pełną łączność pomiędzy wszystkimi instrumentami muzycznymi i dowolnymi komputerami przyjęto specjalny rodzaj transmisji nazywany MIDI (z angielskiego: *Musical Instrument Digital Interface*). Urządzenia wyposażone w system MIDI



Rys. 1

można ze sobą łączyć i przesyłać między nimi informacje. Dotyczy to zarówno transmisji między komputerem i urządzeniem muzycznym, jak również między samymi urządzeniami muzycznymi (rys. 1). Standard MIDI umożliwia łączność z 16 niezależnymi, wspólnie połączonymi urządzeniami, potencjalne możliwości współpracy są zatem olbrzymie. Poprzez MIDI jesteśmy w stanie programować syntezatory, perkusje elektryczne, wzmacniacze estradowe, efekty elektroniczne (np. delay) itp.

Każdy, kto choć trochę interesował się muzyką elektroniczną zauważył, że muzycy posługują się całymi zestawami instrumentów, które znakomicie ze sobą i z muzykiem współpracują. To zgranie jest możliwe dzięki specjalnemu urządzeniu: **sekwencerowi**. Sekwencer w określonym (uprzednio zaprogramowanym) tempie wysyła do połączonych z nim instrumentów rozkazy generacji dźwięków i ustala odpowiednie brzmienia. Z naszego punktu widzenia wykonuje zapisany przez muzyka program.

Do chwili obecnej, na łamach miesięczników związanych z komputerami, pojawiły się artykuły opisujące parametry poszczególnych dźwięków. Bardzo ciekawym i polecanym przez nas jako wstęp jest ciąg artykułów Jacka Jędrzejowskiego o skomputeryzowanej muzyce, jaki ukazał się w „Informiku" (dodatek do „Młodego Technika").

W naszym artykule chcemy pokazać muzykom amatorom, jak można programować utwory za pomocą komputera. Tak napisane utwory będą odtwarzane, bądź za pomocą wbudowanych w komputer generatorów dźwięku, bądź przez instrumenty elektroniczne (np. syntezatory i perkusje).

Pragniemy w tym miejscu wtrącić bardzo ważną uwagę: każdy komputer nadaje się do transmisji MIDI i może służyć jako sekwencer dla instrumentów elektronicznych. Z powodzeniem wypróbowaliśmy komputery ZX Spectrum i Sinclair QL, jako sterowniki syntezatorów CASIO CZ 1000, KORG POLY 800, KORG DW 6000 YAMAHA DX 7, ROLAND & YUNO 1 i perkusji elektronicznych ROLAND TR 505, YAMAHA RX 15.

Tworzenie muzyki za pomocą komputera upodabnia się do pisania programu w określonym języku wyższego rzędu (np. Basicu, Pascalu itp.) i dlatego dla wyjaśnienia kilku ważnych pojęć, którymi będziemy się posługiwać, wykorzystamy następujące porównanie.

STRUKTURA PROGRAMU

1) każdy program składa się z rozkazów,
2) rozkazy w programie są tak ułożone, że komputer dokładnie wie, w jakiej kolejności ma je wykonać,
3) dobry program ma budowę strukturalną i składa się z podprogramów (cegiełek), które połączone w odpowiedni sposób budują program.

Jeżeli przeprowadzimy następujące powiązania znaczeń:

a) dźwięk — rozkaz,
b) takt — podprogram,
c) utwór — program,

to otrzymamy:

STRUKTURA UTWORU

1) każdy utwór składa się z dźwięków,
2) dźwięki są wykonywane w ściśle określonej kolejności,
3) utwór ma budowę strukturalną i składa się z taktów.

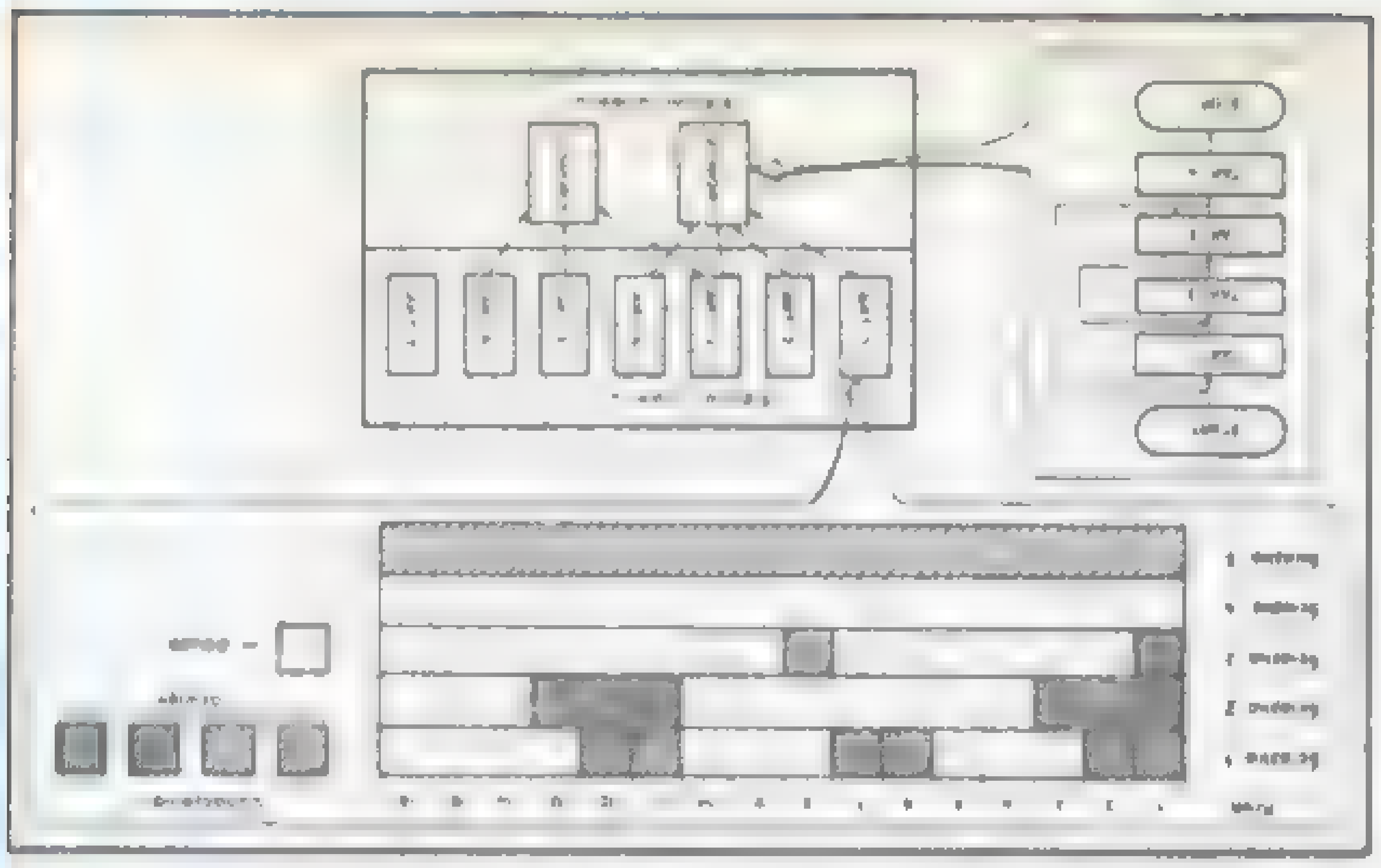
Każdy programista wie, jak ważnym narzędziem w jego pracy jest edytor. Za jego pomocą tworzymy, poprawiamy i rozbudowujemy program.

Nasz sekwencer posiada dwa niezależne edytory: edytor taktu (pozwala na budowanie i korekcję taktów) i edytor utworu (za jego pomocą z istniejących taktów składamy utwór).

Podstawową strukturą sekwencera przedstawia rysunek 2

a) stronę edycyjną obsługują wyżej opisane edytory,
b) nadrzędną strukturą programową jest utwór (jednocześnie zapisanych utworów





Rys 2

może być kilka). Każdy utwór zbudowany jest z dowolnej ilości taktów, które mogą się wzajemnie zapętlać (powtarzać) określoną ilość razy. Zapętlenia mogą wnikać w głąb siebie, analogicznie, jak pętle FOR—NEXT w programie komputerowym.

c) podrzędną strukturą programową jest takt. Jest to nierozdzielna sekwencja dźwięków. Takt określa kilka parametrów

parametr 1: ilość kroków — w pewien sposób łączy się z metrum taktu i informuje ile maksymalnie dźwięków może zmienić się na jednym śladzie. Na rys. 2 przedstawiony został przykładowy takt posiadający 16 kroków. Oznacza to, że na jednym śladzie można umieścić 16 kolejno zmieniających się dźwięków (nazywanych szesnastkami), bądź 8 dwukrotnie dłuższych dźwięków (ósemek), albo dźwięków o innych długościach przedzielonych przerwami (pauzami). Może wystąpić ścieżka pusta (przykładowo ścieżka 4) lub ścieżka generująca dźwięk ciągły (tak jak ścieżka 5)

parametr 2: tempo — informuje jak szybko mają następować po sobie kolejne kroki. Każdy takt może posiadać inne tempo

parametr 3: ilość ścieżek — każda ścieżka zawiera informacje przekazywane jednemu źródłu dźwięku (może być to generator w syntezatorze, lub pojedynczy instrument w perkusji elektronicznej). Ilość ścieżek zależy od liczby niezależnych źródeł dźwięku wykorzystywanych w danym takcie. Ponieważ sekwencer przystosowany jest do sterowania kilkoma instrumentami jednocześnie, stąd ścieżki można w dowolny sposób przypisać poszczególnym instrumentom (np. ścieżki 1—4 mogą obsługiwać syntezator, ścieżki 5—8 generatory komputera, a pozostałe perkusję)

Pragniemy na zakończenie pierwszej części artykułu wyjaśnić, że prezentowany sekwencer ma dosyć ubogą strukturę i należy go analizować bardziej od strony dydaktycznej niż eksploatacyjnej. Stanowi on jednak dobrą bazę do rozbudowy przez poszczególnych użytkowników. Jego prostota zapewnia przejrzystość działania, a nam umożliwia przedstawienie podstawowych problemów programowania utworów i ich transmisji w systemie MIDI

Zwracamy się w tym miejscu do czytelników zainteresowanych opisanymi problemami o kontakt z redakcją. „IKS" może stać się czasopismem, na łamach którego, tematyka związana z programowaniem muzyki za pomocą komputera znajdzie godne miejsce. Warto tę szansę wykorzystać









(Foto: Jan Zelman)

Programowanie mikroprocesora 6502

Henryk KRASUSKI

Każdy użytkownik mikrokomputera ATARI XL/XE często irytuje się, obserwując realizację programów napisanych w języku BASIC — są one wykonywane powoli. Czy można zwiększyć szybkość ich działania? Oczywiście, istnieje na to sposób. Jest nim programowanie w języku asemblera.

Na wstępie pragnę uczynić jedno zastrzeżenie: nie namawiam użytkowników ATARI do zaniechania programowania w BASIC-u. ATARI BASIC jest dobrym i bardzo wygodnym narzędziem programowania, a to, że niektóre operacje wykonuje powoli, jest jego wadą, charakterystyczną dla wszystkich języków interpretacyjnych, którą łatwo wyeliminować, włączając do programu w języku BASIC procedury w kodzie maszynowym. Operacja ta jest prosta, nawet dla początkującego programisty dzięki instrukcji języka BASIC X=USR (...), za pomocą której można wywołać podprogram maszynowy i przekazać do niego wiele parametrów.

Jest rzeczą oczywistą, że trzeba wiedzieć, w jakiej sytuacji włączenie procedury w kodzie maszynowym przyniesie oczekiwane korzyści, tzn. wydatnie przyspieszy działanie programu, a w jakiej zabieg ten okaże się nieopłacalny. Na pewno nie warto programować operacji arytmetycznych, czy arytmetyki zmiennoprzecinkowej w kodzie maszynowym. Istnieją natomiast obszary, w których zastosowanie kodu maszynowego zwiększa szybkość działania programu od kilku do kilkunastu razy, a w skrajnych przypadkach nawet 1000 razy. Obszarami tymi są:

— generowanie dźwięków;
— wyprowadzanie dużych zbiorów informacji na ekran;
— pętle programowe w języku BASIC, realizujące przesłania pomiędzy bajtami pamięci, zmieniające zawartość obszarów pamięci oraz wyszukujące z nich bajty o określonej zawartości;
— wykonywanie operacji logicznych na pojedynczych bitach.

W ostatnim przypadku mniej istotna jest szybkość działania, lecz fakt, że operacje tego typu są niemożliwe do wykonania z poziomu języka BASIC.

Tworzenie programów w kodzie maszynowym wymaga znajomości mikroprocesora MOSTEC 6502, jego architektury, listy rozkazów i sposobu ich wykonywania oraz podstawowych procedur systemu operacyjnego. Ponadto przydatny jest do tego asembler (nie jest niezbędny), tj. program tłumaczący rozkazy z postaci mnemonicznej na ich kod maszynowy. Godnym polecenia jest pakiet ATARI Assembler Editor, zawierający oprócz asemblera, edytor — umożliwiający pisanie i poprawianie programów oraz debugger — program przeznaczony do testowania napisanych programów maszynowych. Zaletą tego pakietu jest możliwość współpracy z magnetofonem i stacją dysków, co z punktu widzenia wielu użytkowników ATARI nie posiadających stacji dysków jest rzeczą o istotnym znaczeniu.

Na początek kilka słów o architekturze mikroprocesora 6502. Do jej elementów należą:

a) 8-bitowa szyna danych;
b) 16-bitowa szyna adresowa;
c) jeden akumulator — rejestr A;
d) dwa 8-bitowe rejestry indeksowe — rejestr X i Y;

Rejestry te można również wykorzystywać, podobnie jak akumulator, tzn. można przechowywać w nich dane, przesyłać za ich pośrednictwem dane pomiędzy bajtami pamięci, itp.;

e) rejestr stanu mikroprocesora, zwany również rejestrem flagowym lub rejestrem F;
f) stos mikroprocesora. Stanowi go pierwsza strona pamięci mikrokomputera, tj. bajty o adresach od 256 do 511 ($100 — $1FF). Ze stosem związany jest 9-bitowy wskaźnik stosu — rejestr S;
g) licznik programu — 16-bitowy rejestr PC (ang. Program Counter);
h) dwie linie przerwań mikroprocesora;
i) strona zerowa pamięci mikrokomputera, tj. bajty o adresach od 9 do 255 ($00 — $FF). Czytelnicy, którzy już są zapoznani z problematyką programowania mikroprocesora 6502, mogą być zaskoczeni zaliczeniem strony zerowej pamięci do jego architektury. W literaturze na temat 6502 strona zerowa pamięci nie jest traktowana jako element architektury. Jednak ze względu na istnienie specjalnego trybu adresowania dostępnego dla wielu rozkazów 6502, tzw. adresowania strony zerowej, uważam, że stanowi ona istotny jej element. Ponadto cały mechanizm adresowania pośredniego realizowany jest przez 6502 przy wykorzystaniu bajtów tej strony — co również przemawia za traktowaniem jej jako elementu architektury.

Na bliższe przedstawienie zasługują trzy elementy spośród wyżej wymienionych, a mianowicie: stos i wskaźnik stosu, rejestr F i licznik programu.

Stos jest bardzo ważnym elementem każdego systemu mikrokomputerowego. Można go sobie wyobrazić jako zbiór rejestrów wykorzystywanych przez mikroprocesor przy obsłudze przerwań, do chwilowego przechowywania danych pobranych z pamięci, przy wykonywaniu procedur wywoływanych z wykonywanego programu, itp. Mikroprocesor 6502 stosuje specyficzną metodę wprowadzania i odczytywania informacji ze stosu — zwaną LIFO (ang. last-in, first-out). Metoda ta polega na tym, że informacja zapisana do stosu jako pierwsza, odczytana zostanie jako ostatnia, natomiast informacja wprowadzona do stosu jako ostatnia, będzie pierwszą przy odczycie. Adres stosu, do którego zapisano ostatni element, zawarty jest we wskaźniku stosu (dokładnie jest to ten adres zwiększony o jeden). Wskaźnik stosu jest jedynym 9-bitowym rejestrem mikroprocesora 6502. Jego najbardziej znaczący bit jest zawsze równy jeden. Gdy stos jest pusty, wskaźnik stosu wskazuje adres 511 ($1FF). Przy wprowadzeniu kolejnego bajtu do stosu wskaźnik stosu jest zmniejszony o jeden aż do wartości 256 ($100). Próba wprowadzenia do stosu informacji przy wskaźniku równym 256 prowadzi do przepełnienia stosu i w sytuacji, gdy wartość wskaźnika nie jest kontrolowana, konsekwencją tego może być załamanie systemu. Wywoływanie podprogramów maszynowych z języka BASIC realizowane jest z wykorzystaniem stosu mikroprocesora. Instrukcja:

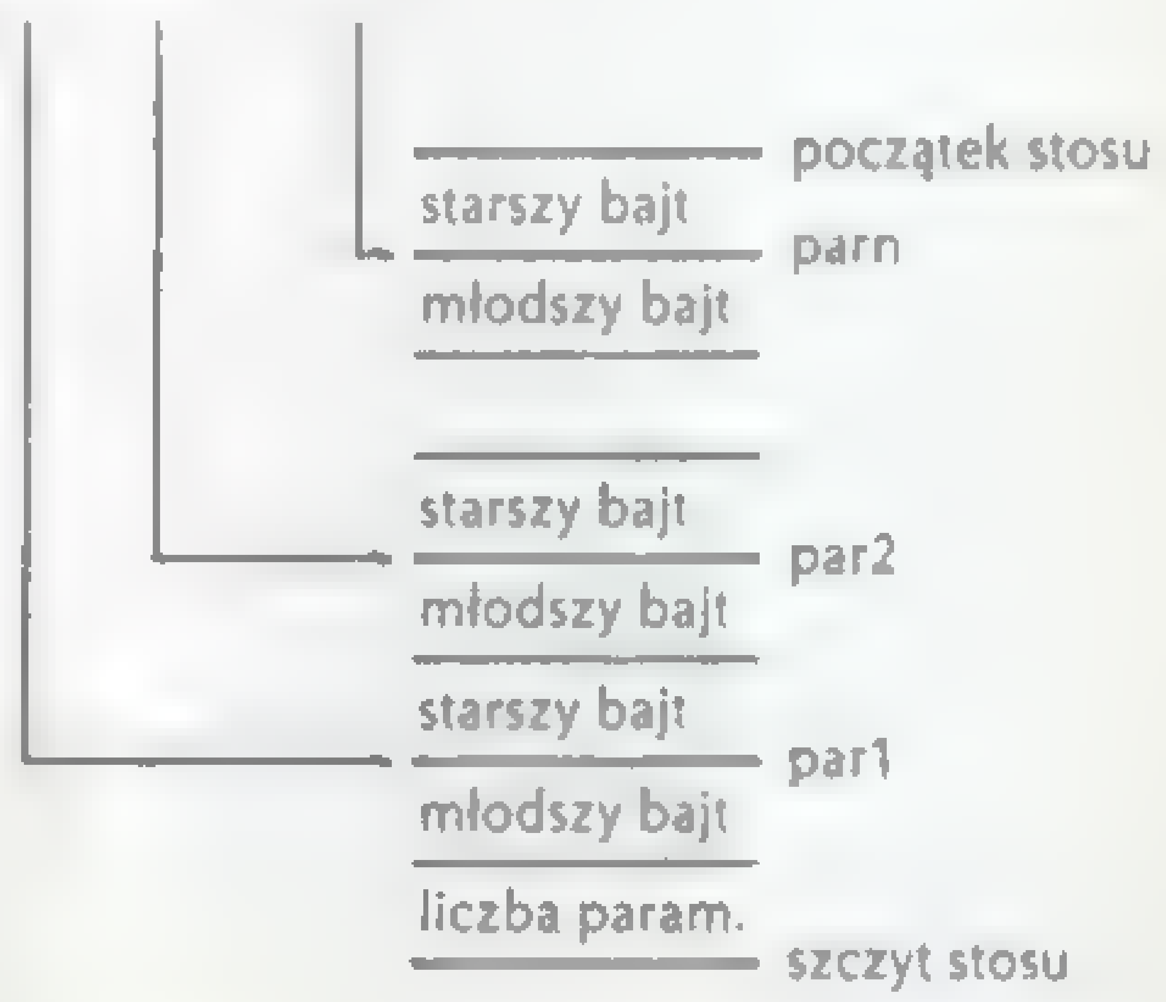
X=USR(adres,par1, par2,...,parn)

działa w sposób następujący:

— na stos przesyłane są dwubajtowe parametry w kolejności odwrotnej od tej, w jakiej zostały wymienione w instrukcji USR, tzn. najpierw wysyłany jest parn i wskaźnik stosu jest zmniejszany o 2;
— kolejno przesyłane są pozostałe parametry... par2, par1. Każdy z nich zajmuje dwa bajty stosu;
— na końcu na stos wysyłana jest jednobajtowa liczba przesłanych parametrów.

Ilustruje to poniższy schemat:

X=USR(adres,par1,par2,...,parn)



Wywołany podprogram maszynowy, przed rozpoczęciem realizacji przypisanej mu funkcji, musi pobrać ze stosu przekazane parametry. Wykonuje to sekwencją rozkazów PLA i STA (patrz lista rozkazów). Nawet w sytuacji, gdy do podprogramu maszynowego nie są przekazywane żadne parametry musi on

usunąć ze stosu bajt zawierający ich liczbę — w tym przypadku zero. Zatem każdy podprogram maszynowy, przeznaczony do współpracy z językiem BASIC musi wykonać rozkaz PLA jako pierwszy.

Wykonany podprogram maszynowy może przekazywać z powrotem do języka BASIC wynik zrealizowanej funkcji. Może to czynić w dwojaki sposób. Jeżeli wynik zrealizowanej funkcji mieści się na dwóch bajtach, to podprogram maszynowy może umieścić go w bajtach strony zerowej o adresach $D4 (młodszy) i $D5 (starszy). Przekazany w ten sposób wynik jest dostępny z poziomu języka BASIC w zmiennej X użytej przy wywołaniu podprogramu maszynowego. Natomiast, jeżeli wynik działania podprogramu maszynowego nie da się przekazać w dwóch bajtach pamięci (np. posortowana tablica stałych tekstowych), to może on być przekazany do języka BASIC w zarezerwowanym dla tego celu obszarze pamięci, którego adres powinien być przekazany do podprogramu maszynowego jako jeden z parametrów.

Rejestr stanu mikroprocesora zawiera szereg wskaźników, które mogą być wykorzystywane przez program maszynowy. Zawartość niektórych z nich może być zmieniana przez programistę dla specyficznych celów, jednak z reguły ich zawartość jest modyfikowana przez mikroprocesor, aby poinformować programistę o aktualnym stanie procesu przetwarzania. Rejestr ten można zobrazować poniższym schematem:

|N|V| |B|D|I|Z|C| rejestr F

Znaczenie poszczególnych wskaźników jest następujące:

N — wskaźnik znaku. Jest ustawiony (N=1), gdy po wykonaniu ostatniego rozkazu w bajcie zawierającym wynik, najbardziej znaczący bit jest równy jeden. Gdy po wykonaniu ostatniego rozkazu najbardziej znaczący bit wyniku jest zerem, wskaźnik N jest zgaszony (N=0). Wskaźnik ten jest ustawiany również przez procedury systemu operacyjnego, obsługujące urządzenia zewnętrzne w przypadku niepomyślnego zakończenia przesłania danych z/do tych urządzeń.

V — wskaźnik przepełnienia. Jest ustawiany w sytuacji, gdy wynik ostatniego rozkazu nie mieści się na siedmiu bitach i do jego przedstawienia musi być użyty najbardziej znaczący bit, który normalnie jest bitem znaku.

B — wskaźnik przerwania programowego. Jest ustawiany, gdy miroprocesor wykona rozkaz przerwania BRK (patrz lista rozkazów).

D — wskaźnik dziesiątego trybu pracy mikroprocesora. Jest ustawiany i gaszony przez programistę. Gdy jest ustawiony mikroprocesor traktuje liczby, na których wykonuje operacje arytmetyczne nie jak liczby binarne, a jak liczby w kodzie BCD.

I — wskaźnik przerwań sprzętowych. Programista ustawiając ten wskaźnik blokuje przerwania tego typu.

Z — wskaźnik zera. Ustawiany jest, gdy wynikiem ostatniego wykonanego rozkazu jest zero. Ustawiany jest również, gdy bajt zawierający zero zostanie pobrany z pamięci do akumulatora lub jednego z rejestrów indeksowych.

C — wskaźnik przeniesienia. Jest ustawiany, gdy wynik dodawania lub odejmowania nie mieści się w jednym bajcie. Na przykład: gdy dodamy do siebie dwie liczby 128 i 129, to otrzymamy 257. W takiej sytuacji wskaźnik C zostałby ustawiony, ponieważ liczby 257 nie da się zapisać na ośmiu bitach. Wskaźnik może być ustawiony również przez rozkazy przesuwające w lewo lub w prawo zawartość bajtów (patrz lista rozkazów).

Do efektywnego łączenia podprogramów maszynowych z programem w języku BASIC wystarczy posługiwanie się czterema z wyżej wymienionych wskaźników: N, Z, C i V.

Licznik programu jest 16-bitowym rejestrem, w którym mikroprocesor przechowuje adres następnego rozkazu, który ma być wykonany. Licznik programu zbudowany jest z dwóch rejestrów 8-bitowych PCL i PCH, natomiast efektywny adres rozkazu do wykonania obliczany jest według znanej zależności adres = (PCL) + 256 * (PCH).

Zestaw rozkazów mikroprocesora składa się z 56 rozkazów. W większości rozkazów można wyodrębnić dwa elementy: kod operacji do wykonania (kod rozkazu) i operand. Kod rozkazu jest liczbą z zakresu 0—255 i zawsze zajmuje jeden bajt pamięci. Operand wskazuje argument rozkazu, tj. bajt w pamięci mikrokomputera lub rejestr mikroprocesora, na którym operacja ma być wykonana. Operand może zajmować jeden albo dwa bajty pamięci lub może w ogóle nie występować. Stąd rozkazy mikroprocesora mogą mieć różną długość — od jednego do trzech bajtów.

```
bajt    n       n+1     n+2
| kod |                              rozkaz jednobajtowy
| kod | |operand|                    rozkaz dwubajtowy
| kod | o p e|r a n d|               rozkaz trzybajtowy
```

Wiele rozkazów 6502 posiada kilka trybów adresowania. Tryb adresowania zawiera informację o sposbie interpretacji operandu i o jego wielkości. W dalszej części artykułu przedstawione zostały wszystkie tryby adresowania rozkazów mikroprocesora 6502. Dla każdego trybu podany został sposób zapisu operandu wymagany przez pakiet ATARI Assembler Editor oraz sposób, w jaki obliczana jest efektywna wartość argumentu rozkazu. W każdym trybie adresowania operand, o ile występuje, może być liczbą szesnastkową (poprzedzoną znakiem $ lub liczbą dziesiętną.

1. Adresowanie natychmiastowe

kod # operand

Operand jednobajtowy. Argumentem rozkazu jest operand. Np.: rozkaz LDA # 20 powoduje, że do akumulatora zostanie wprowadzona liczba 20.

2. Adresowanie absolutne

kod operand

Operand dwubajtowy. Argumentem rozkazu jest zawartość bajtu, którego adres jest równy wartości operandu.

3. Adresowanie absolutne indeksowane

kod operand, X
kod operand, Y

Operand dwubajtowy. Argumentem rozkazu jest zawartość bajtu o adresie powstałym przez dodanie do wartości operandu zawartości jednego z rejestrów indeksowych X lub Y.

4. Adresowanie strony zerowej

kod operand

Operand jednobajtowy. Argumentem rozkazu jest zawartość bajtu ze strony zerowej. Adres tego bajtu jest równy wartości operandu.

5. Adresowanie strony zerowej indeksowane

kod operand, X

Operand jednobajtowy. Argumentem rozkazu jest zawartość bajtu ze strony zerowej o adresie równym wartości operandu zwiększonej o zawartość rejestru indeksowego X. Jak widać, ten tryb adresowania może używać jako rejestru indeksowego wyłącznie rejestru X. Jednakże istnieją dwa rozkazy stanowiące wyjątek od tej zasady. Są to LDX (załaduj do rejestru X) i STX (prześlij zawartość rejestru X). Rozkazy te mogą używać jako rejestru indeksowego, w tym trybie adresowania — rejestru Y.

6. Adresowanie absolutne pośrednie

kod (operand)

Operand dwubajtowy. Istota adresowania pośredniego polega na tym, że operand nie wskazuje bezpośrednio adresu argumentu, jak to miało miejsce w wyżej wymienionych trybach adresowania, lecz bajt w pamięci — wskaźnik, z którego pobierany jest adres argumentu. Jest to sytuacja analogiczna do tej, która ma miejsce przy wykorzystaniu instrukcji GOTO z języka BASIC, zawierającej zamiast numeru linii nazwę zmiennej, do której wcześniej wpisano numer linii. Jedynym rozkazem mikroprocesora 6502, w którym dozwolone jest użycie tego trybu adresowania, jest rozkaz skoku JMP. Na przykład: rozkaz JMP ($1000) nie skacze bezpośrednio do rozkazu zawartego w bajcie o adresie $1000, lecz jest realizowany w sposób następujący:

— pobierana jest zawartość bajtu o adresie równym wartości operandu i bajtu następnego, tj. o adresie o jeden większym, w przykładzie są to bajty o adresach $1000 i $1001;

— zawartość tych dwóch bajtów traktowana jest jako 16-bitowy adres obliczany według zależności: ($1000)+($1001)*256;

— skok następuje do rozkazu zawartego w bajcie o obliczonym adresie.

Wszystkie inne rozkazy, które mogą używać adresowania pośredniego, muszą jednocześnie używać rejestrów indeksowych X lub Y. Istnieją dwa tryby adresowania pośredniego wykorzystujące te rejestry. Są one przedstawione poniżej.

7. Adresowanie indeksowane pośrednie

kod (operand, X)

Operand jednobajtowy. Ten tryb adresowania wymaga użycia rejestru X. Jego zawartość dodawana jest do wartości operandu. Suma wskazuje adres bajtu na stronie zerowej. Wskazywany

bajt i bajt następny zawierają 16-bitowy adres argumentu rozkazu.

8. **Adresowanie pośrednie indeksowane**

kod (operand),Y

Operand jednobajtowy. Tryb ten wymaga użycia rejestru indeksowego Y. Adres argumentu rozkazu wyznaczany jest podobnie jak w trybie przedstawionym w punkcie 7. Zasadnicza różnica polega na tym, że zawartość rejestru nie jest dodawana do wartości operandu, a do 16-bitowego adresu, pobranego z dwóch bajtów, z których pierwszy wskazywany jest przez operand. Mówiąc inaczej: w adresowaniu indeksowanym pośrednim wartość indeksu dodawana jest do wartości operandu przed wyznaczeniem efektywnego adresu argumentu rozkazu (preindeksacja). Natomiast w adresowaniu pośrednim indeksowanym wartość indeksu dodawana jest po wyznaczeniu adresu (postindeksacja).

9. **Adresowanie względne**

kod operand

Operand jednobajtowy. Ten tryb adresowania dozwolony jest jedynie dla rozkazów skoków tzw. krótkich. Operand traktowany jest jako liczba dodatnia, gdy jest mniejszy od 128 lub jako liczba ujemna — równa 256 minus jego wartość, gdy jest nie mniejszy od 128. Liczba ta dodawana jest do zawartości licznika programu, wyznaczając adres, do którego zostanie wykonany skok. Ponieważ licznik programu zawsze wskazuje adres następnego do wykonania rozkazu, a rozkazy skoków „krótkich" zajmują dwa bajty pamięci, to za ich pomocą „skoczyć" można maksymalnie o 129 (127+2) bajtów do przodu i 126 (—128+2) bajtów do tyłu od adresu, w którym znajduje się rozkaz skoku. Podprogram maszynowy, w którm wykorzystano wyłącznie rozkazy skoków z tym trybem adresowania jest relokowalny, tzn. będzie działał poprawnie bez względu na obszar pamięci, w którym zostanie umieszczony. Podprogramy tak skonstruowane mogą być umieszczane w stałych tekstowych języka BASIC. Natomiast podprogramy, w których zastosowano rozkazy skoków tzw. długich (JMP i JSR), muszą rezydować w ściśle określonych miejscach pamięci — są programami nierelokowalnymi. Umieszczenie tych programów w nieodpowiednich obszarach pamięci powoduje ich złe działanie. Wyjątek stanowi użycie rozkazów JSR i JMP do wywołania procedur systemu operacyjnego — podprogram maszynowy jest w tym przypadku również relokowalny.

10. **Adresowanie akumulatora**

kod A

W tym trybie adresowania nie występuje operand. Argumentem rozkazu jest zawartość akumulatora. ATARI Assembler Editor wymaga, aby jako operand wpisywać literę A. W innych asemblerach wymaganie takie może nie istnieć.

11. **Adresowanie wewnętrzne**

kod

W tym trybie adresowania również nie występuje operand. Argumentem rozkazu z tym trybem adresowania jest z reguły zawartość jednego z rejestrów mikroprocesora lub wskaźnika z rejestru flagowego.

W przedstawionej poniżej liście rozkazów mikroprocesora 6502 opisane zostały wszystkie rozkazy dostępne dla programisty. Zostały one podzielone na kilka grup. Jako kryterium podziału zastosowano stopień podobieństwa funkcji realizowanych przez różne rozkazy. Dla każdego rozkazu, obok jego nazwy mnemonicznej, podano krótki opis realizowanej funkcji, tryby adresowania, z którymi rozkaz może być użyty, kody szesnastkowe i dziesiętne rozkazu, liczbę zajmowanych bajtów pamięci oraz wskaźniki rejestru flagowego, których stan może ulec zmianie po wykonaniu rozkazu.

ROZKAZY ZMIENIAJĄCE W SPOSÓB BEZPOŚREDNI STAN WSKAŹNIKÓW REJESTRU FLAGOWEGO

Rozkazy z tej grupy dają programiście możliwość bezpośredniej zmiany zawartości wskaźników C, D, I oraz V rejestru F. Każdy z nich zajmuje jeden bajt pamięci i może być użyty wyłącznie z wewnętrznym trybem adresowania. Rozkazami tymi są:

CLC — zeruj wskaźnik przeniesienia C;
CLD — zeruj wskaźnik dziesiętnego trybu pracy D;
CLI — zeruj wskaźnik przerwań I;
CLV — zeruj wskaźnik przepełnienia V;
SEC — ustaw wskaźnik przeniesienia;
SED — ustaw wskaźnik dziesiętnego trybu pracy;
SEI — ustaw wskaźnik przerwań.

Kody szesnastkowe i dziesiętne rozkazów:

CLC — $18/24 SEC — $38/56
CLD — $D8/216 SED — $F8/248
CLI — $58/88 SEI — $78/120
CLV — $B8/184

ROZKAZY SKOKÓW „KRÓTKICH" (WZGLĘDNYCH)

Rozkazy te umożliwiają przeniesienie sterowania w programie maszynowym maksymalnie o 129 bajtów do przodu i 126 bajtów do tyłu od ich własnego adresu w zależności od stanu jednego ze wskaźników V, Z, C i N. Każdy z nich zajmuje dwa bajty pamięci i może być użyty tylko ze względnym trybem adresowania. Nie oddziałują one na żaden ze wskaźników rejestru flagowego. Rozkazami tymi są:

BCC — skocz, gdy wskaźnik C=0;
BCS — skocz, gdy wskaźnik C=1;
BVC — skocz, gdy wskaźnik V=0;
BVS — skocz, gdy wskaźnik V=1;
BNE — skocz, gdy wskaźnik Z=0;
BEQ — skocz, gdy wskaźnik Z=1;
BPL — skocz, gdy wskaźnik N=0;
BMI — skocz, gdy wskaźnik N=1;

Kody szesnastkowe i dziesiętne rozkazów:

BCC — $90/144 BCS — $BO/176 BCV — $50/80
BVS — $70/112 BNE — $DO/208 BEQ — $FO/240
BPL — $10/16 BMI — $30/48

ROZKAZY SKOKÓW „DŁUGICH" (BEZWZGLĘDNYCH)

Są dwa rozkazy umożliwiające przeniesienie sterowania w programie maszynowym w dowolne jego miejsce. Rozkazami tymi są: JMP i JSR.

JMP — skocz do adresu określonego przez operand.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	JMP 3000	$4C/76	3
pośrednie	JMP (3000)	$6C/108	3

JMP nie oddziałuje na żadne wskaźniki rejestru F. Używając tego rozkazu z pośrednim trybem adresowania, należy zwrócić uwagę, aby określony przez operand wskaźnik zawierający adres skoku nie znajdował się na granicy dwóch stron pamięci. Na przykład: w rozkazie JMP ($OBFF) młodszy bajt wskaźnika jest ostatnim bajtem strony $OB, a starszy bajt pierwszym bajtem strony $OC. W takiej sytuacji adres skoku jest wyznaczany przez mikroprocesor 6502 w sposób błędny. Natomiast, gdy wskaźnik z adresem skoku nie jest położony na granicy stron pamięci — JMP z pośrednim trybem adresowania działa prawidłowo.

JSR — wykonaj podprogram o adresie określonym przez operand.

Licznik programu jest zwiększony o 2 ((PC)=(PC)+2)) i przesyłany jest na stos. Następnie sterowanie przekazywane jest do rozkazu, którego adres określany jest przez operand. Powrót z podprogramu (tj. wykonanie instrukcji następującej po JSR) realizowany jest za pomocą rozkazu RTS, który powinien być ostatnim rozkazem wywołanego podprogramu. Rozkaz JSR nie oddziałuje na żadne wskaźniki rejestru flagowego. Jedyny dostępny dla tego rozkazu tryb adresowania to adresowanie absolutne, np.: JSR $E456. Kod rozkazu: $20/32. JSR zajmuje trzy bajty pamięci.

ROZKAZY PRZESYŁAJĄCE ZAWARTOŚĆ BAJTU Z PAMIĘCI DO REJESTRÓW A, X, Y

Są trzy takie rozkazy:

LDA — załaduj do akumulatora bajt z pamięci;
LDX — załaduj do rejestru X bajt z pamięci;
LDY — załaduj do rejestru Y bajt z pamięci;

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l.zaj. bajtów
LDA			
natychmiastowe	LDA #14	$A9/169	2
absolutne	LDA 3000	$AD/173	3
absolutne indeksowane X	LDA 3000, X	$BD/189	3
absolutne indeksowane Y	LDA 3000,Y	$B9/185	3
strony zerowej	LDA $A9	$A5/165	2
strony zerowej indeks. X	LDA $A9,X	$B5/181	2
indeksowane pośrednie	LDA ($A9,X)	$A1/161	2
pośrednie indeksowane	LDA ($A9),Y	$B1/177	2
LDX			
natychmiastowe	LDX #14	$A2/162	2
absolutne	LDX 3000	$AE/174	2
absolutne indeksowane Y	LDX 3000, Y	$BE/190	3
strony zerowej	LDX $A9	$A6/166	2
strony zerowej indeks. Y	LDX $A9,Y	$B6/182	2
LDY			
natychmiastowe	LDY #14	$AO/160	2
absolutne	LDY 3000	$AC/172	3
absolutne indeksowane X	LDY 3000, X	$BC/188	3
strony zerowej	LDY $A9	$A4/164	2
strony zerowej indeks. X	LDY $A9, X	$B4/180	2

Rozkazy LDA, LDX, LDY oddziałują na wskaźniki N i Z. Mogą ustawić wskaźnik N, gdy ładowany do rejestru bajt ma najbardziej znaczący bit równy 1. Wskaźnik Z zostanie ustawiony, gdy ładowany do rejestru bajt zawiera zero.

ROZKAZY PRZESYŁAJĄCE ZAWARTOŚĆ REJESTRÓW A, X, Y DO PAMIĘCI

Są to rozkazy komplementarne w stosunku do rozkazów z poprzedniej grupy. Również są trzy takie rozkazy.

STA — zapamiętaj zawartość akumulatora w pamięci;
STX — zapamiętaj zawartość rejestru X w pamięci;
STY — zapamiętaj zawartość rejestru Y w pamięci;

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
STA			
absolutne	STA $D300	$8D/141	3
absolutne indeksowane X	STA $D300, X	$9D/157	3
absolutne indeksowane Y	STA $D300, Y	$99/153	3
strony zerowej	STA 240	$85/133	2
strony zerowej indeks. X	STA 240, X	$95/149	2
indeksowane pośrednie	STA (240, X)	$81/129	2
pośrednie indeksowane	LDA (240), Y	$91/145	2
STX			
absolutne	STX $D300	$8E/142	3
strony zerowej	STX $FO	$86/134	2
strony zerowej indeks. Y	STX $FO, Y	$96/150	2
STY			
absolutne	STY 3000	$8C/140	3
strony zerowej	STY 129	$84/132	2
strony zerowej indeks. X	STY 129, X	$94/148	2

Rozkazy STA, STX, STY nie oddziałują na żadne wskaźniki rejestru flagowego.

ROZKAZY PRZESŁAŃ POMIĘDZY REJESTRAMI A, X, Y

Rozkazy te najczęściej służą do chwilowego przechowania zawartości jednego rejestru w innym. Można również przechowywać zawartość rejestrów w pamięci. Jednak to drugie rozwiązanie jest gorsze. Rozkazy przesyłające zawartość rejestrów z/do pamięci wykonywane są w czasie średnio dwukrotnie dłuższym i zajmują więcej pamięci.

TAX — prześlij zawartość akumulatora do rejestru X;
TAY — prześlij zawartość akumulatora do rejestru Y;
TXA — prześlij zawartość rejestru X do akumulatora;
TYA — prześlij zawartość rejestru Y do akumulatora;

Kody rozkazów:
TAX — $AA/170 TAY — $A8/168
TXA — $8A/138 TYA — $98/152

Do tej grupy należałoby również zaliczyć dwa rozkazy umożliwiające programiście zmianę zawartości wskaźnika stosu (rejestru S). Zmiana zawartości tego rejestru nie może być dokonana w sposób bezpośredni, a jedynie poprzez rejestr X. Rozkazami tymi są:

TSX — prześlij zawartość ośmiu mniej znaczących bitów rejestru S do rejestru X;
TXS — prześlij zawartość rejestru X do ośmiu mniej znaczących bitów rejestru S;

Każdy z tych rozkazów zajmuje jeden bajt pamięci i może być użyty wyłącznie z wewnętrznym trybem adresowania. Oddziałują one, za wyjątkiem TXS, na wskaźniki N i Z rejestru flagowego. TXS może zmienić stan wszystkich wskaźników rejestru F.
TSX — $BA/186 TXS — $9A/154

ROZKAZY MODYFIKUJĄCE ZAWARTOŚĆ REJESTRÓW INDEKSOWYCH

Rejestry indeksowe wykorzystywane są w programie maszynowym do konstruowania pętli, w których służą jako modyfikatory adresów przetwarzanych obszarów pamięci i jednocześnie jako licznik pętli. Podobną funkcję spełniają indeksy tablic w języku BASIC. Licznik każdej pętli programowej wymaga modyfikacji — zmniejszania lub zwiększania o wartość kroku po każdorazowym wykonaniu ciągu rozkazów zawartych w pętli. Mikroprocesor 6502 dostarcza programiście bardzo wygodnych rozkazów do realizacji tego zadania. Rozkazami tymi są:

INX — zwiększ o jeden zawartość rejestru X;
DEX — zmniejsz o jeden zawartość rejestru X;
INY — zwiększ o jeden zawartość rejestru Y;
DEY — zmniejsz o jeden zawartość rejestru Y;

Kody rozkazów:
INX — $E8/232 DEX — $CA/202 INY — $C8/200
DEY — $88/136

Każdy z tych rozkazów zajmuje jeden bajt pamięci i może być użyty wyłącznie z wewnętrznym trybem adresowania. Ich wykonanie może zmienić stan wskaźników N i Z rejestru flagowego. Do tej grupy rozkazów należałoby również zaliczyć dwa rozkazy przeznaczone do zmniejszania lub zwiększania o jeden zawartości bajtu z pamięci:

INC — zwiększ o jeden zawartość bajtu z pamięci o adresie wskazanym przez operand;
DEC — zmniejsz o jeden zawartość bajtu o adresie wskazanym przez operand;

Dzięki nim w łatwy sposób można wykorzystać bajt z pamięci jako licznik pętli programowej.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
INC			
absolutne	INC $D300	$EE/238	3
absolutne indeksowane X	INC $D300, X	$FE/254	3
strony zerowej	INC 240	$E6/230	2
strony zerowej indeks. X	INC 240, X	$F6/246	2
DEC			
absolutne	DEC $D300	$CE/206	3
absolutne indeksowane X	DEC $D300, X	$DE/222	3
strony zerowej	DEC 240	$C6/198	2
strony zerowej indeks. X	DEC 240, X	$D6/214	2

Rozkazy INC i DEC mogą zmienić stan wskaźników N i Z rejestru flagowego.

ROZKAZY PORÓWNUJĄCE ZAWARTOŚĆ REJESTRÓW A, X, Y Z ZAWARTOŚCIĄ PAMIĘCI

Rozkazy z tej grupy odejmują zawartość bajtu z pamięci lub (gdy użyte są z natychmiastowym trybem adresowania) wartość operandu od zawartości rejestru. W zależności od wyniku odejmowania zmieniają one stan wskaźników N, Z i C.
N — ustawiany jest, gdy bajt w pamięci jest większy od zawartości rejestru;
Z — ustawiany jest, gdy porównywane bajty są równe;
C — ustawiany jest, gdy bajt w pamięci jest mniejszy od zawartości rejestru;
Należy zaznaczyć, że rozkazy te nie zmieniają zawartości porównywanych bajtów. Rozkazami tymi są:

CMP — porównaj zawartość akumulatora z zawartością bajtu z pamięci;
CPX — porównaj zawartość rejestru X z zawartością bajtu z pamięci;
CPY — porównaj zawartość rejestru Y z zawartością bajtu z pamięci;

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
CMP			
natychmiastowe	CMP #14	$C9/201	2
absolutne	CMP 3000	$CD/205	3
absolutne indeksowane X	CMP 3000,X	$DD/221	3
absolutne indeksowane Y	CMP 3000,Y	$D9/217	[illegible]
strony zerowej	CMP $A9	$C5/197	[illegible]
strony zerowej indeks. X	CMP $A9,X	$D5/213	2
indeksowane pośrednie	CMP ($A9,X)	$C1/193	2
pośrednie indeksowane	CMP ($A9,Y)	$D1/209	[illegible]
CPX			
natychmiastowe	CPX #14	$E0/224	2
absolutne	CPX 3000	$EC/236	3
strony zerowej	CPX $A9	$E4/228	2
CPY			
natychmiastowe	CPY #14	$C0/192	2
absolutne	CPY 3000	$CC/204	3
strony zerowej	CPY $A9	$C4/196	2

ROZKAZY WYKONUJĄCE OPERACJE NA STOSIE MIKROPROCESORA

PHA — prześlij akumulator na stos i zmniejsz o jeden wskaźnik stosu;
PHP — prześlij rejestr flagowy na stos i zmniejsz o jeden wskaźnik stosu;
PLA — prześlij bajt ze szczytu stosu do akumulatora i zwiększ o jeden wskaźnik stosu;
PLP — prześlij bajt ze szczytu stosu do rejestru flagowego i zwiększ o jeden wskaźnik stosu;
Rozkazy te są rozkazami jednobajtowymi i mogą być użyte tylko z wewnętrznym trybem adresowania. PLA może zmienić stan wskaźników N i Z rejestru flagowego, natomiast PHP i PHA nie oddziałują na żadne wskaźniki. PLP może zmienić stan wszystkich wskaźników.
Kody rozkazów:

PHA	$48/72	PLA	$68/104
PHP	$08/8	PLP	$28/40

ROZKAZY WYKONUJĄCE OPERACJE ARYTMETYCZNE I LOGICZNE

ADC — dodaj zawartość bajtu z pamięci lub wartość operandu (adresowanie natychmiastowe) do zawartości akumulatora. Wynik dodawania pozostaje w akumulatorze. Wynik dodawania może być zwiększony o jeden, gdy przed wykonaniem rozkazu ustawiony był wskaźnik C. Dlatego zawsze przed użyciem tego rozkazu należy zerować wskaźnik C (rozkaz CLC), chyba że wymagana jest korekta wyniku (dodawanie liczb binarnych zapisanych na kilku bajtach). ADC może zmienić stan wskaźników N, V, Z i C rejestru flagowego.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	ADC #$80	$69/105	2
absolutne	ADC $D300	$6D/109	3
absolutne indeksowane X	ADC $D300,X	$7D/125	3
absolutne indeksowane Y	ADC $D300,Y	$79/121	3
strony zerowej	ADC 240	$65/101	2
strony zerowej indeks X	ADC 240,X	$75/117	2
indeksowane pośrednie	ADC (240,X)	$61/97	2
pośrednie indeksowane	ADC (240),Y	$71/113	2

AND — wykonaj iloczyn logiczny (AND) zawartości bajtu z pamięci i zawartości akumulatora. Rozkaz porównuje kolejno wszystkie bity bajtu z pamięci i akumulatora. Gdy porównywane bity są równe jeden, to odpowiadający im bit wyniku jest również równy jeden. W każdym innym przypadku bit wyniku jest zerem. Wynik rozkazu pozostaje w akumulatorze.
Przykład:

```
bit      7 6 5 4 3 2 1 0
         1 0 1 1 1 1 0 1   bajt z pomięci
AND      0 0 0 0 1 1 0 0   bajt z akumulatora
         — — — — — — — —
         0 0 0 0 1 1 0 0   wynik w akumulatorze
```

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	AND #$80	$29/41	2
absolutne	AND $D300	$2D/45	3
absolutne indeksowane X	AND $D300,X	$3D/61	3
absolutne indeksowane Y	AND $D300,Y	$39/57	3
strony zerowej	AND 240	$25/37	[illegible]
strony zerowej indeks. X	AND 240,X	$35/53	[illegible]
indeksowane pośrednie	AND (240,X)	$21/33	[illegible]
pośrednie indeksowane	AND (240),Y	$31/49	2

AND oddziałuje na wskaźniki N i Z rejestru flagowego.

ASL — przesuń zawartość bajtu z akumulatora lub z pamięci o jedną pozycję bitową w lewo. Najbardziej znaczący bit przesuwanego bajtu przenoszony jest do wskaźnika C, bit najmniej znaczący jest zerowany. Wykonywana operacja przesuwania w lewo odpowiada mnożeniu zawartości bajtu przez dwa. ASL oddziałuje na wskaźnik N, Z i C.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	ASL $D300	$0E/14	3
absolutne indeksowane X	ASL $D300,X	$1E/30	3
strony zerowej	ASL 240	$06/6	2
strony zerowej indeks. X	ASL 240,X	$16/22	2
akumulatora	ASL A	$0A/10	1

BIT — wykonaj iloczyn logiczny zawartości bajtów z pamięci i akumulatora. Rozkaz ten działa podobnie jak rozkaz AND, z tą tylko różnicą, że nie zmienia zawartości porównywanych bajtów i oddziałuje na inne wskaźniki rejestru F. BIT zapala wskaźnik Z, gdy bajty są równe, ponadto do wskaźnika N kopiowany jest siódmy bit bajtu z pamięci, a do wskaźnika V — szósty bit tego bajtu. Na stan pozostałych wskaźników BIT nie oddziałuje.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	BIT $D300	$2C/44	3
strony zerowej	BIT 240	$24/36	2

EOR — wykonaj logiczną różnicę symetryczną (exclusive OR) na zawartościach bajtu z pamięci i akumulatora. Rozkaz ten porównuje kolejno bity bajtu z pamięci i akumulatora. Jeże-

li porównywane bity są jedynkami, to odpowiadający im bit wyniku jest zerem; gdy są zerami to odpowiadający im bit wyniku jest również zerem; natomiast w sytuacji, gdy przynajmniej jeden z porównywanych bitów jest jedynką, to odpowiadający im bit wyniku jest jedynką.

Przykład:

```
bit    7 6 5 4 3 2 1 0
       1 0 1 1 1 1 0 1   bajt z pamięci
EOR    0 0 0 0 1 1 0 0   bajt z akumulatora
       — — — — — — — —
       1 0 1 1 0 0 0 1   wynik w akumulatorze
```

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	EOR #$80	$49/73	2
absolutne	EOR $D300	4D/77	3
absolutne indeksowane X	EOR $D300,X	$5D/93	3
absolutne indeksowane Y	EOR $D300,Y	$59/89	3
strony zerowej	EOR 240	$45/69	[illegible]
strony zerowej indeks. X	EOR 240,X	$55/85	2
indeksowane pośrednie	EOR (240,X)	$41/65	2
pośrednie indeksowane	EOR (240),Y	$51/81	2

LSR — przesuń zawartość bajtu z pamięci lub z akumulatora o jedną pozycję bitową w prawo. Najmniej znaczący bit przesuwanego bajtu przenoszony jest do wskaźnika C, bit najbardziej znaczący jest zerowany. Wykonywana operacja przesuwania w prawo odpowiada dzieleniu zawartości bajtu przez dwa. LSR oddziałuje na wskaźnik N, Z i C. Ustawiony wskaźnik C świadczy o powstałej w wyniku dzielenia reszcie.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	LSR $D300	$4E/78	3
absolutne indeksowane X	LSR $D300,X	$5E/94	3
strony zerowej	LSR 240	$46/70	2
strony zerowej indeks. X	LSR 240,X	$56/86	2
akumulatora	LSR A	$4A/74	1

ORA — wykonaj sumę logiczną (OR) na zawartościach bajtów z pamięci i akumulatora. Rozkaz porównuje kolejno wszystkie bity bajtów z pamięci i akumulatora. Jeżeli porównywane bity są zerami, to odpowiadający im bit wyniku jest również zerem. Dla każdej innej kombinacji porównywanych bitów bit wyniku jest jedynką. Wynik operacji pozostawiony jest w akumulatorze.

Przykład:

```
bit    7 6 5 4 3 2 1 0
       1 0 1 1 1 1 0 1   bajt z pamięci
EOR    0 0 0 0 1 1 0 0   bajt z akumulatora
       — — — — — — — —
       1 0 1 1 1 1 0 1   wynik w akumulatorze
```

Rozkaz może zmienić stan wskaźników N i Z rejestru F.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	ORA #$80	$09/9	2
absolutne	ORA $D300	$0D/13	3
absolutne indeksowane X	ORA $D300,X	$1D/29	3
absolutne indeksowane Y	ORA $D300,Y	$19/25	3
strony zerowej	ORA 240	$05/5	2
strony zerowej indeks. X	ORA 240,X	$15/21	2
indeksowane pośrednie	ORA (240,X)	$01/1	2
pośrednie indeksowane	ORA (240),Y	$11/17	2

ROL — „obróć" zawartość bajtu z pamięci lub akumulatora o jedną pozycję w lewo. Rozkaz działa podobnie jak ASL. Zawartość „obracanego" bajtu przesuwana jest o jedną pozycję bitową w lewo, przy tym do bitu zerowego wprowadzana jest zawartość wskaźnika C (ASL zeruje ten bit), a bit siódmy kopiowany jest do wskaźnika C i stanowi jego nową wartość. Rozkaz ROL wspólnie z rozkazem ASL używane są do mnożenia przez dwa liczb zapisanych na kilku bajtach. Poza wskaźnikiem C rozkaz ROL może zmienić stan wskaźników N i Z.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	ROL $D300	$2E/46	3
absolutne indeksowane X	ROL $D300,X	$3E/62	3
strony zerowej	ROL 240	$26/38	2
strony zerowej indeks. X	ROL 240,X	$36/54	2
akumulatora	ROL A	$2A/42	1

ROR — „obróć" zawartość bajtu z pamięci lub akumulatora o jedną pozycję bitową w prawo. Rozkaz działa podobnie jak LSR. Zawartość „obracanego" bajtu przesuwana jest o jedną pozycję bitową w prawo, przy tym do bitu siódmego wprowadzana jest zawartość wskaźnika C (LSR zeruje ten bit), a bit zerowy kopiowany jest do wskaźnika C i stanowi jego nową wartość. Rozkaz ROR wspólnie z rozkazem LSR używane są do dzielenia przez dwa liczb zapisanych na kilku bajtach. Poza wskaźnikiem C rozkaz ROR może zmienić stan wskaźników N i Z.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	ROR $D300	$6E/110	3
absolutne indeksowane X	ROR $D300,X	$7E/126	3
strony zerowej	ROR 240	$66/102	2
strony zerowej indeks. X	ROR 240,X	76/118	2
akumulatora	ROR A	$6A/106	1

SBC — odejmij zawartość bajtu z pamięci lub wartość operandu (adresowanie natychmiastowe) do zawartości akumulatora. Wynik odejmowania pozostaje w akumulatorze. Wynik ten może być zmniejszony o jeden, gdy przed wykonaniem rozkazu zgaszony był (równy zero) wskaźnik C. Dlatego zawsze przed użyciem tego rozkazu należy „zapalić" wskaźnik C (rozkaz SEC). SBC może zmienić stan wskaźników N, V, Z i C rejestru flagowego.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	SBC #$80	$E9/233	2
absolutne	SBC $D300	$ED/237	3
absolutne indeksowane X	SBC $D300,X	$FD/253	3
absolutne indeksowane Y	SBC $D300,Y	$F9/249	3
strony zerowej	SBC 240	$E5/229	[illegible]
strony zerowej indeks. X	SBC 240,X	$F5/245	2
indeksowane pośrednie	SBC (240,X)	$E1/225	2
pośrednie indeksowane	SBC (240),Y	$F1/241	2

POZOSTAŁE ROZKAZY

BRK — rozkaz ten generuje przerwanie programowe i jest stosowany przy testowaniu poprawności działania skonstruowanego programu maszynowego. W testowanym programie umieszcza się, w odpowiednich miejscach, rozkaz BRK. Gdy zostanie on wykonany realizacja testowanego programu zostanie przerwana. Wówczas za pomocą odpowiedniej procedury monitorującej można „obejrzeć" zawartość rejestrów mikroprocesora i określonych obszarów pamięci, oceniając na ich podstawie poprawność działania wykonanego programu bądź jego fragmentu. Rozkaz ten zapala wskaźnik B rejestru F. Rozkaz BRK zajmuje jeden bajt pamięci i może być stosowany jedynie z wewnętrznym trybem adresowania. Jego kod: $00/0.

NOP — nic nie rób. Rozkaz ten zatrzymuje na okres dwóch cykli zegarowych pracę mikroprocesora. Może być stosowany do budowania pętli opóźniających lub można nim zastąpić, po zakończeniu testowania, rozkazy BRK wstawione w tym celu w program maszynowy. NOP zajmuje jeden bajt pamięci, nie oddziałuje na żadne wskaźniki rejestru F i może być użyty tylko z wewnętrznym trybem adresowania. Jego kod: $EA/234.

RTI — „wróć z przerwania". Rozkaz ten służy jako „wyjście" z procedur obsługujących niektóre rodzaje przerwań mikroprocesora. Odtwarza on, zapamiętane na stosie, rejestr F i licznik programu i aktualizuje wskaźnik stosu. RTI zajmuje jeden bajt pamięci i musi być używany tylko z wewnętrznym trybem adresowania. Jego kod: $40/64.

RTS — „wróć z podprogramu". Rozkaz ten służy jako „wyjście" z podprogramów wywoływanych rozkazem JSR. Odtwarza on, zapamiętany na stosie, licznik programu i zwiększa wartość wskaźnika stosu o dwa. Nie oddziałuje na żadne wskaźniki rejestru F. Zajmuje jeden bajt pamięci i jest dostępny tylko z wewnętrznym trybem adresowania. Jego kod: $60/96.

Umieszczone poniżej przykłady podprogramów powinny ułatwić zrozumienie zasad tworzenia procedur w języku maszynowym. W tym celu kody źródłowe tych procedur opatrzone zostały licznymi komentarzami. Ostatni program, napisany w BASIC-u, przeznaczony jest do tworzenia linii DATA, zawierających powstały w wyniku asemblacji kod maszynowy zapisany na taśmie lub dysku. Utworzone linie DATA program ten również zapisuje na taśmę lub dysk. Można je dołączyć instrukcją ENTER do tworzonego programu w języku BASIC.

H. KRASUSKI

Bibliografia:

1. ZAKS Rodney: Programming the 6502, 6502 Series-Volume. SYBEX Inc., 1980 r.
2. LAVENTHAL L.A.: 6502 Assembly Language Programming. Osborne. McGraw-Hill Inc., 1979 r.


```
01        .OPT OBJ
02 ;Ta linia zawiera informacje dla assemblera
03        *= $0600   ;od tego adresu
04 ;asembler umiesci kod wynikowy podprogramu
10 ;***********************************************
20 ;             Podprogram 1                     *
30 ;***********************************************
40 ;podprogram umozliwia   odczyt/zapis z   dyskietki
50 ;sektora o dowolnym numerze.
60 ;Sposob wywolania z jezyka BASIC:
70 ;    X=USR(ADRES,NRD,FUN,ADR(SEK$),NRSEK)
80 ;gdzie:
90 ;NRD    - zmienna zawierajaca nr drive'u
0100 ;FUN    - kod funkcji do wykonania
0110 ; FUN=82  odczyt sektora
0120 ; FUN=87  zapis sektora
0130 ;SEK$   - zmienna tekstowa o dlugosci 128 bajtow
0140 ;z/do ktorej zapisywany/odczytywany jest sektor
0150 ;NRSEK   - zmienna zawierajaca nr sektora
0160 ;do odczytu/zapisu
0170 ;Po wykonaniu podprogramu zmienna X zawiera 1
0180 ;gdy odczyt/zapis zakonczyl sie pomyslnie
0190 ;kod bledu w przeciwnym wypadku
0200 ;
0210 ;deklaracja zmiennych dla programu maszynowego
0220 ;
0230 NRD     = $0301 ;nr drive'u
0240 FUN     = $0302 ;kod funkcji
0250 STATUS  = $0303 ;wynik wykonania
0260 SEKM    = $0304 ;adres SEK$ -mlodszy bajt
0270 SEKS    = $0305 ;adres SEK$ -starszy bajt
0280 NRSM    = $030A ;nr sektora -mlodszy bajt
0290 NRSS    = $030B ;nr sektora -starszy bajt
0330         PLA      ;usun ze stosu liczbe param.
0340         PLA      ;pobierz ze stosu st. bajt NRD
0350         PLA      ;pobierz ze stosu ml. bajt NRD
0360         STA NRD  ;zapamietaj go w zmiennej NRD
0370         PLA      ;zrob to samo z
0380         PLA      ;kodem funkcji do wykonania
0390         STA FUN  ;zapamietaj kod funkcji w FUN
0400         PLA      ;st.bajt adresu SEK$
0410         STA SEKS ;zapamietaj w SEKS
0420         PLA      ;ml. bajt tego adresu
0430         STA SEKM ;zapamietaj w SEKM
0440         PLA      ;st. bajt numeru sektora
0450         STA NRSS ;zapamietaj w NRSS
0460         PLA      ;a mlodszy bajt
0470         STA NRSM ;zapamietaj w NRSM
0480         JSR $E453  ;skocz do procedury SIOV
0490         LDA STATUS ;wynik zap/odcz.do akumulatora
0500         STA $D4    ;zapamietaj wynik w bajtach
0510         LDA #0     ;$D4 i $D5 strony zerowej
0520         STA $D5
0530         RTS        ;wyjdz z podprogramu
0540         .END
```

```
10 ;***********************************************
20 ;            Podprogram 2                      *
30 ;***********************************************
40 ;Podprogram zamienia komunikaty /stale i zmienne
50 ;tekstowe/ z trybu "normalnego" na "invers video"
60 ;i odwrotnie. UWAGA: dlugosc komunikatu
70 ;nie moze przekraczac 255 bajtow
80 ;Sposob wywolania z jezyka BASIC
90 ;    X=USR(ADRES,ADR(KOM$),LEN(KOM$))
0100 ;gdzie:
0110 ;KOM$ stala lub zmienna tekstowa, ktora ma byc
0120 ;zmieniona
0130 ;
0140      PLA          ;usun ze stosu liczb. param.
0150      PLA          ;pob. ze stosu st. bajt adresu
0160      STA 204      ;KOM$ i zapam. go w bajcie 204
0170      PLA          ;a mlodszy bajt tego adresu
0180      STA 203      ;zapamietaj w bajcie 203
0190      PLA          ;dlugosc stalej /tylko jeden
0200      PLA          ;bajt/ zapamietaj w bajcie
0210      STA 205      ;205
0220      LDY #0       ;zero do rejestru Y
0230 ZAM  LDA (203),Y  ;zaladuj kolejny bajt stalej
0240 ;                 KOM$ do akumulatora
0250      EOR #$80     ;zmien tryb
0260      STA (203),Y  ;przeslij zmieniony bajt
0270 ;                 z powrotem na jego miejsce
0280      INY          ;zwieksz o jeden liczbe zmie-
0290 ;                 nionych bajtow
0300      CPY 205      ;czy zamienione wszystkie
0310      BNE ZAM      ;nie, to zamien kolejny
0320      RTS          ;tak, to wyjdz z podprogramu
0330      .END
```

```
10 ;***********************************************
20 ;              Podprogram 3                    *
30 ;***********************************************
40 ;Podprogram przesyla dane tworzace obraz
50 ;z dowolnego obszaru pamieci do obszaru
60 ;pamieci ekranu
62 ;tryb w jakim wyswietlany jest obraz musi byc
64 ;ustawiony z poziomu jezyka BASIC instr. GRAPHIC
65 ;przed wywolaniem podprogramu
90 ;Sposob wywolania z jezyka BASIC:
0100 ;      X=USR(ADRES,ADR(DANE0$),LB)
0110 ;gdzie:
0120 ;DANE0$ - stala tekstowa zawierajaca dane
0130 ;tworzace obraz
0135 ;LB - dlugosc DANE0$ w bajtach
0140 ;
0150 SKADM = $D6     ;mlodszy bajt adresu stalej DANE0$
0160 SKADS = $D7     ;starszy bajt tego adresu
0162 LBLOK = $D9     ;liczba blokow 256 bajtowych
0164 RESZT = $D8     ;reszta bajtow do przeslania
0166 ;(LBLOK*256)+RESZT = liczbie bajtow do przesl.
0170       PLA       ;usun ze stosu bajt z liczba param.
0180       PLA       ;pob. ze stosu st. bajt adresu
0190       STA SKADS;i zapamietaj go w SKADS
0200       PLA       ;pob. ze stosu ml. bajt adresu
0210       STA SKADM;zapamietaj go w SKADM
0212       PLA       ;pob. liczbe blokow
0213       STA LBLOK;zapamietaj ja w LBLOK
0214       PLA       ;pobierz reszte
0215       STA RESZT;zapamietaj ja w RESZT
0220       LDX LBLOK;liczba blokow do rej. X
0230       LDY #0    ;zero do rej. Y
0240 P1    LDA (SKADM),Y ;pob. kolejny bajt z DANE0$
0250       STA (88),Y    ;zapamietaj go w obszarze
0260 ;               pamieci ekranu
0270       DEY       ;czy przeslany caly blok
0280       BNE P1    ;nie, to skocz do etykiety P1
0290       INC SKADS;tak, zwieksz o 1 st. bajt adresu
0300 ;               DANE0$
0310       INC 89    ;zwieksz o jeden starszy bajt
0320 ;               adresu pamieci ekranu
0330       DEX       ;odejmij 1 od liczby blokow
0335       BMI KON   ;zakoncz przesylanie
0340       BNE P1    ;skocz do P1, gdy liczba blokow
0350 ;               nie jest zerem
0352       LDY RESZT;reszta do rej. Y
0354       BNE P1    ;gdy RESZT<>0 to przesylaj
0360 KON   RTS       ;gdy zero wyjdz z podprogramu
0370       .END
0380 ;UWAGA: podprogram ten bardzo szybko zmienia
0390 ;i wyswietla ekrany w trybie graficznym.
0400 ;Jezeli uzywa sie go do wyswietlania ekranow
0410 ;w trybie tekstowym, to DANE0$ musi zawierac
0420 ;kody PEEK/POKE wyswietlanych znakow a nie
0430 ;ich kody ATASCII.
```

```
1 REM ******************************
2 REM *        PROGRAM 4           *
3 REM ******************************
10 ? CHR$(125)
20 ? "Program tworzy linie DATA z kodu"
30 ? "maszynowego zapisanego na tasmie"
40 ? "lub dysku. Zbior z kodem
maszynowym"
50 ? "musi miec format wymagany przez
DOS"
100 DIM CIO$(6),BUF$(10000),LINIA$(92),
F$(14),KOD$(3)
110 CIO$(1)=CHR$(104):CIO$(2)=CHR$(162)
:CIO$(3)=CHR$(32)
120 CIO$(4)="L":CIO$(5)="V":CIO$(6)=CHR
$(228)
130 TRAP 130:? "Podaj urzadzenie i nazwe
"
131 ? "zbioru z kodem maszynowym: "
:INPUT F$
140 IF F$(1,2)<>"C:" AND F$(1,2)<>"D:"
THEN ? CHR$(253):GOTO 130
150 TRAP 2000:CLOSE #2:OPEN #2,4,0,F$
160 IOCB=864:POKE IOCB+2,7
165 SB=INT(ADR(BUF$)/256):MB=ADR(BUF$)-
SB*256
170 POKE IOCB+4,MB:POKE IOCB+5,SB
175 POKE IOCB+8,0:POKE
IOCB+9,PEEK(742)-PEEK(145)
180 X=USR(ADR(CIO$)):CLOSE #2:TRAP
40000
185 AD=ADR(BUF$)
190 IF PEEK(AD)<>255 AND PEEK(AD+1)
<>255 THEN 2100
195 AP=PEEK(AD+3)*256+PEEK(AD+2)
200 AK=PEEK(AD+5)*256+PEEK(AD+4)
205 IF AK-AP+1>10000 THEN GOTO 2200
210 ? :? CHR$(253);CHR$(253);"zbior
";F$;" OK"
215 TRAP 215:? :? "Podaj urzadzenie i
nazwe zbioru"
220 ? "do ktorego beda wpisane linie
DATA"
225 F$="":INPUT F$
230 IF F$(1,2)<>"C:" AND F$(1,2)<>"D:"
THEN ? CHR$(253):GOTO 215
236 ? "podaj poczatkowy nr linii DATA:
":INPUT NRL
237 ? "podaj krok numeracji: ":INPUT
KROK
239 TRAP 2300:CLOSE #2:OPEN
#2,8,0,F$:TRAP 40000
240 FOR I=0 TO AK-AP STEP 20
245    FOR J=1 TO 92:LINIA$(J)=" ":NEXT
J
250    J=LEN(STR$(NRL))
255    LINIA$(1,J)=STR$(NRL):J=J+2:
LINIA$(J)="DATA":J=J+6
260    FOR K=0 TO 19
270     N=PEEK(AD+6+I+K):KOD$="000":KOD$
(4-LEN(STR$(N)))=STR$(N)
275     LINIA$(J)=KOD$:J=J+3
280     IF K<>19 THEN
LINIA$(J)=",":J=J+1
285    NEXT K
300    LINIA$(J)=CHR$(155):PRINT
#2,LINIA$ :NRL=NRL+KROK:NEXT I
305 CLOSE #2
310 ? CHR$(253);CHR$(253)
315 ? :? "Gotowe..."
320 ? "Chcesz utworzyc nastepny zbior
(T/N)"
325 CLOSE #1:OPEN #1,4,0,"K:":GET #1,K
330 IF K<>84 THEN END
335 CLR :GOTO 20
2000 ? CHR$(253);"BLAD NR: ";PEEK(195);"
PRZY ODCZYCIE ZBIORU ";F$
2010 CLR :GOTO 20
2100 ? CHR$(253);"ZBIOR ";F$;" NIE JEST
W FORMACIE DOS-a"
2102 ? "NAGLOWEK: ";PEEK(AD);" ";PEEK
(AD+1);" ";
2104 ? PEEK(AD+2);" ";PEEK(AD+3);" ";
PEEK (AD+4);" ";PEEK(AD+5)
2110 CLR :GOTO 20
2200 ? CHR$(253);"ZBIOR ";F$;" ZAJMUJE
WIECEJ NIZ 10000 BAJTOW"
2210 CLR :GOTO 20
2300 ? CHR$(253);"BLAD NR ";PEEK(195);"
PRZY ZAPISIE ZBIORU ";F$
2310 CLR :GOTO 20
```



KTO POMOŻE?

Czy stara, wysłużona Odra 1305 może uchodzić za symbol nowoczesności? — Pytanie retoryczne i dlatego kierownictwo Wojskowych Zakładów Uzbrojenia nr 1 w Krakowie zaczęło rozglądać się za czymś nowszym. Oczywiście nie po to tylko, by mieć nowy symbol. — *Widzimy szansę przejścia „do przerobu" przez nasz ośrodek wielu prac planistycznych i obliczeniowych.* — *Co na tym zyskają zakłady?* — *Przede wszystkim zmniejszymy zatrudnienie w administracji* — zapewnia dyrektor płk Ireneusz **Rosiak**. A nie jest to błacha sprawa, ponieważ każdy pracownik kosztuje zakłady 650 tys. zł rocznie. Na tę sumę składa się pensja, różne składki, obciążenia, podatki — i to bez względu na to, czy ten ktoś pracuje, czy też — przykładowo — przebywa na zwolnieniu lekarskim. — *A komputer nie choruje i jest nadzwyczaj dyspozycyjny* — tę prawdę krakowianie odkryli już dawno.

Oprócz redukcji administracji możliwe jest zwiększenie efektywności gospodarki materiałowej i kadrowej. Lepsze wykorzystanie dwóch najważniejszych czynników produkcji w zakładzie, to dodatkowy zysk... Tak się bowiem składa, że z maszyn i urządzeń niewiele da się już „wycisnąć", ponieważ wykorzystanie czasu pracy maszyn wynosi 96 proc.!

Ale to nie wszystko. Dyrekcja widzi możliwość „zaprzęgnięcia" komputera do planowania produkcji. Nie chodzi przy tym o sporządzanie prostych zestawień, lecz o coś na pograniczu fantastyki — oczywiście w naszych krajowych wyrunkach. Tym „czymś" byłyby dodatkowe korzyści wynikające nie z produkcji w ogóle, lecz z produkcji wykonanej w określonym czasie. Co to daje? — Lepiej jest wykorzystywany potencjał przedsiębiorstwa i zadowoleni klienci, ponieważ np. remont sprzętu odbywa się w dogodnych terminach dla odbiorcy.

Wszystko to być może... Póki co, nie ma jednak chętnych do „skomputeryzowania" zakładów. Podstawową przeszkodą jest to, że przedsiębiorstwo wykonuje produkcję dla potrzeb sił zbrojnych, a więc występuje problem tajemnicy wojskowej i państwowej. Ale to jeszcze nie wszystko. Krakowianie nie byliby sobą, gdyby nie liczyli pieniędzy. Dyrekcja ma więc następujący pogląd na sprawę: komputeryzacja — tak, ale... I tu następuje lista życzeń: muszą być oszczędności na funduszu płac, powinien ulec przyspieszeniu obieg informacji, kolejna sprawa, to eliminacja niektórych prac i dokumentów, udoskonalenie ewidencji materiałowej i zmniejszenie poziomu zapasów.

Warto zwrócić również uwagę i na to, że w WZU dostrzegane są dwie grupy korzyści. Jedne — wymierne — dadzą się wyliczyć jeszcze przed uruchomieniem systemu. I pewnie właśnie one zadecydują o jego wyborze i wprowadzeniu. Ale pomysłodawcy liczą również na korzyści, które w żaden sposób nie dadzą się obliczyć, ani oszacować ex ante. Niewymierne teraz, powinny jednak w przyszłości powiększyć masę zysku. Wynikać będą przede wszystkim z ogólnej poprawy organizacji pracy w przedsiębiorstwie i trafności decyzji podejmowanych w oparciu o dokładniejsze dane. Raz może to być potrzeba nowych uruchomień, innym zaś razem podjęcie produkcji deficytowych części zamiennych lub zabiegi o wykonanie usług dla kontrahenta zagranicznego. Tych możliwości może oczywiście być o wiele więcej.

Mógłby ktoś powiedzieć, że problem przed którym stanęły Wojskowe Zakłady Uzbrojenia nr 1 w Krakowie jest ich własnym kłopotem. Przecież jest to samodzielne, samorządne i samofinansujące się przedsiębiorstwo, więc niech sobie jakoś samo radzi. Ale ten nieobcy w naszej rzeczywistości społecznej sposób myślenia nie uwzględnia właśnie społecznych korzyści płynących z rozumnego traktowania problemów gospodarczych. Bo rzecz nie dotyczy tylko tej jednej sprawy w jednym z wielu przedsiębiorstw i stąd pytanie w tytule, adresowane do czytelników w mundurach.

Andrzej MĘDYKOWSKI

LIGA MYŚLĄCYCH

ZADANIE 1

Kosiarze mieli skosić dwie łąki. Rano wszyscy kosiarze zaczęli kosić większą łąkę, a po obiedzie podzielili się. Połowa pozostała i kosiła pierwszą łąkę, którą do wieczora skosili, druga zaś połowa poszła kosić drugą łąkę, której powierzchnia równała się połowie powierzchni pierwszej łąki. Ilu było kosiarzy, skoro wiemy, że resztę drugiej łąki skosił jeden kosiarz w ciągu następnego dnia?

ZADANIE 3

Kolumna pojazdów o długości l = 2 km porusza się z prędkością V_k = 20 km/h. Od czoła kolumny do ostatniego pojazdu i z powrotem jedzie łącznik na motocyklu z prędkością V_t = 60 km/h. Ile czasu potrzebuje łącznik na przebycie drogi tam i z powrotem oraz jaką drogę przebędzie w tym czasie?

ZADANIE 2

W pierwszym półroczu roku 1987 produkcja spadła o 12,2 proc. w stosunku do produkcji pierwszego półrocza roku 1986, liczba pracowników zaś zwiększyła się o 8,3 proc. O ile procent spadła wydajność jednego pracownika?

ZADANIE 4

Z trzech naczyń, w kształcie sześcianów, pierwsze jest o 1 dm wyższe niż drugie, a drugie o 1 dm wyższe od trzeciego. Kiedy drugie naczynie napełnimy wodą z pierwszego naczynia, a trzecie wodą z drugiego, to w pierwszym naczyniu będzie o 12 litrów wody więcej niż w drugim naczyniu. Jakie są wymiary naczyń?

ZADANIE 5

Podczas pierwszej jazdy samochodem zużyto 20 proc. benzyny, która znajdowała się w zbiorniku paliwa. Podczas drugiej jazdy zużyto 10 proc. ilości benzyny, która pozostała w zbiorniku po pierwszej jeździe. Po dwóch jazdach pozostało w zbiorniku 9 litrów benzyny. Ile litrów benzyny znajdowało się w zbiorniku przed pierwszą jazdą?

Rozwiązania zadań prosimy przesyłać do redakcji do końca czerwca br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

OCALONY

Robert JASTRZĘBSKI

Nastąpił drugi dzień po wybuchu. Szedłem przez ogromną równinę w poszukiwaniu przedmiotów, którym wybuch nadał fantastyczne, a zarazem przerażające kształty. Ciągle potykałem się o jakieś stopniowe elementy metalowe, pochodzące prawdopodobnie z konstrukcji budowlanych miasta, jakie istniało tu przed eksplozją. Przez gęstą warstwę pyłów powybuchowych i zgęstniałych chmur z trudem przebijało się szare i blade światło słoneczne

Uparcie szedłem naprzód. Chciałem za wszelką cenę znaleźć jakieś schronienie i odszukać choćby jednego żyjącego człowieka. Głód dawał mi się we znaki. Po godzinie zaczęło się ściemniać. Pod szarym sklepieniem chmur przesuwały się wolno czarne, kłębiaste warstwy dziwnego pyłu Skierowałem się w stronę odległych wzgórz

W pewnej chwili na jednym z wzniesień zobaczyłem oślepiający błysk. Zjawisko to powtórzyło się jeszcze trzykrotnie. Pomyślałem, że musi być tam człowiek. Zacząłem biec. Po paruminutowym biegu upadłem. Byłem bardzo zmęczony. W oddali znowu zajaśniało ostre światło. Rozbłysnęło trzy razy i zgasło Poderwałem się z ziemi

Po pewnym czasie z trudem dotarłem do pierwszego wzgórza. Było nagie, usłane mnóstwem skurczonych kamieni. W niektórych miejscach jaśniały szkliste stopione skały Zacząłem się wspinać. Raz po raz ziemia wymykała mi się spod nóg. Ręce pokaleczyłem sobie o okruchy skalne Chciałem za wszelką cenę dotrzeć do źródła owego tajemniczego światła

Wspinając się coraz wyżej, odczułem, że zrobiło się nieco chłodniej. Zdumiałem się, skąd te wahania temperatury. Odpowiedź przyszła niebawem. Z szarego sklepienia chmur runął potok deszczu. Ale nie był to zwykły deszcz. Krople były gęste, jakby zmieszane z pyłem. Sprawiało to wrażenie, że zamiast kropli wody, spadają z chmur krople oleju. Po upływie paru sekund mój podarty kombinezon pokrył się gęstą i lepką substancją

Próbowałem dalej się wspinać, lecz ślizgałem się jak na lodzie. Traciłem równowagę i upadłem w płynącą po zboczu olejową ciecz. Podczas jednej z takich prób zacząłem zjeżdżać w dół zbocza. Zbocze urwało się i spadłem. Straciłem przytomność

Gdy się obudziłem — było już prawie ciemno. Deszcz nie padał. Z ledwością odlepiłem się od podłoża. Oleista ciecz zgęstniała, cały byłem oblepiony szaro-brunatną masą Głowa bolała mnie okropnie. Obolały i potłuczony powoli stanąłem na nogi. Rozejrzałem się. Półka skalna była mała i wąska Poniżej było piętnastometrowe urwisko. O zejściu w dół nie było mowy. Wejście na górę też było niemożliwe

Usiadłem zrozpaczony. Zapadła noc. Było bardzo zimno. Póki starczyło mi sił starałem się nie zasnąć. Lecz nie trwało to zbyt długo Po parunastu minutach zmożył mnie sen

Nagle odczułem wyraźną zmianę temperatury. Zrobiło się ciepło. Otworzyłem oczy Obok mnie ze szczeliny skalnej wydobywał się jaskrawy promień świetlny. Po chwili znikł. Wstałem i podszedłem do szczeliny Krzyknąłem do wnętrza. Odpowiedziało tylko echo Chciałem krzyknąć jeszcze raz

Oślepił mnie blask potężnej wiązki światła. Wszystko co dotychczas mnie otaczało, zaczęło znikać. Całą przestrzeń przeniknęło światło. Nie mogłem dojrzeć nawet własnej postaci. W czasie trwania tej jasności do moich uszu dobiegał cichy, a zarazem dziwny dźwięk. W zasadzie nie słyszałem tego uszami. Miałem wrażenie, że przenikał on bezpośrednio do mojego mózgu. Poczułem, że moje niewidzialne ciało przesuwa się Nie potrafiłem określić, w jakim kierunku się przemieszczam, ale najwyraźniej odczuwałem ruch. Czułem, że przemieszczam się coraz szybciej. Jasność zaczęła się zmieniać w przestrzeń falującą całą gamą kolorów. Barwy wirowały jak szalone. Za jakiś czas zaczęły się oddalać. Całą przestrzeń powoli wypełniały pasma o ciemnych kolorach. Koncentrując się stworzyły nieprzenikalną ciemność

Dźwięczenie ustało. Zewsząd otaczała mnie czerń. Czułem się bardzo rozluźniony, lekki, nie odczuwałem ciężaru swojego ciała. Nie wiem jak długo trwałem w tym stanie Pojęcie czasu jakby przestało istnieć

Później zacząłem odczuwać jak z tego dziwnego stanu przechodzę w stan snu. Powróciło poczucie własnego ciała

Obudziłem się. Leżałem na dziwnej elastycznej srebrnej płycie zawieszonej w powietrzu. Płyta znajdowała się pośrodku wielkiej, białej sali. Nad moją głową wisiała wirująca, czerwona kula. Spojrzałem na siebie Byłem ubrany w czarny jednolity kombinezon wykonany z połyskliwego materiału Kombinezon nie miał żadnych kieszeni, ani zapięć. Stanowił z moim ciałem jakby całość. Nawet twarz miałem zasłoniętą, lecz wszystko widziałem i słyszałem bardzo dokładnie

Wstałem z płyty i zacząłem się rozglądać. Wokół mnie nic nie było prócz białych ścian, lekko nachylonych ku górze. Żadnych okien, ani drzwi. Zacząłem iść przed siebie Zauważyłem, że czerwona kula przesuwa się wraz ze mną

Podszedłem do jednej ze ścian, wyciągnąłem rękę. Dłoń zniknęła w mlecznej przestrzeni. Cofnąłem rękę. Dłoń pojawiła się z powrotem. Wtem, coś mi kazało bym szedł naprzód. Wokół mnie nie było nikogo. Pomyślałem, by sprawdzić czy pozostałe ściany zachowują się w podobny sposób jak ta, przy której stoję Chciałem się obrócić, gdy do mojego mózgu dotarł ponownie ten sam rozkaz

Zacząłem się zastanawiać, kto ma tak potężną siłę, by wniknąć do mojego mózgu i przekazywać mi polecenia w postaci myśli Spojrzałem do góry. Nademną wciąż wirowała czerwona kula. Domyślałem się, że to właśnie ona jest moim telepatycznym stróżem.

Ruszyłem naprzód. Przeniknąłem przez mglistą ścianę. Ukazała się tu podobna do poprzedniej sala, tyle że miała kolor niebieski. I znowu otrzymałem polecenie, żeby iść do przodu. Przechodziłem przez kolejne ściany i kolejne kolorowe sale

Zaczęła mnie denerwować ta wędrówka. Stanąłem i spojrzałem na kulę. Nadal wirowała nad moją głową, pulsując czerwonym światłem. Podskoczyłem, by jej dotknąć Była nieuchwytna. Zacząłem pytać, gdzie jestem, kim ona jest. Odpowiedzi nie było Znowu kazano mi iść przed siebie.

Przekroczyłem kolejną ścianę, znalazłem się w ogromnym pomieszczeniu. Nie posiadało ono ani ścian, ani podłogi i sufitu. Wokoło otaczała mnie czarna przestrzeń. Spojrzałem na kulę. Zauważyłem jak zmienia swój kolor z czerwonego w żółty, potem w biały. Później stała się przezroczysta. Wtem, z miejsca, gdzie wirowała, strzelił w otchłań jaskrawy promień. Jednak zanim zniknęła, przekazała mi swój ostatni rozkaz, abym przygotował się na spotkanie z komputerem czasu

W miejscu, gdzie zniknął promień, przejawił się jaskrawo świecący punkt. Punkt zaczął się powiększać. Blask był tak potężny, że zmusił mnie do zamknięcia oczu. Po paru sekundach odczułem, że świecenie zmalało. Otworzyłem oczy. Ujrzałem potężną błękitną kulę. Pulsowała ona tajemniczym blaskiem, jednakże nie mogłem nigdzie dojrzeć opuszczającego ją promieniowania

Chciałem podejść do kuli, lecz nie mogłem zrobić kroku, chciałem coś powiedzieć, lecz nie mogłem wydusić z siebie słowa. Patrzyłem na kulę i nie mogłem oderwać od niej wzroku. Wtem ujrzałem na jej powierzchni skupiające się cienkie pasemka promieni świetlnych

Nagle pasemka zniknęły w jednym punkcie. W moją stronę zaczęła zbliżać się wstęga światła. Wniknęła ona we mnie. Nic nie poczułem. Po pewnej chwili doznałem wrażenia, że błękitna kula mnie pochłania Obraz, jaki dotychczas widziałem przed so-

bą zaczął się pokrywać gęstą mgłą. Po paru sekundach straciłem z oczu kulę...

Przede mną roztaczała się mleczna przestrzeń. Szum zaczął wzrastać. Niebawem był tak głośny, że trudno było wytrzymać. Myślałem, że stracę słuch. Nagle szum się urwał. Miałem uczucie, że zaczynam się przemieszczać. W miejsce mlecznej przestrzeni napłynęły niezrozumiałe obrazy. Nie mogłem dociec co one przedstawiały. Ich treść była jakaś fantastyczna, występowały na nich różnokolorowe elementy. Obrazy te powoli wirowały i nakładały się na siebie.

Z każdą chwilą prędkość wirowania rosła. Gdy prędkość była już nieskończenie wielka, cała ta dotychczas kolorowa mozaika zlała się w jedną ciemną otchłań.

Po chwili poczułem chłód. Doznałem wrażenia, że na czymś leżę. Zrobiło się ciepło. Ocknąłem się. Otworzyłem oczy, podniosłem głowę i rozejrzałem się dokoła. Leżałem znowu na półce skalnej, na którą spadłem, lecz zamiast otaczających mnie wtedy nagich skał, zewsząd otaczała mnie bujna roślinność. Spojrzałem w dół, w stronę równiny. Był to przepiękny widok. Tam, gdzie kiedyś były stopione i skruszone kamienie, teraz rosła zielona i soczysta trawa.

Wyłaniały się z niej potężne drzewa, pokryte listowiem olbrzymich rozmiarów. Gdzieniegdzie strzelały z liściastej gęstwiny wysokie, o fantastycznej formie rośliny, przypominające drzewa iglaste. Cała równina zdawała się być dziewiczym, zielonym oceanem.

Okolica wydawała mi się znajoma, ale jakże inaczej wyglądała ona przedtem, po eksplozji. Gdzie podziały się tajemnicze kule i pomieszczenie, w którym niedawno przebywałem? Ile czasu mogłem tam być? Czy to był fakt, czy tylko sen? Dręczyło mnie pytanie o pochodzenie tego tajemniczego komputera. Kto go stworzył? Był to komputer czasu — może więc to maszyna albo to „coś" przetransportowało mnie w czasie? Ale w przeszłość, czy w przyszłość?

Chyba nigdy nie znajdę na te pytania odpowiedzi. Spojrzałem na siebie. Ponownie byłem ubrany w swój stary i podarty kombinezon. Po tajemniczym, czarnym ubraniu nie było śladu. Przypomniałem sobie, że tutaj, w skale musi być szczelina, z której wydobywał się promień świetlny. Podszedłem do skały. Rozgarnąłem wiszące liście. Jednakże nic nie znalazłem, prócz kamiennej ściany, porośniętej mchem. Po szczelinie nie było śladu.

Odwróciłem się i podszedłem nad urwisko. Z łatwością zszedłem w dół po gąszczu korzeni i lian. Zanurzyłem się w wysokiej trawie i gęstym, wilgotnym listowiu. Podniosłem z ziemi solidnie wyglądający kij i zacząłem przedzierać się przez zieloną ścianę traw i krzewów. Słońce świeciło jaskrawym blaskiem, obdarzając swoim ciepłem i światłem zieloną równinę. Chwyciłem kij mocno w dłoń i ruszyłem przed siebie na spotkanie nowego świata.

PROGRAM SPECTRUM 49, 50

Dźwięki

```
  Program demonstruje dźwięki w
zakresie częstotliwości pomiędzy
0.10 Hz i 14000 Hz.
  Wysokość dźwięku określa się
bezpośrednio w Hz lub w skali od
-60 do +69.
  Dla dźwięków różniących się o
mniej niż 10% od częstotliwości
nominalnej program podaje nazwę
dźwięku.
  Program nie przyjmuje błędnych
danych, np. liter.

 100 PAPER 4: BORDER 4: CLS
 102 DIM p$(4): DIM t$(12,7): FO
R t=1 TO 12: READ t$(t,3 TO 5):
NEXT t: RESTORE
 104 DATA " c ","cis"," d ","dis
"," e "," f ","fis"," g ","gis",
" a ","ais"," h "
 110 PRINT #0;AT 1,10; PAPER 7;
BRIGHT 1;" Ton / Hz "
 120 PAUSE 0: LET i$=INKEY$: GO
TO 130*(i$="h")+160*(i$="t")+120
*(i$<>"h" AND i$<>"t")
 130 INPUT TAB 8; PAPER 7; BRIGH
T 1;" 0.10< Hz <14000 "; LINE h$
: FOR n=1 TO LEN h$: IF NOT (h$(
n)>="0" AND h$(n)<="9" OR h$(n)=
".") THEN GO TO 130
 135 NEXT n: LET hz=VAL h$: IF h
z<0.10 OR hz>14000 THEN GO TO 13
0
 140 LET to=12*LN (hz/261.625565
)/LN 2
 150 GO TO 180
 160 INPUT TAB 8; PAPER 7; BRIGH
T 1;" -60< Ton < 69 "; LINE o$:
FOR n=1 TO LEN o$: IF NOT (o$(n)
>="0" AND o$(n)<="9" OR o$(n)=".
" OR o$(n)="-") THEN GO TO 160
 161 NEXT n
 165 LET to=VAL o$: IF to<-60 OR
to>69 THEN GO TO 160
 170 LET hz=261.625565*2↑(to/12)
 179 REM wydruk i BEEP
 180 CLS : LET d=100+to: IF (d-I
NT to)<100.1 OR (d-INT to)>100.9
 THEN GO SUB 300
 190 LET s$=STR$ to:
     LET a$=p$+"Ton "+s$+p$
 200 PRINT AT 5,(32-LEN a$)/2;
PAPER 6; BRIGHT 1;a$
 210 LET s$=STR$ hz:
     LET b$=p$+s$+" Hz"+p$
 220 PRINT AT 17,(32-LEN b$)/2;
PAPER 6; BRIGHT 1;b$
 230 BORDER 6: BEEP .5,to: BORDE
R 4
 240 PAUSE 20:
 IF INKEY$<>"" THEN GO TO 110
 250 GO TO 230
 299 REM podprogram
 300 LET t=61+INT (to+.1)
 310 IF t>12 THEN LET t=t-12: GO
 TO 310
 320 PRINT AT 11,13; BRIGHT 1; P
APER 7; INK 2;t$(t): RETURN
9998 REM zapis na taśmę
9999 CLEAR : SAVE "Ton/Hz" LINE
100: VERIFY "": GO TO 100
```

Klawiatura




```
  0>REM Program KLAWIATURA
 80 BORDER 4: CLS : BRIGHT 0
 90 FOR n=0 TO 7: READ bt: POKE
USR "a"+n,bt: NEXT n: REM a=
 100 DATA 255,195,195,0,129,195,
231,255
 110 FOR n=60000 TO 60100: POKE
n,250: NEXT n
 120 FOR n=0 TO 16: READ a: POKE
60000+a,n: NEXT n
 130 DATA 49,81,87,51,69,52,82,8
4,54,89,55,85,56,73,79,48,80
 140 FOR n=0 TO 16: READ a: POKE
61000+n,a: NEXT n
 150 DATA 1,3,5,7,9,10,12,15,16,
18,19,21,22,24,27,28,30
 160 DIM a$(32): LET pam=50000:
LET tab=60000: LET poz=61000: PO
KE 23658,8
 170 BRIGHT 1
 180 FOR n=11 TO 15:
                    PRINT AT n,0;"
 ▀▀    ▀▀ ▀▀    ▀▀ ▀▀ ▀▀    ▀▀ "
   NEXT n
 190 PRINT AT 16,0;"
 2     3  4     5  7  6     0 "
"
 195 PRINT AT 17,0; a$,a$;a$,"
  Q  W  E  R  T  Y  U  I  O  P
",a$
 200 FOR n=16 TO 232 STEP 24: PL
OT n,0: DRAW 0,86: NEXT n
 210 GO SUB 390
 220 BRIGHT 0
 230 PAUSE 0: PRINT AT 2,10; BRI
GHT 1;" REJESTRACJA ": PRINT #0;
AT 0,0;a$;"            X = ODTWARZA
NIE       "
 240 FOR n=1 TO 9999: PRINT AT 9
,15; BRIGHT 1;n
 250 LET a=250
 260 IF INKEY$="X" THEN GO TO 31
0
 270 IF INKEY$<>"" AND PEEK (COD
E INKEY$+tab)<>250 THEN LET t=CO
DE INKEY$: LET a=PEEK (tab+t)
 280 IF a<250 THEN PRINT AT 10,P
EEK (poz+a); BRIGHT 1;CHR$ 144:
BEEP .2,a-2: PRINT AT 10,PEEK (p
oz+a);" "
 290 IF a=250 THEN PRINT AT 8,15
;"  "
 300 POKE pam+n,a: NEXT n
 310 PRINT AT 2,10; BRIGHT 1; IN
VERSE 1;" ODTWARZANIE ": BORDER
4: GO SUB 400: FOR m=1 TO n-1: L
ET p=PEEK (pam+m): PRINT AT 8,15
; BRIGHT 1; INVERSE 1;m+1;
 320 IF p<250 THEN PRINT AT 10,P
EEK (poz+p); BRIGHT 1; INVERSE 1
;CHR$ 144: BEEP .2,p-2: PRINT AT
 10,PEEK (poz+p);" "
 330 IF p=250 THEN PAUSE 1
 340 NEXT m
 350 PRINT #0;AT 0,0;a$;"
POWTORZYC ( T/N ) ?      "
 360 PAUSE 0: IF INKEY$="T" THEN
 PRINT AT 8,15;"    ": GO SUB 40
0: GO TO 310
 370 BORDER 1: FOR n=2 TO 9: PRI
NT AT n,0;a$: NEXT n: GO SUB 400
: GO SUB 390: GO TO 230
 390 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175:
 PLOT 0,87: DRAW 255,0: RETURN
 400 PRINT #0;AT 0,0;a$,a$: RETU
RN
 900 CLEAR : SAVE "KLAWIATURA" L
INE 70: BEEP .5,-10: CLS : PRINT
 '"Verify  KLAWIATURA": VERIFY
"": RESTORE : GO TO 70
```

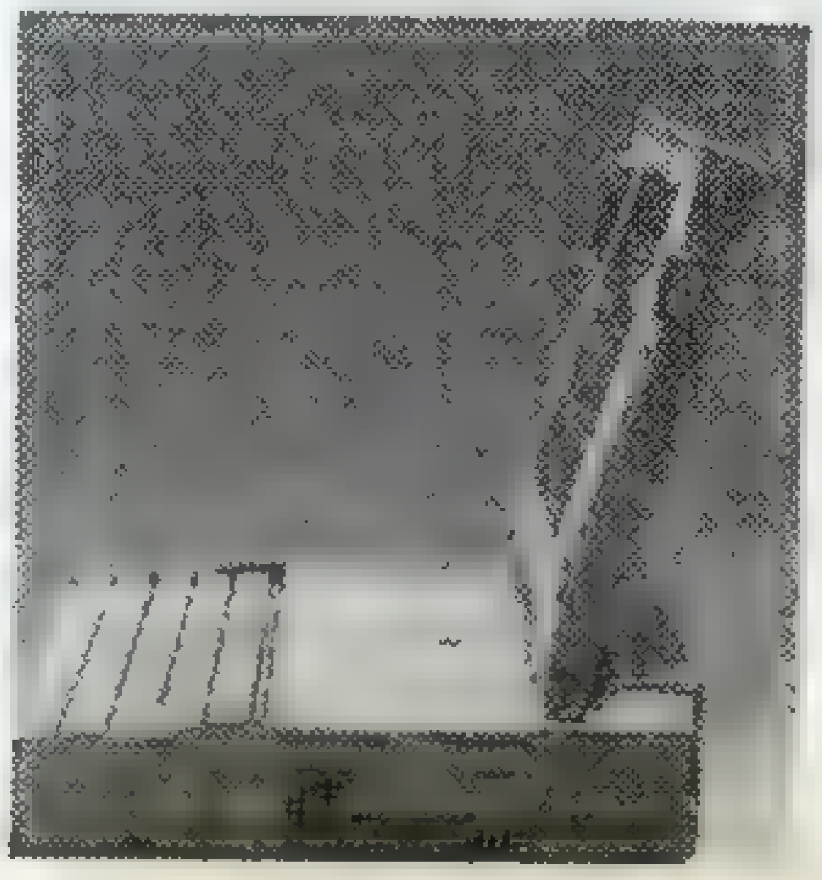
J. Rubinowicz

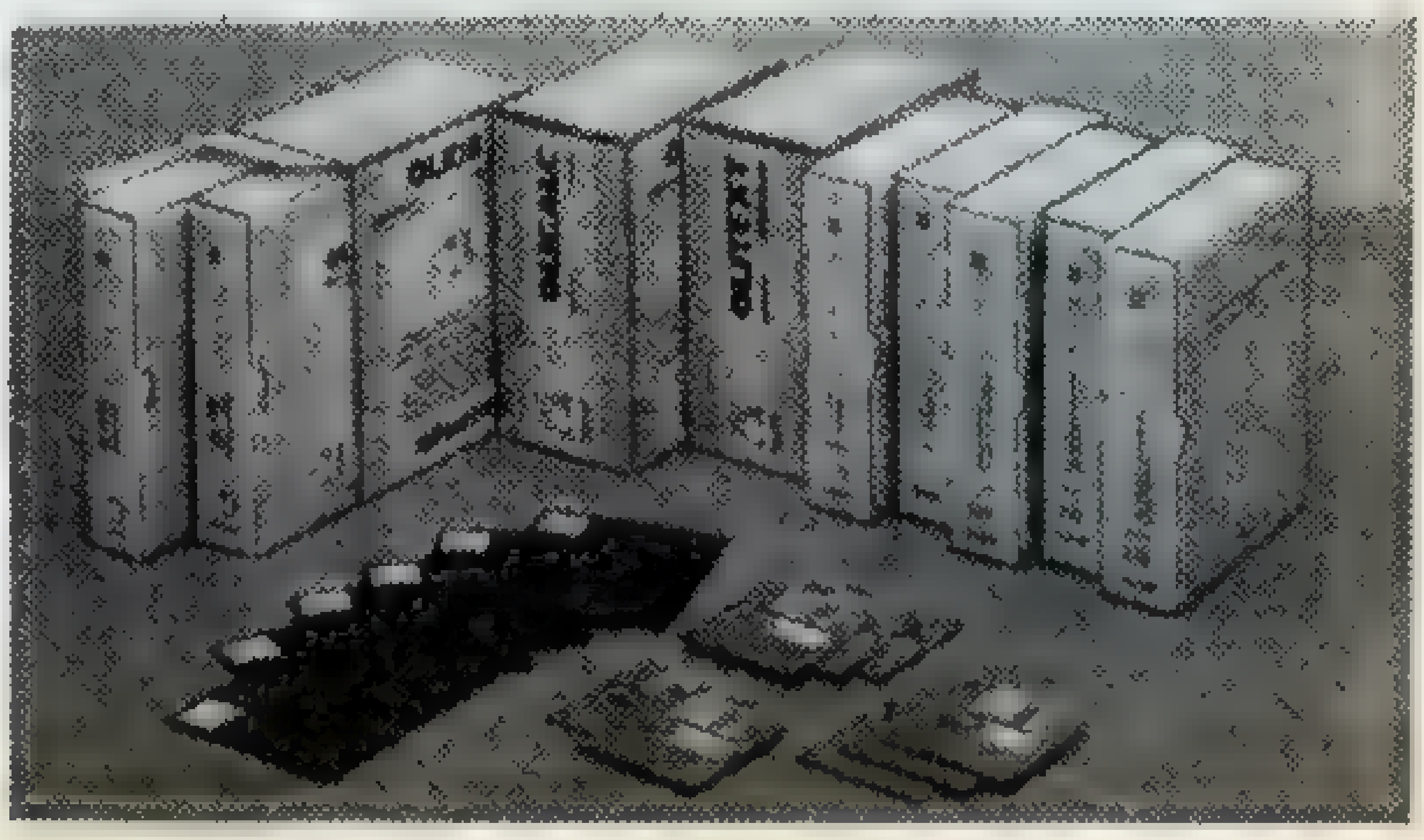
Spis najciekawszych programów

Poniżej przedstawiamy spis najciekawszych programów narzędziowych jakie ukazały się w IKS-ie, BAJTKU, KOMPUTERZE i MIKROKLANIE (od pierwszych numerów) na SPEKTRUM, AMSTRADA, COMMODORE i ATARI.
W tabeli podano nazwę prezentowanego programu, jego funkcję i źródło tj.numer pisma (dla IKS-a literka 'Z' oznacza - ZESZYT PROGRAMÓW)/rok ukazania się/skrót nazwy pisma, gdzie:

IKS - INFORMATYKA.KOMPUTERY SYSTEMY,
BAJ - BAJTEK,
KOM - KOMPUTER,
MIK - MIKROKLAN.

Lp	NAZWA PROGRAMU/FUNKCJA - JĘZYK PROGRAMU	ŹRODŁO
1	2	3
	Z X - S P E K T R U M	
1	2	3
1	/Przenoszenie programów (zbiorów danych) ze SPEKTRUM na AMSTRADA - BASIC	Z3/87/IKS
2	ASEMBLER/Symulacja prawdziwego assemblera na SPEKTRUM - ASSEMBLER	Z2/87/IKS
3	KATALOG/Katalogowanie gier - BASIC	Z4/87/IKS
4	COMPACT/Zmniejsza zajętość pamięci,zwiększa szybkość działania programów -BASIC	2/86/IKS
5	/Zapamiętywanie 100 obrazów w 48k RAM-u SPEKTRUM - BASIC	5/86/IKS
6	/Zwiększanie tablicy zmiennych - BASIC	6/86/IKS
7	/Edytor znaków - BASIC	2/87/IKS
8	/Test monitora - BASIC	4/87/IKS
[illegible]	/Zwiększanie napisów na ekranie - BASIC	[illegible]/BAJ
10	/64 kolumny tekstu na ekranie - BASIC	7/86/BAJ
11	KATALOG/Spis programów na taśmie - BASIC	10/86/BAJ
12	/Renumeracja linii programowych - BASIC	11/86/BAJ
13	/Rozszerzanie tablic w pamięci - BASIC	4/87/BAJ
14	/śledzenie wykonywania programu - BASIC	11/87/BAJ
15	/Wyszukiwanie i tworzenie tablic występowania zmiennych w programie - BASIC	3/87/KOM
16	/Nagrywanie programu w dowolnym kodzie, umieszczanie pod wskazanym adresem-BASIC	3/87/KOM
17	/Sortowanie tablicy znakowej dwuwymiarowej - BASIC, - ASSEMBLER	5/87/KOM
18	/Tworzenie linii DATA z kolejnymi bajtami programu maszynowego - BASIC	12/87/KOM
	A M S T R A D	
1	2	3
1	/Przenoszenie programów (zbiorów danych) ze SPEKTRUM na AMSTRADA - BASIC	Z2/87/IKS
2	Kopiowanie ekranu monitora - BASIC	4/87/IKS
3	/Zniesienie protekcji odczytu - BASIC	6/87/IKS
[illegible]	/Odtwarza skasowany plik na dysku -BASIC	7/86/BAJ
5	/Katalog dysku - BASIC	8/86/BAJ
6	/Analiza nazw zmiennych - BASIC	8/86/BAJ
7	Zdejmowanie zabezpieczenia programów napisanych w BASIC-u - BASIC	9/86/BAJ
8	MONITOR/Przeglądanie i modyfikacja pamięci - BASIC	11/86/BAJ
9	/Emulator BASIC-a 1.1 dla CPC 464 -BASIC	9/87/BAJ 10/87/BAJ
	C O M M O D O R E	
1	2	3
1	DISASEMBLER/Tłumaczy instrukcje z kodu wewnętrznego na język symboliczny- BASIC	5/86/IKS
2	MANIAKTURBO/Usprawnienie interpretera	3-4/86BAJ
3	/Katalog dysku na C64 - BASIC	9/86/BAJ
4	TSLCOPY/Kopiowanie na dysk programow z pamięci komputera - BASIC	11/86/BAJ
[illegible]	LISTER/Zmiana szerokości wydruku	11/86/BAJ
6	/Łączenie programów w BASIC-u - BASIC	4/87/BAJ
7	TURBO16/Przyspieszone ładowanie programów na C16 i C116 - BASIC	5/87/BAJ
8	AUTONUMBER/Automatyczne numerowanie linii programów w BASIC-u - BASIC	5/87/BAJ
9	/Połączenie C64 z drukarkami D-100 lub DZM-180 - BASIC	6/87/BAJ
10	/Przypisywanie rozkazów lub funkcji klawiszom funkcyjnym - BASIC	6/87/BAJ
11	/Katalog dysku	6/87/BAJ
12	/Przenoszenie obrazu z ekranu na drukarkę dla C16/116 PLUS/4	9/87/BAJ
13	DOS+/Rozszerza funkcje DOS-u (C64)-BASIC	10/87/BAJ
14	DISK-PROTECTOR/Zabezpiecza pliki przed skasowaniem - BASIC	11/87/BAJ
15	/Usprawnienie programu TURBO16 - BASIC	11/87/BAJ
16	/Obsługa dysku MS-DOS dla C128 - BASIC	9/87/KOM
17	TURBOTAPE64/Przyśpiesza operacje zapisu, odczytu i weryfikacji programów na taśmę magnetyczną dla C64 - BASIC	2/86/MIK
	A T A R I	
1	2	3
1	/Miniassembler - BASIC	Z1/87/IKS
2	/Automatyczny start programów - BASIC	Z2/87/IKS
3	DYSKMENU/Przeglądanie dyskietki i automatyczne uruchamianie programu - BASIC	Z2/87/IKS
[illegible]	/Symulacja funkcji assemblera - BASIC	Z4/87/IKS
5	/Konwerter języka maszynowego - BASIC	Z4/87/IKS
6	DISASEMBLER/Tłumaczy instrukcje z kodu wewnętrznego na język symboliczny -BASIC	Z4/87/IKS
7	/Edytor - BASIC	4/87/IKS
8	MINI-DOS/Realizuje funkcje DOS-u - BASIC	6/87/IKS
9	/Kopiowanie programów napisanych w języku wewnętrznym - BASIC	6/87/IKS
10	/Tworzenie procedur wyświetlania - BASIC	7-8/87IKS
11	/Kopiowanie programów - BASIC	7-8/87IKS
12	/Test stacji dysków - BASIC	7-8/87IKS
13	DISASEMBLER/Tłumaczy instrukcje z kodu wewnętrznego na język symboliczny -BASIC	10/87/IKS
14	SUMY KONTROLNE/Ułatwia wprowadzanie, drukowanie programów - BASIC	11/87/IKS
15	/Wyświetla na ekranie mapę dysku - BASIC	11/87/IKS
16	/Edytor kodu maszynowego - BASIC	12/87/IKS
17	RENUMERATOR/Renumeracja linii programowych programu - BASIC	5-6/86BAJ
[illegible]	TAPECOPIER/Kopiuje programy taśmowe	11/86/BAJ
19	SYMULATOR 6502/Analiza działania programu w kodzie maszynowym - BASIC	2/87/BAJ
20	/Katalog dysku, usuwanie plików - BASIC	2/87/BAJ
21	/Kasetowy system operacyjny - BASIC	5/87/BAJ
22	SPEED TRANS/Przemieszczanie danych w pamięci komputera - BASIC	8/87/BAJ
23	/Dynamiczne przeglądanie pamięci - BASIC	8/87/BAJ
24	/Przenosi zawartości zbiorów dyskowych "do wnętrza" instrukcji DATA - BASIC	2/87/KOM





DIGITAL INFORMATION DISPLAY – cyfrowy monitor ekranowy,
DIGITAL INPUT – wejście cyfrowe,
DIGITAL INTEGRATED CIRCUIT – cyfrowy układ scalony,
DIGITAL OPTICAL RECORDING – zapis optyczny cyfrowy,
DIGITAL OUTPUT – wyjście cyfrowe,
DIGITAL PLOTTER – ploter, pisak cyfrowy, urządzenie rysujące,
DIGITAL PRINTER – drukarka cyfrowa,
DIGITAL READ-OUT – odczyt cyfrowy,
DIGITAL RECORDING – zapis cyfrowy,
DIGITAL REPRESENTATION OF DATA – cyfrowe przedstawienie danych,
DIGITAL SORT – sortowanie metodą cyfrową,
DIGITAL-TO-ANALOG CONVERTER – konwerter cyfrowo-analogowy,
DIGITIZE – przekształcać na postać numeryczną, dyskretyzować,
DIGITIZER – przetwornik analogowo-cyfrowy,
DIGIT KEYBOARD – klawiatura cyfrowa,
DIGIT POINT – przecinek (np. dziesiętny),
DIGIT SAFEGUARDING CODE – kod zabezpieczający cyfry,
DIMENSION – wymiar,
DIMENSIONLESS NUMBER – liczba bezwymiarowa,
DIP SWITCH – zestaw przełącznikowy wzdłużny,
DIRECT – bezpośredni, także: kierować,
DIRECT ACCESS – dostęp bezpośredni,
DIRECT ACCESS DEVICE – urządzenie o bezpośrednim dostępie,
DIRECT-ACCESS STORAGE – pamięć o dostępie bezpośrednim,
DIRECT ADDRESS – adres bezpośredni,
DIRECT ADDRESSING – adresowanie bezpośrednie (adres argumentu bezpośrednio za rozkazem),
DIRECT ARGUMENT – argument bezpośredni, występuje bezpośrednio po rozkazie,
DIRECT CODE – kod prosty,
DIRECT COMPUTER DISPLAY – ekranopis,
DIRECT CONTROL – sterowanie bezpośrednie,
DIRECT DIGITAL CONTROL – sterowanie cyfrowe bezpośrednie,
DIRECT DRIVE – synchronizacja bezpośrednia,
DIRECT-DRIVE MONOCHROME MONITOR – monitor monochromatyczny o bezpośredniej synchronizacji,
DIRECTED BEAM – konturowe rozwinięcie (kreślenie) obrazu,
DIRECTED BEAM CRT DISPLAY DEVICE – monitor konturowy,
DIRECTED BEAM DEVICE – konturowe urządzenie graficzne,
DIRECTED BEAM DISPLAY DEVICE – patrz: DIRECTED BEAM DEVICE,
DIRECT INFORMATION – informacja adresowana, informacja indywidualna,
DIRECT INPUT-OUTPUT – bezpośrednie przesyłanie informacji między mikroprocesorem a urządzeniami zewnętrznymi,

DIRECT JUMP – skok bezpośredni, także: rozkaz skoku bezpośredniego,
DIRECT LINE – patrz: IMMEDIATE LINE,
DIRECT MEMORY ACCESS CONTROLLER – sterownik DMA (bezpośredniego dostępu do pamięci),
DIRECT MEMORY ACCESS – bezpośredni dostęp do pamięci,
DIRECT MODE – patrz: IMMEDIATE MODE,
DIRECT NUMERICAL CONTROL (DNC) – bezpośrednie sterowanie numeryczne,
DIRECT SUBTRACT COUNTER – licznik odejmujący,
DIRECT (DATA) TRANSMISSION – transmisja (danych) bezpośrednia,
DIRECTED SET – zbiór skierowany,
DIRECTION – kierunek (zwiększanie lub malenie adresów),
DIRECTORY – informator, katalog, kartoteka, skorowidz
DISABLE – wyłączyć z działania, unieruchomić,
DISABLE INTERRUPTS – wyłącz przerwania,
DISABLE PULSE – impuls blokujący, impuls zamykający,
DISADVANTAGE – wada, ujemna strona,
DISASSEMBLE – patrz: DISADVANTAGE,
DISCARD – odrzucić, zaniechać,
DISCONNECT – odłączać, rozłączać,
DISCONTINUE – przerywać,
DISCONTINUITY – brak ciągłości, przerwa,
DISCONTINUOUS – przerywany, nieciągły,
DISCREPANCY – rozbieżność, różnica (np. pomiarów),
DISCRETE – dyskretny, nieciągły, odrębny, pojedynczy,
DISCRETE CHANNEL – kanał dyskretny, kanał ziarnisty,
DISCRETE INFORMATION SOURCE – dyskretne źródło informacji,
DISCRETE REPRESENTATION OF DATA – dyskretne przedstawienie danych,
DISCRIMINATE – rozróżniać,
DISCUSSION – analiza, dyskusja, omówienie (np. wyników),

DISJUNCTION – alternatywa,
DISK – dysk magnetyczny,
DISK AUXILIARY ROUTINE – program dyskowy pomocniczy,
DISK CARTRIDGE – pakiet, dysk wymienny,
DISK DRIVE – napęd dyskietkowy, napęd dyskowy,
DISKETTE – dyskietka, dysk elastyczny,
DISK FACE – strona dysku,
DISK LAYOUT – rozplanowanie układu dysku,
DISK MEMORY – pamięć (magnetyczna) dyskowa,
DISK OPERATING SYSTEM – system operacyjny dyskowy,
DISK PACK – patrz: DISK CARTRIDGE,
DISK PARAMETER BLOCK – blok parametrów dyskowych,
DISK STORAGE – patrz: DISK MEMORY,
DISK STORAGE DUMP – wydruk zawartości dysku, wypis zawartości dysku,
DISK STORAGE EXTRACT – wyciąg z dysku,
DISK STORAGE PRINT-OUT – patrz: DISK STORAGE DUMP,
DISK TRACK – ścieżka dyskowa,
DISK TRANSFER AREA – obszar (bufor) transmisji dyskowych,
DISK UNIT – jednostka pamięci dyskowej,
DISPARITY – rozbieżność,
DISPATCHER – program rezydent,
DISPATCHING – koordynacja, także: służba dyspozytorska,
DISPERSED STORAGE – zapamiętywanie rozproszone,
DISPLACE – przemieszczać, przesuwać, umieścić,
DISPLACEMENT – przesunięcie, także: nazwa adresu używanego przy adresowaniu indeksowym,

DISPLAING – obrazowanie,
DISPLAY – zobrazowanie, przedstawienie, także: monitor obrazowy, przedstawiać, obrazować,
DISPLAY ADAPTER – adapter ekranu,
DISPLAY BACKGROUND – tło (podstawa) obrazu,
DISPLAY BUFFER – bufor obrazu, pamięć obrazu,
DISPLAY COMMAND – komenda graficzna,
DISPLAY CONSOLE – konsola graficzna,
DISPLAY CYCLE – cykl odnawiania, cykl odświeżania,
DISPLAY DATA – dane graficzne,
DISPLAY DEVICE – graficzne urządzenie wyjściowe,
DISPLAY ELEMENT – element obrazu (np. punkt, odcinek, znak alfanumeryczny, itp.),
DISPLAY ENTITY – obiekt graficzny (zbiór elementów obrazu, którymi można operować jako całością),
DISPLAY FILE – plik graficzny,
DISPLAY FOREGROUND – przedni plan obrazu,
DISPLAY FRAME – kadr obrazu,
DISPLAY GENERATOR – generator obrazu (przetwarza zawartość bufora obrazu na obraz fizyczny),
DISPLAY IMAGE – kod kadru,
DISPLAY INSTRUCTION – komenda graficzna,
DISPLAY ORDER – rozkaz graficzny,
DISPLAY PAGE – stronica obrazowa,
DISPLAY PANEL – panel graficzny, konsola graficzna oprogramowana w sposób umożliwiający pracę w trybie interakcyjnym,
DISPLAY PRIMITIVE – element obrazu (np. punkt, odcinek, znak alfanumeryczny, itp.),
DISPLAY SCOPE – wielkość obrazu, rozmiar rysunku,
DISPLAY SPACE – przestrzeń operacyjna urządzenia, (obszar, którego zawartość jest przedstawiona na powierzchni obrazowania),
DISPLAY SURFACE – powierzchnia obrazowania (nośnik, na którym tworzony jest obraz fizyczny),
DISPLAY UNIT – monitor ekranowy, ekranopis, także: wskaźnik radarowy,
DISPLAY WRITER – element kreślący obraz,
DISPOSE OF – rozporządzać (czymś), dysponować,
DISREGARD – ignorować, pomijać, nie zważać (na coś),
DISSEMINATE – rozpowszechniać, rozprowadzać (np. informacje),
DISSEMINATION – rozpowszechnianie, rozprowadzanie,
DISSIMILAR – niepodobny, różny,
DISTANCE CONTROL – sterowanie zdalne,
DISTINCTION – różnica,
DISTORT – wypaczyć, uszkodzić, zniekształcić,
DISTORTION – spaczenie, wypaczenie, wykrzywienie, zniekształcenie,
DISTORTION FACTOR – współczynnik zniekształcenia,
DISTRIBUTE – rozdzielać,
DISTRIBUTED PROCESSING – przetwarzanie rozproszone,
DISTRIBUTING BOARD – tablica rozdzielcza,
DISTRIBUTION – rozłożenie, rozkład, rozprowadzenie,
DISTURB SIGNAL – sygnał zakłócający,
DISTURBANCE – zakłócenie, zaburzenie,
DIVERGE – odchylać się, odbiegać,
DIVERGENCY – dywergencja, rozbieżność,
DIVIDE – dzielić, także: podziel,
DIVIDEND – dzielna,
DIVIDER – dzielnik,
DIVISION – dzielenie, znak dzielenia, także: w COBOLu: dział (sekcja),
DIVISION SIGN – znak dzielenia,
DIVISION WITH ROUND-OFF – dzielenie z zaokrągleniem,
DMAC – patrz: DIRECT MEMORY ACCESS CONTROLLER,

DML – patrz: DATA MANIPULATION LANGUAGE,
DNC – patrz: DIRECT NUMERICAL CONTROL,
DO – patrz: DIGITAL OUTPUT,
DOC – patrz: DIAGNOSTICS ON-CHIP,
DOCUMENT – dokument,
DOCUMENT-FREE DATA PROCESSING – przetwarzanie danych bezdokumentowe,
DOCUMENT PROCESSING – przetwarzanie dokumentów,
DOCUMENT READER – czytnik dokumentów,
DOCUMENT SAFEGUARDING – ochrona dokumentów,
DOCUMENTATION – dokumentacja,
DOMAIN – dziedzina, obszar,
DOMAIN OF MAP – dziedzina odwzorowania,
DOR – patrz: DIGITAL OPTICAL RECORDING,
DOT AND ELEMENT – pseudoelement "I",
DOT DIAGRAM – wykres punktowy,
DOT MATRIX – matryca punktów,
DOT MATRIX CHARACTER GENERATOR – generator mozaikowy znaków,
DOT MATRIX PRINTER – drukarka mozaikowa,
DOT MEMORY – pamięć punktowa,
DOT OR ELEMENT – pseudoelement "LUB",
DOT PRINTER – patrz: DOT MATRIX PRINTER,
DOT PRODUCT – iloczyn skalarny,
DOUBLE CURRENT – prąd dwukierunkowy,
DOUBLE DENSITY (DD) – podwójna gęstość, także: sposób zapisu na dysku,
DOUBLE-DENSITY FLOPPY DISK – dyskietka o podwójnej gęstości zapisu,
DOUBLE-LENGTH NUMBER – liczba o podwójnej długości,
DOUBLE PRECISION – podwójna precyzja,
DOUBLE-PRECISION COMPUTATION – liczenie z podwójną precyzją,
DOUBLE-PRECISION NOTATION – zapis podwójnej precyzji,
DOUBLE SIDED (DS) – dwustronny, także: zapis po obu stronach dysku,
DOUBLE-WIRE COMMUNICATION – przesyłanie dwuprzewodowe,
DOWNLOAD – ładowanie w dół, ładowanie na dolne pozycje (np. rejestru),
DOWNLOADING – ładowanie zdalne, ładowanie skrośne, ładowanie w dół,
DOWN-TIME – okres wyłączenia, okres niesprawności (urządzenia), czas przestoju, przestój spowodowany awarią sprzętu,
DOWN-TIME OPERATION – czynność związana z usunięciem uszkodzenia,
DOWN-UP COUNTER – licznik rewersyjny (zliczający w obydwu kierunkach),
DPB – patrz: DISK PARAMETER BLOCK,
DPL – patrz: DESCRIPTOR PRIVILEGE LEVEL,
DRAGGING – wodzenie (np. przesuwanie elementu obrazu za piórem świetlnym),
DRAM – patrz: DYNAMIC RANDOM ACCESS MEMORY,
DRAWING AREA – wielkość (rozmiar) powierzchni obrazowania,
DRAW – rysować, także: wyciągać wniosek,
DRAWBACK – cecha ujemna, wada,
DRAWING SIZE – wielkość obrazu, rozmiar rysunku,
DRIVE – urządzenie, napęd,
DRIVER – program obsługi urządzenia, moduł sterujący,
DROP – opuszczać, także: spadek (np. napięcia),
DROP CABLE – kabel dołączeniowy,
DROP IN – sygnał nie zapisany (dot. zapisu na nośniku magnetycznym),

DROP OUT – sygnał nie odebrany (dot. zapisu na nośniku magnetycznym),
DROPPED BIT – bit opuszczony, bit zagubiony,
DRUM – bęben,
DRUM MEMORY – pamięć (magnetyczna) bębnowa,
DRUM PLOTTER – pisak bębnowy, ploter bębnowy,
DRUM PRINTER – drukarka bębnowa,
DRY RUN – przebieg próbny,
DS – patrz: DATA SEGMENT,
DT-EQUIPMENT (DTE) – urządzenie transmisji danych, UTD,
DTA – patrz: DISK TRANSFER AREA,
DTE – patrz: DT-EQUIPMENT,
DU – patrz: DISK UNIT,
DUALCODE – kod dwójkowy,
DUAL FULL ADDER – sumator pełny dwójkowy,
DUAL PAPER FEED – przesuw podwójny, posuw podwójny,
DUAL-PURPOSE KEY – klawisz dwuczynnościowy,
DUAL-TRACK RECORDIND – zapisywanie dwuścieżkowe,
DUE DATE – termin,
DUMB TERMINAL – "niemy terminal",
DUMMY – element fikcyjny, upozorowany, sztuczny,
DUMMY ARGUMENT – argument formalny,
DUMMY DEVICE – urządzenie pozorne,
DUMMY INDEX – niemy indeks,
DUMMY INSTRUCTION – rozkaz pusty,
DUMMY NAME – nazwa pusta,
DUMMY ORDER – rozkaz fikcyjny,
DUMMY PARAMETER – parametr formalny,
DUMMY ROUTINE – ślepy (bezczynny) podprogram,
DUMMY SECTION – pseudosekcja,
DUMMY VARIABLE – zmienna fikcyjna,

DUMP – przeniesienie całego lub częściowego obszaru pamięci komputera do innego obszaru pamięci lub na nośnik zewnętrzny,
DUMP CHECK – sprawdzenie wydruku pamięci,
DUMPING – przenoszenie zawartości pamięci komputera, przekazywanie,
DUMPING PROGRAM – program post-mortem,
DUMPING SYSTEM CONTROL AREA – kopiowanie obszaru sterowania,
DUODECIMAL – dwunastkowy (system liczenia),
DUPLEX – transmisja jednoczesna, transmisja dupleksowa, dupleks,
DUPLEX CHANNEL – kanał dupleksowy,
DUPLEX OPERATION – patrz: DUPLEX,
DUPLICATE – kopia, także: powielać, reprodukować, powtarzać,
DUPLICATING PROGRAM – program duplikujący, program kopiujący,
DUPLICATING PUNCH – dziurkarka powtarzająca się, dziurkarka reprodukująca,
DUPLICATION – kopiowanie, powielanie,
DUPLICATION CHECK – kontrola przez duplikację,
DUPLICATION METHOD – metoda powielania,
DURABILITY – trwałość,
DURATION – czas trwania,
DUSTPROOF BAG – pyłoszczelna torebka (np. do przechowywania taśmy magnetycznej),
DUTY CYCLE – cykl obowiązkowy, cykl pracy,
DUTY PROGRAM – program usługowy, program użytkowy,
DX (DATA REGISTER) – rejestr danych,
DYADIC – dwuczłonowy, podwójny,
DYNAMIC IMAGE – przedni plan obrazu,
DYNAMIC MAIN STORAGE ALLOCATION – przydzielanie dynamiczne pamięci operacyjnej,
DYNAMIC MAPPING SYSTEM – system z odwzorowaniem dynamicznym,
DYNAMIC PROGRAM RELOCATION – przesunięcie dynamiczne programu,
DYNAMIC PROGRAMMING – programowanie dynamiczne,
DYNAMIC RANDOM ACCESS MEMORY – dynamiczna pamięć o dostępie bezpośrednim,
DYNAMIC SKEW – przekos dynamiczny,
DYNAMIC STORAGE – pamięć dynamiczna,
DYNAMIC TESTING TECHNIQUE – testowanie dynamiczne,

E

E^2 – patrz: ELECTRICALLY ERASABLE DEVICE,
EA – patrz: EFFECTIVE ADDRESS,
EAC – patrz: END-AROUND CARRY,
EARLY FAILURE PERIOD – okres uszkodzeń początkowych,
EARN – patrz: EUROPEAN ACADEMIC RESEARCH NETWORK,
EAROM – patrz: ELECTRICALLY ALTERABLE ROM,
EARTH – uziom, zwarcie doziemne, także: uziemiać,
EARTH BAR – szyna uziemiająca,
EARTH TERMINAL – zacisk uziomowy,
EBCDI-CODE – patrz: EXTENDED BINARY-CODED DECIMAL INTERCHANGE CODE,
EBCDIC – patrz: EXTENDED BINARY-CODED DECIMAL INTERCHANGE CODE,
EBNF – patrz: EXTENDED BACKUS-NAUR FORM,
ECC – patrz: ERROR CORRECTION CODE,
ECGB – patrz: ENHANCED COLOR GRAPHICS BOARD,
ECHO – echo, potwierdzenie poprawności odbioru wysłanej informacji,
ECHO-CHECK – kontrola zwrotna, echo kontrolne,
ECHO-PLEX – echo-pleks,
ECHO-PRINTING – drukowanie echa,
ECHO SUPPRESSOR – tłumik echa,
ECHOING THE CHARACTER – potwierdzenie poprawności odbioru wysłanego znaku,
ECMA – patrz: EUROPEAN COMPUTER MANUFACTURES' ASSOCIATION,
ECTL – patrz: EMITTER-COUPLED TRANSISTOR LOGIC,
EDC – patrz: ERROR DETECTION/CORRECTION,
EDGE – krawędź (np. karty, taśmy papierowej),
EDGE-NOTCHED CARD – karta (kartotekowa) selekcyjna nacinana,
EDGE PATH – droga krawędziowa, ścieżka krawędziowa,
EDGE-PUNCHED CARD – karta (kartotekowa) obrzeżnie dziurkowana,
EDGE-TRIGGERED WRITE – zapis przełączany krawędzią,
EDGEWISE – wzdłuż krawędzi, na kant,
EDIT – redagować, także: edytor,
EDIT-DIRECTED – redagowanie kierowane (sterowane),
EDITING – redagowanie wyników, przygotowywanie informacji,
EDITING KEY – klawisz edycji,
EDITING STATEMENT – dyrektywa edycji,
EDITING TERMINAL – terminal w skomputeryzowanym systemie wydawniczym,
EDITION – wydanie,
EDITOR – edytor, program redagujący,

Enigma

Za pomocą tego programu można kopiować gry i programy użytkowe do pojemności 33 kB pamięci. W zależności od stosowanego nośnika oryginału oraz jego wtórnika mamy do wyboru trzy możliwości kopiowania tj.: z taśmy na taśmę; z dysku na taśmę i z taśmy na dysk. Obsługa programu jest bardzo prosta i polega na wykonywaniu poleceń zobrazowanych na ekranie monitora. Program składa się z trzech części. Pierwsza to zwiastun, po wprowadzeniu którego komputer oczekuje na wpisanie z klawiatury instrukcji RUN. Instrukcja ta uruchamiając zwiastuna umożliwia wprowadzenie części drugiej (objaśnienie) oraz programu głównego w formie gotowej do wykonywania kopii. Do sterowania programem wykorzystywane są cztery klawisze:
T — taśma lub tak w zależności od pytania; D — dysk; N — nie i SPACE do potwierdzenia wyboru.

H. JANKOWSKI

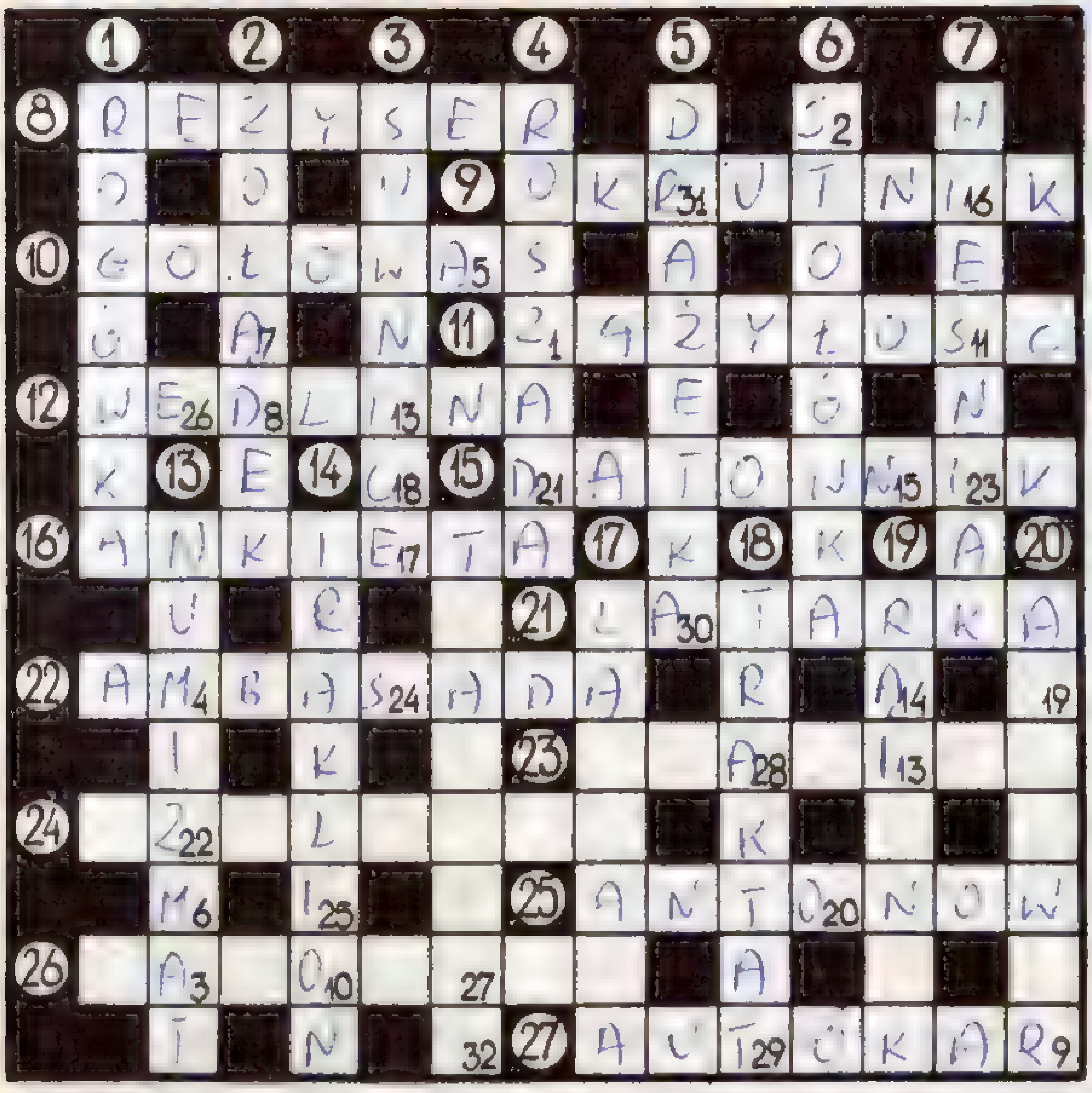
```
VH 0 POKE 16,64:POKE 53774,64:POKE
   709,14:POKE 710,0:POKE 752,1
FB 1 ? CHR$(125):POSITION 10,19:? "
   Fanom ATARI-Redakcja IKS-a":FOR
   I=1572 TO 1600:READ A:POKE I,A:N
   EXT I:U=USR(1572)
MH 2 DATA 169,12,141,252,2,169,1,14
   1,68,2,162,253,154,169,183,72,16
   9,84,72,169,4,32,182,187,169,255
   ,76,4,187

WX 3 GRAPHICS 0:POKE 16,64:POKE 537
   74,64:POKE 710,0:POKE 752,1:POKE
   40002,8:DL=PEEK(560)+256*PEEK(5
   61):POKE DL+6,7
ZP 4 FOR I=14 TO 20:POKE DL+I,6:NEX
   T I:POKE 709,14:POKE 82,0:? "
   ***************************"
WZ 5 ? "     atari-enigma
   *****************(X-1987)**":POSI
   TION 24,4:? "- kopiuje programy
   o pojemnosci ";
KJ 6 POSITION 34,5:? "do 33 kBajt,"
   :POSITION 24,6:? "w/g jednej z w
   ybranych relacji :":? :? "    ta
   sma  tasma"
VJ 7 ? "     tasma   dysk":? "    dy
   sk   tasma":? :? "       Procesor
   MOSTEK 6502C (HALT)":POSITION 2
   4,16
JF 8 ? "POWODZENIA !":FOR I=1536 TO
   1571:READ A:POKE I,A:NEXT I:FOR
   I=1601 TO 1607:READ A:POKE I,A:
   NEXT I:I=USR(1572)
KG 9 DATA 104,104,104,162,16,157,66
   ,3,104,157,69,3,104,157,68,3,104
   ,157,73,3,104,157,72,3
JP 10 DATA 32,86,228,132,212,169,0,
   133,213,133,77,96
NC 11 DATA 104,169,0,141,65,6,96
SS 20 CLR :OPEN #2,4,0,"K:":OPEN #3
   ,8,0,"E:"
AE 25 POKE 16,64:POKE 53774,64:POKE
   82,2:POKE 709,14:POKE 710,0
NG 30 DIM F$(16),FR$(16),FL$(6),AD(
   9),ST(9)
FX 40 AP=PEEK(14)+256*PEEK(15)
SX 50 DL=PEEK(560)+256*PEEK(561).
PZ 60 ST=AP+54:L=DL-ST-6
RU 70 U=USR(1601)
ZK 90 ? "}"
ZC 91 LN=90:OP=1:?
GY 95 LZ=0:AD(0)=0:ST(0)=ST:KOP=0
UU 100 ? "Pochodzenie programu: ";C
   HR$(212);"asma - ";CHR$(196);"ys
   k ? ";
LF 110 GET #2,AN:IF AN<>68 AND AN<>
   84 THEN 110
XZ 115 DYSK=0:PUT #3,AN:?
MK 120 IF AN=68 THEN DYSK=1
DC 125 IF  NOT DYSK THEN F$="C:":DY
   R=128:GOTO 160
UV 127 GOSUB 900
YM 130 ? :? "Nazwa programu: ";:INP
   UT F$
II 135 IF F$="" THEN ? CHR$(28);CHR
   $(156);CHR$(28);:GOTO 130
RB 140 GOSUB 2000
DN 150 DYR=0
WW 160 RT=160:K=0:TRAP 5000
CY 162 KOP=KOP+1:ST(KOP)=ST(KOP-1)+
   AD(KOP-1)+10:LZ=LZ+AD(KOP-1)+10
XA 170 OPEN #1,4,DYR,F$
QJ 175 X=USR(1536,7,ST(KOP),L-LZ)
MJ 178 CLOSE #1
JG 180 TRAP 40000
PI 185 IF X=1 OR X=136 THEN 240
UO 190 IF DYSK THEN ? :? "Stacja dy
   skowa";
SE 200 IF  NOT DYSK THEN ? :? "Magn
   etofon";
JL 210 ? " nie pracuje."
YU 215 RT=160 K=1:KOP=KOP-1:GOTO 51
   .
PR 240 IF X=1 THEN ? :? "Zbyt dlugi
   program !":IF KOP>1 THEN KOP=KO
   P-1.00000085:GOTO 3000
OX 241 IF X=1 AND KOP=1 THEN GOTO 9
   1
QP 250 AD=PEEK(856)+256*PEEK(857)
RJ 251 AD(KOP)=AD
MC 253 ? :? "Pojemnosc ";KOP;" czes
   ci - ";AD;" bajt":IF DYSK THEN 2
   65

PM 258 IF KOP=9 THEN 3000
XP 260 ? "Czy wgrywac dalej ? ";
?? 261 GET #2,AN:IF AN=78 AND KOP>1
    THEN ? :GOTO 3000
GA 262 IF AN=84 THEN ? "Tak":GOTO 1
   60
YY 263 IF AN=78 AND KOP=1 THEN ? "N
   ie":GOTO 265
QB 264 GOTO 262
RO 265 D=DYSK
JR 269 LN=268:OP=0:DYSK=0
CR 270 ? :? "Wgrac na ";CHR$(212);"
   asme czy ";CHR$(196);"ysk ? ";
FV 275 GET #2,AN:IF AN<>68 AND AN<>
   84 THEN 275
XZ 280 DYSK=0:PUT #3,AN:?
?? 290 IF AN=68 THEN DYSK=1
DV 295 IF  NOT DYSK THEN DYR=128:F$
   ="C:":GOTO 360
?? 300 DYR=0:GOSUB 900
OG 302 IF  NOT D THEN 330
IY 303 ? "Poprzednia nazwa: ";F$
?? 305 ? :? "Czy zmieniasz nazwe ?
   ";
PH 310 GET #2,AN:IF AN<>78 AND AN<>
   84 THEN 310
MH 320 ? :IF AN=78 THEN 360
US 330 ? :? "Nowa nazwa: ";:INPUT F
   $
MQ 335 IF F$="" THEN ? CHR$(28);CHR
   $(156);CHR$(28);:GOTO 330
RF 350 GOSUB 2000
FX 360 I=1
SZ 400 GOSUB 7000
MT 420 ? :? "Skopiowano !"
LM 430 ? :? "Czy kopiowac dalej ? "
VZ 440 GET #2,AN:IF AN<>78 AND AN<>
   84 THEN 440
UP 450 IF AN=84 AND KOP=1 THEN 268
NS 455 IF AN=84 AND KOP>1 THEN 3000
ST 460 GOTO 90
UH 500 ? :? "Pomylka ";PEEK(195)
MJ 510 POKE 732,0:POKE 764,255:POKE
    53279,7:POKE 752,1:?
QZ 511 IF (KOP>1 OR K) AND OP THEN
   ? "[OPTION] zachowuje sie poprze
   dnia czesc"
OT 512 ? "[SELECT] powtorz ostatni
   I/O proces"
OP 513 ? "[START] powrot do ATARI-E
   NIGMA"
NV 514 ? "[HELP]  powtorz ostatnie
   I/O dzialanie"
FR 520 IF PEEK(732)=17 THEN POKE 75
   2,1:GOTO LN
BE 522 IF PEEK(53279)=6 THEN POKE 7
   52,1:GOTO 90
WF 523 IF (KOP>1 OR K) AND OP AND P
   EEK(53279)=3 THEN POKE 752,1:GOT
   O 3000
ZA 525 IF PEEK(53279)=5 THEN POKE 7
   52,1:GOTO RT

UE 530 POKE 764,255:GOTO 520
TP 900 ? :? "Podaj numer stacji: ";
ZS 910 GET #2,AN:DRN=AN-48:IF DRN<1
    OR DRN>4 THEN 910
LK 915 PUT #3,AN:?
IW 920 FR$="D":IF DRN>1 THEN FR$(LE
   N(FR$)+1)=STR$(DRN)
ZI 925 FR$(LEN(FR$)+1)=":":FL$=FR$:
   FL$(LEN(FL$)+1)="*.*"
QQ 930 ? :? "Czytnik ";DRN;" szuka
   w spisie ":?
QL 935 RT=935:TRAP 500
LG 940 CLOSE #1:OPEN #1,6,0,FL$:CLO
   SE #1
JV 945 TRAP 40000
ZP 950 RETURN
UB 2000 F=0
MB 2010 FOR I=1 TO LEN(F$)
SM 2015 F=F+(F$(I,I)=":")
EW 2020 NEXT I
GM 2025 IF F=0 THEN FR$(LEN(FR$)+1)
   =F$:F$=FR$
AJ 2030 RETURN
FK 3000 POKE 201,5:LN=3000:OP=0
PJ 3005 ? :? CHR$(17);:FOR A0=1 TO
   22:? CHR$(18);:NEXT A0:? CHR$(5)
VU 3010 FOR I=1 TO KOP
GR 3020 ? ,"| Czesc ";I;" - ";AD(I)
   ;
PV 3030 POKE 85,24:? " bajt |"
FD 3040 NEXT I
UC 3050 ? ,CHR$(26);:FOR A0=1 TO 22
   :? CHR$(18);:NEXT A0:? CHR$(3)
CY 3060 ? :? "Wgrywam !":?
AU 3070 F$="C:":DYR=128
QA 3075 POKE 752,1
WP 3080 FOR I=1 TO KOP
HQ 3090 ? I;" ";
ZH 3100 GOSUB 7000
EW 3110 NEXT I
OH 3115 POKE 752,1:?
PH 3120 GOTO 420
CK 5000 KOP=KOP-1:GOTO 500
VP 7000 RT=7000:TRAP 500
LN 7010 IF I>1 THEN POKE 764,12
SH 7020 OPEN #1,8,DYR,F$
YV 7030 X=USR(1536,11,ST(I),AD(I))
NB 7040 CLOSE #1
OQ 7050 TRAP 40000
LV 7060 IF X=1 THEN 7090
TG 7070 IF DYSK THEN ? "Pomylka ";X
SN 7080 IF  NOT DYSK THEN ? "Pomylk
   a ";X
EE 7085 RT=7090:GOTO 510
BG 7090 RETURN
```

KRZYŻÓWKA NR 4



POZIOMO: 8) Krzysztof Zanussi, 9) był nim Neron, 10) niedoświadczony młokos, młodzik, 11) komitywa, poufałość, 12) salami albo kabanos, 15) przyrząd używany w kasach biletowych lub na poczcie, 16) badanie statyst. służące do obserwacji statystycznej, 21) podręczne źródło światła, 22) placówka dyplomatyczna, 23) przeciwdziałanie, 24) Fiodor (1873—1938), rosyjski śpiewak światowej sławy, bas, 25) radziecki konstruktor lotniczy, twórca m.in. samolotu AN-22, 26) część ringu, 27) turystyczny środek lokomocji.

PIONOWO: 1) zewnętrzna, przezroczysta błona gałki ocznej, 2) między przełykiem a jelitem, 3) uniwersalne urządzenia transportowe, dźwignice, 4) manewr szachowy, 5) tabletka, 6) zakład zbiorowego żywienia, 7) chłop, 13) dawna moneta lub medal, mająca wartość zabytkową, 14) największe miasto i port Krety, 15) morska żegluga nieregularna, 17) port morski we wsch. części Cypru, 18) rozprawa naukowa, 19) bylina oleista uprawiana na obszarach tropikalnych dla nasion i do wyrobu oleju rycynowego, 20) Żyd Wieczny Tułacz, ze średniowiecznej legendy.

Litery z kratek ponumerowanych dodatkowo w prawym dolnym rogu od 1 do 32 utworzą hasło, które wystarczy nadesłać jako rozwiązanie zadania na kartkach pocztowych pod adresem redakcji, do końca czerwca, **naklejając kupon „IKS"-a** Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe

KUPON ▶

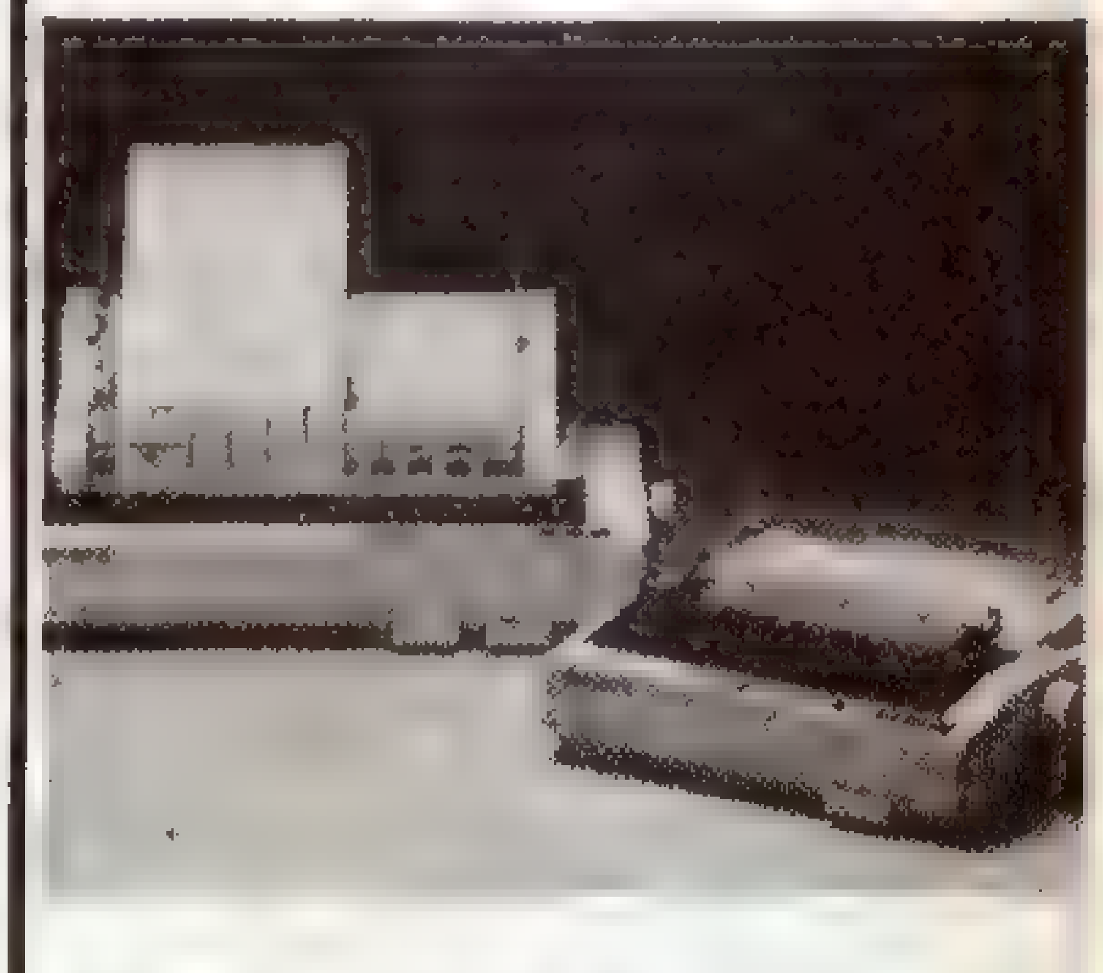
I. K. S.

MIKROCIEKAWOSTKI

star



NR-10 I NR-15



NB 24-10 I NB 24-15



ND-10 I ND-15

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1117. Nr ind. 361682. U-40

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 ZŁ

DODATEK „ŻOŁNIERZA WOLNOŚCI" Nr 5/1988 ISSN 0860—2794

Propozycje

Żywioł i koniunktura — czyli komputery sprowadzane przez Prosystem w sklepach Centralnej Składnicy Harcerskiej — strona 15.

Czy przypadkowo skasowane zbiory można odtworzyć? — odpowiedź na stronie 22.

Informatyczny słownik angielsko-polski — Uwaga to może się przydać — strona 29.







Nauka • Technika

Zachodnioniemiecka firma SIGMA-SPORT zaprezentowała na wystawie w Nowym Jorku rower wyposażony w komputer, który pokazuje cykliście aktualną prędkość jazdy, szybkość średnią i maksymalną, jaką rowerzysta może na danej trasie rozwinąć przy swoich możliwościach fizycznych, liczbę przejechanych kilometrów, siłę nacisku na pedały – a także liczbę skurczów serca rowerzysty.

■ ■ ■

W Japonii opracowano elektroniczny system poszukiwania abonentów, umożliwiający błyskawiczne odnalezienie danej osoby i uzyskanie z nią połączenia telefonicznego. Ludzie, którzy z racji obowiązków służbowych wędrują po różnych pomieszczeniach – wkładają karteczkę ze swym kodem do specjalnej przystawki aparatu telefonicznego, który jest akurat w pobliżu. Informacja ta przekazywana jest natychmiast do komputera, który steruje pracą centrali telefonicznej.

Ciąg dalszy na str. 8



Programuję w ciemno...

W numerze

Drugi rok „Stobit-u" — str. 3
Transputer — str. 4
W szponach Atari — str. 6—8
Tłumacz grafiki — str. 9
Wskaźniki i funkcje — str. 10—12
Awaryjne przerywanie programu ZX — str. 13
Wesołe zadania — str. 14
Żywioł i koniunktura — str. 15
Infosystem — str. 16—17
Lilavati — str. 18
Jak odzyskać skasowane zbiory — str. 22
Tajemnicze błędy — str. 24
Elektroliza — str. 25
Giełda pomysłów — str. 31
Krzyżówka — str. 32

Klasówka przy klawiaturze

Zanim fala namiętności mikrokomputerowych dotarła do Polski wraz ze stosunkowo tanimi modelami Spectrum, Atari i Amstrada, informatycy, nauczyciele i wiele innych osób zaczęło sobie zadawać pytanie. „Kiedy te cudowne maszynki dotrą do polskich szkół?" W to, że wcześniej, czy później muszą dotrzeć, mało kto wątpił. Komputer stał się synonimem nowoczesności, a jakby na to nie spojrzeć, próg nowoczesności, legendarny XXI wiek, przekroczy za 12 lat. Dzisiejsze 6-latki będą ostatnim pokoleniem osiągającym pełnoletność (według dzisiejszych zasad, w „starym" XX wieku). Czyż zatem można pozwolić, by ich starsi koledzy, nie zetknęli się podczas szkolnych lekcji z klawiaturą komputera, by nie poznali jego zasad pracy i nie przyzwyczaili się do rewolucji niesionej przez masowo stosowaną informatykę?

Jak to najczęściej bywa, wówczas gdy można zarobić trochę pieniędzy, pojawiają się energiczne firmy. Podobnie było z dostawą mikrokomputerów do polskich szkół. Pierwsze oferty wpłynęły od firm zagranicznych oraz spółek z kapitałem mieszanym. Zwyciężyła jednak koncepcja rodzimego komputera szkolnego. Dwa lata temu rozstrzygnięty został konkurs na polski mikrokomputer edukacyjny — wygrała konstrukcja opracowana przez zespół z Zakładu Badań Operacyjnych i Systemów Komputerowych Instytutu Automatyki Politechniki Poznańskiej. Mikrokomputer otrzymał nazwę Junior, nazwę-symbol, podkreślającą jego przeznaczenie. Zastosowany jest w nim mikroprocesor 8-bitowy Zilog Z-80A, ten sam co w najpopularniejszych mikrokomputerach 8-bitowych (Spectrum, Amstrad). Nowsze konstrukcje wykorzystują mikroprocesory 16- i 32-bitowe. Nie w tej jednej dziedzinie decydują finanse i dylemat produkować tanie Juniory, czy nie produkować wcale. Wybór niewielki i trudno się dziwić, że zdecydowano się na pierwszy wariant, jednak Junior jest droższy niż dostępne na rynku 8-bitowce sprowadzane z zagranicy.

Dwa lata, jakie upłynęły od rozstrzygnięcia konkursu na komputer edukacyjny, to krótki okres. Przez te dwa lata uruchomiona została produkcja we wrocławskich zakładach Elwro. Jesienią ubiegłego roku ruszyła produkcja seryjna, w grudniu około 2 tysiące jednostek centralnych czekało na odbiorcę. Nie były to kompletne zestawy, takie jakich sobie życzył klient — Ministerstwo Edukacji Narodowej. Dla fabryki były to już gotowe produkty, co gorsze, że w świetle przepisów, za ich składowanie fabryka powinna zapłacić karę (gdyby nie sprzedano ich w 1987 roku). Po licznych monitach, 31 grudnia, za zgodą Ministerstwa Edukacji Narodowej, wyspecjalizowane przedsiębiorstwo zaopatrujące szkoły w pomoce naukowe Cezas, zakupiło kłopotliwą partię Juniorów. Nowy rok rozpoczął klient — Ministerstwo Edukacji od problemów ze skompletowaniem zestawów. Serwis będzie musiał dodatkowo jeździć do szkół, by zamontować brakujące urządzenia peryferyjne — wiadomo, kto poniesie koszty.

W naszych warunkach dwuletni okres uruchamiania produkcji jest małym rezultatem. Te same dwa lata, jeśli zmierzymy nimi rozwój informatyki, zmieniają swoją wymowę. Widać to na specjalistycznych targach sprzętu komputerowego. Pojawiły się nowe konstrukcje komputerów, urządzeń zewnętrznych. ZX Spectrum dwa lata temu uważany był za komputer, dziś traktuje się go jak zabawkę. Magnetofon kasetowy, jako pamięć zewnętrzna został wyparty przez stacje dysków elastycznych. Junior, choć jeszcze praktycznie nie dotarł do szkół, w błyskawicznym tempie starzeje się. Do dnia dzisiejszego nie zostało zakończone testowanie serii próbnych, nadal są problemy ze skompletowaniem pełnych zestawów zamawianych przez Ministerstwo Edukacji.

Szkolna konfiguracja składa się z zestawu sieciowego (ośmiu komputerów wykorzystujących wspólny dysk elastyczny i drukarkę) oraz dwóch dodatkowych, autonomicznych komputerów z dyskami elastycznymi i drukarkami. Podczas pracy komputerów w sieci, jeden z nich (nauczycielski), steruje pracą dysku, drukarki, pozostałe komputery (uczniowskie) pracują jak normalne ZX Spectrum wyposażone w pamięć dyskową i drukarkę. Żaden z użytkowników pracując przy swojej klawiaturze i monitorze, nie odczuwa, że w tym samym czasie z urządzeń zewnętrznych korzystają inni użytkownicy. Przewaga komputerów pracujących w sieci nad pracą pojedynczych komputerów, polega na tym, że można między nimi przeprowadzać transmisję danych, można wykorzystywać mniej urządzeń peryferyjnych, a to obniża koszty. W przypadku Juniorów działających w lokalnej sieci mikrokomputerowej Junet, praca uczniów na komputerze może być obserwowana przez nauczyciela na ekranie jego komputera. Pomaga mu to o tyle w pracy, że nie będzie musiał chodzić po klasie, by patrzeć, jak radzą sobie wychowankowie.

Junior i sieć Junet to pierwszy etap informatyki — sprzęt drugi, to oprogramowanie. Pracą sieci zarządza system operacyjny CP/J, wzorowany na najbardziej rozpowszechnionym dla mikrokomputerów ośmiobitowych dyskowym systemie operacyjnym CP/M. Oprogramowanie przechowywane w pamięci EPROM komputera zawiera edytor, interpreter języka Basic i programy sterujące pracą urządzeń zewnętrznych. Junior jest zgodny ze Spectrum firmy Sinclair Research Ltd. — wszystkie programy napisane dla tego najpopularniejszego mikrokomputera, bez żadnych adaptacji, przeróbek mogą być wykorzystywane w Juniorze. System operacyjny CP/J oraz oprogramowanie systemowe umożliwiają stosowanie najbardziej rozpowszechnionych języków i systemów programowania: Basic, Logo, Pascal, Fortran, C, Forth,

makroasemblery mikroprocesorow Intel 8080, 8085 i Z80.

O ile nad zakupem komputerów czuwają kuratoria (chodzi o to, by nie dopuścić do zakupu modeli nie kompatybilnych ze sobą), to przy zakupie i doborze programów edukacyjnych szkoły są samodzielne. Powstał „Katalog oprogramowania dydaktycznego" — informator dla nauczycieli o tym, jakie programy znajdują się w zbiorach resortu edukacji narodowej. Wybór konkretnego programu zależy od nauczyciela, od jego metod nauczania i doświadczenia.

Informatyka w szkole nie może ograniczać się wyłącznie do sali „komputerowej", z Juniorów powinni korzystać nauczyciele innych przedmiotów — jak z normalnych pomocy naukowych, dlatego potrzebne są dodatkowe, autonomiczne egzemplarze. Z jednej strony mikrokomputer na lekcjach przedmiotów ścisłych może być wykorzystywany jako narzędzie do obliczeń, z drugiej przy dobrych programach edukacyjnych nauka może być atrakcyjną zabawą. Jak twierdzi Zbigniew Rogowski, doradca ministra ds. informatyki brakuje dobrych programow edukacyjnych. Przy ich pisaniu należy połączyć doświadczenia pedagogów i informatyków. Bez tego zawsze będzie czegoś brakować — tak jak w programie Ortografia — zasób słownictwa nie został dobrany pod kątem uczniow szkoły podstawowej, do ktorych był adresowany.

Od 1985 roku prowadzone są kursy przygotowujące nauczycieli do korzystania z mikrokomputerow. Przyjęto zasadę, że informatyki nauczać będą ludzie z przygotowaniem specjalistycznym, by korzystać z komputera na lekcji wystarczy krótsze przeszkolenie. Przedstawiciel producenta zapewniał mnie, że oprogramowanie sieci Junet jest na tyle proste w obsłudze, iż podczas szkoleń organizowanych dla nauczycieli, wystarczała godzina, by samodzielnie mogli prowadzić zajecia z wykorzystaniem sieci, pod warunkiem, że dany nauczyciel wiedział, co można pokazać w konkretnym programie. Oczywiście by poznać sieć należy poświęcić znacznie więcej czasu, trzeba dodatkowo poznać Juniora. Z pewnością nie wszystkich nauczycieli uda się przekonać do korzystania z informatyki — dyrektor Rogowski od swojego francuskiego odpowiednika wie, że we Francji w 1985 roku 30 proc. nauczycieli odcinało się od komputerów, wybierając tradycyjne metody nauczania. Jak będzie u nas, na razie zwolennicy nowoczesności wprowadzają komputery do szkół, kiedy skończą, zobaczymy.

D.O.

JUNET, czyli sieć lokalna mikrokomputerów ELWRO—800 JUNIOR, zapewnia współpracę kilkudziesięciu mikrokomputerów. Dzięki sieci możliwe jest wspólne wykorzystywanie pamięci dyskowych i drukarek przez wiele mikrokomputerów. Możliwe jest też przesyłanie obrazów, tekstow, programów oraz wybranych obszarow pamięci operacyjnej z jednych mikrokomputerów do drugich. Poszczególne komputery pracujące w sieci mogą mieć zróżnicowane uprawnienia do korzystania z niej. Na przykład nauczyciel korzystając z mikrokomputera nauczycielskiego czyli uprzywilejowanego w sieci JUNET, może rozsyłać programy do mikrokomputerów uczniowskich oraz kontrolować prace uczniów, kopiując bez ich wiedzy zawartość ekranu wybranego mikrokomputera na własny ekran. Funkcje te są niedostępne w mikrokomputerach uczniowskich, ale uczniowie poprzez sieć JUNET mogą korzystać z pamięci dyskowych i drukarki mikrokomputera nauczycielskiego. Informacja przesyłana w sieci może być kierowana do jednego, wybranego odbiorcy, do określonych grup odbiorców lub do wszystkich odbiorców. Szybkość transmisji wynosi 64 kb/s. Sieć ma architekturę typu „wspolna szyna", dzięki czemu dołączenie kolejnych mikrokomputerów nie wymaga żadnych zmian we wcześniejszych połączeniach. Do sieci JUNET może być dołączony profesjonalny mikrokomputer ELWRO—800 jako jednostka dużej mocy obliczeniowej.

Drugi rok „Stobit-u"

„Co kupić? Magnetowid, czy komputer" — zastanawiał się jesienią 1986 roku Wiesław Rypina, kierownik Domu Kultury „Stokłosy". Nie wiadomo, jaką podjąłby decyzję, gdyby wśród zapytanych o to osób nie było Jana Ejgerda, zawodowego informatyka mieszkającego w pobliżu. Jak każdy fachowiec pasjonujący się swoją dziedziną, odpowiedział — „komputery, z pewnością będzie z nich większy pożytek. Powstanie klub komputerowy, dzieciaki będą się mogły czegoś pożytecznego nauczyć". W ten sposób powstała sekcja komputerowa „Stobit", nazwa powstała z połączenia części nazwy warszawskiego osiedla Sto-kłosy i jednostki informacji — bit.

Początki, jak to początki, nie były łatwe. Gdyby było odwrotnie, to nie wspominano by o nich z rozrzewnieniem. Zaczęto się od dwóch Unipolbritów, bez stacji dysków. Chętnych było zbyt wielu, osiedle młode, dużo rodzin z dziećmi, do centrum stolicy daleko, a tu pod bokiem taka atrakcja. Organizatorzy przyjęli założenie, że z jednej klawiatury podczas zajęć mogą korzystać dwie osoby. Stąd łatwo było wyliczyć, że przy dwóch komputerach, ośmiu godzinach zajęć tygodniowo można przyjąć 32 osoby. Od razu utworzona została lista rezerwowa z której zapisane osoby miały pierwszeństwo przy przyjmowaniu. Najłatwiej było zwiększyć liczbę zajęć. W miarę upływu czasu kupiono więcej sprzętu — stację dysków do Unipolbritów, Amstrada CPC 464 i Amstrada CPC 6128 z kolorowymi monitoram.



Pod koniec zajęć Agnieszka lubi sobie troszkę pograć.

„Stobit" rozpoczął drugi rok działalności — nadal funkcjonuje lista rezerwowa, co świadczy, że zajęcia prowadzone są w sposób ciekawy. Wśród uczestników zajęć wielu ma własne komputery, do nich należą Rafał Tkaczyński, Paweł Wróbel. Oni szukają czegoś więcej niż zabawy przy klawiaturze, chcą poznać zasady pracy komputera, języki programowania. Czasem trafia do klubu samouk, jak Jakub Poleć, który samodzielnie rozgryzł podstawowe języki programowania, obecnie uczy się asemblera. Przyszedł do klubu, bo „zawsze tu jest fajnie, można z kimś porozmawiać, powymieniać programy, doświadczenia".

Większość przychodzących do „Stobitu" pierwszy raz zetknęła się z mikrokomputerami. Dla nich, 8-, 12-latków, (w tym wieku jest najwięcej uczestników) spotkania na zajęciach są szansą poznania informatyki. Jak stwierdził jeden z uczestników, „u mnie w szkole nauczyciele prawie nic nie wiedzą o komputerach. Pani bibliotekarka powiedziała, że to takie urządzenie, w którym po naciśnięciu guzika, wypisują się różne informacje". Dzieci śmieją się z takiego określenia komputera, dla nich dawno przestał być czarną skrzynką z guzikiem.

Tekst i zdjęcia: D.O.

Transputer — współczesna alternatywa komputera optycznego?

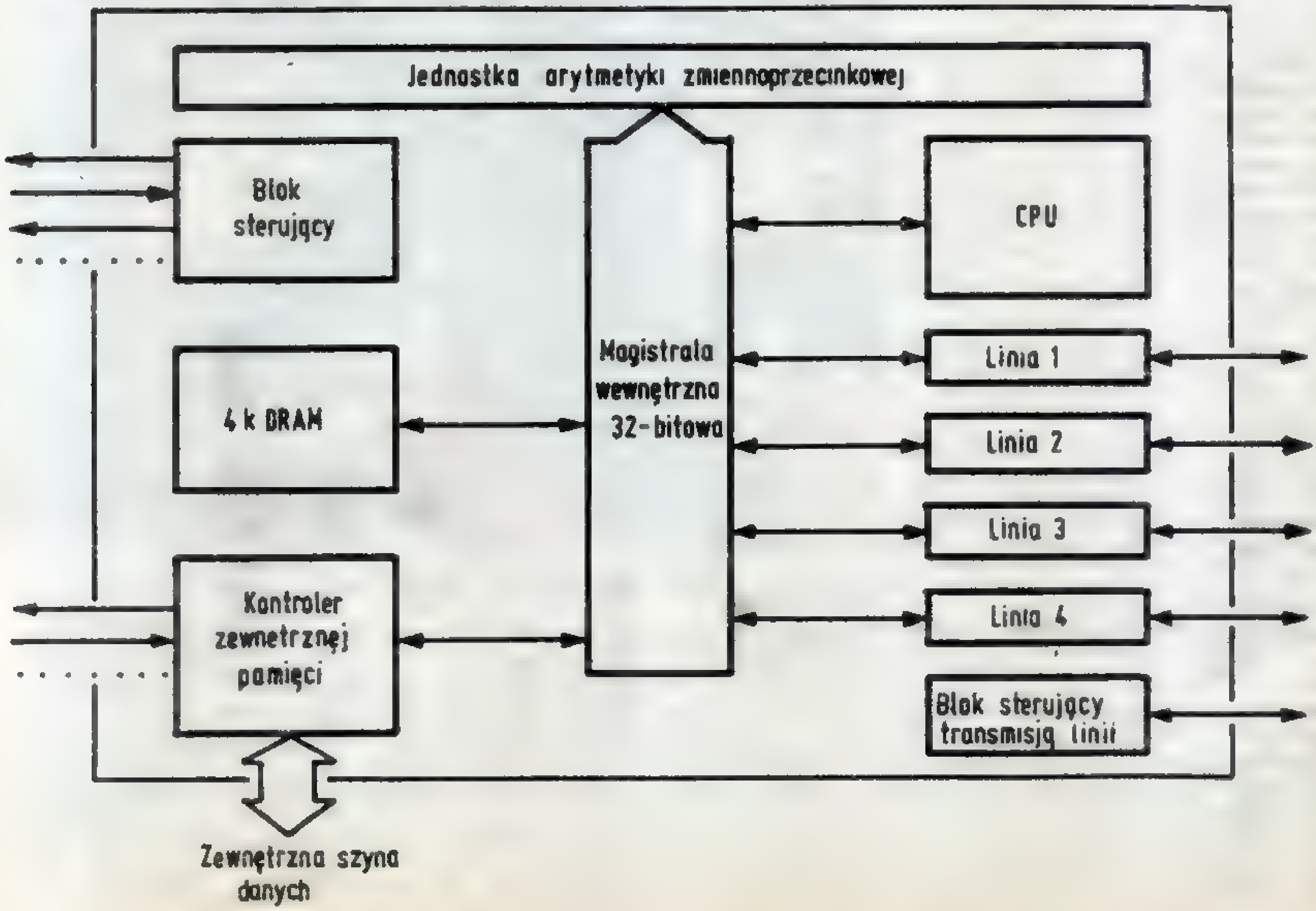
Już od dawna naukowcy, projektanci i konstruktorzy w swoich pracowniach toczą walkę, aby pokonać kolejne bariery techniki i technologii. Postęp badań w holografii [1], rozwój techniki laserowej i optoelektroniki upoważniał do stwierdzeń, że w niedalekiej przyszłości nastąpi pełna adaptacja fal optycznych jako nośnika informacji. Oczekiwano, że w telekomunikacji oraz w teleinformatyce elektronikę uzupełni lub całkowicie zastąpi nowa dziedzina — **fotonika**. Jednakże osiągnięcia współczesnej elektroniki, w szczególności pojawienie się w latach siedemdziesiątych, a następnie burzliwy rozwój mikroprocesorów dokonały rewolucyjnych zmian w wielu dziedzinach życia.

Komputery optyczne

Pomimo że techniczne opracowania pamięci holograficznych nie mają znaczenia przemysłowego, stały się one punktem zwrotnym w badaniach nad wykorzystaniem zjawisk dyfrakcji i interferencji fal świetlnych w opracowaniach komponentów architektonicznych komputerów optycznych (procesorów, pamięci, kanałów transmisji, przetworników WE/WY). Motorem tych zainteresowań były teoretyczne możliwości tj. niezwykła szybkość przesyłu i przetwarzania informacji zakodowanych w postaci fal świetlnych, milionkrotne zwiększenie pojemności kanałów transmisji oraz spodziewana, niewiarygodnie duża gęstość upakowania informacji w odpowiednio skonstruowanych pamięciach. W pracach badawczo-konstrukcyjnych napotkano na całkowicie nowe i skomplikowane problemy natury technologicznej i technicznej. Aktualne ich rozwiązania są niezadowalające. W przeciwieństwie do procesów elektronicznych procesy optyczne są pozbawione cech uniwersalności dla konkretnej realizacji układowej, mogą z dużą szybkością realizować tylko określoną klasę operacji. Zasadniczymi elementami architektury komputerów optycznych są lasery. Przede wszystkim stanowią one źródło światła, a także często decydują o precyzyjności innych komponentów architektury. Pożądane jest, aby wytwarzana wiązka charakteryzowała się bardzo dużą spójnością czasowo-przestrzenną, wysoką stabilnością częstotliwości, dużą mocą i dobrą podatnością na modulację. Z drugiej strony współczesne źródła światła powinny mieć odpowiednio niewielkie gabaryty i wykazywać się dużą żywotnością oraz sprawnością energetyczną. Wymagania powyższe są trudne do spełnienia. Dla przykładu lasery półprzewodnikowe są wielkości łebka od szpilki, niestety dyskwalifikuje je mała moc i niska spójność wytwarzanej wiązki. Natomiast lasery gazowe cechujące się idealnymi parametrami emitowanego światła wykazują dużą podatność na uszkodzenia oraz niską sprawność energetyczną. Wielkie trudności sprawia konstruktorom wykonanie niektórych specjalistycznych elementów i przetworników optycznych o wysokiej precyzyjności w technologii monolitycznej optyki zintegrowanej (scalonej).

Współczesne osiągnięcia i kierunki rozwoju

Aktualne badania i prace konstrukcyjne koncentrują się wokół problemów: hybrydyzacji (połączenia) środowisk — optycznego i mikroelektronicznego, dalszego doskonalenia procesów technologicznych oraz rozwoju optoelektroniki. Równolegle próbuje się poprawić zasadnicze parametry współcześnie istniejących systemów. Trwają poszukiwania nowych rozwiązań sprzętowych i systemowych (struktury RISC, komputery jednoukładowe, wieloprogramowość i wielodostęp oraz zrównoleglenie procesów). Dynamiczny rozkwit przeżywa sfera technologii układów scalonych. W jednym chip'ie można umieścić do 256 mikroprocesorów, zaś najnowszy superchip o powierzchni niespełna 14 cm² zawiera 28 milionów mikroelementów. Trzeba było aż 10 lat oczekiwania w sejfach koncernu IBM, aby koncepcja szybkich dysków sztywnych typu Winchester ujrzała światło dzienne. Najnowsze rozwiązania wykorzystują laserowe techniki tworzenia zapisu sygnału cyfrowego i jego odczytu (analogia do fonicznych płyt CD). Firma Sanyo produkuje optyczne stacje hard dysków (SOF 8003D). Na jednej



płycie można pomieścić od 0,5 do 1 GB w zależności od jej średnicy. Firmy 3M oraz Verbatim na targach CeBIT'87 w Hannowerze zaprezentowały dyskietki optyczne umożliwiające wielokrotny zapis i odczyt (pojemność dyskietek od 50 do 400 MB). Odpowiednie stacje są w opracowaniu.

Dość obiecująco przedstawiają się zagadnienia zrównoleglenia procesów. W tej dziedzinie preferowana jest wieloprocesorowość oraz systemy wielokomputerowe. Dzięki temu uzyskuje się pewien poziom współbieżności i wielozadaniowości prowadzonych procesów. Jak wiemy, większość z opracowanych procesów ma strukturę architektoniczną opartą na strukturze zaproponowanej przez Johna von Neumana. Zgodnie z tym licznik rozkazów wymusza operacje według programu zakodowanego w pamięci. Poszczególne operacje są wykonywane w określonej sekwencji kroków. Taka struktura ogranicza szybkość pracy. Dla ominięcia takiej sytuacji stosuje się więcej niż jeden procesor. W typowych rozwiązaniach architektonicznych komputerów procesor centralny steruje i kontroluje poprawność pracy pozostałych komponentów struktury, procesory uzupełniające z zasady wykonują funkcje specjalne (procesor, graphic controller itp.). Technika ta, stosunkowo droga i mało efektywna obciążona jest także niedogodnościami struktury von Neumana. Wieloprogramowość i wielodostęp w niewielkim stopniu poprawiają funkcjonalność struktury. Efekty tych zabiegów są zauważalne (w makroczasie) tylko przy stosunkowo niedużych systemach wykonujących nieskomplikowane algorytmy. Alternatywne rozwiązania dla tych zagadnień stanowią sieci komputerowe i technika decentralizacji „inteligencji" systemu dla procesów słabopowiązanych.

Transputery

W związku z niedogodnościami struktury von Neumana zachodzi pytanie, czy nie można by podzielić program na części, które byłyby wykonywane jednocześnie (w mikroczasie) w różnych procesach. Aby w pełni wykorzystać potencjalne możliwości techniki scalonej potrzebny jest system o dużo większym stopniu równoległego współdziałania niż to jest możliwe w standardowych systemach. W 1985 r. niewielka firma INMOS wprowadziła na rynki zbytu nową jakościowo rodzinę mikro: transputery T212 i T412, które zostały opracowane jako programowalny komponent systemowy specjalnie pod kątem rozwiązania wyżej postawionego zadania. Nazwa ta pochodzi od dwóch słów: TRANSistor oraz comPUTER i jest nawiązaniem do podobieństwa funkcji spełnianych przez transputery w systemach komputerowych z funkcjami tranzystorów w układach elektronicznych.

Transputer posiada procesor centralny, kilka pamięci oraz cztery linie komunikacyjne, będące jednocześnie układem wejściowym i wyjściowym. Do każdej z linii można bezpośrednio podłączyć inny transputer lub poprzez adapter inne urządzenia mikroprocesorowe. Transputer komunikuje się z własną wewnętrzną pamięcią RAM za pomocą równoległej szyny interfejsu z szybkością do 20 Mb/s na każdej linii szyny. Niezależnie od tego 16-bitowy interfejs pamięci lokalnej off-chip T212 posiada stałą konfigurację (z 16-bitową szyną danych). Szybkość przesyłania danych między pamięcią lokalną a współpracującą z pamięcią transputera jest 2-, 3-krotnie mniejsza. 32-bitowy T414 posiada o wiele bardziej złożony interfejs. Umożliwia on 13 różnych konfiguracji realizowanych przez zaszycie w kości lub programowo. Szybkość transmisji na liniach komunikacyjnych standardowo wynosi 10—20 Mb/s zależnie od typu zegara.

W kwietniu 1987 r. na rynkach światowych pojawiło się następne dziecko transputerowej rodziny: IMS T—800, o pełnej 32-bitowej architekturze wewnętrznej. Twórca transputera Iann Barron do struktury mikroukładu włączył jednostkę arytmetyki zmiennoprzecinkowej oraz kilka innych, bardzo interesujących ulepszeń. W ten sposób T—800 jest znacznie szybszy od powszechnie stosowanych konkurentów: Intela 80386, Motoroli 68020, Nat-Semi 320—32 czy Fairchild Clippera. Za wadę transputerów uważa się fakt, że INTEL nie jest ich producentem. Wtedy promocja na rynkach byłaby z pewnością skuteczniejsza. Pogląd ten wydaje się zbyt przesadny. Transputerami interesują się firmy z USA i Japonii. Istnieją liczne ich sprzętowe zastosowania, m.in. IBM stosuje z powodzeniem transputery w sterownikach monitorów kolorowych wysokiej rozdzielczości. T—800 ma moc przetwarzania porównywalną z mocą najpopularniejszego dotychczas supermikrokomputera VAX 860. Nie ma w tym nic dziwnego, gdyż od samego początku transputer był projektowany pod kątem współbieżności procesów w dużych systemach komunikacyjnych. Oczywiście nie jest prawdą, że procesory Intel 80386 i Motorola 68020 nie mogą być używane w równoległych procesach, jednakże ich wewnętrzna architektura oraz oprogramowanie są zwrócone na obsługę przede wszystkim własnego mikrośrodowiska. Trudności realizacyjne wynikają z tego, że procesy komunikacyjne między tego typu CPU odbywają się poprzez zewnętrzne szyny. Planowanie bloków funkcjonalnych składających się z 100 takich mikroprocesorów znacznie komplikuje projekt struktury PCB interfejsów dla tych wszystkich szyn, że zadanie wydaje się niewykonalne. Trudnym zadaniem jest także wykonanie oprogramowania bloku, gdyż jednostki CPU mikroprocesorów znacznie lepiej przystosowane są do pełnienia funkcji zarządzających systemem niż współpracy równoległej ze sobą. To co jest więc istotą przystosowania T800 wynika po części z jego architektury wewnętrznej (Rys. 1).

Transputer jest układem opartym na strukturze RISC, zawierającym wszystkie elementy systemu komputerowego i może on wykonywać procesy bez potrzeby dodatkowych urządzeń rozszerzających niezależnie od sygnału zegarowego. Jego konwencjonalna jednostka CPU posiada podwyższone parametry arytmetyczno-logiczne dzięki zastosowaniu w niej 64-bitowej jednostki operacji zmiennoprzecinkowych, umożliwiających tym zwiększenie szybkości ich realizacji do 1,5 MFLOPS[2] (dla T414 wynosiła ona 100 kFLOPS). W dodatku posiada też własną pamięć on-chip DRAM (2 KB), z którą wymiana jest szybsza (80 Mb/s, w stosunku do 20 Mb/s dla off-chip DRAM). Ponadto wyposażony jest także w 4 linie komunikacyjne, przez które może bezpośrednio dokonywać transmisji z innym transputerem lub poprzez adapter z innymi urządzeniami komputerowymi. Możliwa jest bezpośrednia transmisja między dwiema pamięciami dwóch różnych transputerów. Kod transputera jest bardzo zwarty. Większość wykonywanych instrukcji jest jednobajtowa, i wykonywane są w jednym cyklu. W przyszłości planuje się znaczne zwiększenie złożoności struktur wewnętrznych. Do programowania układów transputerowych służy specjalizowany język wysokiego poziomu OCCAM. Dokładniejsze informacje o tym języku oraz innym oprogramowaniu w następnym odcinku z tego cyklu.

[1] holografia — polega na odtworzeniu rzeczywistego obrazu poprzez działanie falą odniesienia na zarejestrowany hologram (obraz interferencyjny fali odbitej od przedmiotu oraz fali odniesienia). Teoretyczna gęstość zapisu jest bliska 10^{10}b/cm² (uzyskana 10^7), zaś w hologramach przestrzennych 10^{13}b/cm³.

[2] FLOPS (floating-point operation per secound) — operacje zmiennoprzecinkowe na sekundę.

W.M.S.



— Interesuje nas ktoś, kto zna dobrze ten instrument... rys. Puchalski

W y k r e s

Ludwik PIELA · Tomasz MROWIEC

W dzisiejszym odcinku zamieszczamy zgodnie z obietnicą, kompletny program umożliwiający kreślenie wykresów. Tworzy on, wyświetla i drukuje wykresy słupkowe, liniowe lub punktowe. Dane do kreślenia mogą być wprowadzone z klawiatury (gdy nie ma ich zbyt dużo) lub ze zbioru (zbiorów). Dodatkowo akceptuje on dane z programu przetwarzania arkusza obliczeniowego Visicalc. Napisany w języku BASIC może być używany na wszystkich 8-bitowych komputerach Atari posiadających 48KB pamięci wewnętrznej i stacje dysków lub magnetofon.

Mniej zaawansowani użytkownicy mogą używać go bezpośrednio po wprowadzeniu. Bardziej zaawansowani mogą wprowadzić własne ulepszenia i dostosować go do swoich potrzeb.

W swojej aktualnej postaci program może wyświetlić jednocześnie zawartość trzech zbiorów danych zawierających do 100 wartości każdy. Zbiory muszą zawierać jednakową liczbę danych z przedziału [0,900000].

Po wczytaniu danych program pyta o nazwy wykresu i osi Y. Jeśli nie chcemy nadawać nazw, naciskamy <RETURN>. Jeśli kreślimy dwa lub trzy wykresy jednocześnie, musimy wprowadzić nazwy dla każdego z nich. Dodatkowo musimy zaakceptować (ewentualnie zmienić) wartości minimalną i maksymalną oznaczaną na osi pionowej. Następnie program wybiera skalę dla osi poziomej X i prosi o wybranie sposobu jej opisu (litery, miesiące, liczby, bez oznaczania, ewentualnie własne oznaczenia).

Możemy kreślić następujące rodzaje wykresów: **liniowy**, **punktowy** lub **słupkowy**. W przypadku tego ostatniego musimy być świadomi ograniczeń wynikających z szerokości słupków. Na przykład, wykres słupkowy zawartości trzech zbiorów zmieści się na ekranie, jeśli ma nie więcej niż 12 wartości w każdym zbiorze.

Zbiory danych, przechowujące wartości, które chcemy przedstawić w postaci graficznej, możemy przygotować dwoma sposobami. Pierwszy z nich polega na wykorzystaniu jednego z edytorów tekstów, np. AtariWriter, PaperClip lub Speed Script. Musimy przy tym przestrzegać następującej zasady: wszystkie liczby muszą być umieszczone w jednej kolumnie. W tym celu po wprowadzeniu liczby naciskamy klawisz <RETURN>. Zatem dane w zbiorze powinny wyglądać następująco.

```
  887
 2745
   91
13019
  562
 6743
```

W przypadku korzystania ze stacji dysków każdy zbiór ma własną nazwę, nadawaną podczas przechowywania. W przypadku magnetofonu wystarczy zapisać zbiory jeden po drugim (będą one oczywiście bez nazw).

Drugi sposób przygotowania danych do wykreślenia polega na wykorzystaniu programu przetwarzania arkusza obliczeniowego VisiCalc. Z jego pomocą możemy przygotować dane w postaci kolumn liczb. Przechowujemy je w pamięci zewnętrznej za pomocą opcji PF (wydruk do zbioru). Najpierw umieszczamy kursor u góry pierwszej kolumny danych arkusza obliczeniowego. Wprowadzamy rozkaz (PF, a następnie nazwę zbioru (dla zbioru na taśmie magnetycznej podajemy C). Po propozycji „lower right" przesuwamy kursor na koniec kolumny danych i naciskamy <RETURN>. Musimy powtórzyć te kroki dla każdej przechowywanej kolumny danych.

Istnieje jeszcze jeden sposób wprowadzania danych do programu WYKRES, przeznaczony głównie do wprowadzania niewielkiej liczby danych dla celów testowania, bądź też poznania możliwości programu. Po propozycji podania nazwy zbioru wprowadzamy „E". Teraz wartości możemy wprowadzać za pomocą klawiatury. Wprowadzanie danych kończymy naciskając CTRL—3 (znak EOF).

Możliwości programu nie będziemy opisywać. Po wprowadzeniu danych są one prezentowane w postaci menu i bardzo łatwo można je sprawdzić samemu.

```
HK 10 REM WYKRESY - cz. 1
OI 20 BRK=1:IF PEEK(53279)=5 THEN
    BRK=0
EA 40 DIM A$(20),F$(15),G$(37),T$
   (40),XL$(300),M$(72),TL$(40),Y
   L$(20),K$(25)
JA 50 DIM CL$(1),Y(100,2),R(100),
   B(100,2),PTS(3)
NE 60 M$="STYLUTMARKWIMAJCZELIPSI
   EWRZPAZLISGRU":M$(37)=M$:CL$=C
   HR$(125)
GF 70 GRAPHICS 0:POKE 82,2:POKE 8
   3,39:CLOSE #1:OPEN #1,4,0,"K:"
   :GOSUB 1930
DO 80 ? CL$:POSITION 17,1:? "WYKR
   ESY":?
VJ 90 ? :? "Program ten kresli wy
   kresy skladajace sie maksymaln
   ie ze 100 punktow na    podst
   awie danych ze zbioru."
NF 95 GOSUB 2000
WR 100 ? :? "Czy chcesz wykaz zbi
   orow dyskowych              (
   T/N)";
FI 110 GOSUB 1280:IF  NOT A THEN
   140
BK 120 ? CL$:TRAP 310:CLOSE #2:OP
   EN #2,6,0,"D:*.*"
CX 130 INPUT #2,A$:? A$;"  ";:IF
   A$(5,9)<>"FREE" THEN 130
KR 140 CLOSE #2:? :? :? "Ile wykr
   esow kreslic jednoczesnie
                  (1-3)";
BG 150 GET #1,FILES:FILES=FILES-4
   8:IF FILES<1 OR FILES>3 THEN 1
   50
HA 160 ? FILES:? :A=0:YMAX=0:YMIN
   =900000:G$=" ":G$(36)=" ":G$(2
   )=G$
SO 170 REM LADOWANIE ZBIOROW
MQ 180 A=A+1:? "Podaj nazwe zbior
   u ";A;"  ";:INPUT #16,A$
ZC 190 F$=A$:TRAP 310:CLOSE #2:OP
   EN #2,4,0,F$:B=0:TRAP 250
NO 200 INPUT #2,C:IF C>900000 THE
   N ? "ZBYT DUZA WARTOSC !":GOTO
    100
RV 210 IF C<0 THEN ? "DANE UJEMNE
    !":GOTO 100
DY 220 B=B+1:Y(B,A-1)=C:IF YMAX<C
    THEN YMAX=C
PA 230 IF YMIN>C THEN YMIN=C
UF 240 IF B<100 THEN 200
AS 250 IF B=0 THEN A=A-1:? "BLAD
   LADOWANIA !":GOTO 180
AO 260 PTS(A)=B:G$(A*12-11)=A$:IF
    A<FILES THEN 180
EL 270 IF FILES=1 THEN 320
TF 280 IF FILES=2 AND PTS(1)=PTS(
   2) THEN 320

MU 290 IF FILES=3 AND PTS(1)=PTS(
   2) AND PTS(2)=PTS(3) THEN 320
HB 300 ? " ZBIORY ZAWIERAJA ROZNA
    ILOSC DANYCH !":GOTO 100
MN 310 ? :? "BLAD NR ";PEEK(195):
   GOTO 100

KL 315 REM WYKRESY - cz.2
UX 320 GOSUB 1530:REM SKALOWANIE
WC 330 GOSUB 1800:REM NADANIE NAZ
   W
WD 340 REM MENU
HM 350 TRAP 40000:POKE 702,64:GRA
   PHICS 0:GOSUB 1930:POKE 82,11:
   ? :? "    M E N U":? "    -----
   ----"
AH 360 ? :? "1. HISTOGRAM":? "2.
   WYKRES LINIOWY":? "3. WYKRES P
   UNKTOWY"
PE 370 ? "4. ZMIANA SKALI":? "5.
   ZMIANA NAZW":? "6. NOWE DANE":
   ? "7. KONIEC PROGRAMU"
IO 380 ? :? " CO WYBIERASZ?";:POK
   E 82,2
DV 390 GET #1,A:GH=A-48:IF GH<1 O
   R GH>7 THEN 390
FB 400 ? GH:IF GH=4 THEN GOSUB 15
   30:GOTO 350
NT 410 IF GH=5 THEN GOSUB 1800:GO
   TO 350
LJ 420 IF GH=6 THEN RUN
PJ 430 IF GH=7 THEN ? CL$:END
DK 440 IF GH>1 THEN ? :? :? "CZY
   KRESLIC PIONOWE LINIE (T/N)";:
   GOSUB 1280:VL=A
QZ 450 REM
YA 460 GRAPHICS 8+16:GOSUB 1930:C
   OLOR 1
KD 470 PLOT 56,0:DRAWTO 56,155:DR
   AWTO 319,155
PL 480 FOR A=LOW TO HI STEP (HI-L
   OW)/DIV:C=155-(A-LOW)*SCALE
CB 490 PLOT 50,C:DRAWTO 55,C:T$=S
   TR$(A)
NA 500 X=6-LEN(T$):Y=C-3:GOSUB 74
   0
IC 510 FOR B=59 TO 317 STEP 3:PLO
   T B,C:NEXT B
CX 520 NEXT A
IH 530 IF TL$<>"" THEN T$=TL$:X=2
   0-INT(LEN(T$)/2):Y=0:GOSUB 740
ID 540 IF YL$<>"" THEN FOR A=1 TO
    LEN(YL$):T$=YL$(A,A):X=0:Y=72
   -LEN(YL$)*4+A*8:GOSUB 740:NEXT
    A
ME 550 IF XLABEL=5 THEN 600
YI 560 A=1:FOR B=1 TO PTS:A$=XL$(
   A,A+LL-1)
```

```
LO 570 FOR C=1 TO LL:T$=A$(C,C):X=A(B)/8:Y=149+C*7:GOSUB 740:NEXT C
IQ 580 A=A+LL:IF A>LEN(XL$) THEN A=1
DV 590 NEXT B
ST 600 IF FILES>1 THEN ON FILES-1 GOSUB 840,900
HV 610 IF VL>0 AND GH>1 THEN FOR A=1 TO PTS:FOR B=154 TO 9 STEP -5:PLOT A(A),B:NEXT B:NEXT A
KZ 620 ON GH GOSUB 980,1040,1040
RJ 630 T$="(M)enu":X=0:Y=160:GOSUB 740:T$="(D)ruk":X=0:Y=168:GOSUB 740
KH 640 GET #1,A:IF A<>77 AND A<>68 THEN 640
OF 650 IF A=77 THEN 350
UI 660 COLOR 0:FOR A=0 TO 55:PLOT A,160:DRAWTO A,174:NEXT A:COLOR 1:TRAP 630

QY 1515 REM WYKRESY - cz.3
TX 1520 REM SKALOWANIE
CG 1530 ? CL$:? "Ustawienie skali osi Y"
MB 1540 ? :? "Podaj dolna wartosc skali.":? "0-";YMIN;") 0";
YS 1550 ? "0";:LOW=0:TRAP 1570:INPUT LOW:IF LOW<0 OR LOW>YMIN THEN 1540
FH 1560 REM OBLICZENIE SKALI
HZ 1570 HI=YMAX-LOW
NW 1580 POWER=0
QP 1590 IF HI>=100 THEN HI=HI/10:POWER=POWER+1:GOTO 1590
HN 1600 IF HI<10 THEN HI=HI*10:POWER=POWER-1:GOTO 1600
TY 1610 HI=INT(HI+0.99)
PE 1620 IF HI/12=INT(HI/12) THEN DIV=12:GOTO 1660
JT 1630 IF HI/10=INT(HI/10) THEN DIV=10:GOTO 1660
FB 1640 IF HI/8=INT(HI/8) THEN DIV=8:GOTO 1660
ZF 1650 HI=HI+1:GOTO 1620
EM 1660 IF POWER>0 THEN FOR A=1 TO POWER:HI=HI*10:NEXT A
OY 1670 IF POWER<0 THEN FOR A=POWER TO -1:HI=HI/10:NEXT A
TR 1680 SCALE=144/HI
NV 1690 HI=HI+LOW
OT 1700 ? :? "Gorna wartosc na osi Y ";HI;".":? "Zgadzasz sie (T/N)?";
KQ 1710 GOSUB 1280:IF A THEN 1760
MY 1720 TRAP 1720:? :? "Podaj gorna granice (wieksza niz ":? YMAX;")";:INPUT HI
NF 1730 IF HI<YMAX OR HI>900000 THEN 1720
TT 1740 HI=HI-LOW:GOTO 1580
IZ 1750 REM
JV 1760 ? :? "Przeliczam punkty ...":PTS=PTS(1):FOR A=1 TO PTS:FOR B=1 TO FILES
XA 1770 A(A)=INT(245/PTS*(A-0.5)+65):A(A)=INT(A(A)/2)*2:B(A,B-1)=INT(155.5-(Y(A,B-1)-LOW)*SCALE)
EO 1780 NEXT B:NEXT A:RETURN

SD 1790 REM WYKRESY - cz.4
NQ 1800 ? CL$:? "Nadawanie nazw (";PTS;" punktow )"
RU 1810 ? :? "Podaj nazwe wykresu.":INPUT TL$
QE 1820 ? :? "Podaj nazwe osi Y.":INPUT YL$
MP 1830 IF FILES=1 THEN 1890
YU 1840 ? :? "Podaj oznaczenia dla zbiorow danych.":? "(Maksymalnie 8 znakow)"
RS 1850 ? :K$=" ":K$(24)=" ":K$(2)=K$:G$(37)=" "
OZ 1860 FOR A=1 TO FILES:? "Zbior -";A;" (";G$(A*12-11,A*12);")";
VM 1870 INPUT A$:IF A$<>"" THEN K$(A*8-7)=A$
QQ 1880 NEXT A:K$(25)=" "
YI 1890 ? :? "Oznaczenie osi X  Wlasne ,Litery , Miesiace , Numery , lub Bez opisu ?";
SA 1900 POKE 702,64:GET #1,XLABEL:GOSUB 1980:XLABEL=XLABEL-74:IF XLABEL<1 OR XLABEL>5 THEN 1900
EX 1910 ? CHR$(XLABEL+74):IF XLABEL<5 THEN ON XLABEL GOSUB 1310,1440,1460,1490
AX 1920 RETURN
XA 1930 POKE 710,12:POKE 712,12:POKE 709,2:POKE 77,0
IY 1940 IF BRK THEN POKE 16,112:POKE 53774,112
BG 1950 RETURN

NF 725 REM WYKRESY - cz.5
FR 730 REM WYDRUK TEKSTU W GR.8
XO 740 X=INT(X):Y=INT(Y):COLS=40:START=(PEEK(89)*256+PEEK(88))+Y*COLS+X
HY 750 FOR E1=1 TO LEN(T$):E3=ASC(T$(E1))
XR 760 IF E3<32 OR (E3>127 AND E3<160) THEN E3=E3+64:GOTO 780
CS 770 IF E3>31 AND E3<96 THEN E3=E3-32
WM 780 CHARSET=PEEK(756)*256+E3*8
TG 790 FOR E2=7 TO 1 STEP -1
IA 800 POKE START+E2*COLS,PEEK(CHARSET+E2):NEXT E2
YD 810 X=X+1:IF X>COLS THEN START=START+COLS*8:X=0
JC 820 START=START+1:NEXT E1:RETURN

SO 1300 REM WYKRESY - cz.6
KM 1310 ? CL$:? "Oznaczenia na osi x dla ";PTS;" punktow"
JO 1320 ? :? "ILE ZNAKOW (1-3)?";
ZR 1330 TRAP 1330:GET #1,LL:LL=LL-48:IF LL<1 OR LL>3 THEN 1330
DP 1340 ? LL:? :? "Podaj oznaczenie dla punktow.          Nacisnij RETURN Jesli chcesz"
QZ 1350 ? "powielic oznaczenia.":A=1
XP 1360 ? :? "Podaj oznaczenie punktu ";A;".   >";:B=1
IQ 1370 GET #1,C:IF C=155 AND B=1 AND A>1 THEN ? "POWIEL":RETURN
ZU 1380 IF C<32 OR C>123 THEN 1370
KJ 1390 XL$(A*LL-LL+B)=CHR$(C):? CHR$(C);
LA 1400 IF B<LL THEN B=B+1:GOTO 1370
ZE 1410 IF A<PTS THEN A=A+1:GOTO 1360
SX 1420 ? :RETURN
OM 1430 REM LITERY
NP 1440 FOR A=65 TO 90:XL$(A-64)=CHR$(A):NEXT A:LL=1:RETURN
EV 1450 REM MIESIACE
EB 1460 TRAP 1460:? :? "Miesiac poczatkowy (1-12)";:INPUT A:A=INT(A):IF A<1 OR A>12 THEN 1460
MA 1470 LL=3:B=A*3-2:XL$=M$(B,B+35):RETURN
JC 1480 REM LICZBY
OD 1490 TRAP 1490:? :? "Zaczynamy od NR (0-900)";:INPUT A:A=INT(A):IF A<0 OR A>900 THEN 1490
NM 1500 XL$=" ":XL$(300)=" ":XL$(2)=XL$
OQ 1510 LL=LEN(STR$(A+PTS)):FOR B=1 TO PTS+1:XL$(B*LL-LL+1)=STR$(A+B-1):NEXT B:RETURN

OY 825 REM WYKRESY - cz.7
QZ 830 REM
GQ 840 T$="Znak   -             -":T$(9)=K$(1,8):T$(21)=K$(9,16):X=7:Y=182:GOSUB 740
ON 850 IF GH>1 THEN X=100:Y=185:F=1:GOSUB 1110:X=204:F=2:GOSUB 1110:RETURN
SC 860 FOR A=106 TO 110:PLOT A,182:DRAWTO A,188:NEXT A
RB 870 PLOT 202,188:DRAWTO 202,182:DRAWTO 206,182:DRAWTO 206,188:DRAWTO 202,188:PLOT 203,185:DRAWTO 205,185
ZU 880 RETURN
TZ 890 REM TRZY ZBIORY
YC 900 T$="Znak                -":T$(9)=K$(1,8):T$(21)=K$(9,16):T$(33)=K$(17,24)
RS 910 X=0:Y=182:GOSUB 740
ZX 920 IF GH>1 THEN X=52:Y=185:F=1:GOSUB 1110:X=148:F=2:GOSUB 1110:X=244:F=3:GOSUB 1110:RETURN
EE 930 FOR A=50 TO 54:PLOT A,182:DRAWTO A,188:NEXT A
KI 940 PLOT 146,188:DRAWTO 146,182:DRAWTO 150,182:DRAWTO 150,188:DRAWTO 146,188:PLOT 147,185:DRAWTO 149,185
NH 950 PLOT 242,188:DRAWTO 242,182:DRAWTO 246,182:DRAWTO 246,188:PLOT 244,182:DRAWTO 244,188
ZR 960 RETURN
FL 970 REM HISTOGRAM
KR 980 FOR A=1 TO PTS
FL 990 IF FILES=1 THEN B=-2:GOSUB 1190
KL 1000 IF FILES=2 THEN B=-4:GOSUB 1180:B=2:GOSUB 1190
N2 1010 IF FILES=3 THEN B=-8:GOSUB 1180:B=-2:GOSUB 1190:B=4:GOSUB 1230
CT 1020 NEXT A:RETURN
FT 1030 REM WYKRES PUNKTOWY I LINIOWY
XP 1040 FOR A=0 TO FILES-1:FOR B=1 TO PTS
BJ 1050 X=A(B):Y=B(B,A):F=A+1:GOSUB 1110
NQ 1060 NEXT B:NEXT A
II 1070 IF GH=3 THEN RETURN
FL 1080 FOR A=0 TO FILES-1:PLOT A(1),B(1,A):FOR B=2 TO PTS:DRAWTO A(B),B(B,A)
ED 1090 NEXT B:NEXT A:RETURN
WO 1100 REM KRESLENIE PUNKTOW
FB 1110 COLOR 0:PLOT X,Y-2:DRAWTO X,Y+2:COLOR 1
CQ 1120 IF F=1 THEN PLOT X,Y:PLOT X,Y-1:PLOT X+1,Y-1:DRAWTO X+1,Y+1:DRAWTO X-1,Y+1:DRAWTO X-1,Y-1
BE 1130 IF F=2 THEN PLOT X,Y:PLOT X-1,Y-2:DRAWTO X+1,Y-2:PLOT X+2,Y-1:DRAWTO X+2,Y+1
IK 1140 IF F=2 THEN PLOT X+1,Y+2:DRAWTO X-1,Y+2:PLOT X-2,Y+1:DRAWTO X-2,Y-1
MG 1150 IF F=3 THEN PLOT X-2,Y-2:DRAWTO X+2,Y+2:PLOT X+2,Y-2:DRAWTO X-2,Y+2
AT 1160 RETURN
IT 1170 REM
DE 1180 FOR C=B TO B+4:PLOT A(A)+C,B(A,0):DRAWTO A(A)+C,155:NEXT C:RETURN
JS 1190 COLOR 0:FOR C=A(A)+B+1 TO A(A)+B+3:PLOT C,B(A,1):DRAWTO C,154:NEXT C:COLOR 1
GA 1200 PLOT A(A)+B,155:DRAWTO A(A)+B,B(A,1):DRAWTO A(A)+B+4,B(A,1):DRAWTO A(A)+B+4,155
QK 1210 IF B(A,1)<152 THEN FOR C=B(A,1) TO 153 STEP 3:PLOT A(A)+B,C:DRAWTO A(A)+B+4,C:NEXT C
AJ 1220 RETURN
OQ 1230 IF B(A,2)>153 THEN 1250
TR 1240 COLOR 0:PLOT A(A)+B+1,B(A,2):DRAWTO A(A)+B+1,154:PLOT A(A)+B+3,B(A,2):DRAWTO A(A)+B+3,154:COLOR 1
KO 1250 PLOT A(A)+B,155:DRAWTO A(A)+B,B(A,2):DRAWTO A(A)+B+4,B(A,2):DRAWTO A(A)+B+4,155
BU 1260 PLOT A(A)+B+2,B(A,2):DRAWTO A(A)+B+2,155:RETURN
WA 1280 ? "T";:POKE 702,64:GET #1,A:IF A<>78 AND A<>84 AND A<>155 THEN 1280
QW 1285 IF A=155 THEN A=84
CU 1290 ? CHR$(A):A=(A-78)/11:RETURN

PN 670 REM WYKRESY - cz.8
LD 680 CLOSE #5:OPEN #5,8,0,"P:"
GN 690 TRAP 1960:DM=PEEK(88)+PEEK(89)*256:DM=DM+40*191
TC 700 FOR X=DM TO DM+39
```

```
KK 705 LA$=CHR$(K0):LA$(192)=LA$:
   LA$(2)=LA$
KT 710 W=USR(1536,X,ADR(LA$)):? #
   5;CHR$(27);"9";CHR$(27);"A";CH
   R$(0);CHR$(192);LA$:NEXT X
NQ 715 CLOSE #5
RO 720 TRAP 40000:GOTO 630
GK 1960 GRAPHICS 8+16+16:GOSUB 19
   30:POKE 712,64:POKE 752,1
JI 1965 ? CHR$(125);CHR$(29);CHR$
   (29);"          ***DRUKARKA ODLAC
   ZONA***"
NZ 1970 FOR X=1 TO 90:NEXT X:GRAP
   HICS 8+16+32:GOSUB 1930:GOTO 6
   30
HP 1980 IF XLABEL=87 THEN XLABEL=
   [illegible]
KT 1990 IF XLABEL=66 THEN XLABEL=
   79
CM 1995 RETURN
BV 2000 RESTORE 2030:FOR T=1 TO 6
   1:READ Q:POKE 1535+T,Q
ML 2010 NEXT T:DIM LA$(193):RETUR
   N
CW 2030 DATA 104,104,141,21,6,104
   ,141,20,6,104,141,27,6,104,141
   ,26,6,160,193,173,255,255,136,
   240,35,141,255,255,238
QQ 2040 DATA 26,6,240,21,173,20,6
   ,56,233,40,141,20,6,144,4,24,7
   6,19,6,206,21,6,76,19,6,238,27
   ,6,76,33,6,96
```

Dla lepszego poznania programu, w celu ewentualnego zmodyfikowania go, omówmy krótko funkcje realizowane przez poszczególne segmenty programu.

Linie 80—310 realizują wprowadzanie danych. W ramach tego segmentu linie 120—130 odczytują i wyprowadzają na ekran katalog zbiorów dyskietki, oczywiście tylko wtedy, gdy chcemy sprawdzić jej zawartość. Wprowadzanie danych realizowane jest w liniach 200—310. W trakcie tej czynności następuje kontrolowanie zakresu wartości danych oraz wyznaczanie wartości maksymalnej i minimalnej (potrzebne później podczas skalowania). Segment ten wprowadza dane z podanej liczby zbiorów i przechowuje je w tablicy Y. Sprawdza także, czy liczba danych we wszystkich zbiorach jest jednakowa.

Kolejną czynnością wykonywaną przez program jest skalowanie wartości w celu odpowiedniego rozmieszczenia ich na ekranie. W tym momencie znane są wartości minimalna i maksymalna. Użytkownik otrzymuje propozycję wybrania położenia osi X, od 0 do Ymin, w zależności od potrzeb. Realizują to linie programu 1530—1550. Następnie w liniach 1570—1690 dobierany jest najlepszy współczynnik skalowania. Po wyznaczeniu go użytkownik otrzymuje propozycję górnej granicy na osi Y (linie 1700—1740) i może zgodzić się z tym lub nie. W przypadku braku akceptacji możemy sami wybrać górną granicę, która musi być większa od Ymax, i następuje ponowne dobieranie współczynnika skalowania. W liniach 1760—1780 następuje przeliczenie wartości danych na współrzędne ekranu oraz wybranie symbolu oznaczającego na wykresie punkty z każdego zbioru.

Następnie program przechodzi do przygotowania opisu wykresu (linie 1800—1920). W tym celu pyta użytkownika o nazwę wykresu, oznaczenie osi Y, oznaczenie dla danych z poszczególnych zbiorów oraz sposób opisu osi X. W ostatnim przypadku możemy wybrać jedno z oznaczeń standardowych — miesiące, litery, liczby, bez opisu — lub wprowadzić własne. Mogą one mieć od 1 do 3 znaków. Poszczególne opisy osi X realizowane są przez podprogramy zawarte w liniach 1310—1510.

Po wykonaniu powyższych czynności program prezentuje na ekranie swoje możliwości (linie 350—380), które możemy wybrać wprowadzając cyfrę z przedziału 1—7. W przypadku wybrania opcji 1, 2 lub 3 przechodzimy do wykonania czynności przygotowawczych, tj.: wykreślenia osi współrzędnych i opisania ich (linie 460—610). Jak widać dosyć często wywoływany jest podprogram rozpoczynający się od linii 740. Jest to segment (procedura) drukujący tekst na ekranie w trybie graficznym 8.

Wykres kreślony jest, w zależności od wybranej opcji, przez różne segmenty programu. I tak, wykres słupkowy kreślony jest przez instrukcje zawarte w liniach 980—1020 i 1180—1260, wykres punktowy: 1040—1070 i 1110—1160, a wykres liniowy: 1040—1090 i 1110—1160

Po narysowaniu wykresu możemy przejść do menu lub wykonać kopię ekranu na drukarce. Informują o tym napisy w lewym dolnym rogu ekranu. Wydruk zawartości ekranu realizowany jest przez instrukcje zawarte w liniach 680—720, które wykorzystują procedurę w języku maszynowym, umieszczoną na stronie 6 przez instrukcje w liniach 2000—2040.

W programie WYKRES dostarczone są „domyślne" odpowiedzi na część pytań. Jeśli nam one odpowiadają wystarczy nacisnąć klawisz <RETURN>, jeśli nie, to należy podać własną propozycję, już bez naciskania klawisza <RETURN>. Dotyczy to głównie pytań, na które odpowiadamy TAK (T) lub NIE (N). Poprawną realizację takiego rozwiązania zapewnia podprogram umieszczony w liniach 1280—1290, który przekazuje odpowiedź w postaci wartości logicznej (A=1 jeśli TAK, A=0 jeśli NIE).

Działania programu nie można przerwać za pomocą klawisza BREAK. Zapewnia to podprogram w liniach 1930—1950, który musi być wywoływany po każdym użyciu instrukcji GRAPHICS. Istnieje jednak możliwość wyłączenia zakazu przerywania działania programu. Wystarczy uruchomić go z wciśniętym klawiszem <SELECT>. Realizują to instrukcje w linii 20

Do wprowadzenia programu WYKRES proponujemy wykorzystać Edytor BASIC'a („IKS" nr 4 z 1987 roku).

Tomasz MROWIEC
Ludwik PIELA

CPC

Po wczytaniu i uruchomieniu programu gdy będziesz chciał wydrukować tekst z ekranu napisz CALL &A000.

```
10 REM Kopiuje tekst z ekranu na
   drukarke
20 MEMORY &9FFF:FOR x=&A000 TO
   &A03C
30 READ a$:POKE x,VAL("&"+a$):NEXT
40 DATA cd,81,bb,cd,11,bc,3c,47
50 DATA 3e,0a,cb,27,10,fc,47,0e
60 DATA 19,21,01,01,cd,75,bb,c5
70 DATA cd,60,bb,cd,2c,a0,3e,09
80 DATA cd,5a,bb,10,f3,c1,0d,c8
90 DATA c5,3e,0a,cd,36,a0,3e,0d
100 DATA cd,36,a0,c3,18,a0,cd,2b
110 DATA bd,d8,c3,2c,a0
```

Nauka • Technika

W Japonii opracowano nowego typu suchy elektrolit, który umożliwia zminiaturyzowanie akumulatora. Wytwarzany jest on w formie niezwykle cienkiej kartki, dzięki czemu sam akumulator może mieć grubość nie przekraczającą 0,1 mm. Forma elektrolitu sprawia, że nowego typu akumulator jest odporny zarówno na niskie, jak i wysokie temperatury - w granicach od -60 do +100°C. Może on być ponadto eksploatowany w próżni na statkach kosmicznych. Miniakumulatory będą mogły być powszechnie stosowane zarówno w odbiornikach radiowych i magnetofonach, jak i nowoczesnych kartach kredytowych z wbudowanymi układami scalonymi. Obecnie prowadzone są testy doboru odpowiednich elektrod do płaskiego elektrolitu

Ciąg dalszy na str. 22

Tłumacz grafiki

Przy współpracy C-64 z drukarką inną niż Commodore lub nie posiadającą interfejsu do współpracy z tym komputerem może pojawić się problem polegający na tym, że drukarka nie będzie drukować poprawnie programów zawierających grafikę C-64.

Ponizszy program tłumaczy grafikę C-64 na język angielski lub kod ASCII, które „rozumie" i drukuje poprawnie kazda drukarka

```
1 PRINT 'TLUMACZ'
2 PRINT CHR$(147)CHR$(5):POKE 53281,0:FO
R X=0TO5:PRINT:NEXT
3 PRINT:PRINT TAB(17)'TLUMACZ'
4 FOR X=0TO3000:NEXT
15 CLR:OPEN 15,8,15
16 U$=CHR$(145):RT$=CHR$(18):LS$='
   [26 SPACJI]          '
17 RO$=CHR$(146):WT$=CHR$(5):RD$=CHR$(28
):N$=CHR$(0):POKE 53280,0:GOTO 700
60 INPUT#15,D1,D2$,D3,D4:RETURN
200 IF A=32 THEN A$=' ':RETURN
201 IF A=17THEN A$='[CSRS D]':RETURN
203 IF A=145THEN A$='[CRSR B]':RETURN
204 IF A=147THEN A$='[CLR/HOME]':RETURN
205 IF A=157THEN A$='[CRSR L]':RETURN
206 IF A=19THEN A$='[HOME]':RETURN
207 IF FT$='S'ANDA=13THENPRINT#1:RETURN
208 IF A=13THENA$='[RETURN]':RETURN
209 IFA=14 THENA$='(LOWER CASE]':RETURN
210 IF A=5THENA$='[WHT]':RETURN

211 IF A=18THENA$='[REV ON]':RETURN
212 IF A=20THEN A$='[DEL]':RETURN
213 IF A=28THEN A$='[RED]':RETURN
214 IF A=30THEN A$='[GREEN]':RETURN
215 IF A=31THEN A$='[BLUE]':RETURN
216 IF A=141THEN A$='[SHIFT RETURN]':RET
URN
217 IFA=142THENA$='[UPPER CASE]':RETURN
218 IF A=144THENA$='[BLACK]':RETURN
219 IF A=146THEN A$='[REV OFF]':RETURN
220 IF A=148THEN A$='[INST]':RETURN
221 IF A=156THEN A$=[PUR]':RETURN
222 IF A=158THEN A$='[YEL]':RETURN
223 IF A=159THEN A$='(CYN]':RETURN
224 IF A=255THEN A$='[PI]':RETURN
225 IF A=133THEN A$='[F1]':RETURN
226 IF A=134THEN A$='[F3]':RETURN
227 IF A=135THEN A$='[F5]':RETURN
228 IF A=136THEN A$='[F7]':RETURN
229 IF A=137THEN A$='{F2]':RETURN
230 IF A=138THEN A$='[F4]':RETURN
231 IF A=139 THEN A$='[F6]':RETURN

232 IF A=140 THEN A$='[F8]':RETURN
233 IF A=160 THEN A$=' ':RETURN
234 IF A=151 THEN A$='[GREY 1]':RETURN
235 IF A=152 THEN A$='[GREY 2]':RETURN
236 IF A=153 THEN A$='[GREY 3]':RETURN
237 IF A=129 THEN A$='[ORANGE]':RETURN
238 IF A=150 THEN A$='[LT RED]':RETURN
239 IF A=153 THEN A$='(LT GREEN]':RETURN
240 IF A=154 THEN A$='[LT BLUE]':RETURN
241 IF A=149 THEN A$='[BROWN]':RETURN
250 X$=STR$(A):X$=MID$(X$,2,3):A$='[CHR$
('+X$+')]':RETURN
300 IF A=153 THEN A$='PRINT':RETURN
301 IF A=151 THEN A$='POKE':RETURN
302 IF A=152 THEN A$='PRINT#':RETURN
303 IF A=129 THEN A$='FOR':RETURN
304 IF A=131 THEN A$='DATA':RETURN
305 IF A=137 THEN A$='GOTO':RETURN
306 IF A=139 THEN A$='IF':RETURN
307 IF A=141 THEN A$='GOSUB':RETURN
308 IF A=142 THEN A$='RETURN':RETURN
309 IF A=163 THEN A$='TAB(':RETURN
310 IF A=164 THEN A$='TO':RETURN
311 IF A=175 THEN A$='AND':RETURN
312 IF A=176 THEN A$='OR':RETURN
313 IF A=199 THEN A$='CHR$':RETURN
314 IF A=140 THEN A$='RESTORE':RETURN
315 IF A=170 THEN A$='+':RETURN
316 IF A=171 THEN A$='-':RETURN
317 IF A=172 THEN A$='*':RETURN
318 IF A=173 THEN A$='/':RETURN
319 IF A=174 THEN A$='[']':RETURN
320 IF A=177 THEN A$=')':RETURN
321 IF A=178 THEN A$='=':RETURN
322 IF A=179 THEN A$='(':RETURN
323 IF A=194 THEN A$='PEEK':RETURN
324 IF A=167 THEN A$='THEN':RETURN
325 IF A=133 THEN A$='INPUT':RETURN
326 IF A=132 THEN A$='INPUT#':RETURN
327 IF A=143 THEN A$='REM':RETURN
328 IF A=159 THEN A$='OPEN':RETURN
329 IF A=160 THEN A$='CLOSE':RETURN

330 IF A=169 THEN A$='STEP':RETURN
331 IF A=201 THEN A$='RIGHT$':RETURN
332 IF A=147 THEN A$='LOAD':RETURN
333 IF A=203 THEN A$='GO':RETURN
340 IF A=128 THEN A$='END':RETURN
341 IF A=130 THEN A$='NEXT':RETURN
342 IF A=134 THEN A$='DIM':RETURN
343 IF A=135 THEN A$='READ':RETURN
345 IF A=138 THEN A$='RUN':RETURN
346 IF A=144 THEN A$='STOP':RETURN
347 IF A=145 THEN A$='ON':RETURN
348 IF A=148 THEN A$='SAVE':RETURN
349 IF A=149 THEN A$='VERIFY':RETURN
350 IF A=150 THEN A$='DEF':RETURN
351 IF A=154 THEN A$='CONT':RETURN
352 IF A=155 THEN A$='LIST':RETURN
353 IF A=156 THEN A$='CLR':RETURN
354 IF A=157 THEN A$='CMD':RETURN
355 IF A=197 THEN A$='VAL':RETURN
356 IF A=161 THEN A$='GET':RETURN
357 IF A=165 THEN A$='FN':RETURN

358 IF A=166 THEN A$='SPC(':RETURN
359 IF A=168 THEN A$='NOT':RETURN
360 IF A=180 THEN A$='SGN':RETURN
361 IF A=181 THEN A$='INT':RETURN
362 IF A=182 THEN A$='ABS':RETURN
363 IF A=184 THEN A$='FRE':RETURN
364 IF A=184 THEN A$='FRE':RETURN
365 IF A=185 THEN A$='POS':RETURN
366 IF A=186 THEN A$='SQR':RETURN
367 IF A=187 THEN A$='RND':RETURN
368 IF A=188 THEN A$='LOG':RETURN
369 IF A=189 THEN A$='EXP':RETURN
370 IF A=190 THEN A$='COS':RETURN
371 IF A=191 THEN A$='SIN':RETURN
372 IF A=192 THEN A$='TAN':RETURN
373 IF A=193 THEN A$='ATN':RETURN
374 IF A=195 THEN A$='LEN':RETURN
375 IF A=196 THEN A$='STR$':RETURN
376 IF A=158 THEN A$='SYS':RETURN
377 IF A=198 THEN A$='ASC':RETURN
378 IF A=200 THEN A$='LEFT$':RETURN
379 IF A=202 THEN A$='MID$':RETURN

380 IF A=146 THEN A$='WAIT':RETURN
381 IF A=162 THEN A$='NEW':RETURN
382 IF A=136 THEN A$='LET':RETURN
383 IF A=255 THEN A$='[PI]':RETURN
390 X$=STR$(A):X$=MID$(X$,2,3):A$='CHR$(
'+X$+')':RETURN
700 PRINTCHR$(147):PRINT:PRINT:PRINT:PRI
NT:PRINT
730 NF$='':INPUT'NAZWA PLIKU';NF$
732 IF LEN(NF$)<1 OR LEN(NF$)>16 THEN PR
INT U$U$:GOTO 730
740 FT$='':PRINT'RODZAJ PLIKU:'RT$'P'RO$
'RG LUB 'RT$'S'RO$'EQ';:INPUT FT$
742 IF FT$<>'P' THEN IF FT$<>'S'THEN PRI
NT U$U$:GOTO 740
750 FD$='':PRINT'WYDRUK NA'RT$'E'RO$'KRA
N LUB 'RT$'D'RO$'RUKARKE';:INPUT FD$
752 IF FD$<>'E'THEN IF FD$<>'D'THEN PRIN
T U$U$:GOTO 750
754 IF FD$='E' THEN FD=3
756 IF FD$='D' THEN FD=4
760 SK=0:PRINT'ILE BITOW DO POMINIECIA
'RT$'(RETURN)'RO$' =0':INPUT SK
800 PRINT CHR$(147):FOR X=0TO4:PRINT
810 PRINT RT$'POSZUKIWANIE: 'NF$' ,'FT$
830 IF FD=3 THEN TY$='EKRAN'
835 IF FD=4 THEN TY$='DRUKARKE'
840 PRINT TAB(3)'WYPROWADZENIE NA: 'TY$
845 PRINT TAB(5)RT$' ( F1 = PRZERWA )'
850 CLOSE 8:OPEN 8,8,8,'0:'+NF$+','+FT$+
',R'
852 IF SK>0 THEN FOR X=0TO SK:GET#8,A$:N
EXT:X=0
855 CLOSE 1:OPEN 1,FD
856 PRINT#1,NF$
860 GOSUB 60:IF A>0 THEN PRINT CHR$(147)
D1;D2$,D3;D4
865 IF FT$='P'THEN GET#8,A$,B$
870 IF FT$='P'THEN GET#8,A$,B$
874 IF FT$='S' THEN 905
875 GET#8,C$,D$:F1=ASC(C$+N$):F2=ASC(D$+
N$)
880 IF ST=0 THEN PRINT#1,(F2*256)+F1;
890 GET#8,A$:A=ASC(A$+N$):IFA>0THENC=0
891 GETBK$:IFBK$=CHR$(133)THEN GOTO1000
892 IFA=0THENFL=0:PRINT#1:C=C+1:GOTO870
893 IF ST=0THEN 900
895 IF ST=64 THEN 1000
896 PRINT'STAN : 'ST:GOTO 1000
900 IF FL=0 AND A=34 THEN FL=1:GOTO905
901 IFFL=1ANDA>93THENGOSUB 200:GOTO 910
902 IF A=34 THEN FL=0
903 IF A<33 THEN GOSUB 200:GOTO 910
904 IF A>93 THEN GOSUB 300:GOTO 910
905 PRINT#1,CHR$(A);:A=0:GOTO 890
910 PRINT#1,A$;:GOTO 890
1000 PRINT#1:GOSUB 60
1001 PRINTCHR$(5):CLOSE1:CLOSE8:CLOSE15
1010 IF D1>0 THEN PRINT'BLAD-'
1011 PRINT'STAN DYSKIETKI:'D1;D2$,D3;D4
1015 PRINT:PRINT'CZY CHCESZ SPROBOWAC JE
SZCZE RAZ ?(T/N)'
1020 GETQ$:IF Q$='' THEN 1020
1030 IF Q$='T' THEN CLR:GOTO 15

1040 END
READY.
```

Program moze współpracować również z VIC-20, należy tylko opuścić lub odpowiednio zmienić linie 1 i 20 (ustalenie koloru tablicy i ramki).

Z programu mogą korzystać (niestety) tylko posiadacze stacji dysków. Tłumaczyć (listować) i drukować mozna za jego pomocą pliki programowe (PRG) i sekwencyjne (SEQ), zarówno na ekran, jak i na drukarkę. Mozliwy jest także skok do środka pliku przed przystąpieniem do drukowania, co pozwala na zaoszczędzenie papieru.

Działanie programu mozemy w dowolnej chwili przerwać klawiszem F1.

Program moze być bardzo przydatny dla osób, które:

— posiadają drukarkę inną niż Commodore,

— posiadają drukarkę Commodore, ale chcą lepiej poznać sposób zapisu znaków graficznych,

— mają trudności w zrozumieniu grafiki Commodore i nie posiadają drukarki

Opracował: **Tadeusz CISEK**

Wskaźniki i funkcje w języku „C"

H. KRASUSKI Z. DYJAK

Język C staje się coraz częściej stosowanym narzędziem programowania. Jego użytkownicy podkreślają wiele zalet tego języka, takich jak: szczególna przydatność do tworzenia oprogramowania systemowego, możliwość definiowania i przetwarzania nowoczesnych struktur danych oraz prostota i zwartość budowanych programów. **Inną jego zaletą, być może najistotniejszą jest to, że nie jest on związany z żadnym konkretnym sprzętem i napisane w nim programy mogą być uruchamiane na dowolnym komputerze, dla którego opracowano kompilator tego języka.**

Intensywny wzrost liczby mikrokomputerów użytkowanych zarówno przez amatorów, jak i profesjonalistów znacznie przybliża ich użytkownikom możliwość programowania w języku C, tym bardziej że jego kompilatory dostępne są w zestawie oprogramowania prawie każdego systemu mikrokomputerowego, w tym również mikrokomputerów domowych, jak AMSTRAD czy ATARI. Warto zatem zainteresować się bliżej tym językiem.

Język C jest językiem programowania ogólnego zastosowania. Jednym z celów jego zaprojektowania było wypełnienie luki istniejącej pomiędzy takimi językami jak BASIC a językiem asemblera. Język C stworzony przez programistów systemowych stanowi wysokopoziomową alternatywę dla języków asemblerowych. Jest on znacznie szybszy i oferuje programiście więcej możliwości niż BASIC (np. wykonywanie operacji logicznych na pojedynczych bitach, tworzenie programów strukturalnych), a jednocześnie jest mniej „błędotwórczy" i bardziej wydajny niż język asemblera. Programy napisane w języku C, mimo iż działają nieco wolniej niż programy w kodzie maszynowym, są jednak znacznie łatwiejsze w tworzeniu i testowaniu. Takie elementy języka jak: wskaźniki, funkcje rekurencyjne i wygodne w użyciu struktury sterujące czynią nawet potencjalnie skomplikowany program łatwym do zakodowania i przetestowania. Co więcej, programy w nim napisane, w przeciwieństwie do programów w języku asemblera mogą być przeniesione na dowolny inny komputer kosztem naprawdę niewielkich modyfikacji.

Programiści poznający język C, w sytuacji gdy wcześniej posługiwali się takimi językami, jak: PASCAL, FORTRAN, czy BASIC są początkowo zaskoczeni zastosowaniem w nim kilku rozwiązań, które trudno znaleźć w innych językach programowania. Szczególnie dwa z nich zasługują na bliższe przedstawienie — wskaźniki i funkcje, w tym również funkcje biblioteczne.

Funkcje biblioteczne w języku C stanowią zbiór procedur dostarczanych z kompilatorem (biblioteka kompilatora), realizujących operacje wejścia/wyjścia programu, obliczenia numeryczne itp. Można zatem zapytać: cóż jest odmiennego w bibliotece funkcji języka C w odniesieniu do innych języków programowania? Różnica polega na tym, że w innych językach programowania instrukcje realizujące wyżej wymienione operacje stanowią element języka, nieodłącznie z nim związany. Natomiast w języku C funkcje nie są elementem języka. Na przykład programując w języku BASIC można użyć instrukcji PRINT do wyprowadzenia danych na urządzenie zewnętrzne, mając świadomość, że instrukcja ta jest elementem tego języka. Natomiast w języku C nie ma instrukcji przeznaczonej do wyprowadzania danych, mało tego, w ogóle nie ma w nim instrukcji przeznaczonych do obsługi tzw. wejścia/wyjścia programu. Jak zatem takie operacje realizować w tym języku? Otóż do wykonywania tych operacji służą w języku C funkcje zawarte w bibliotece kompilatora. Każdy kompilator języka C zawiera bibliotekę funkcji zdefiniowanych przez twórców języka oraz pewną liczbę funkcji specyficznych dla konkretnej implementacji sprzętowej. Na przykład: kompilator języka C opracowany dla mikrokomputerów ATARI XL/XE (DEEP BLUE C /DBC/) zawiera bibliotekę funkcji realizujących arytmetykę zmiennoprzecinkową. Jest to biblioteka specyficzna dla tego kompilatora, której istnienie spowodowane jest niestandardowym sposobem wykonywania obliczeń zmiennoprzecinkowych na tym sprzęcie. W kompilatorach dedykowanych dla mikrokomputerów, w których zastosowano standardowe liczby zmiennoprzecinkowe taka biblioteka w ogóle nie istnieje. Obok niej kompilator DBC posiada standardowe funkcje języka C takie jak: printf[], strlen[], getchar[], putchar[], tolower[], toupper[] i inne. Funkcje o takich nazwach występują we wszystkich kompilatorach języka C, bez względu na sprzęt, na który je opracowano. Konsekwencją faktu, że funkcje nie są elementami języka C są drobne różnice w sposobie ich realizacji na konkretnym typie mikrokomputera.

Jednak funkcje biblioteczne nie są tematem niniejszego artykułu. Konstrukcja całego programu w języku C oparta jest o idee funkcji. Każdy program w tym języku musi zawierać przynajmniej jedną funkcję. Oczywiście może również zawierać ich wiele. Ogólnie, funkcjami w języku C nazywane są, definiowane przez programistę, procedury lub podprogramy realizujące określoną operację, na przykład: wyprowadzenie wyników obliczeń na ekran, obliczenia wartości wyrażenia itp. Funkcje mogą być oddzielnie kompilowane i testowane, a następnie w postaci półskompilowanej dołączane do programu podczas fazy linkowania. Ta cecha języka stanowi decydujący czynnik jego „przewagi" nad innymi językami programowania oraz w znacznym stopniu ułatwia i przyspiesza proces tworzenia programu. Jej istotę można zobrazować następującym przykładem: każdy dobrze skonstruowany program przed zakończeniem działania powinien zapytać użytkownika, czy naprawdę chce zakończyć pracę. W przypadku negatywnej odpowiedzi program powinien uruchomić się ponownie. Przy użyciu takich języków programowania, jak BASIC czy PASCAL zakodowanie takiej ewentualności w programie wymaga kilku lub kilkunastu instrukcji. Niestety instrukcji tych musimy ponownie użyć pisząc nowy program, jeśli chcemy aby kończył on się podobnie. Jak problem ten można rozwiązać w języku C? Piszemy funkcję, która będzie realizowała zadanie pytania i analizowanie odpowiedzi użytkownika. Przykładem takiej funkcji jest funkcja o nazwie czydalej[] zamieszczona w programie i (patrz dalsza część artykułu). Funkcję tę kompilujemy i testujemy, a jej kod półskompilowany wprowadzamy do biblioteki własnych funkcji. W każdym nowym programie, gdy będziemy chcieli użyć tej funkcji, to wystarczy ją tylko wywołać, natomiast nie ma konieczności pisania jej ponownie. Oczywiste jest, że jej kod półskompilowany musimy dołączyć podczas linkowania do każdego programu, w którym funkcję tę będziemy wywoływać. To zaś nie stanowi żadnego problemu, gdyż proces linkowania wykonywany jest przez specjalny program dołączony do kompilatora.

Stałe i zmienne programowe, na których funkcja wykonuje operacje nazywane są w języku C argumentami funkcji. Liczba argumentów, które można przekazać do funkcji jest zależna od implementacji sprzętowej kompilatora. Praktycznie jest to liczba dosyć duża, np.: wspomniany wyżej kompilator DBC umożliwia przekazanie do funkcji aż 126 argumentów. Argumenty przekazywane są do funkcji poprzez wartość i ich wartość nie może być zmieniona przez funkcję. Oznacza to, że każda funkcja tworzy własne, prywatne kopie wartości argumentów, na których wykonuje operacje. Kopie te są niszczone w momencie, gdy funkcja wykona przypisane jej zadanie. Wynik obliczeń wykonanych przez funkcję nazywany jest „wartością zwracaną przez funkcję". Każda funkcja w języku C może zwrócić do miejsca wywołania tylko jedną wartość. W tym miejscu można postawić pytanie: jak zatem przetwarzać tablice za pomocą funkcji? Z powyższego wynika, że przekazanie do funkcji, jako argumentów, elementów tablicy jest praktycznie niemożliwe. I owszem, jest to prawda, gdy chcielibyśmy przekazać do funkcji wartości poszczególnych elementów tablicy. Jednak problem ten można rozwiązać inaczej. Po prostu zamiast przekazywać do funkcji wartości elementów tablicy, można przekazać jako argument funkcji, adres początku tej tablicy. W ten sposób doszliśmy do pojęcia „wskaźnik".

Wskaźnikiem w języku C jest zmienna, zawierająca adres innej zmiennej. Sposób operowania zmiennymi programu za pomocą ich adresów jest początkowo szokujący dla programisty przyzwyczajonego do używania zmiennych poprzez ich nazwy, jak ma to miejsce w językach wysokopoziomowych. Jednak mechanizm wskaźników, co pokażemy w dalszej części, w niektórych sytuacjach jest znacznie wygodniejszy w stosowaniu i umożliwia tworzenie bardziej zwartego i efektywnego tekstu źródłowego programu.

Język C oferuje programiście również klasyczny sposób przetwarzania tablic, polegający na odwoływaniu się do ich elementów poprzez indeksy. Taką metodę przetwarzania tablic zastosowano w programie 1. Należy w tym miejscu zaznaczyć, że wszystkie programy

zawarte w niniejszym artykule zostały napisane przy wykorzystaniu kompilatora Deep Blue C przeznaczonego dla mikrokomputerów ATARI XL/XE.

Program 1 realizuje funkcję przestawiania elementów tablicy znakowej (łańcucha) w taki sposób, że pierwszy znak staje się ostatnim, drugi przedostatnim, itd. Łańcuch wczytywany jest z klawiatury do tablicy wyraz [120] za pomocą funkcji gets[] z biblioteki AIO kompilatora DBC. Operacja przestawiania znaków realizowana jest w pętli:

```
for(i=0, j=dl—i;i<j; i++,j——)$(
    znak=wyraz[i];
    wyraz[i]=wyraz [j];
    wyraz[j]=znak;
    $);
```

i, j są indeksami „maszerującymi" wzdłuż tablicy wyraz [], i — od jej początku do końca, j — od końca do początku. Zmienna dl zawiera długość tablicy wyraz [] (liczbę znaków). Cały program składa się z dwóch funkcji main[] i czydalej {}. Zadaniem funkcji czydalej{} jest tylko eleganckie zakończenie programu. Natomiast zasadnicza część

Program 1

```
/*
 * program Przestawianka
 */
main()$(
char znak,wyraz[120],
int i,j,dl,
 do$(
 printf("\nNapisz dowolny wyraz:\n");
 dl=gets(wyraz);
 if(dl<=0) continue,
 printf("\nNapisales:\n");
   for(i=0,i<=dl-1;i++)
        printf(" %c",wyraz[i]),
/* teraz przestaw znaki we wprowadzonym wyrazie */
   for(i=0,j=dl-1,i<j,i++,j--)$(
     znak=wyraz[i],
     wyraz[i]=wyraz[j],
     wyraz[j]=znak,
    $)   /* koniec petli for  */
/* wyswietl przestawiony wyraz    */
 printf("\nA tak to wyglada po przestawieniu\n");
   for(i=0,i<=dl-1;i++)
      print("  %c",wyraz[i]);
  $)   /* koniec do  */
 while(czydalej()),
$)  /* koniec main  */
/*
 * funkcja czydalej()
 */
czydalej()$(
int stat,odp,T,t,N,n;
char *u,od,zer,i;
i='1',
Od='4',
zer='0';
u="K ",
T=84;   /* kod litery T */
t=116;  /* kod litery t */
N=78,   /* kod litery N */
n=110,  /* kod litery n */
stat=open(i,od,zer,u);     /* otwarcie "K:" z iocb 1    */
if(stat!=1)$(
   printf(" \n\gBlad otwarcia K. ");
   return;
$)     /* koniec if   */
do$(   /* poczatek do */
printf("\nCzy chcesz kontynuowac?");
odp=cgetc(i);
if(odp!=T && odp!=t && odp!=N && odp!=n)$(
  printf("\n\gOdpowiedz albo T albo N");
  continue,
 $)  /* koniec if */
$)   /* koniec do */
while(odp!=T && odp!=t && odp!=N && odp!=n);
close(i);
return((odp==T || odp==t) ? 1 : 0);
$) /* koniec czydalej() */
```

programu tj.: wczytywanie łańcucha znaków z klawiatury, przestawianie jego elementów oraz wyświetlanie go w postaci wczytanej i po przestawieniu realizowana jest w funkcji main[]. Widać, że pętla ta poza odmiennym sposobem jej zorganizowania (for(...)) nie różni się od pętli realizujących podobną funkcję, tworzonych na przykład w języku BASIC.

Załóżmy, że w tworzonych programach będziemy musieli dosyć często wykonywać operacje przestawiania elementów tablicy znaków. Dobrze byłoby zatem przenieść operację przestawiania do funkcji, której argumentami byłby adres tablicy i liczba jej elementów. Nazwiemy tę funkcję przestaw[]. Wówczas, w miejsce omówionej wyżej pętli możemy umieścić jej wywołanie postaci:

przestaw(tablica, liczbael);

Aby taką funkcję napisać musimy posłużyć się wskaźnikami.

Zmienne wskaźnikowe lub krótko wskaźniki, podobnie jak wszystkie inne zmienne w języku C, muszą być zadeklarowane przed ich użyciem. Każdy wskaźnik posiada typ związany z obiektem (zmienną, tablicą), który wskazuje. Deklaracja: char* U; nakazuje kompilatorowi języka C utworzenie wskaźnika typu znakowego, tzn. takiego, który wskazywał będzie obiekty tego samego typu. Natomiast deklaracja: int* i; nakazuje kompilatorowi utworzenie wskaźnika typu całkowitoliczbowego, tzn., że wskazywane przez niego obiekty są zmiennymi lub elementami tablic tego samego typu. Sam fakt zadeklarowania w programie zmiennych wskaźnikowych nie wiąże jeszcze wskaźnika z tablicą. Aby wskaźnik można było w programie sensownie wykorzystać musi on zostać prawidłowo zainicjowany, tzn. musi mu zostać przypisana wartość będąca adresem jednego z obiektów programu. Używanie wskaźników bez nadania im wartości początkowych jest częstym błędem popełnianym przez początkujących programistów w języku C. Wskaźniki są dla programisty alternatywną metodą przetwarzania dużych obszarów danych. Alternatywną, gdyż jak wynika z przedstawionego programu i obszary takie można przetwarzać stosując metodę indeksowania elementów, z których składają się te obszary. Kompilatory języka C realizując deklarację zmiennej wskaźnikowej nie kontrolują, czy został zadeklarowany obiekt, który ma być wskazywany przez wskaźnik. Stąd tak istotne jest poprawne inicjowanie wartości wskaźników.

Jeżeli umieścimy w programie deklarację:

char tablica [100]

i zadeklarujemy wskaźnik tego samego typu:

char * wsk;

to instrukcja:

wsk = & tablica[0]

inicjuje wskaźnik wsk na zerowy element tablicy, innymi słowy: przypisuje wskaźnikowi wsk adres zerowego (elementy tablic w języku C numerowane są od zera) elementu tablicy. Poprawne w języku C jest również inicjowanie wskaźnika w taki sposób:

wsk = & tablica[i]; gdzie i — zmienna typu int

Operator „&" jest operatorem adresu, powodującym przypisanie zmiennej wskaźnikowej wartości będącej adresem zmiennej stojącej po jego prawej stronie.

W języku C nazwa tablicy jest wskaźnikiem. Kompilator C każde odwołanie do elementu tablicy realizuje posługując się wskaźnikiem do jej początku. Zatem poprawne jest również zainicjowanie wskaźnika instrukcją przypisania:

wsk=tablica;

należy zwrócić uwagę, że w tym przypadku w instrukcji została użyta nazwa tablicy bez nawiasów [..], w których podaje się wartość indeksu elementu tablicy.

Jeżeli w programie zadeklarujemy tablice i zmienną wskaźnikową o takiej samej nazwie, np.:

char tablica[100]
char * tablica;

Funkcja 1

```
/*
 * funkcja przestaw()
 */
przestaw(wyraz,dlugosc)
char *wyraz;
int dlugosc;
$( /* poczatek funkcji */
char znak;
char *koniec;
dlugosc-=1;
koniec=wyraz+dlugosc;
for(; wyraz<koniec; wyraz++, koniec--)$(
 znak=*wyraz;
 *wyraz=*koniec;
 *koniec=znak;
 $) /* for */
return;
$) /* koniec funkcji przestaw() */
```

to ostatnia deklaracja jest jednocześnie poprawnym zainicjowaniem wskaźnika o nazwie tablica (wskazuje on początkowy element tablicy o tej samej nazwie).

Z faktu, że nazwa tablicy jest traktowana w języku C jak wskaźnik wynika, że deklaracja wskaźnika jest równoważna deklaracji tablicy. NP.: jeżeli zadeklarujemy wskaźnik

char * kom;

to jest to równoważne deklaracji tablicy o nieokreślonej liczbie elementów. Oczywistym jest, że po takiej deklaracji wskaźnik ten jeszcze niczego nie wskazuje. Instrukcja: kom=„Naciśnij dowolny klawisz" wprowadzi do tego wskaźnika adres tekstu zawartego pomiędzy znakami „...".

```
program 2
/*
 * program  Przestawianka
 */
main()$(
char znak,wyraz[120];
int i,j,dl;
 do$(
 printf("\nNapisz dowolny wyraz \n"),
 dl=gets(wyraz),
 if(dl<=0) continue;
 printf("\nNapisales \n"),
   for(i=0;i<=dl-1;i++)
        printf(" %c",wyraz[i]),
/* teraz przestaw znaki we wprowadzonym wyrazie */
 przestaw(wyraz,dl);
/* wyswietl przestawiony wyraz   */
 printf("\nA tak to wyglada po przestawieniu\n"),
   for(i=0;i<=dl-1,i++)
       printf(' %c",wyraz[i]),
  $)   /* koniec do  */
 while(czydalej());
$)  /* koniec main  */
```

Funkcja przestaw[] napisana z wykorzystaniem zmiennych wskaźnikowych ma postać:

Funkcja ta wykorzystuje dwa wskaźniki o nazwach wyraz i koniec. Wskaźnik wyraz wskazuje początek tablicy o nazwie wyraz, a wskaźnik koniec — koniec też i tablicy. Zatem wskaźniki te pełnią podobną rolę jak indeksy i, j w programie 1. Instrukcja:

znak=*wyraz;

przypisuje zmiennej o nazwie znak element tablicy wyraz wskazywany przez wskaźnik wyraz; natomiast instrukcja:

wyraz= *koniec;

przypisuje elementowi tablicy wskazywanemu przez wskaźnik wyraz element wskazywany przez wskaźnik koniec, czyli przy pierwszym obiegu pętli do pierwszego elementu tablicy zostanie wprowadzony ostatni, w drugim przebiegu pętli do drugiego elementu zostanie wprowadzony przedostatni, itd.

Z analizy działania tej funkcji wynika, że na wskaźnikach można wykonywać operacje arytmetyczne, jest to zgodne z prawdą. Wskaźniki są adresami. Zatem wykonywanie na nich operacji dodawania i odejmowania jest wykonywaniem operacji na adresach. Wyrażenia w pętli for[] wyraz++ i koniec—— zwiększają i zmniejszają o jeden odpowiednio wskaźniki wyraz i koniec. To zwiększanie o jeden nie należy interpretować jako matematyczne dodanie jedynki. Operacja zwiększenia (zmniejszenia) wskaźnika o jeden oznacza, że po jej wykonaniu wskazuje on następny (poprzedni) element tablicy. Mając to na uwadze nie wolno w programie w języku C dodawać lub odejmować od siebie wskaźników różnych typów np.: char i int. Wynika to z faktu, że zmienne typu int i char są różnej długości.

Możemy teraz napisać nową postać programu 1, używając skonstruowanej funkcji przestaw[].

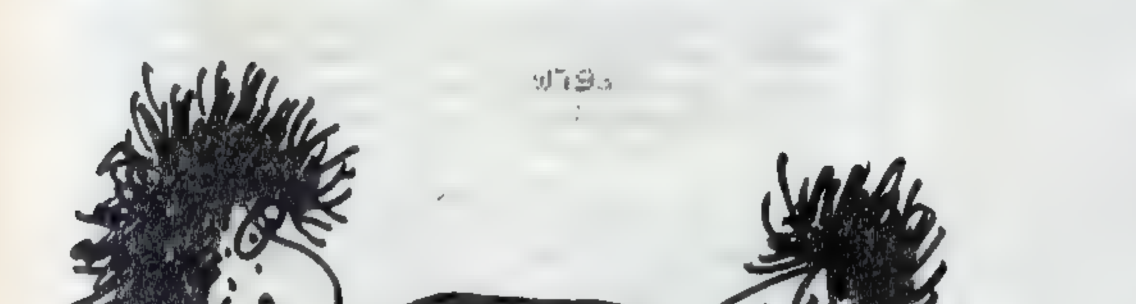
Reasumując, należy stwierdzić, że wskaźniki i funkcje są elementami języka C, które decydują, że język ten posiada wiele walorów w porównaniu z innymi językami wysokopoziomowymi. Funkcje z możliwością ich oddzielnego kompilowania i tworzenia z ich kodów półskompilowanych bibliotek, są szczególnie cennym mechanizmem. Mechanizm ten stanowi niejako możliwość definiowania własnych, specyficznych „instrukcji", które raz zdefiniowane mogą być zawsze używane.

Wskaźniki natomiast, w połączeniu z cechami instrukcji sterujących języka C, umożliwiają tworzenie bardzo zwartego kodu źródłowego programu. Możliwą do uzyskania zwartość ilustruje przedstawiana prawie we wszystkich publikacjach na temat tego języka instrukcja:

while(*s++=*t++);

która kopiuje łańcuch znaków t do łańcucha s. Czy można zaprogramować to krócej w innym języku?

H. KRASUSKI
Z. DYJAK



— Ale nie ma takiego doświadczenia w wagarowaniu, jak my!

Nauka • Technika

W Japonii opracowano nowy typ komputera optycznego (określony mianem neuro-komputera), który jest w stanie nauczyć się jak odróżniać obrazy. Jego budowa oparta jest na układach elektronicznych skonstruowanych w ten sposób, że imitują one strukturę neuronów ludzkiego mózgu. Nowy komputer optyczny, znajdujący się dopiero w początkowym stadium rozwoju, jest w stanie rozpoznać znaki i kształty, a następnie w miarę dokładnie je odwzorować. W początkowym stadium uczenia się komputer optyczny odtwarza znaki czy kształty w sposób niedokładny. Jednakże po przeanalizowaniu różnic pomiędzy znakami 'na wejściu' i 'na wyjściu' komputer koryguje swoją pamięć i ostatecznie odwzorowuje właściwe znaki. W dodatku, gdy znak został całkowicie zapamiętany przez komputer, jest on w stanie wybrać ten sam znak spośród innych, nawet jeśli jest on nieco zniekształcony. Taka funkcja kojarzenia, zdaniem japońskich naukowców, oznacza, że w przyszłości udoskonalone komputery optyczne będą mogły czytać ręcznie pisane znaki.

Ciąg dalszy na str. 22

Awaryjne przerwanie programu ZX

Często podczas działania programu przewidujemy opcję jego zatrzymania (przerwania) przez naciśnięcie określonego klawisza.

W Basic'u uzyskujemy to przez kontrolę klawiatury rozkazem INKEY$. Ta metoda ma jednak swoje wady: przede wszystkim reakcja programu na naciśnięty klawisz istnieje tylko podczas wykonywania instrukcji INKEY$, to znaczy, że aby reakcja na naciśnięty klawisz była szybka, to ta instrukcja musi wykonywać się często. Po drugie: program jest wolniejszy, gdyż każde wykonanie instrukcji INKEY$ pochłania określony odcinek czasu. Stąd jest to metoda w wielu przypadkach nieekonomiczna.

Istnieje jednak inny sposób, wykorzystujący fakt, że co 20 ms komputer kontroluje klawiaturę wykorzystując przerwania w trybie pierwszym (IM 1).

Prezentowany program pracuje w trybie drugim (IM 2). Co 20 ms następuje nie tylko kontrola klawiatury, ale również test, czy naciśnięto zadeklarowany klawisz. Jeśli tak, to zostaje wymuszony skok do określonej przez użytkownika linii programu. Skok do zadanej linii programu uzyskuje przez wstawienie do zmiennej systemowej NEWPPC numeru tej linii.

Procedura nie działa podczas wykonywania rozkazów GO TO, GO SUB, RETURN ponieważ instrukcje te zmieniają wartość NEWPPC. Nie jest to jednak poważne ograniczenie, gdyż zmienna systemowa LAST_K pamięta ostatnio wciśnięty klawisz i zaraz po wykonaniu wyżej wymienionych rozkazów nastąpi przerwanie.

Współpraca z programem:

RANDOMIZE USR 65529 — włącza tryb pierwszy przerwań, wyłącza działanie programu

RANDOMIZE USR 65517 — włącza tryb drugi przerwań, uruchamia program

komórki o adresach 65526 i 65527 zawierają numer linii w Basic'u, do której nastąpi skok podczas wymuszenia przerwania

komórka o adresie 65528 zawiera kod znaku klawisza wymuszającego przerwanie

UWAGA: jeżeli nastąpi wymuszenie przerwania do nieistniejącej linii, wtedy nastąpi przerwanie wykonywania programu i ukaże się komunikat systemu.

Krzysztof POŹNIAK

```
  10 REM ***********************
  11 REM
  12 REM    Krzysztof Poźniak
  13 REM
  14 REM   awaryjne przerwanie
  15 REM        programu
  16 REM
  17 REM                 ©1988
  18 REM
  19 REM ***********************
 100 CLEAR 65488
 110 LET s=0
 111 FOR k=65489 TO 65535
 112 READ l
 113 LET s=s+l
 114 POKE k,l
 115 NEXT k
 116 IF s<>5279 THEN BEEP 1,10:
PRINT "zle dane!!": STOP
 120 BEEP 1,10
 125 LET nrlinii=9000
 126 POKE 65526,nrlinii-256*INT
(nrlinii/256): POKE 65527,INT (n
rlinii/256)
 130 PRINT : PRINT "nacisnij ten
 klawisz, ktorym    zatrzymasz p
rogram"
 135 PAUSE 0
 140 LET x$=INKEY$
 145 POKE 65528,CODE x$
 150 PRINT "teraz program przerw
iesz klawiszem: ";x$;" i uruchom
isz od linii ";nrlinii;" obsluge
 przerwania"
 151 FOR k=1 TO 100: NEXT k: PRI
NT "wcisnij ENTER": PAUSE 0:
 155 RANDOMIZE USR 65517
 200 FOR k=1 TO 10000: PRINT AT
10,10;"licznik :";k: NEXT k: STO
P
1000 DATA 205,56,0,245,229,33,24
8,255,58,8,92,190,32,11,42,246,2
55,34,66,92,62,1,50,68,92,225,24
1,201,62,59,237,71,237,94,201,24
,219,40,35,115,237,86,201,0,0,0,
24,5279
9000 RANDOMIZE USR 65529: PRINT
: PRINT "zostala uruchomiona obs
luga przerwania"
```

```
00012  ; program wykorzystuje przerwa-
00013  ; nia maskowalne, ustawia tryb 2
00014  ; przerwan.
00015  ; program nie jest relokowalny.
00016  ;
00017  rej.i   equ  59 ;wartosc rejestru
00018  ;                 przerwan
00019  ;
00020  ; adres procedury ROM-u
00021  keyb    equ  38h ;kontrola kla-
00022  ;                 wiatury
00023  ;
00024  ; adresy zmiennych systemowych
00025  newppc equ  5c42h
00026  nsppc  equ  5c44h
00027  last_k equ  5c08h
00028  ;
00029          org  65489
00030  ;
00031  ; rozszerzenie przerwania
00032  ; program wykonywany w czasie
00033  ; przerwania
00034  przerw call keyb
00035         push af
00036         push hl
00037         ld   hl,adrznk
00038         ld   a,(last_k)
00039         cp   (hl)
00040         jr   nz,koniec
00041         ld   hl,(nrlin)
00042         ld   (newppc),hl
00043         ld   a,1
00044         ld   (nsppc),a
00045  koniec pop  hl
00046         pop  af
00047         ret
00048  ;
00049  ; inicjalizacja trybu 2
00050  tryb2  ld   a,rej.i
00051         ld   i,a
00052         im   2
00053         ret
00054  adrprw jr   przerw
00055  ;
00056  ; zmienne procedury
00057  nrlin  dw   9000 ;numer linii
00058  adrznk db   's'  ;kod znaku
00059  ;
00060  ; inicjalizacja trybu 1
00061  tryb1  im   1
00062         ret
00063         org  65535
00064         jr   13
00065         end
```

PROGRAM 53

ZX SPECTRUM

Wesołe zdania

Ten program ma charakter typowo rozrywkowy.

Generuje za pomocą generatora liczb pseudolosowych (RND) zdania złożone z przypadkowo wybranych słów. Dowcip i sens tych zdań zależą wyłącznie od programisty, który w liniach DATA umieszcza zestawy wyrazów.

Chciałbym nadmienić, że pierwszą wersję tego programu napisał mój kolega z Klubu Mikrokomputerowego „HOBBYTE" Krzysztof Herman.

Krzysztof POŹNIAK

```
  10 REM ***********************
  11 REM *                     *
  12 REM *  Krzysztof Pozniak  *
  13 REM *                     *
  14 REM *    wesole zdania    *
  15 REM *                     *
  16 REM *               ©1988 *
  17 REM *                     *
  18 REM ***********************
  20 CLS
  30 RESTORE 9000
  40 LET n=5: REM ilosc wyrazow
                  w zdaniu
  50 LET k=10  REM ilosc wyrazow
                  w zestawie
  60 LET i=10: REM maksymalna
                  dlug. wyrazu
  70 DIM t$(n,k,i)
  80 DIM d(n,k)
  90 REM wczytywanie danych
 100 FOR m=1 TO n
 110 FOR l=1 TO k
 120 READ w$
 130 LET dlug=LEN w$
 140 IF dlug>i THEN PRINT "wyraz
 '";w$;"' jest za dlugi": STOP
 150 LET t$(m,l)=w$
 160 LET d(m,l)=dlug
 170 NEXT l
 180 NEXT m
 190 REM generowanie zdan
 200 FOR m=1 TO n
 210 LET x=1+INT (k*RND)
 220 PRINT t$(m,x,1 TO d(m,x));"
 ";
 230 NEXT m
 240 PRINT
 250 GO TO 200
9000 REM zbior wyrazow
9010 REM
9020 REM wyraz pierwszy
9030 DATA "zwariowany","zakochan
y","krzywy","kulawy","wesoly","z
ly","gluchy","tchorzliwy","glodn
y","smieszny"
9040 REM wyraz drugi
9050 DATA "tancerz","piosenkarz"
,"zajac","wilkolak","jasiu","paj
ac","naukowiec","szef","bandyta"
,"wariat"
9060 REM wyraz trzeci
9070 DATA "ugryzl","zamalowal","
podgladal","kochal","ugotowal","
zjadl","obrabowal","pocalowal","
wyrzucil","wynalazl"
9080 REM wyraz czwarty
9090 DATA "okropnego","wielkiego
","pieknego","garbatego","stareg
o","zielonego","odwaznego","zgni
lego","madrego","smutnego"
9100 REM wyraz piaty
9110 DATA "amanta","pokrake","do
ktora","muzyka","smoka","pijaka"
,"poete","upiora","ciecia","mura
rza"
```

Ochrona danych podczas wykonywania instrukcji CLEAR

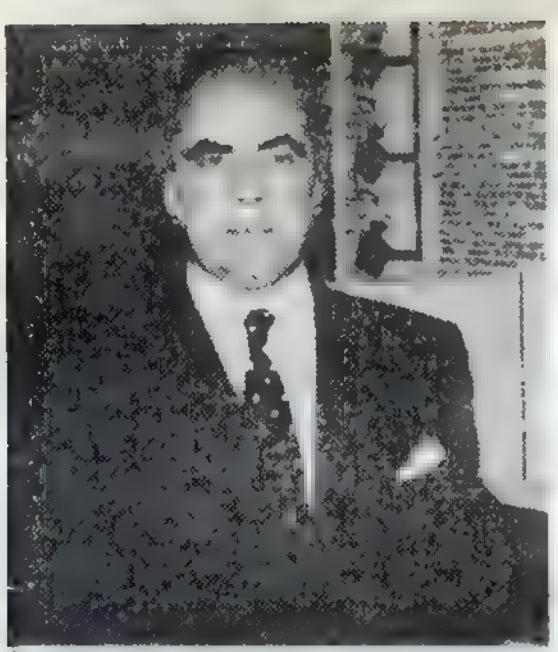
Prezentowany program, a dokładnie linia 70, pozwala na ochronę danych podczas rezerwowania miejsca instrukcją CLEAR. Mechanizm ochrony polega na przypisaniu obszarowi programu obszaru zmiennych na czas działania instrukcji CLEAR w linii 70. Po wykonaniu tej instrukcji odtwarza się stare granice obszarów.

```
   1 REM ***********************
   2 REM
   3 REM    Krzysztof Pozniak
   4 REM
   5 REM  Ochrona danych podczas
   6 REM    wykonywania CLEAR
   7 REM
   8 REM                   ©1988
   9 REM ***********************

  10 CLEAR 50000
  20 LET a=1
  30 PRINT "zmienna a=";a
  40 PRINT "CLEAR=";PEEK 23730+2
56*PEEK 23731
  50 PRINT #0;"wcisnij dowolny k
lawisz"
  60 PAUSE 0
  70 POKE 23728,PEEK 23627: POKE
 23729,PEEK 23628: LET x=PEEK 23
641+256*PEEK 23642-1: POKE 23627
,x-256*INT (x/256): POKE 23628,I
NT (x/256): CLEAR 40000: POKE 23
627,PEEK 23728: POKE 23628,PEEK
23729
  80 PRINT "zmienna a=",a
  90 PRINT "CLEAR=";PEEK 23730+2
56*PEEK 23731
```

Uwaga: gdyby argument instrukcji CLEAR spowodował błąd systemu, wtedy samemu należy ustawić granicę obszaru zmiennych pobierając tę wartość z komórek pamięci o adresach 23728 i 23729. Komórki te nie są używane przez system.

Krzysztof POŹNIAK



Dyrektor Hans Hofer

Żywioł i koniunktura

Rozmowa z dyrektorem firmy PROSYSTEM COMPUTER TECHNIC — HANSEM HOFEREM

● Jaki jest pana przepis na udany interes w informatycznej branży?

— Nie ma chyba uniwersalnej zasady. Prosystem jest firmą handlową, stąd dla nas najważniejsza jest jakość sprzedawanego towaru i sprawność obsługi klienta. Dodać należałoby jeszcze skuteczną reklamę, dobry serwis, no i oczywiście znajomość rynku.

● I języka.

— Nie ma w tym nic dziwnego. W Polsce bywam 25 razy w roku. Oczywiście język nie jest jak widać najsilniejszą moją stroną. Chociaż bardzo pomaga w pracy.

● Ile już lat pańska firma poznaje Polskę?

— Dziesięć.

● Będzie zatem jubileusz.

— W maju.

● Czy udany?

— Nasze kontakty handlowe są z roku na rok coraz większe. Sprowadzaliśmy i sprowadzamy do Polski odzież z najwyższym znakiem jakości. O trzech lat zajmujemy się również sprzętem komputerowym. Swoim zasięgiem obejmujemy też NRD, Czechosłowację i Związek Radziecki.

● Który z tych krajów jest najpoważniejszym klientem?

— Bez wątpienia Związek Radziecki. To bardzo wymagający, ale i bardzo chłonny rynek. W Polsce mamy znacznie większą konkurencję, powiedziałbym nawet, że informatykę opanował żywioł.

● I dobra koniunktura.

— W tej chwili największe interesy robi się sprzedając sprzęt, gdy tymczasem na całym świecie niezwykle drogie jest oprogramowanie. W Polsce najdroższe nawet programy kopiuje się bezkarnie.

● Dlatego w waszej ofercie znajdują się jedynie komputery i urządzenia peryferyjne?

— Sprzedawanie oprogramowania jest dziś jeszcze nieopłacalne. Nie pomogą zabezpieczenia przed kopiowaniem, jeśli nie ma odpowiednich przepisów chroniących prawa autorskie twórców programów. Teraz można sprzedać kilka egzemplarzy, a już za parę dni pojawią się pierwsze kopie, którymi będą handlować już inni — a my jesteśmy bezsilni. Dlatego wolimy sprzęt.

● W sklepach Centralnej Składnicy Harcerskiej prowadzącej sprzedaż waszych towarów często pojawiają się nowe modele komputerów, ale nie wszystkie znikają?

— Początkowo sądziliśmy, że minikomputery MSX mogą być przeznaczone do szkół. Dyskusja o tym, jaki sprzęt ma trafić do uczniów trwała długo. Teraz nasza oferta, to głównie urządzenia profesjonalne.

● Za złotówki?

— Ilość sprzętu sprzedawanego za złotówki jest ograniczona. Natomiast wszystko możemy dostarczyć za dewizy. W tej chwili w sklepach składnicy można kupić na przykład mikrokomputery BONDWELL BW 38-2, BONDWELL BW-8s, SVI-838 X'PRESS 16, a także stacje dysków, dyski twarde, monitory, drukarki, dyskietki.

● To zestaw standardowy.

— To prawda. Sprzęt o podobnym standardzie sprzedaje prawie każda z wielu działających spółek. Ale sądzę, że nasza pełna oferta jest znacznie atrakcyjniejsza.

● Co proponujecie?

— Przede wszystkim aplikacje profesjonalne, na przykład systemy komputerowego wspomagania projektowania inżynierskiego, tworzone na 32-bitowych komputerach. W skład oprogramowania wchodzą pakiety grafiki trójwymiarowej, programy służące do projektowania obwodów drukowanych i przygotowania klisz do produkcji płytek drukowych, a także systemy sterowania obrabiarkami numerycznymi.

● Dziedzina ich zastosowań jest bardzo wąska.

— Ale właśnie tam coraz częściej wykorzystuje się sprzęt komputerowy. Mamy też coś dla dziennikarzy, a raczej dla wydawców — systemy komputerowej edycji składu i drukowania tekstu i grafiki. Dziennikarzom może się też przydać telefax i telefon z możliwością transmisji obrazu, a urządzenie to uzyskało już atest i jest dopuszczone do zastosowania w krajowej sieci telefonicznej.

● Zainteresowanie sprzętem jest jednak proporcjonalne do możliwości jego zastosowania. Co zatem sprzedaje się najlepiej?

— Wszystko, co utrzymane jest w standardzie IBM. Sądzę, że już niedługo wzrośnie zainteresowanie sieciami i szybkimi, 32-bitowymi komputerami wyposażonymi w zegar o częstotliwości 24 MHz. Moi klienci coraz częściej bardzo konkretnie formułują swoje wymagania, są dobrymi partnerami.

● I tylko takich właśnie panu życzę. Dziękuję za rozmowę.

Wiesław CETERA



Telefax (Foto: S. Bibulski)

INFOSYS

Jeśliby mierzyć skalę informatyzacji naszego kraju częstotliwością i rozmachem imprez handlowych to bez wątpienia znaleźlibyśmy się w europejskiej czołówce, a już z pewnością na pierwszym miejscu wśród krajów RWPG. Po „Komputerze 88", kameralnych wystawach w warszawskim „Forum" handlową galę organizował Poznań. Od 25 do 29 kwietnia na terenie Międzynarodowych Targów Poznańskich trwały Międzynarodowe Targii Elektroniki, Telekomunikacji i Techniki Komputerowej „Infosystem '88". Jest to impreza, która w ubiegłym roku gościła we wrocławskiej Hali Ludowej. Zorganizowano ją z inicjatywy „Przeglądu Technicznego" w 120--lecie tej redakcji, co bez wątpienia dobrze podkreślało charakter gazety — organu programowo związanego z postępem. „Infosystem '87" rozbudził wiele namiętności, a patronat, jaki nad nim roztoczył Urząd Postępu Technicznego i Wdrożeń wróżył mu różową raczej przyszłość.

Przez rok impreza dojrzała i znacznie rozszerzyła swą tematykę. Hala Ludowa zrobiła się zbyt ciasna, a i wyspecjalizowana instytucja, jaką są Międzynarodowe Targi Poznańskie potrafi informatyczną fiestę znacznie lepiej przygotować — co już na wstępie podkreśla słuszność tezy, iż żadna działalność nie znosi amatorstwa, o czym warto pamiętać w kontekście różnorodnej działalności informatycznej.

Na „Infosystem '88" przywieziono sprzęt, urządzenia i systemy komputerowe przeznaczone dla wielu dziedzin. Na stoiskach znalazły się także urządzenia teletransmisyjne, radiokomunikacyjne i radiolokacyjne, urządzenia technologiczne dla przemysłu elektronicznego, podzespoły i oprogramowanie. Oferowano też usługi w zakresie przetwarzania danych, a także kompletacji i montażu sprzętu, literaturę fachową

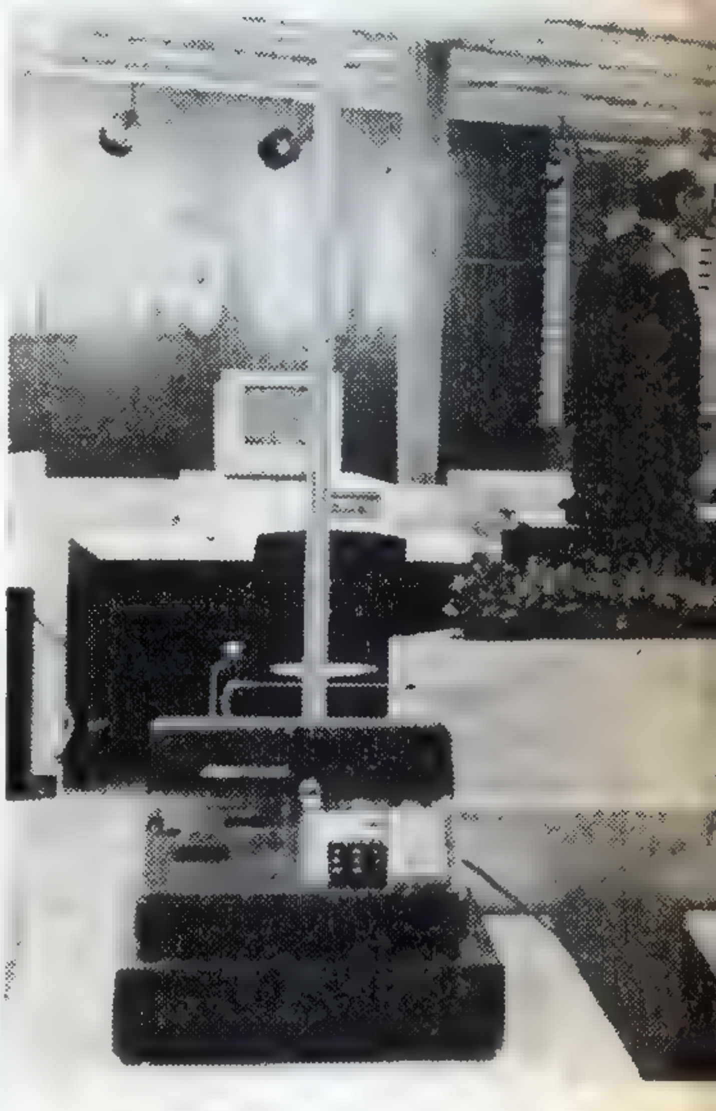
Z zapowiedzi wynikało, że będzie to impreza atrakcyjna, na najwyższym światowym poziomie. Ponad dwieście anonsowanych firm zwiastować miało lepsze jutro dla rozwoju informatyki. Tymczasem nie było ICL — firmy znanej i zadomowionej w Polsce, z udziału w imprezie zrezygnowała firma Olivetti.

Nie sądzę jednak, aby te fakty odbiły się na poziomie naszego rynku informatycznego, nic bowiem nie jest w stanie powstrzymać wzbierającej ciągle fali informatyzacji, która jest przecież synonimem postępu dla jednych i... dobrych interesów dla drugich

Sprzęt czyli pieniądze

Samopoczucie informatyków miał poprawić rządowy program informatyki, obejmujący siedem potężnych tematów i 17 zamówień rządowych. W 1987 roku na ten cel przeznaczono 50 mld złotych, co jest sumą raczej niewielką, chociażby w porównaniu z wydatkami na alkohol. „Statystycznej" flaszki nie można przeliczyć na komputer, choć kursy obu tych rzeczy trzymają się w Pewexie trzeźwo. Rozsądek stracić można dopiero wtedy, kiedy uspołecznieni i prywatni pośrednicy przeliczają ceny światowe na rodzimą walutę. Polski przemysł elektroniczny jest niedoinwestowany, jego udział w globalnej produkcji wynosi ok. 2,2 procenta (wśród krajów socjalistycznych wyprzedzamy jedynie Albanię i Rumunię) Nasza szansa na przodownictwo w tej dziedzinie (jeszcze nie tak dawno w RWPG wyprzedzał nas jedynie Związek Radziecki) przepadła, a możliwości odrobienia zaległości, realnie oceniając, nie są największe Tymczasem potrzeby rosną: do końca tego stulecia w oświacie będzie potrzeba 700 tys. mikrokomputerów, w szkolnictwie wyższym 30 tys., w przemyśle i budownictwie ok. 190 tys., zaś w transporcie i komunikacji ok. 20 tys. Tego, niemal miliona urządzeń nie wyprodukuje z pewnością „ELWRO", a i tak nie jest to produkcja stwarzająca konkurencję, ani jakościową, ani cenową Tak, tak ceny komputerów na Zachodzie spadają średnio o kilkanaście procent rocznie (oczywiście nie chodzi o nowości) — tymczasem u nas jest odwrotnie.

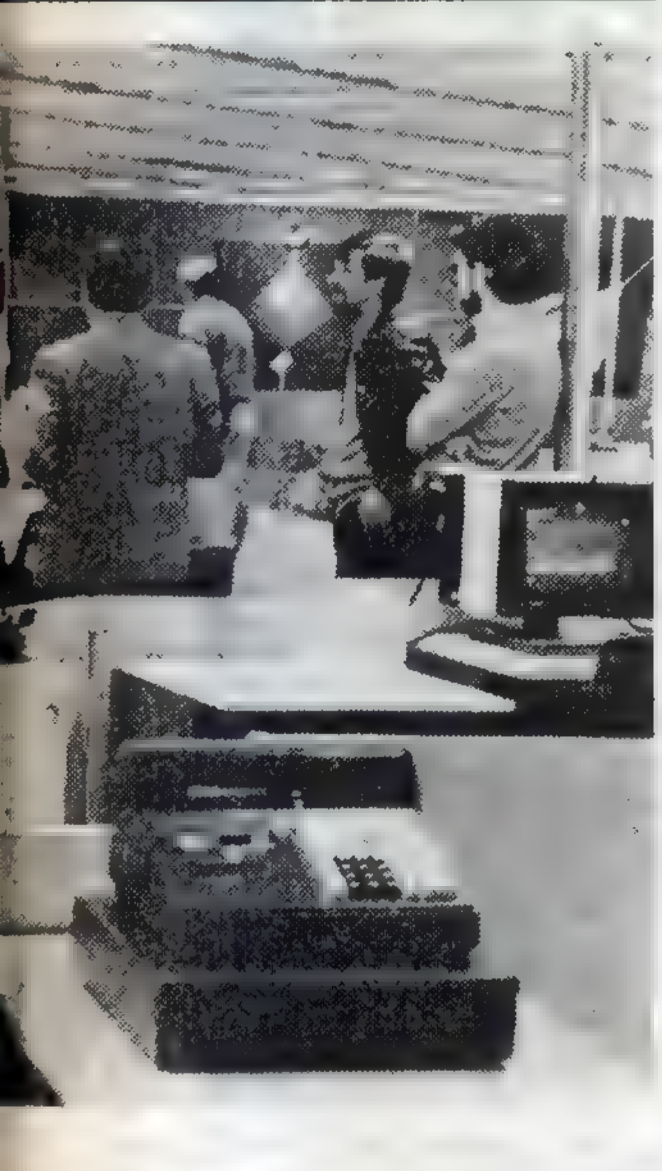
Komputery są towarem poszukiwanym, stąd też dobre interesy robią dziś ci, którzy pośredniczą w ich sprzedaży. Listę takich pośredników można ciągnąć w nieskończoność W tej branży zarabia się pieniądze szybko. Na szczęście jest to zdrowy rynek, który sprawnie eliminuje firmy niesolidne, udzielające na przykład fikcyjnych gwarancji na sprzęt, którego nikt nie potrafi naprawić.







STEM '88







Rzetelni dostawcy są znani. Wśród nich niestety najwyższe ceny mają firmy będące jednostkami gospodarki uspołecznionej, a wyjątek stanowi tu Centralna Składnica Harcerska, która sprzedaje sprzęt sprowadzany przez firmę Prosystem — na ogół niezłej jakości (zgodny z IBM, produkowany na Tajwanie) i nawet tani.

Co i za ile sprzedaje się gdzie indziej? Wystarczy wziąć jakąkolwiek gazetę — znaleźć kolumnę ogłoszeń i sprawdzić telefonicznie. Jeśli ktoś myśli poważnie o kupnie komputera, to cierpliwość może być nagrodzona. Komputery są. Trzeba mieć tylko pieniądze. Tyle sprzęt.

a teraz oprogramowanie czyli dżungla

Komputer wart kilka milionów jest dopiero początkiem drogi informatyki. Ta kupa wysokogatunkowego złomu zda się na niewiele, jeśli brak jest odpowiednich programów, a one swą wartością przewyższają często kilkakrotnie wartość komputera. Takie relacje są na całym świecie, na którym należymy niestety do nielicznej grupy państw nieprzestrzegających praw autorskich twórców programów komputerowych. Mamy bez wątpienia zdolnych i bardzo zdolnych programistów. Nasi „twórcy" (choć cudzysłów w tym miejscu jest raczej niepotrzebny) potrafią ominąć każde zabezpieczenie programu, potrafią dokonać niezbędnych w nim przeróbek, a dokumentację szybko przetłumaczyć. W efekcie pojawiają się takie anonse: „do zakupionego u nas sprzętu dołączamy bogate oprogramowanie" i tu pojawia się obszerna lista programów, które przedstawiają często o wiele większą wartość niż składany w pośpiechu tajwański komputer.

Poważni kupcy spoza naszych granic niechętnie umieszczają w swojej ofercie oprogramowanie. Sprzedaż kilku egzemplarzy po prostu się nie opłaca, bowiem już wkrótce pojawią się pierwsze kopie za połowę ceny i taniej.

Oczywiście mamy w dziedzinie tworzenia oprogramowania niewątpliwe sukcesy. Bez wątpienia na czoło wysuwa się tu Studio Kajkowskich, które nawet konie mówiące ludzkim głosem znają, zapraszając z ekranów telewizorów do salonu firmy w Sopocie.

Proponowane ceny są bezkonkurencyjne (!) podobno, jak i jakość proponowanego oprogramowania. W tym przypadku jednak największe bodaj znaczenie ma dobre imię firmy, która jest już w kraju znaną, a ludzie, którzy ją stworzyli mogą z pewnością zapisać plus po stronie sukcesu.

Jakakolwiek forma wymieniania wystawców „Infosystemu 88" jest daleko niepełna — bo przecież w targach udział wzięło ponad 150 firm.

Pozostańmy zatem przy ofertach, w każdej z nich znajdowały się żelazne pozycje: kadry, magazyny, księgowość i finanse. Już pobieżny przegląd skłaniał do refleksji. Wiele z nich napisanych było głównie z myślą o programiście, a nie o użytkowniku. Złośliwi żartowali, że były to propozycje zgodne z zasadą trzech Z: Zrobić, Zainkasować, Zapomnieć. Racja, bo czasami i wstyd na to patrzeć. Niestety pewnie się zdarzy, że ktoś to kupi (ceny powyżej 400 tys. za sztukę).

Wśród ciekawych propozycji programowych odnotować warto edytor tekstowy TAG przygotowany przez Spółdzielnię Pracy Informatyków — „Infoserwis". TAG jest graficznym, pełnoekranowym edytorem tekstów o prostej obsłudze, jasnej komunikacji z użytkownikiem. Kosztuje 180 000. Zainteresowanie wzbudził też podręczny słownik polsko-angielski i angielsko-polski oferowany przez Computech z Krakowa. Producenci żądali za niego 300 000 mając świadomość, że tak naprawdę są to dwa autonomiczne programy umożliwiające dopiero wprowadzenie haseł słownika. Ale i tu cierpliwość może być nagrodzona. Inna firma DORAN, która w tegorocznym Infosystemie udziału nie bierze, proponuje podobny słownik, wypełniony już 15 tysiącami haseł i z dobrym oprogramowaniem, umożliwiającym podobno współpracę z edytorami tekstów za 200 000. I jak tu nie wierzyć w prawdziwą konkurencję, która jestem przekonany sprawi, że rynek informatyczny będzie jednym z najzdrowszych. Czego sobie i Czytelnikom życzę.

W. CETERA

Lilavati (5)

„Kadryl Lucasa"

Danuta KWASIŻUR
Mieczysław SKONIECZNY

Kolejny, piąty już, odcinek opisujący rozrywki matematyczne zgromadzone przez S. Jeleńskiego w „Lilavati" poświęcony będzie prostej grze planszowej nazwanej przez autora „Kadrylem Lucasa". Nazwa gry pochodzi od nazwiska matematyka francuskiego Edwarda Lucasa.

Gra jest przeznaczona dla 1 osoby, ale można także szukać rozwiązania w zespole kilku graczy.

Prezentowany program przedstawia zasady gry na ekranie informacyjnym, jak pokazano na poniższym rysunku.

```
NA KWADRACIE 49-POLOWYM ROZSTAWIONE SA
KRAZKI CZERWONE I CZARNE.
POLE SRODKOWE JEST WOLNE. KORZYSTAJAC Z
TEGO POLA NALEZY PRZEPROWADZIC CZERWONE
KRAZKI NA MIEJSCE CZARNYCH, A CZARNE NA
MIEJSCE CZERWONYCH.
MOZNA PRZESUWAC KRAZEK ALBO NA SASIADU-
JACE POLE WOLNE ALBO TEZ PRZESKAKIWAC
JEDNYM KRAZKIEM PRZEZ DRUGI SASIEDNI
GDY POZA NIM ZNAJDZIE SIE POLE PUSTE.



NACISNIJ DOWOLNY KLAWISZ
```

Następnie po naciśnięciu dowolnego klawisza ukazuje się plansza gry oraz informacje dodatkowe:

a) w górnym wierszu:

— dotychczasowy rekord (ilość punktów odpowiada ilości posunięć);

— aktualny wynik (ilość posunięć w bieżącej grze);

b) w wyszczególnionym oknie:

— kolumna i rząd pionka, który chcemy przesuwać;

— kolumna i rząd pola, na które chcemy przesunąć pionka.

Przed wykonaniem każdego ruchu należy wprowadzić te parametry wybierając numerycznie: kolumnę i rząd (wg oznaczeń planszy). Taką sytuację przedstawia poniższy rysunek:

(oczywiście pionki mają dwa kolory: czarny i czerwony).

Przed wykonaniem każdego ruchu programu bada, czy jest on dopuszczalny (zgodny z zasadami gry). W wypadku wprowadzenia błędnych parametrów mikrokomputer wyświetla jeden z komunikatów:

POLE ZAJĘTE;

PRZESUNIĘCIE NIEMOŻLIWE;

i powtórnie wraca w stan oczekiwania na parametry.

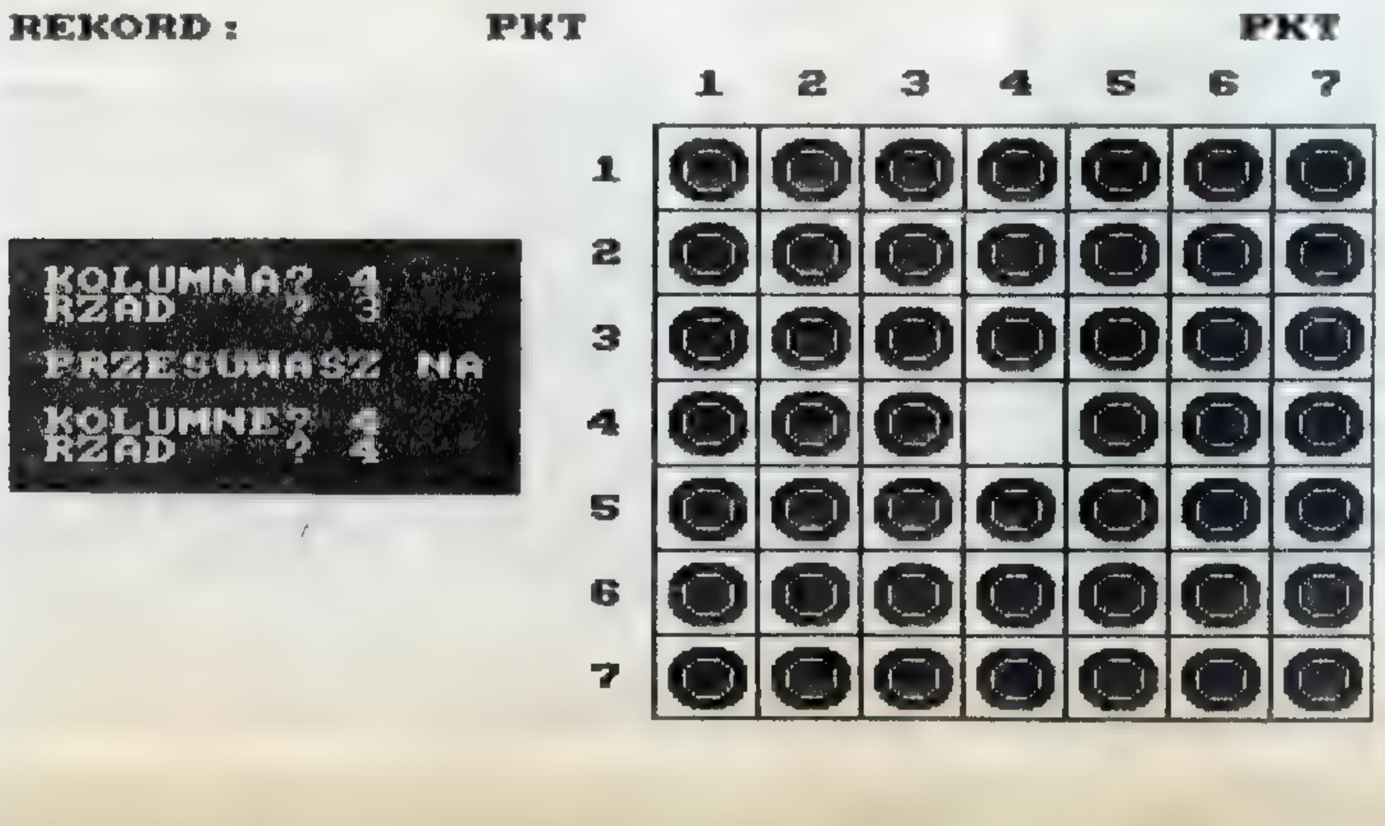
W programie opracowanym w języku BASIC mikrokomputera AMSTRAD CPC 6128 wyszczególniono 2 tablice:

TABL(7,7) — zawiera wartości 0, 1 lub 2 dotyczące aktualnego rozstawienia pionków (0 — pole puste, 1 — pionek czarny, 2 — pionek czerwony);

TABLP(7.7) — zawiera rozstawienie końcowe pionków i służy do badania, czy gra zakończyła się sukcesem.

Poszczególne fragmenty programu realizują następujące czynności:

```
10 MODE 1:INK 0,26:INK 1,0:INK 2,6:INK 3,14:PAPER 3:BORDER 14:CLS:PEN
1
20 PRINT "NA KWADRACIE 49-POLOWYM ROZSTAWIONE SA  KRAZKI CZERWONE I CZ
ARNE.            POLE SRODKOWE JEST WOLNE. KORZYSTAJAC Z TE
GO POLA NALEZY PRZEPROWADZIC CZERWONE KRAZKI NA MIEJSCE CZARNYCH, A CZ
ARNE NA MIEJSCE CZERWONYCH."
30 PRINT "MOZNA PRZESUWAC KRAZEK ALBO NA SASIADU- JACE POLE WOLNE ALBO
 TEZ PRZESKAKIWAC   JEDNYM KRAZKIEM PRZEZ DRUGI SASIEDNI     GD
Y POZA NIM ZNAJDZIE SIE POLE PUSTE."
40 PEN 2:LOCATE 1,20:PRINT "NACISNIJ DOWOLNY KLAWISZ"
50 IF INKEY$="" THEN GOTO 50
60 PEN 1:MODE 1:CLS:WINDOW £1,20,40,5,25:PAPER £1,0:CLS £1
70 rek=0:gl=0
80 PAPER 3:PEN 2:LOCATE 1,1:PRINT "REKORD:        PKT "
90 GRAPHICS PEN 1:PLOT 302,336:DRAWR 336,0:DRAWR 0,-336:DRAWR -336,0:D
RAWR 0,336
100 FOR k=0 TO 288 STEP 48
110 PLOT 350+k,334:DRAWR 0,-334
120 PLOT 302,288-k:DRAWR 334,0
130 NEXT k
140 SYMBOL AFTER 125
150 SYMBOL 125,0,0,0,0,3,7,15,15
160 SYMBOL 126,0,0,0,255,255,255,129,126
170 SYMBOL 127,0,0,0,0,192,224,240,240
180 SYMBOL 128,30,29,29,29,29,29,29,30
190 SYMBOL 129,120,184,184,184,184,184,184,120
200 SYMBOL 130,15,15,7,3
210 SYMBOL 131,126,129,255,255,255
220 SYMBOL 132,240,240,224,192
230 b$(1)=CHR$(125)+CHR$(126)+CHR$(127)
240 b$(2)=CHR$(128)+CHR$(143)+CHR$(129)
250 b$(3)=CHR$(130)+CHR$(131)+CHR$(132)
260 DIM tabl(7,7)
270 DIM tablp(7,7)
280 FOR k=1 TO 7
290 FOR i=1 TO 7
```



```
300 READ tablp(k,i)
310 NEXT i
320 NEXT k
330 DATA 1,1,1.1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,0,1,1,1,2,2,2
,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
340 RESTORE 390:FOR k=1 TO 7
350 FOR i=1 TO 7
360 READ tabl(k,i)
370 NEXT i
380 NEXT k
390 DATA 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,0,2,2,2,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
400 pkt=0:FOR k=1 TO 7
410 FOR i=1 TO 7
420 b=tabl(k,i):GOSUB 910
430 NEXT i
440 NEXT k
450 PAPER 3:PEN 1:FOR j=1 TO 7
460 LOCATE 18+j*3,3:PRINT USING "£";j;
470 LOCATE 18,3+j*3:PRINT USING "£";j;
480 NEXT j
490 PAPER 3:PEN 1:LOCATE 31,1:PRINT "      PKT "
500 WINDOW £2,1,15,9,17:PAPER £2,1:CLS £2:PEN £2,0
510 PKT=0
520 CLS £2:PEN 0:PAPER 1:LOCATE 1,10:INPUT " KOLUMNA";K1
530 IF k1>7 OR k1<1 THEN GOTO 520
540 INPUT " RZAD    ";RZ1
550 IF rz1>7 OR rz1<1 THEN GOTO 540
560 PRINT:PRINT " PRZESUWASZ NA"
570 LOCATE 1,15:INPUT " KOLUMNE";K2
580 IF k2>7 OR k2<1 THEN GOTO 570
590 INPUT " RZAD    ";RZ2
600 IF rz2>7 OR rz2<1 THEN GOTO 590
610 IF TABL(RZ2,K2)=0 THEN GOTO 630
620 CLS £2:SOUND 1,358:LOCATE 1,13:PRINT " POLĘ ZAJETE":FOR T=1 TO 800
:NEXT T:GOTO 520
630 IF K1=K2 AND RZ1=RZ2-1 THEN GOTO 720
640 IF K1=K2 AND RZ1=RZ2-2 THEN GOTO 720
650 IF K1=K2 AND RZ1=RZ2+1 THEN GOTO 720
660 IF K1=K2 AND RZ1=RZ2+2 THEN GOTO 720
670 IF K1=K2-1 AND RZ1=RZ2 THEN GOTO 720
680 IF K1=K2-2 AND RZ1=RZ2 THEN GOTO 720
690 IF K1=K2+1 AND RZ1=RZ2 THEN GOTO 720
700 IF K1=K2+2 AND RZ1=RZ2 THEN GOTO 720
710 CLS £2:SOUND 1,358:LOCATE 1,12:PRINT " PRZESUNIECIE":PRINT " NIEMO
ZLIWE":FOR T=1 TO 800:NEXT T:GOTO 520
720 K=RZ1:I=K1:B=0:GOSUB 910:
730 K=RZ2:I=K2:B=TABL(RZ1,K1):GOSUB 910:
740 TABL(RZ2,K2)=TABL(RZ1,K1):TABL(RZ1,K1)=0
750 PKT=PKT+1
760 PAPER 3:PEN 1:LOCATE 31,1:PRINT USING "££££";PKT
770 FOR i=1 TO 7
780 FOR j=1 TO 7
790 IF tabl(i,j)<>tablp(i,j) THEN GOTO 520
800 NEXT j
810 NEXT i
820 CLS £2:PRINT £2:PRINT £2," BRANO !":PRINT £2:PRINT £2,"UDALO SIE "
":PRINT £2
830 FOR k=5 TO 14:SOUND 1,258:SOUND 1,320:BORDER k:FOR t=1 TO 10:NEXT
t:NEXT k
840 IF g1=1 THEN  g1=1:PEN 0:PAPER 3:LOCATE 9,1:PRINT USING "£££££";re
k:GOTO 210
850 IF PKT<REK THEN PRINT £2,"USTALILES NOWY REKORD":REK=PKT:PEN 0:PAP
ER 3:LOCATE 9,1:PRINT USING "£££££";rek
860 :INPUT £2,"GRASZ DALEJ ? (T/N)";K$
870 IF K$="T" OR K$="t" THEN GOTO 340
880 IF K$="N" OR K$="n" THEN END
890 GOTO 210
900 IF INKEY$="" THEN GOTO 900
910 PAPER 0:PEN b:FOR a=0 TO 2:LOCATE 17+i*3,2+a+k*3:PRINT b$(a+1);:NE
XT a
920 GRAPHICS PEN 1:PLOT 302+i*48,382-k*48:DRAWR 0,-46:DRAWR -48,0:RETU
RN
```

10—50 — wyświetlenie ekranu informacyjnego;
60—510 — rysowanie planszy, deklaracja tablic i nadanie wartości początkowych zmiennych;
520—600 — wprowadzanie przez grającego parametrów do przesunięcia pionka;
610—710 — badanie, czy przesunięcie jest dopuszczalne (zgodne z zasadami gry);
720—760 — realizacja przesunięcia pionka na planszy, wykorzystująca podprogram 910—920 — rysowania pionka na wybranym polu.

Wytrwałym Czytelnikom, którzy wprowadzą ten program do pamięci mikrokomputera życzymy ustalenia rekordu trudnego do pobicia. Autor „Lilavati" podaje, że minimalna ilość posunięć, po których gra kończy się sukcesem wynosi 120 pkt.

Po zakończeniu jednej gry (k ~nunikat: BRAWO! UDAŁO SIĘ! i dodatkowy efekt dźwiękowy) program pyta czy „GRASZ DALEJ?". Można wtedy przystąpić do nowej gry naciskając klawisz T.

Prosty program graficzny dla mikrokomputera AMSTRAD CPC 6128

Mikrokomputery serii AMSTRAD 6128 posiadają stosunkowo bogate oprogramowanie narzędziowe. Jedną z zalet tego komputera są spore możliwości graficzne, które pozwalają tworzyć ciekawe i nieraz skomplikowane obrazy. Pełne możliwości graficzne CPC 6128 udostępniają użytkownikowi także pakiety programów narzędziowych jak np. SPRITE, czy AUTOCAD. Dla początkującego użytkownika są one często skomplikowane, trudne w użyciu i mało komunikatywne (konwersacja prowadzona jest w języku angielskim). Wszystko to sprawia, że informatyk-amator nie wykorzystuje tych pakietów, a chcąc tworzyć ekrany graficzne sięga do możliwości języka BASIC. Tym samym traci sporo czasu, wykorzystując instrukcje SYMBOL, DRAW, PLOT osiąga wreszcie zamierzony cel. Nabywa przy tym przekonania, że aby osiągnąć ciekawe efekty graficzne trzeba się sporo namęczyć.

Dlatego też początkującym użytkownikom proponujemy proste narzędzie, które pozwoli im szybko uzyskiwać na ekranie różne kompozycje. Przy pewnej wprawie w obsłudze tego programu można pokusić się o tworzenie nawet bardziej skomplikowanych obrazów.

```
10 MODE 1:INK 0,0:INK 1,26:PAPER 0:CLS:PEN 1:INPUT"tryb (0,1,2)";tryb
15 INPUT "ramka (0-26)";ram
17 BORDER ram
20 p1=0
30 MODE tryb
40 DIM TRYBY(3,2)
50 FOR I=1 TO 3:READ TRYBY(I,1),TRYBY(I,2):NEXT i
60 DATA 20,16,40,4,80,2
70 WINDOW £1,1,TRYBY(TRYB+1,1),1,1
80 N1=0
90 DIM kol(tryby(tryb+1,2))
100 FOR v=1 TO tryby(tryb+1,2):kol(v)=-1:NEXT v
120 CLS £1:INPUT"CZYTAC ZBIÓR(t/n)";k$
130 IF k$="n" OR k$="N" THEN GOTO 400
140 IF k$="T" OR k$="t" THEN GOTO 160
150 GOTO 120
160 CLS £1:INPUT £1,"TLO";TLO
170 KOL(1)=TLO:N1=N1+1
180 FOR I=2 TO TRYBY(TRYB+1,2)
190 CLS £1:PRINT £1,"KOLOR";I-1;:INPUT £1,K
200 IF K=-1 THEN GOTO 240
210 kol(i)=k:N1=N1+1
220 NEXT I
230 GOTO 270
240 FOR J=I TO TRYBY(TRYB+1,2)
250 KOL(J)=-1
260 NEXT J
```

```
270 INPUT £1,"PODAJ NAZWE";ZB$
280 CLS£1:LOAD zb$
290 PEN 1:CLS £1:PRINT £1,"P-POMOC"
295 REM 'wybor opcji'
300 A$=INKEY$
310 IF A$="" THEN GOTO 300
320 IF A$="R" OR A$="r" THEN GOTO 440
330 IF A$="K" OR A$="k" THEN GOTO 430
340 IF A$="M" OR A$="m" THEN GOTO 870
350 IF A$="F" OR A$="f" THEN GOTO 850
360 IF A$="z" OR A$="Z" THEN GOTO 820
370 IF A$="P" OR A$="p" THEN GOTO 1120
380 IF A$="d" OR A$="D" THEN GOTO 1370
390 GOTO 300
400 CLS £1:INPUT £1,"TLO";TLO
410 INK 0,TLO:PAPER 0:CLS
420 KOL(1)=TLO
430 GOSUB 1040
440 GOSUB 480
450 PLOT x,y:GOSUB 660
460 GOTO 290
470 END
475 REM 'sterowanie kursorem'
480 |SCREENCOPY,2,1
490 x=0:y=0:p1=1:PLOT x,y
500 A$=INKEY$
510 IF A$="" THEN GOTO 500
520 IF INKEY(6)=0 THEN RETURN
530 IF ASC(a$)=240 THEN GOTO 580
540 IF ASC(a$)=241 THEN GOTO 600
550 IF ASC(a$)=242 THEN GOTO 620
560 IF ASC(a$)=243 THEN GOTO 640
570 GOTO 500
580 IF y=398 THEN GOTO 500
590 |SCREENCOPY,1,2:y=y+2:PLOT x,y:GOTO 500
600 IF y=0 THEN GOTO 500
610 |SCREENCOPY,1,2:y=y-2:PLOT x,y:GOTO 500
620 IF x=0 THEN GOTO 500
630 |SCREENCOPY,1,2:x=x-2:PLOT x,y:GOTO 500
640 IF x=638 THEN GOTO 500
650 |SCREENCOPY,1,2:x=x+2:PLOT x,y:GOTO 500
655 REM 'podprogram rysowania plotami'
660 A$=INKEY$
670 IF A$="" THEN GOTO 660
680 IF INKEY(6)=0 THEN RETURN
690 IF ASC(a$)=240 THEN GOTO 740
700 IF ASC(a$)=241 THEN GOTO 760
710 IF ASC(a$)=242 THEN GOTO 780
720 IF ASC(a$)=243 THEN GOTO 800
730 GOTO 660
740 IF y=398 THEN GOTO 660
750 y=y+2:PLOT x,y:GOTO 660
760 IF y=0 THEN GOTO 660
770 y=y-2:PLOT x,y:GOTO 660
780 IF x=0 THEN GOTO 660
790 x=x-2:PLOT x,y:GOTO 660

800 IF x=638 THEN GOTO 660
810 x=x+2:PLOT x,y:GOTO 660
815 REM 'zapis zbioru na dysku'
820 CLS £1:INPUT £1,"PODAJ NAZWE";ZB$
830 CLS£1:SAVE ZB$,B,&C000,&4000
840 END
845 REM 'wypelnianie kolorem'
850 GOSUB 1040
860 GOSUB 480:MOVE X+2,Y:FILL N:y=0:x=0:GOTO 290
865 REM 'mazanie'
870 GRAPHICS PEN 1:GOSUB 480
880 A$=INKEY$
890 IF A$="" THEN GOTO 880
900 IF INKEY(6)=0 THEN GRAPHICS PEN 0:PLOT X,Y:GRAPHICS PEN 1:GOTO 290
910 IF ASC(a$)=240 THEN GOTO 960
920 IF ASC(a$)=241 THEN GOTO 980
930 IF ASC(a$)=242 THEN GOTO 1000
940 IF ASC(a$)=243 THEN GOTO 1020
950 GOTO 880
960 IF y=398 THEN GOTO 880
970 GRAPHICS PEN 0:PLOT X,Y:y=y+2:GRAPHICS PEN 1:PLOT x,y:GOTO 880
980 IF y=0 THEN GOTO 880
990 GRAPHICS PEN 0:PLOT X,Y:y=y-2:GRAPHICS PEN 1:PLOT x,y:GOTO 880
```



Ponieważ podczas pracy, program będzie przechowywał w pamięci kilka obrazów ekranu, dlatego też niezbędne jest wprowadzenie przed jego uruchomieniem programu BANKMAN.

Po inicjacji pracy program żąda odpowiedzi na pytanie „Tryb?" — Należy wprowadzić liczbę z przedziału 0—2. W wybranym trybie będzie można maksymalnie wykorzystać:
0—16 kolorów;
1—4 kolory;
2—2 kolory.

Następne pytanie brzmi: „Czytać zbiór (t/n)?". Jeżeli jest to kolejny etap pracy mający na celu dokończenie obrazu zapisanego wcześniej na dysku, to należy podać nazwę tego obrazu, jaką nadał użytkownik przy poprzednim seansie.

W dalszej kolejności podajemy kolory:
— tła;
— ramki;
— poszczególnych atramentów.

Wprowadzamy liczby całkowite z przedziału 0—26 będące kodami pełnej palety barw tego mikrokomputera.
Po podaniu ww. parametrów wstępnych, możemy korzystać z zestawu następujących opcji programu:

R — rysowanie obrazu,
K — zmiana koloru atramentu,
M — mazanie nakreślonych linii,
F — wypełnianie obiektów kolorami,
Z — zapis obrazu na dysk,
P — pomoc (wyświetlenie pełnego zestawu opcji),
D — drukowanie zawartości ekranu na drukarce.



```
1000 IF x=0 THEN GOTO 880
1010 GRAPHICS PEN 0:PLOT X,Y:x=x-2:GRAPHICS PEN 1:PLOT x,y:GOTO 880
1020 IF x=638 THEN GOTO 880
1030 GRAPHICS PEN 0:PLOT X,Y:x=x+2:GRAPHICS PEN 1:PLOT x,y:GOTO 880
1035 REM 'wybor koloru'
1040 CLS £1:INPUT £1,"KOLOR";K
1050 FOR I=2 TO tryby(tryb+1,2)
1060 IF KOL(I)=K THEN GRAPHICS PEN I-1:N=I-1:CLS £1:GOTO 1110
1070 NEXT I
1080 N1=N1+1
1090 IF N1>TRYBY(TRYB+1,2)-1 THEN PRINT £1,"MASZ JUZ ";:PRINT £1,USING "££";tryb
y(tryb+1,2);:PRINT £1," KOL. !":FOR T=1 TO 1500:NEXT
T:GOTO 1040
1100 kol(n1+1)=k:INK N1,K:N=N1:GRAPHICS PEN N:CLS £1
1110 RETURN
1115 REM 'pomoc'
1120 CLS £1:|SCREENCOPY,2,1
1130 MODE 1:PEN 1:PRINT "K-ZMIANA KOLORU":PRINT "R-RYSOWANIE":PRINT "M-MAZANIE":
PRINT "F-WYPELNIANIE KOLOREM":PRINT "Z-ZAPIS NA DYSK
U"
1280 LOCATE 1,20:PRINT "NACISNIJ DOWOLNY KLAWISZ"
1290 IF INKEY$="" THEN GOTO 1290
1300 MODE TRYB
1310 FOR K=1 TO TRYBY(TRYB+1,2)
1320 IF KOL(K)=-1 THEN GOTO 1350
1330 INK K-1,KOL(K)
1340 NEXT K
1350 WINDOW £1,1,TRYBY(TRYB+1,1),1,1:|SCREENCOPY,1,2
1360 GOTO 290
1365 REM 'drukowanie ekranu na drukarce'
1370 CLS £1
1380 MEMORY &9FFF
1390 FOR i=&A000 TO &A0B0
1400 READ v:POKE i,v:t=t+v
1410 NEXT i
1420 IF t<>20961 THEN PRINT "BLAD"
1430 DATA 205,166,160,62,27,205,157,160
1440 DATA 62,49,205,157,160,205,186,187
1450 DATA 205,231,187,50,180,160,17,0
1460 DATA 0,33,143,1,34,178,160,62
1470 DATA 7,50,177,160,62,10,205,157
1480 DATA 160,62,13,205,157,160,62,27
1490 DATA 205,157,160,62,76,205,157,160
1500 DATA 62,127,205,157,160,62,2,205
1510 DATA 157,160,14,0,58,177,160,71
1520 DATA 229,197,213,205,240,187,209,193
1530 DATA 33,180,160,190,225,55,32,1
1540 DATA 167,203,17,43,43,16,233,58
1550 DATA 177,160,254,7,40,7,175,203
1560 DATA 17,203,17,203,17,121,205,157
1570 DATA 160,19,229,33,127,2,55,237
1580 DATA 82,225,56,5,42,178,160,24
1590 DATA 193,35,124,181,40,32,43,17
1600 DATA 0,0,34,178,160,62,7,189
1610 DATA 32,146,124,180,32,142,62,4
1620 DATA 50,177,160,24,135,205,46,189
1630 DATA 56,251,205,43,189,201,62,27
1640 DATA 205,157,160,62,64,205,157,160,201
1650 CALL &A000
1660 END
```

OPCJA R

Po jej wybraniu należy na pytanie „KOLOR" wprowadzić kod liczbowy koloru, którym będziemy rysować. W sytuacji, kiedy mamy już maksymalną liczbę kolorów w zadeklarowanym trybie pojawi się stosowny komunikat. Następnie na ekranie pojawi się w lewym dolnym rogu punkt w wybranym kolorze. Przesuwanie punktu do miejsca, w którym chcemy rozpocząć rysowanie realizujemy za pomocą klawiszy funkcyjnych (→, ←, ↑, ↓). Po dotarciu punktu na żądane miejsce wciskamy klawisz ENTER. Od tego momentu klawisze funkcyjne powodują przesuwanie punktu, który kreśli linię. Zakończenie rysowania wybranym kolorem lub aktualnego fragmentu obrazu — klawisz ENTER.

OPCJA K

Po wybraniu tej opcji pojawia się pytanie „KOLOR". Po podaniu kodu liczbowego koloru program bada, czy użytkownik nie przekroczył już limitu barw dla aktualnego trybu pracy. Jeśli tak, to pojawia się komunikat: SĄ JUŻ N KOLORY (gdzie N = 2,4 lub 16) i pada ponowne pytanie „KOLOR". Jeżeli użytkownik wprowadzi dopuszczalny kod liczbowy barwy, to program powraca do kreślonego obrazu ze zaktualizowanym kolorem pióra graficznego.

OPCJA M

Przebieg tej opcji jest analogiczny do opcji R w części dotyczącej poruszaniem piksela. Po naprowadzeniu piksela na żądane miejsce i naciśnięciu ENTER klawisze (↑, ↓, →, ←) sterują kierunkiem wymazywania.



OPCJA F

Po naprowadzeniu punktu na wybrane miejsce (jak w opcjach R, M) i wciśnięciu ENTER następuje wypełnianie obiektu, w którym znalazł się punkt kolorem jego brzegu. (Wywoływana jest instrukcja FILL).

OPCJA Z

Po wybraniu tej opcji program pyta: „PODAJ NAZWĘ". Po zaakceptowaniu podanej nazwy klawiszem ENTER następuje zapisanie aktualnego obrazu na dysku w postaci tzw. „screcn'a"

Uwaga: Nazwa obrazu powinna być 8-znakowa, rozpoczynać się od litery i zawierać litery lub cyfry.

OPCJA P

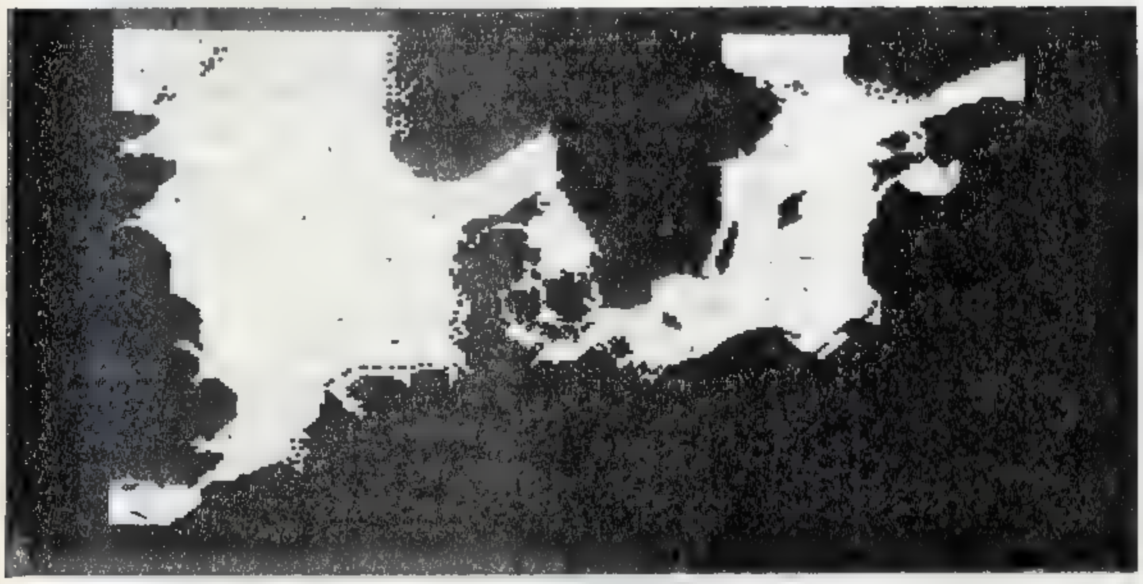
Po naciśnięciu klawisza P niknie komponowany obraz i na ekranie pojawia się pełen zestaw dopuszczalnych opcji.

OPCJA D

Zalecana jest tylko wtedy, gdy zestaw mikrokomputerowy, na którym pracujemy ma podłączoną drukarkę. Po jej wybraniu jest tworzona kopia ekranu na papierze oczywiście w wersji czarno-białej.

Poniższe rysunki powstały za pomocą tego programu i opcji D. Obrazują one różne stopnie trudności, a jednocześnie spore możliwości tego programu graficznego.





Jak odzyskać skasowane zbiory?

Może się zdarzyć, że w czasie pracy na mikrokomputerze skasujemy przypadkowo ważny zbiór na dyskietce lub jeszcze gorzej zostaną zniszczone lub uszkodzone sektory zawierające wykaz zbiorów na dysku, tzw kartoteka dysku (ang. directory) Czy w takiej sytuacji należy uznać za stracone skasowany zbiór lub całą zawartość dysku? Otóż nie, informacje te są do odzyskania, a umozliwia to prezentowany program o nazwie DYSK—KOPIER

Na wstępie trochę teorii. Na każdej dyskietce DOS rezerwuje pewien obszar przeznaczony na zapisanie wspomnianej wyżej kartoteki dysku. Są to sektory o numerach 361—368. Jeżeli kasujemy zbiór, to DOS nie „wymazuje" zawartości sektorów, które kasowany zbiór zajmował, lecz usuwa jego nazwę z kartoteki i aktualizuje zawartość sektora o numerze 360 (tzw. VTOC). A zatem mimo skasowania, zawartość skasowanego zbioru na dyskietce fizycznie nadal istnieje, aż do momentu, gdy zostanie zniszczona przez zapisanie nowych informacji. Problem polega na zidentyfikowaniu sektorów zawierających informacje należące do skasowanego zbioru. Jest to również możliwe, gdy poznamy bliżej strukturę pojedynczego sektora. Jak wiadomo, każdy sektor składa się ze 128 bajtów ponumerowanych od 0 do 127. Pierwsze 125 bajtów (0—124) zawierają informacje użytkownika, natomiast bajty 125, 126 i 127 zarezerwowane są dla DOS—a. Nas interesuje bajt 125, ponieważ jego sześć najstarszych bitów zawiera liczbę (numer) odpowiadającą kolejności wpisu nazwy zbioru, do którego należy dany sektor, w kartotece dyskietki. Po tym wstępie wszystko jest już jasne. Jeżeli mamy na dyskietce dziesięć zbiorów i skasowaliśmy na przykład zbiór piąty, to aby odzyskać jego zawartość należy wybrać z dyskietki wszystkie sektory, które w bajcie 125 mają wpisaną liczbę 4 (numeracja wpisów w kartotece zaczyna się od zera). Po wybraniu należy je zapisać na nową dyskietkę i stracony zbiór jest już odzyskany.

```
1 REM **************************
2 REM PROGRAM KOPIUJACY
3 REM **************************
9 REM MODUL STERUJACY
10 DIM S$(50),TOZ$(50),T$(200),P
US$(128),SE$(23040),SE(180),P$(1
28),INP$(3)
11 PUS$(1)=CHR$(0):PUS$(128)=PUS
$:PUS$(2)=PUS$
12 SE$(1)=CHR$(0):SE$(23040)=SE$
:SE$(2)=SE$
15 POKE 599,0
20 DO=720:VE=0:POKE 1913,80:ND=1
:ZL=1:PISZ=15000:ZROD=18000
25 OPEN #1,4,0,"K:":OPEN #2,4,0,
"E:"
26 GOSUB 29000
30 GOSUB 30000
35 GOSUB 58
40 GOSUB 500
50 GOTO 100
52 X=USR(ADR(S$),ND,360,ADR(P$),
82)
```

Nauka • Technika

Nowy superkomputer skonstruowano w laboratorium SANDIA w amerykańskim stanie Nowy Meksyk. Jest on zdolny do rozwiązywania skomplikowanych zadań naukowych tysiąc razy szybciej od zwykłych komputerów. Posiada 1024 procesory rozwiązujące jednocześnie różne części większego zadania obliczeniowego. Każdy z tych procesorów odgrywa rolę oddzielnego komputera

W czasie prób polecono superkomputerowi przygotowanie dokładnego opisu napięć powstających w rdzeniu metalowym znajdującym się pod obciążeniem. Maszyna dała odpowiedź po tygodniu obliczeń - gdy zwykły komputer potrzebowałby na to około 20 lat.

■ ■ ■

Brytyjskie Ministerstwo Komunikacji zaaprobowało plan stworzenia systemu informacji drogowej przekazywanej za pomocą umieszczonych przy jezdniach emiterów podczerwieni do komputerów znajdujących się w samochodach. Informacje mają dotyczyć przejezdności dróg i nasilenia ruchu. na ich podstawie komputer poinformuje kierowcę o optymalnej trasie przejazdu i najdogodniejszej prędkości

■ ■ ■

```
53 SE=ADR(P$):GS=PEEK(SE+1)+256*
PEEK(SE+2):IF GS=1010 THEN RETUR
N
54 GOSUB 61:POSITION 2,20:? "* *
 * Dysk nie w formacie DOS 2.5 *
 * "
55 FOR G=1 TO 500:NEXT G:GOTO 40
58 X=USR(ADR(TOZ$),ADR(T$),LEN(T
$),0,AD+12):RETURN
60 POKE 752,0:RETURN
61 POKE 752,1:RETURN
100 TRAP 40000:GET #1,A
101 IF A=155 THEN END
102 IF A<66 OR (A>69 AND A<76) O
R (A>77 AND A<>86 AND A<>88) THE
N 100
108 A=A-65
110 GOTO 500+A*500
500 GOSUB 61:? CHR$(125)
510 POSITION 2,1:? "* * * D Y S
K - K O P I E R * * * "
520 POSITION 3,3:? " M E N U
 [RETURN]- WYJSCIE Z MENU"
525 FOR LI=3 TO 11:POSITION LI,4:
? CHR$(13);:NEXT LI

530 POSITION 3,6:? "[";CHR$(66);
"] KOPIUJ SEKTORY [";CHR$(67);"]
 KOPIUJ DYSK"
540 POSITION 3,8:? "[";CHR$(68);
"] DRIVE ZRODL.   [";CHR$(69);"]
 DRIVE KOPI."
550 POSITION 3,10:? "[";CHR$(76)
;"] ODZYSK ZBIORU  [";CHR$(77);"
] TYP DOS-A "
560 POSITION 3,12:? "[";CHR$(86)
;"] VERIFY          [";CHR$(88);"
] FORMATUJ."
700 POKE 559,34:RETURN
999 REM MODUL KOPIOWANIA SEKTORO
W
1000 ? CHR$(125):POSITION 3,2:?
"* * * KOPIOWANIE SEKTOROW * * *
"
1005 GOSUB 60:CZ=1
1010 TRAP 1010:POSITION 2,5:? "N
umer sektora poczatkowego: ";:IN
PUT #2,SS
1020 TRAP 40000:TRAP 1020:POSITI
ON 2,7:? "Numer sektora koncoweg
o:   ";:INPUT #2,ZS
1030 IF ZS<SS THEN 1010
1040 IF ZS>720 THEN GOSUB 52
1050 Y=DO:DO=ZS
1055 GOSUB 61:POSITION 8,5:? CHR
$(156);CHR$(156);CHR$(156)
1060 GOTO 1520
1070 DO=Y:CZ=0
1400 GOTO 40
1498 REM MODUL KOPIOWANIA CALEGO
 DYSKU
1500 ? CHR$(125):POSITION 3,2:?
"    * * * KOPIOWANIE DYSKU * *
*"
1510 SS=1
1515 IF DO=1040 THEN GOSUB 52
1520 GOSUB 61
1530 T=1
1540 POSITION 13,5:? "SEKTOR:  0"
1550 POSITION 13,7:? "BUFOR:   0"
1560 GOSUB ZROD
1600 FOR SEK=SS TO DO
1605 AR=ADR(SE$)+(T-1)*128
1610 X=USR(ADR(S$),ND,SEK,AR,82)

1620 IF SE$((T-1)*128+1,T*128)=P
US$ THEN 1633
1630 SE(T)=SEK
1632 T=T+1
1633 POSITION 21,5:? SEK
1636 POSITION 21,7:? T-1
1640 IF T=181 THEN GOSUB PISZ:GO
SUB ZROD
1700 NEXT SEK
1750 IF T<>0 THEN GOSUB PISZ
1850 IF CZ=1 THEN 1070
1900 GOTO 40
2000 ZL=ZL+(ZL=1)-(ZL=2):T$(31,3
1)=STR$(ZL):GOSUB 58
2100 GOTO 100
2500 ND=ND+(ND=1)-(ND=2):T$(71,7
1)=STR$(ND):GOSUB 58
2600 GOTO 100
5998 REM MODUL ODZYSKIWANIA ZBIO
ROW
6000 ? CHR$(125):POSITION 3,2:?
"    * * * ODZYSKIWANIE ZBIORU *
* *"
6005 GOSUB 60
6010 TRAP 6010:POSITION 3,4:? "P
odaj numer zbioru: ";:INPUT FIL
6015 GOSUB 61
6020 IF DO=1040 THEN GOSUB 52
6030 T=1:GES=0
6040 POSITION 13,5:? "SEKTOR:  4"
6050 POSITION 13,7:? "BUFOR:   0"
6100 FOR SEK=4 TO DO
6103 POSITION 21,5:? SEK
6105 AR=ADR(SE$)+(T-1)*128
6110 X=USR(ADR(S$),ND,SEK,AR,82)
6120 IF SE$((T-1)*128+1,T*128)=P
US$ THEN 6200
6122 IF SE$(T*128,T*128)=CHR$(0)
 THEN 6200
6123 SE=ADR(SE$(T*128-2,T*128-2)
)
6124 FN=INT(PEEK(SE)/4)
6125 IF FN<>FIL THEN 6200
6127 POKE SE,PEEK(SE)-FIL*4
6130 SE(T)=SEK
6135 POSITION 21,7:? T
6140 T=T+1:GES=GES+1:IF T=181 TH
EN GOSUB PISZ:GOSUB ZROD
6200 NEXT SEK
6250 GOSUB PISZ
6260 P$(1)=CHR$(0):P$(128)=P$:P$
(2)=P$
6270 P$(1,1)="B":P$(3,3)=CHR$(IN
T(GES/256)):P$(2,2)=CHR$(GES-(IN
T(GES/256)*256))
6280 P$(5,5)=CHR$(INT(SE(1)/256)
):P$(4,4)=CHR$(SE(1)-(INT(SE(1)/
256)*256))
6290 P$(6,16)="ZBIORODZYSK"
6300 X=USR(ADR(S$),ZL,361,ADR(P$
),87)
6340 GOTO 40
6500 DO=DO+320*(DO=720)-320*(DO=
1040)
6510 IF DO=1040 THEN T$(182,192)
=" 2.0 -[2.5]":GOTO 6530
6520 T$(182,192)="[2.0]- 2.5 "
6530 GOSUB 58
6600 GOTO 100
11000 IF VE=1 THEN POKE 1913,80:
T$(142,152)=" TAK -[NIE]":GOSUB
58:VE=0:GOTO 11100
11010 T$(142,152)="[TAK]- NIE ":
GOSUB 58:POKE 1913,87:VE=1
11100 GOTO 100
12000 REM MODUL FORMATUJE DYSKIE
TKE
12010 POSITION 2,14
12020 IF DO=720 THEN ? "Formatow
anie w drive #";ZL;" dla DOS-a 2
.0 (720 sektorow)":LS=0
12030 IF DO=1040 THEN ? "Formato
wanie w drive #";ZL;" dla DOS-a
2.5 (1040 sektorow)":LS=128
12040 POSITION 2,16:? "Napisz T
aby formatowac"
12050 GET #1,A:IF A<>84 THEN POS
ITION 2,17:? "Formatowanie przer
wane":GOTO 100
12070 IF ZL=1 THEN INP$="D1:"
12080 IF ZL=2 THEN INP$="D2:"
12090 TRAP 40000:XIO 254,#5,LS,0
,INP$

12100 POSITION 2,17:? "Formatowa
nie zakonczone":FOR I=1 TO 300:N
EXT I
12150 ? CHR$(125):GOTO 40
14999 REM MODUL ZAPISU NA DYSK
15000 POSITION 3,20:? "Wloz dysk
 [kopie] do drive'u #";ZL
15010 POSITION 15,22:? "Nacisnij
 [RETURN]"
15020 GET #1,A
15030 IF A<>155 THEN 15020
15035 POSITION 3,20:? CHR$(156);
CHR$(156);CHR$(156)
15040 POSITION 13,10:? "SEKTOR:
"
15050 POSITION 13,12:? "BUFOR:
 "
15060 TRAP 40000:TRAP 40
15100 FOR W=1 TO T-1
15110 SK=SE(W)
15113 POSITION 21,10:? SK
15116 POSITION 21,12:? W
15120 AR=ADR(SE$)+(W-1)*128
15130 X=USR(ADR(S$),ZL,SK,AR,87)
15140 NEXT W
15150 T=1
15155 POSITION 13,7:? "BUFOR:  0
 "
15200 RETURN
18000 POSITION 3,20:? CHR$(253);
CHR$(253);"Wloz dysk [zrodlowy]
do drive'u #";ND
18010 POSITION 15,22:? "Nacisnij
 [RETURN]"
18020 GET #1,A
18030 IF A<>155 THEN 18020
18035 POSITION 3,20:? CHR$(156);
CHR$(156);CHR$(156)
18040 RETURN
28000 REM MODUL DEFINIUJACY STAL
E
28001 REM TEKSTOWE I PODPROGRAMY
28002 REM W KODZIE MASZYNOWYM
29000 POKE 709,15:POKE 710,48
29010 RESTORE 29030
29013 AD=155*256


29015 AD=155*256
29020 FOR I=0 TO 9:READ A:POKE A
D+I,A:NEXT I
29030 DATA 0,0,66,12,155,2,2,2,2
,1
29040 DL=PEEK(560)+PEEK(561)*256
:POKE DL,1:POKE DL+1,0:POKE DL+2
,155
29050 POKE AD+10,PEEK(560)+3:POK
E AD+11,PEEK(561)
29060 RETURN
30000 T$(1,40)="DYSK ZRODLOWY
   - DRIVE #1          "
30001 T$(41,80)="KOPIA
   - DRIVE #1          "
30002 T$(81,120)="
                       "
30003 T$(121,160)="FUNKCJA VERIF
Y       TAK -[NIE]       "
30004 T$(161,200)="TYP DOS-A
      [2.0]- 2.5         "
30020 RESTORE 30050
30030 FOR I=1 TO 50:READ A:S$(I)
=CHR$(A):NEXT I:GOTO 30100
30050 DATA 104,104,104,141,1,3,1
04,141,11,3,104,141,10,3,104,141
,5,3,104,141,4,3,104,104,141,2,3
30051 DATA 32,83,228,173,3,3,133
,212,169,0,133,213,96
30100 RESTORE 30150
30110 FOR I=1 TO 50:READ A:TOZ$(
I)=CHR$(A):NEXT I:GOTO 30200
30150 DATA 104,104,133,204,104,1
33,203,104,104,133,205,104,104,1
33,207,104,133,208,104,101,207,1
33,207
30151 DATA 160,0,177,203,56,233,
32,145,207,200,196,205,208,244,9
6
30152 DATA 0,0,0,0,0,0,0,0,0,0,0
,0
30200 POKE 741,255:POKE 742,154
30300 RETURN
```

Oprócz odzyskiwania zbiorów program DYSK—KOPIER realizuje funkcje kopiowania dyskietek. Może on pracować zarówno

pod kontrolą DOS 2.0, jak i DOS 2.5, w zestawie mikrokomputera z jednym lub dwoma napędami dysków.

Wszystkie opcje programu wyglądają następująco

B — kopiowanie określonej liczby sektorów. Użytkownik podaje numer sektora pierwszego i ostatniego, które mają być skopiowane.
C — kopiowanie całego dysku.
D — przełącznik dla zestawu mikrokomputera z dwiema stacjami dysków — określa, w którym napędzie znajduje się kopiowany dysk.
E — podobnie jak D — określa, w którym napędzie ma być tworzona kopia dysku.
L — odzyskiwanie zbiorów.
M — przełącznik — określa, pod którym DOS-em (2.0 — 2.5) ma pracować program.
V — przełącznik — określa, czy zapis kopii ma być kontrolowany dodatkowo przez odczyt (funkcja VERIFY), czy nie.
X — formatowanie dyskietki.

Podczas kopiowania (odzyskiwania) program wczytuje do bufora w pamięci kolejne sektory. Pojemność bufora ogranicza liczbę wczytywanych sektorów do 180. Po zapełnieniu bufora program zapisuje wczytane sektory na tworzoną kopię i czyta kolejne 180 sektorów itd. itd. Puste sektory nie są zapisywane (kopiowane).

Aby odzyskać zbiór lub kilka zbiorów należy po uruchomieniu programu włożyć do stacji dysków dyskietkę, z której zbiory będziemy odzyskiwać. Następnie wciskamy klawisz L. Program zażąda podania numeru zbioru, który chcemy odzyskać. Po podaniu numeru program rozpocznie wyszukiwanie sektorów posiadających wpisany w bajcie 125 podany numer. Po odnalezieniu wszystkich sektorów (lub zapełnieniu bufora) program zatrzyma się i wyświetli komunikat „Włóż dysk—kopię do drive'u # n (1 lub 2). Należy wówczas włożyć do stacji dysków czystą sformatowaną dyskietkę i wcisnąć RETURN. Na dyskietkę tę zostanie zapisany odzyskany zbiór. Jego nazwą będzie ZBIORODYSK. Zbiór ten należy skopiować (na przykład opcją 0 DOS-a) na inną dyskietkę.

Opracował H. KRASUSKI

Elektroliza

Program przedstawia zjawisko elektrolizy. Pokazuje on między innymi ruch jonów w woltametrze przed i po podłączeniu napięcia do elektrod. Jeśli chcesz to zobaczyć, uruchom ten program, a jeśli chcesz pokazać to swoim kolegom, nagraj ten program na taśmę magnetofonową i zademonstruj go na lekcji fizyki. Twój nauczyciel z pewnością się ucieszy.

```
0 REM ELEKTROLIZA;J.GOLLA,P.GRZE
GANEK, P.WAGNER
5 GRAPHICS 2+16:POSITION 4,3:? #
6;"ELEKTROLIZA"
10 FOR T=1 TO 3000:NEXT T:DIM A$
(1),B$(1),V$(40)
15 GRAPHICS 0:POKE 710,12:POKE 7
12,8:POKE 709,2:POKE 755,0:POSIT
ION 1,6
20 ? "SPACJA-POWROT DO LISTY ROZ
KAZOW":?
30 ? "KLAWISZ SELECT-PRZEPLYW PR
ADU NA"
31 ? "SCHEMACIE":?
35 ? "KLAWISZ START-SCHEMAT ZEST
AWU":?
40 ? "KLAWISZ HELP-RUCH JONOW":?
45 IF PEEK(53279)=6 THEN GOTO 39
0
50 IF PEEK(53279)=5 THEN GOTO 54
0
60 IF PEEK(732)=17 THEN POKE 732
,0:GOTO 660
65 GOTO 45
390 POSITION 0,20:? "CZY SCHEMAT
 MA BYC OPISANY(T-N)";
392 INPUT B$:IF B$<>"T" AND B$<>
"N" THEN GOTO 390
395 A=7:IF B$="T" THEN A=8
400 GRAPHICS A+16:COLOR 1
405 PLOT 10,30:DRAWTO 10,80:DRAW
TO 80,80:DRAWTO 80,30.
410 FOR T=11 TO 79:A=T:IF B$<>"T
" THEN A=2
425 COLOR A:PLOT T,35:DRAWTO T,7
9:NEXT T
430 COLOR 1:FOR T=16 TO 22:PLOT
T,39:DRAWTO T,74:NEXT T
435 COLOR 3:FOR T=68 TO 75:PLOT
T,39:DRAWTO T,74:NEXT T
440 COLOR 1:PLOT 19,39:DRAWTO 19
,5:DRAWTO 45,5:DRAWTO 55,0
442 PLOT 55,5:DRAWTO 130,5:DRAWT
O 130,10:PLOT 130,30
445 FOR T=0 TO 7 STEP 0.5:DRAWTO
 SIN(T)*10+130,COS(T)*10+20:NEXT
 T
450 PLOT 122,26:DRAWTO 130,12:DR
AWTO 138,26:PLOT 124,22:DRAWTO 1
35,22
455 PLOT 130,30:DRAWTO 130,70:DR
AWTO 115,70:PLOT 110,70:DRAWTO 9
0,70
460 DRAWTO 90,25:DRAWTO 72,25:DR
AWTO 72,40
462 PLOT 110,55:DRAWTO 110,85:PL
OT 115,80:DRAWTO 115,60
465 SETCOLOR 1,9,13:SETCOLOR 0,1
2,5:SETCOLOR 2,7,1:SETCOLOR 4,2,
0
467 IF B$="T" THEN GOTO 475
470 SETCOLOR 0,12,1:SETCOLOR 2,7
,15:SETCOLOR 4,2,4:SETCOLOR 1,9,
8:GOTO 530
475 TRAP 475:RESTORE 520:FOR L=1
 TO 13:READ PX,PY,V$
480 DL=PEEK(560)+PEEK(561)*256:D
1=PEEK(DL+4)+PEEK(DL+5)*256
490 FOR U=1 TO LEN(V$):D2=57344+
((ASC(V$(U,U))-32)*8)
500 D3=D1+PY*40+PX+U-1:FOR Z=0 T
O 7:POKE D3+Z*40,PEEK(D2+Z)
515 NEXT Z:NEXT U:NEXT L:B$="
":TRAP 40000
520 DATA 102,5,1-ELEKTROLIT,102,
20,2-AMPEROMIERZ
522 DATA 102,35,3-WYLACZNIK,102,
50,4-ANODA,102,65,5-KATODA,102,8
0,6-BATERIA
525 DATA 13,90,+ -,5,50,1,14,10,
2,6,8,3,14,45,6,2,65,5,9,65,4
530 IF PEEK(764)=33 THEN POKE 76
4,255:GOTO 15
535 GOTO 530
540 GRAPHICS 7+16:SETCOLOR 4,10,
0:RESTORE 650
545 FOR T=1 TO 4:READ A,B,C,D
555 FOR F=A TO B STEP C:E=E+1:CO
LOR E-1:PLOT D,F:IF E=4 THEN E=1
560 NEXT F:NEXT T:COLOR 1:FOR T=
1 TO 4:READ A,B,C,D
575 FOR F=A TO B STEP C:E=E+1:CO
LOR E-1:PLOT F,D:IF E=4 THEN E=1
585 NEXT F:NEXT T:FOR T=1 TO 4:R
EAD A,B,C,D,G
590 FOR F=A TO B STEP C:E=E+0.5:
COLOR E-1:PLOT F,D:DRAWTO F,G
595 IF E=4 THEN E=1
600 NEXT F:NEXT T:FOR T=1 TO 12:
A=RND(0)*40+15:B=RND(0)*30+45
605 FOR F=8 TO 1 STEP -1:E=E+1:C
OLOR E-1:PLOT A+F,B:IF E=4 THEN
E=1
615 NEXT F:A=RND(0)*40+15:B=RND(
0)*30+45
620 FOR F=1 TO 8:E=E+1:COLOR E-1
:PLOT A+F,B:IF E=4 THEN E=1
623 NEXT F:NEXT T:COLOR 3
625 PLOT 5,30:DRAWTO 5,80:DRAWTO
 80,80:DRAWTO 80,30:PLOT 70,40:D
RAWTO 75,40
630 PLOT 21,40:DRAWTO 27,40:PLOT
 24,43:DRAWTO 24,37:PLOT 5,33:DR
AWTO 80,33
635 FOR U=1 TO 15:SETCOLOR 0,10,
U+10:SETCOLOR 1,2,U+5:SETCOLOR 2
,13,9
640 IF PEEK(764)=33 THEN POKE 76
4,255:GOTO 15
645 NEXT U:GOTO 635
650 DATA 45,5,-1,15,5,70,1,150,7
0,20,-1,90,20,45,1,65,15,150
651 DATA 1,5,150,120,-1,70,110,9
0,-1,70,90,65,-1,20
655 DATA 17,13,-1,45,75,67,63,-1
,45,75,113,110,-1,60,80,120,117,
-1,50,90
660 ? CHR$(125):POSITION 3,10:?
"KLAWISZ (Z)  ZAMYKA OBWOD PRADU
"
665 FOR T=1 TO 400:NEXT T
670 POSITION 3,10:? "   PROSZE P
OCZEKAC 15 SEKUND   ":FOR T=1 TO
 400:NEXT T
675 X=PEEK(54279):Y=PEEK(53277):
Z=PEEK(559):O=0:POKE 764,255
680 GRAPHICS 5+16
685 AD=PEEK(106)-16:POKE 106,AD-
16
690 POKE 53277,3:POKE 54279,AD:P
OKE 559,62
695 S=AD*256+1024:FOR I=S TO S+1
024:POKE I,0:NEXT I
700 COLOR 3:FOR T=4 TO 78:PLOT T
,20:DRAWTO T,46:NEXT T:COLOR 1:P
LOT 3,15
705 DRAWTO 3,47:DRAWTO 79,47:DRA
WTO 79,15:COLOR 2
706 FOR T=6 TO 8:PLOT T,15:DRAWT
O T,45:NEXT T:COLOR 1
710 FOR T=73 TO 75:PLOT T,15:DRA
WTO T,45:NEXT T
711 PLOT 8,15:DRAWTO 8,8:DRAWTO
20,8:PLOT 20,5:DRAWTO 20,11:PLOT
 22,3
715 DRAWTO 22,13:PLOT 22,8:DRAWT
O 40,8:PLOT 51,8:DRAWTO 73,8:DRA
WTO 73,15
720 SETCOLOR 1,3,15:SETCOLOR 0,4
,0:SETCOLOR 2,4,8:SETCOLOR 4,4,4
725 GOSUB 900:POKE 704,1:POKE 70
5,1:POKE 706,15
726 POKE 707,15:POKE 53256,1:POK
E 53257,1:POKE 53258,1:POKE 5325
9,17
735 Q=RND(0)*15:H=H-Q:IF H<-80 T
HEN H=30
740 IF PEEK(764)=23 THEN POKE 76
4,255:POKE 53248,60:POKE 53249,6
0:GOTO 742
741 GOTO 745
742 POKE 53250,60:POKE 53251,60:
GOTO 785
745 POKE 53248,100-H:Q=RND(0)*10
:M=M-Q:IF M<-40 THEN M=70
750 POKE 53249,100+M:Q=RND(0)*10
```

```
:N=N-Q:IF N<-80 THEN N=30
755 POKE 53250,100-N
760 Q=RND(0)*24:J=J-Q:IF J<-40 T
HEN J=70:IF PEEK(764)=33 THEN GO
TO 800
765 POKE 53251,100+J:GOTO 735
770 DATA 0,0,0,0,24,24,60,60,102
,102,102,231,129,129,129,129,231
771 DATA 102,102,102,60,60,24,24
,0,0,0,0,0
775 DATA 0,0,0,0,24,24,60,60,126
,126,126,255,129,129,129,129,255
776 DATA 126,126,126,60,60,24,24
,0,0,0,0,0
780 COLOR 3:PLOT 40,1:DRAWTO 50,
1
785 FOR T=1 TO 50:NEXT T:COLOR 3
:O=O+1:PLOT 40,O:DRAWTO 50,O:COL
OR 0
786 PLOT 40,O-1:DRAWTO 50,O-1:IF
 O>=0 THEN COLOR 1:PLOT 12,13:GO
TO 790
788 GOTO 795
790 DRAWTO 17,13:PLOT 60,13:DRAW
TO 66,13:PLOT 63,10:DRAWTO 63,16
:GOTO 800
795 GOTO 785
800 IF PEEK(764)=33 THEN POKE 10
6,PEEK(106)+16:POKE 54279,X:GOTO
 802
801 GOTO 805
802 POKE 53277,Y:POKE 559,Z:POKE
 764,255:GOTO 810
805 GOTO 815
810 FOR T=53248 TO 53255:POKE T,
0:NEXT T:GOTO 15
815 TRAP 825:T=0:T1=73:A1=0:B1=0
:A=0:B=0
820 L=RND(0)*80+80:W=L:A=A+1:RET
URN
825 TRAP 835
830 V=RND(0)*80+80:R=V:B=B+1:RET
URN
835 L=L+1:W=W-1:V=V+1:R=R-1
840 POKE 53248,W:POKE 53251,L:PO
KE 53250,V:POKE 53249,R
845 IF L>180 THEN L=180
850 IF W<60 THEN W=60
855 IF V>180 THEN V=180
860 IF R<60 THEN R=60
865 IF W=60 AND L=180 THEN GOSUB
 820
870 IF V=180 AND R=60 THEN GOSUB
 830
875 IF A=3 THEN A=0:A1=A1+1:COLO
R 1:PLOT T+A1,15:DRAWTO T+A1,45
880 IF B=3 THEN B=0:B1=B1-1:COLO
R 2:PLOT T1+B1,15:DRAWTO T1+B1,4
5
885 IF B1=-7 THEN A1=0:B1=0
890 IF PEEK(764)=33 THEN GOTO 80
0
895 TRAP 40000:GOTO 835
900 S=AD*256+1024+170:RESTORE 77
0
905 FOR I=S TO S+28:READ B:POKE
I,B:NEXT I
910 S=AD*256+1280+140:RESTORE 77
0
915 FOR I=S TO S+28:READ B:POKE
I,B:NEXT I:S=AD*256+1656
916 RESTORE 775:FOR I=S TO S+28:
READ B:POKE I,B
920 NEXT I:S=AD*256+1792+190:RES
TORE 775:FOR I=S TO S+28:READ B
921 POKE I,B:NEXT I:RETURN
```

Uwaga: Zamiast znaku } (nawiasu węzykowego) należy w programie napisać niedostępny dla drukarki znak \ (pochyloną strzałkę). (Otrzymujemy go naciskając najpierw klawisz ESC, a następnie równocześnie klawisze CONTROL oraz CLEAR.)

Jan GOLLA, Piotr GRZEGANEK, Paweł WAGNER
Zespół Szkół Chemiczno-Mechanicznych w Gliwicach

* SUMA KONTROLNA / ETYKIETA *

FY	0	WA	5	NS	10	KL	15
WY	20	JP	30	MP	31	FV	35
DM	40	KC	45	FG	50	NM	60
UW	65	UH	390	KD	392	AV	395
ZC	400	PZ	405	FA	410	MR	425
MV	430	BG	435	LL	440	CX	442
FK	445	PE	450	NB	455	OK	460
HQ	462	KU	465	GC	467	UV	470
XC	475	UX	480	MW	490	ZB	500
QD	515	NF	520	FT	522	NM	525
KU	530	PA	535	KF	540	RG	545
PS	555	EC	560	PS	575	VA	585
MP	590	PY	595	CJ	600	PE	605
KH	615	DH	620	XY	623	NY	625
FS	630	BN	635	KX	640	QA	645
TA	650	AL	651	DX	655	WT	660
FT	665	ZE	670	RH	675	IA	680
KB	685	CE	690	QL	695	PL	700
ZZ	705	AH	706	IU	710	CI	711
HY	715	RD	720	ND	725	OT	726
ER	735	DH	740	SF	741	AC	742
ZI	745	QM	750	RU	755	PP	760
BI	765	ET	770	AA	771	CU	775
LL	776	CP	780	XA	785	ZF	786
VL	788	GS	790	UT	795	TW	800
QQ	801	CF	802	RN	805	QQ	810
HX	815	XF	820	DX	825	FT	830
SB	835	WV	840	RV	845	NN	850
CH	855	IP	860	EW	865	LR	870
AY	875	CO	880	TD	885	DV	890
YT	895	GM	900	QR	905	QM	910
US	915	JN	916	HB	920	ZJ	921

Tajemnicze błędy

Na początek proponuję przeprowadzić dwa eksperymenty:

Program 1:

```
10 FOR k=0 TO 10 STEP 1
20 PRINT k
30 NEXT k
```

Program 2:

```
10 PRINT 1—1+1e—30
20 PRINT 1+1e—30—1
```

W pierwszym przypadku dla Amstrada, Atari i Commodore zamiast dokładnych wartości zaczną wyświetlać się liczby z ogonami dziewiątek, a w Spectrum pętla zatrzyma się na wartości 9.9.

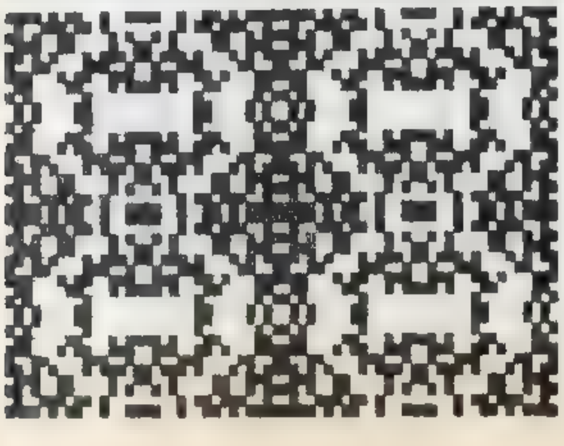
Winę za zaistniałe w obu przypadkach anomalie ponosi dokładność obliczeniowa komputera, a ściślej skończona ilość cyfr po przecinku. Np. 0.1 w zapisie dwójkowym jest nieskończone (tak jak 1/3 w zapisie dziesiętnym). Wobec tego w komputerze jest trzymane przybliżenie 0.1, które podczas wielokrotnego dodawania w pętli potęguje błąd. Jeżeli zamiast 0.1 wstawimy 0.125 (jest ono zapisywane dokładnie), to błędu nie będzie.

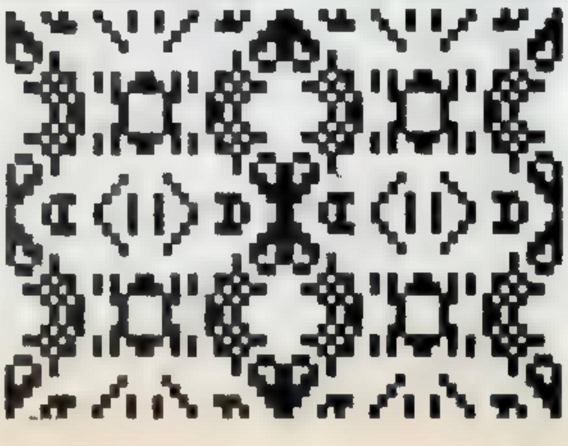
Wniosek: Używać należy kroku dającego się zapisać dokładnie (najlepiej 1):

```
10 FOR k=0 TO 100
20 PRINT k/10
30 NEXT k
```

W drugim przykładzie w linii 20 do 1 dodaje się bardzo mała liczba (1e—30), która ze względu na skończoną ilość liczb po przecinku zostaje obcięta. 1e—30 wymaga postawienia 1 na 30-tym miejscu po przecinku, a komputery nie dopuszczają więcej liczb po przecinku niż 14. Stąd **1+1e—30=1**. Ponieważ dodawanie jest przez komputer wykonywane kolejno, stąd od otrzymanego wyniku zostaje odjęte 1 i powstaje błędny wynik wynoszący 0. Natomiast w linii 10 najpierw zostają od siebie odjęte jedynki dając w efekcie 0, a do 0 zostaje dodana liczba 1e—30. Stąd wynik jest prawidłowy.

Krzysztof POŹNIAK





ZX Spectrum
Semigrafik

```
  1 BORDER 0: CLS : DIM z$(64):
PRINT PAPER 0;z$: DIM a$(15): D
IM b$(15): DIM c$(15): DIM d$(15
): FOR n=1 TO 12: READ a: LET a$
(n)=CHR$ a: READ b: LET b$(n)=CH
R$ b: READ c: LET c$(n)=CHR$ c:
READ d: LET d$(n)=CHR$ d: NEXT n
: RESTORE
  2 LET a=1+INT (RND*15):
LET b=2+INT (RND*5): LET c=13-b:
 LET d=23-b:          LET e=10+b:
 LET f=INT (RND*8):   LET g=16+f:
 LET h=31-f:          LET i=15-f
  3 PRINT
  AT b,f;a$(a);    AT b,g;a$(a);
  AT b,i;b$(a);    AT b,h;b$(a);
  AT e,f;a$(a);    AT e,g;a$(a);
  AT e,i;b$(a),    AT e,h;b$(a),
  AT c,f;c$(a);    AT c,g;c$(a),
  AT c,i;d$(a);    AT c,h;d$(a),
  AT d,f;c$(a);    AT d,g;c$(a),
  AT d,i;d$(a);    AT d,h,d$(a).
  GO TO 2
  4 SAVE "SEMIGRAFIK" LINE 1
  5 DATA
131,131,140,140,140,140,131,131,
137,134,134,137,134,137,137,134,
138,133,138,133,133,138,133,138,
142,141,139,135,139,135,142,141,
143,143,143,143,141,142,135,139,
135,139,141,142,134,137,137,134
 10 FOR n=USR "a" TO USR "b" ST
EP 2: POKE n,170: POKE n+1,85: N
EXT n: LET a$=CHR$ 144+CHR$ 144+
CHR$ 144
 20 BORDER 0: PAPER 0: INK 7: C
LS : LET i=0: LET p=0
 30 DIM t(8): FOR n=1 TO 8: REA
D t(n): NEXT n
 40 DATA 0,2,039,3.86,4.98,7.02
,8.84,10.88,12
 50 FOR m=1 TO 28 STEP 3
 60 FOR n=0 TO 18 STEP 3
 65 IF INKEY$<>"" THEN PAUSE 0
 70 PRINT INK i; PAPER p;AT n,m
;a$;AT n+1,m;a$;AT n+2,m;a$: LET
 p=p+1: IF p=7 THEN LET p=0
 80 NEXT n
 90 PRINT INK i; PAPER p;AT 21,
m;a$: PRINT #0; INK i; PAPER p;A
T 0,m,a$;AT 1,m;a$: LET p=p+1: I
F p=7 THEN LET p=0
100 LET i=i+1: BEEP .01,14+t(i)
110 IF i=8 THEN LET i=0
120 NEXT m: GO TO 50
200 CLEAR : SAVE "kolor 8*10" L
INE 10: BEEP .2,2: VERIFY "": RE
STORE : GO TO 10
1000 REM
   Program demonstruje kolory,
powstałe ze zmieszania 8 kolorów
palety komputera Spectrum
```

PROGRAM XZ SPECTRUM 54

Bakarat

BAKARAT to bardzo popularna w XIX w we Francji gra hazardowa w karty. Jak zwykle w takich przypadkach reguły są proste. Osiem talii kart, bankier, stawki i nieograniczona liczba graczy. Kazdy z nich stawia pewną sumę, lecz rozgrywa tylko ten, który postawił najwięcej. On i bankier otrzymują po 2 karty. Wygrywa ten, którego karty mają najwięcej punktów

Poszczególne karty punktuje się następująco: as — 1, figury i 10 — Ø, blotki — ich wartość. Jeśli suma punktów przekracza 9, to oblicza się jej modulo 9. Gracz rozgrywający wygrywa automatycznie mając 8 lub 9 punktów, (chyba że bankier ma tyle samo), czeka mając 6 lub 7 punktów, musi dobrać kartę mając mniej niż 5 punktów, decyduje o dobraniu karty mając 5 punktów

Bankier zawsze dobiera kartę, jeśli ma początkowo mniej niż 5 punktów. Jeśli rozgrywający wygra, to bank wypłaca wszystkim graczom stawki. Jeśli wygra bankier, to wszyscy tracą postawione pieniądze na jego korzyść.

Zasady gry są nieco uproszczone, co nie przeszkadza przecież w ich modelowaniu. Zresztą powstało wiele odmian bakarata, chociażby CHEMIN DE FER.

Krzysztof MAMCARZ

```
  10 GO SUB 1000
  20 GO SUB 2000
  25 GO SUB 2500
  30 GO SUB 3000
  40 GO SUB 4000
  50 GO SUB 6000
  60 GO TO 25
 100 STOP
1000 REM  GRAFIKI
1010 LET bank=0
1020 DIM k(8,4,13)
1030 LET adres=USR "a"
1040 RESTORE : FOR a=0 TO 31
1050 READ bajtek  POKE adres+a,bajtek
1060 NEXT a
1500 RETURN
1700 DATA 0,108,254,254,124,56,16,0,16,56,124,254,254,84,56,0,16,56,124,254,124,56,16,0,56,56,214,254,214,16,56,0
2000 REM  POCZATEK
2010 INPUT "ilu graczy ?",gracz
2020 IF (gracz<>INT gracz) OR (gracz<1) THEN GO TO 2010
2030 DIM i$(gracz,15): DIM f(gracz): DIM s(gracz)
2040 FOR a=1 TO gracz
2050 INPUT "imie gracza ",(a);": ", LINE i$(a)
2060 PRINT a,i$(a)
2070 LET f(a)=10000
2080 NEXT a
2090 RETURN
2500 REM  START
2510 PAUSE 100: LET c=0: LET max=0: CLS
2520 FOR a=1 TO gracz
2530 IF f(a)<1 THEN LET c=c+1: NEXT a
2535 IF c=gracz THEN PRINT INVERSE 1,"WSZYSCY PRZEGRALI": GO SUB 6000
2540 PRINT "gracz ";a;" : ";f(a)
2550 INPUT (i$(a))'"ile stawiasz ? ";s(a)
2560 IF (s(a)<>INT s(a)) OR (s(a)<1) THEN GO TO 2130
2570 IF s(a)>f(a) THEN GO TO 2110
2580 IF s(a)>max THEN LET max=s(a): LET opoz=a
2590 NEXT a: RETURN
3000 REM  ROZDANIE
3010 CLS : PRINT INVERSE 1;"BANKIER"
3020 LET sumab=0: LET sumag=0
3030 FOR t=1 TO 2
3040 GO SUB 7000: LET sumab=sumab+pkt: GO SUB 7500
3050 NEXT t
3060 PRINT : PRINT INVERSE 1;i$(opoz)
3070 FOR t=1 TO 2
3080 GO SUB 7000: LET sumag=sumag+pkt: GO SUB 7500
3090 NEXT t: PRINT
3100 LET sumab=sumab-9*(sumab>9)
3110 LET sumag=sumag-9*(sumag>9)
3120 RETURN
4000 REM  DECYZJE
4010 IF sumab>7 OR sumag>7 THEN GO TO 4120
4020 IF sumag<5 THEN LET w$="T": GO TO 4070
4030 IF sumag>5 THEN GO TO 4100
4040 INPUT (i$(opoz))'"Czy dobierasz karte ? (T/N)";w$
4060 IF w$="N" OR w$="n" THEN GO TO 4100
4070 IF w$="T" OR w$="t" THEN PRINT : PRINT i$(opoz);" dobiera ";: GO SUB 7000: LET sumag=sumag+pkt-9*(sumag>9): GO SUB 7500: GO TO 4100
4080 GO TO 4040
4100 IF sumab>4 THEN GO TO 4120
4110 PRINT "bankier","dobiera ",: GO SUB 7000: LET sumab=sumab+pkt-9*(sumab>9): GO SUB 7500
4120 PRINT : IF sumab=sumag THEN PRINT INVERSE 1;"R E M I S": RETURN
4130 IF sumab>sumag THEN PRINT INVERSE 1,"BANK WYGRAL": GO TO 4150
4140 PRINT INVERSE 1;i$(opoz),"WYGRAL"
4150 GO SUB 5000
4160 PAUSE 0: CLS
4170 PRINT "STAN GOTOWKI "
4180 PRINT "_______________"
4190 PRINT : PRINT "BANK ",bank
4200 FOR a=1 TO gracz
4210 PRINT i$(a),f(a)
4220 NEXT a
4230 RETURN
5000 REM  WYPLATA
5010 FOR a=1 TO gracz
5020 IF sumab>sumag THEN LET s(a)=-s(a)
5030 LET f(a)=f(a)+s(a)
5040 LET bank=bank-s(a)
5050 NEXT a
5060 RETURN
6000 REM  KONIEC
6010 INPUT "gramy dalej ? (T/N) ";w$
6020 IF w$="T" OR w$="t" THEN RETURN
6040 STOP
7000 REM  LOSOWANIE
7010 LET kolor=INT (RND*4)+1
7020 LET karta=INT (RND*13)+1
7030 FOR a=1 TO 8
7040 IF k(a,kolor,karta)=0 THEN LET k(a,kolor,karta)=1: GO TO 7100
7050 NEXT a: GO TO 7010
7100 IF karta>9 THEN LET pkt=0
7110 IF karta<10 THEN LET pkt=karta
7120 RETURN
7500 REM  RYSOWANIE KARTY
7510 IF karta<11 AND karta>1 THEN LET e$=STR$ karta: GO TO 7560
7520 IF karta=1 THEN LET e$="A"
7530 IF karta=11 THEN LET e$="W"
7540 IF karta=12 THEN LET e$="D"
7550 IF karta=13 THEN LET e$="K"
7560 PRINT e$; INK (0+2*(kolor=1 OR kolor=3));CHR$ (kolor+143)
7570 RETURN
```

Przeliczanie

```
  Program przelicza metryczne
 miary długości na mile, jardy
 stopy i cale z dokładnością do
          1/64 cala

  1 mile = 1760 yds = 5280 ft
  1 yd   = 3 ft     = 36 in
  1 ft   = 12 in

  1 mile = 1609,64 m
  1 km   = 1094 yd 3 19/64 in
--------------------------------
 7 km 253 m 47 cm 4 mm =

 4 mi 328 yd 2 ft 4 7/16 in
--------------------------------
 29 km 53 cm 8 mm =

 18 mi 49 yd 9 1/8 in
--------------------------------
```

```
 100 REM  program MILE/KM
 110 FOR n=0 TO 23: READ a: POKE USR "a"+n,a: NEXT n
 120 DATA 0,0,56,4,60,68,60,6,0,16,20,24,48,16,12,0,3,16,56,64,56,4,120,0
 130 REM a=a b=t c=s
 140 CLS : BORDER 6
 150 PRINT INK 1;"
  Program przelicza metryczne
 miary długości na mile, jardy,
 stopy i cale z dokładnością do
          1/64 cala"
 160 PRINT INK 2;"
  1 mile = 1760 yds = 5280 ft
  1 yd   = 3 ft     = 36 in
  1 ft   = 12 in"
 165 PRINT INK 0;"
  1 mile = 1609.64 m
  1 km   = 1094 yd 3 19/64 in"
 170 FOR t=1 TO 6: BEEP .01,26: BEEP .01,24: BEEP .015,30: NEXT t
 180 PRINT INK 2'"------------------------------------------"
 190 INPUT "kilometry ", LINE k$: IF k$="" THEN LET klm=0
 191 IF k$<>"" THEN LET klm=VAL k$
 192 INPUT "metry "; LINE m$: IF m$="" THEN LET mtr=0
 193 IF m$<>"" THEN LET mtr=VAL m$
 194 INPUT "centymetry ", LINE c$: IF c$="" THEN LET ctm=0
 195 IF c$<>"" THEN LET ctm=VAL c$
 196 INPUT "milimetry ", LINE l$: IF l$="" THEN LET mlm=0
 197 IF l$<>"" THEN LET mlm=VAL l$
 200 LET mm=mlm+10*ctm+1000*mtr+1000000*klm
 210 LET km=0: LET m=0: LET cm=0: LET mm=INT mm
 240 IF mm>9 THEN LET cm=INT (mm/10): LET mm=mm-10*INT (mm/10)
 250 IF cm>99 THEN LET m=INT (cm/100): LET cm=cm-100*INT (cm/100)
 260 IF m>999 THEN LET km=INT (m/1000): LET m=m-1000*INT (m/1000)
 270 PRINT " ";: IF km=0 THEN GO TO 290
 280 PRINT km;" km ";
 290 IF m=0 THEN GO TO 310
 300 PRINT m;" m ";
 310 IF cm=0 THEN GO TO 330
 320 PRINT cm;" cm ";
 330 IF mm=0 THEN GO TO 350
 340 PRINT mm;" mm ";
 350 LET mm=mm+10*cm+1000*m+1000000*km
 360 PRINT "="
 370 LET mi=mm/1608640
 380 PRINT '" "
 390 IF INT mi=0 THEN GO TO 410
 400 PRINT INT mi;" mi ";
 410 LET yd=(mi-INT mi)*1760
 420 IF INT yd=0 THEN GO TO 440
 430 PRINT INT yd;" yd ";
 440 LET ft=(yd-INT yd)*3
 450 IF INT ft=0 THEN GO TO 470
 460 PRINT INT ft;" ft ";
 470 LET in=(ft-INT ft)*12
 480 IF in<1/64 THEN GO TO 170
 490 IF INT in=0 THEN GO TO 510
 500 PRINT INT in;" ";
 510 LET reszta=INT (64*(in-INT in))/64
 520 IF reszta=0 AND in>=1 THEN PRINT "in": GO TO 170
 530 FOR n=1 TO 5
 540 IF reszta*2↑n=INT (reszta*2↑n) THEN PRINT (2↑n*reszta);"/";2↑n;" in": GO TO 170
 550 NEXT n
 560 GO TO 170
9000 SAVE "KM/MILE" LINE 100: BEEP .2,36: CLS : PRINT "Verify KM/MILE": VERIFY "": GO TO 100
```

Przedstawiamy zaktualizowaną listę (por. "IKS" nr. 1/87 oraz nr. 12/87) skrótów, które związane są z mikrokomputerem, łącznie z ich oryginalną interpretacją. Czekamy na inne, Waszym zdaniem, ważne skróty.

W jednym z najbliższych numerów "IKS-a" postaramy się przedstawić uaktualnioną listę skrótów wraz z ich tłumaczeniem (interpretacją) w języku polskim.

ACK - AFFIRMATIVE ACKNOWLEDGMENT,
ACU - AUTOMATIC CALLING UNIT,
ADA - ALL POINTS ADDRESSABLE,
ADLC - ADWANCED DATA LINK CONTROLLER,
ADP - AUTOMATIC DATA PROCESSING,
ADPCM - ADAPTIVE DIFFERENTIAL PCM,
AI - ARTIFICIAL INTELLIGENCE,
AL - AUTOLOAD,
ALE - ADDRESS LATCH ENABLE,
ALU - ARITHMETIC-LOGIC UNIT,
APA - ALL POINTS ADDRESSABLE MODE,
API - AUTOMATIC PROGRAM INTERRUPT,
ARQ - AUTOMATIC REQUEST FOR REPEATS,
ASCII - AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE,
AT - ADVANCED TECHNOLOGY,
AU - ARITHMETIC UNIT,
BASIC - BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE,
BCC - BLOCK CHARACTER CHECK,
BCD - BINARY CODED DECIMAL,
BCD - BLANK COLUMN DETECTION,
BDOS - BASIC DOS,
BER - BIT ERROR RATE,
BEX - BROADBAND EXCHANGE,
BF - BEST FIT ALGORITHM,
BFD - BEST FIT DECREASING ALGORITHM,
BIOS - BASIC INPUT-OUTPUT SYSTEM,
BIU - BUS INTERFACE UNIT,
BORAM - BLOCK-ORIENTED RAM,
BPI - BITS PER INCH,
BPS - BITS PER SECOND,
BSC - BINARY SYNCHRONOUS COMMUNICATIONS,
CAD - COMPUTER AIDED DESIGN,
CAI - COMPUTER-AIDED INSTRUCTION,
CAM - COMPUTER AIDED MANUFACTURE,
CAMAC - COMPUTER AIDED MEASUREMENT AND CONTROL,
CAN - CANCEL CHARACTER,
CAP - COMPUTER AIDED PUBLISHING,
CCP - CONSOLE COMMAND PROCESSOR,
CD ROM - COMPACT DISC ROM,
CGA - COLOR GRAPHICS ADAPTER,
CIM - COMPUTER INPUT MICROFILM,
CISC - COMPLEX INSTRUCTION SET COMPUTER,
CIU - COMPUTER INTERFACE UNIT,
CMOS - COMPLEMENTARY METAL-OXIDE SEMICONDUCTOR,
CNC - COMPUTER NUMERICAL CONTROL,
CPI - CHARACTERS PER INCH,
COM - COMPUTER OUTPUT MICROFILM,
CPL - CURRENT PRIVILEGE LEVEL,
CPM - CONTROL PROGRAM FOR MICROCOMPUTERS (CP/MONITOR),
CPM - CRITHICAL PATH METHOD,
CPS - CHARACTERS PER SECOND,
CPU - CENTRAL PROCESSING UNIT,
CRAM - CARD RANDOM ACCESS MEMORY,
CRC - CYCLICAL REDUNDANCY CHECK,
CRC - CYCLIC REDUNDANCY CHECK CHARACTER,
CROM - CONTROL AND READ-ONLY MEMORY,
CRT - CATHODE-RAY TUBE,
CSL - CONTROL AND SIMULATION LANGUAGE,
CT - COMPUTER TOMOGRAPH,
CTI - COAXIAL TRANSCEIVER INTERFACE,
CU - CONTROL UNIT,
CVSD - CONTINUOUSLY VARIABLE SLOPE DELTAMODULATION,
DAA - DATA ACCESS ARRANGEMENT,
DAS - DATA ACQUISITION SYSTEM,
DBMS - DATABASE MANAGEMENT SYSTEM,
DCE - DATA COMMUNICATION EQUIPMENT,
DCN - DISTRIBUTED COMPUTER NETWORK,
DDCMP - DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL,
DDD - DIRECT DISTANCE DIALING,
DDE - DIRECT DATA ENTRY,
DDL - DATA DESCRIPTION LANGUAGE,
DDS - DATA-PHONE DIGITAL SERVICE,
DEL - DELETE CHARACTER,
DES - DATA ENCRYPTION STANDARD,
DFA - DIGITAL FAULT ANALYSIS,
DG - DATAGRAM,
DI - DIGITAL INPUT,
DIC - DIGITAL INTEGRATED CIRCUIT,
DID - DIGITAL INFORMATION DISPLAY,
DIP - DUAL IN-LINE PACKAGE,
DLE - DATA LINK ESCAPE,
DMA - DIRECT ACCESS MEMORY,
DMAC - DIRECT MEMORY ACCESS CONTROLLER,
DML - DATA MANIPULATION LANGUAGE,
DNC - DIRECT NUMERICAL CONTROL,
DO - DIGITAL OUTPUT,
DOC - DIAGNOSTICS ON-CHIP,
DOR - DIGITAL OPTICAL RECORDING,
DOS - DISC OPERATING SYSTEM,
DPB - DISK PARAMETER BLOCK,
DPL - DESCRIPTOR PRIVILEGE LEVEL,
DRAM - DYNAMIC RANDOM ACCESS MEMORY,
DRAW - DIGITAL READ AND WRITE,
DTA - DISK TRANSFER AREA,
DTE - DATA TERMINAL EQUIPMENT,
DTE - DT-EQUIPMENT,
DU - DISK UNIT,
E^2 - ELECTRICALLY ERASABLE DEVICE,
EA - EFFECTIVE ADDRESS,
EAC - END-AROUND CARRY,
EARN - EUROPEAN ACADEMIC RESEARCH NETWORK,
EAROM - ELECTRICALLY ALTERABLE ROM,
EARS - ELECTRONIC AUDIO RECOGNITION SYSTEM,
EBCDIC - EXTENTED BINARY-CODED DECIMAL INTERCHANGE CODE,
EBNF - EXTENDED BACKUS-NAUR FORM,
ECC - ERROR CORRECTION CODES,
ECD - ENHANCED CONSOLE DRIVER,
ECGB - ENHANCED COLOR GRAPHICS BOARD,
ECMA - EUROPEAN COMPUTER MANUFACTURES' ASSOCIATION,
ECTL - EMITTER-COUPLED TRANSISTOR LOGIC,
EDC - ERROR DETECTION/CORRECTION,
EDP - ELECTRONIC DATA PROCESSING,
EDPC - ELECTRONIC DATA PROCESSING CENTER,
EDPE - ELECTRONIC DATA PROCESSING EQUIPMENT,
EDPM - ELECTRONIC DATA PROCESSING MACHINE,
EDR - EXTERNAL DATA REPRESENTATION,
EDSAC - ELECTRONIC DELAY STORAGE AUTOMATIC CALCULATOR,
EDT - EDITOR,
EDVAC - ELECTRONIC DISCRETE VARIABLE AUTOMATIC CALCULATOR,
EFL - ERROR FREQUENCY LIMIT,
EGA - ENHANCED GRAPHIC ADAPTER,
EIA - ELECTRONIC INDUSTRIES ASSOCIATION,
EIN - EUROPEAN INFORMATION NETWORK,
ELD - ELECTROLUMINESCENT DISPLAY,
ELSI - EXTRA-LARGE-SCALE INTEGRATION,
EMI - ELECTROMAGNETIC INTERFERENCE,
EMOS - ENHANCEMENT MOS,
ENQ - ENQUIRY,
EOB - END OF BLOCK,
EOF - END OF FILE,
EOI - NON-SPECIFIC END OF INTERRUPT,
EOR - END OF RECORD,
EOT - END OF THE TAPE,
EOT - END OF TRANSMISSION,
EPROM - ERASABLE PROGRAMMABLE READ ONLY MEMORY,
E^2PROM - ELECTRICALLY ERASABLE PROGRAMMABLE ROM,
ERA - ERASE CHARACTER,
EROM - ERASABLE ROM,
ERR - ERROR,
ES - EXTRA SEGMENT,
ESC - ESCAPE CHARACTER,
ESD - EXTERNAL SYMBOL DICTIONARY,
ESS - ELECTRIC SWITCHING SYSTEM,
ETB - END OF TRANSMISSION BLOCK CHARACTER,
ETX - END OF TEXT CHARACTER,
EU - EXECUTION UNIT,
EXE - EXECUTE,
FAT - FILE ALLOCATION TABLE,
FAX - FACSIMILE,
FC - FONT-CHANGE CHARACTER,
FCB - FILE CONTROL BLOCK,
FCFS - FIRST-COME FIRST-SERVED,
FCP - FILE CONTROL PROCESSOR,
FCS - FRAME CHECK SEQUENCE,
FD - FILE DESCRIPTION,
FDB - FILE DESCRIPTION BLOCK,
FDM - FREQUENCY DIVISION MULTIPLEXING,
FEP - FRONT END PROCESSOR,
FET - FIELD EFFECT TRANSISTOR,
F/F - FLIP-FLOP,
FF - FIRST FIT ALGORITHM,
FF - FORM FEED,
FFD - FIRST FIT DECREASING ALGORITHM,
FIB - FILE IDENTIFICATION BLOCK,
FIFO - FIRST-IN FIRST-OUT,
FILER - FILE MANAGER,
FILO - FIRST-IN LAST-OUT,
FINUFO - FIRST-IN-NOT-USED-FIRST-OUT,
FLOPS - FLOATING-POINT OPERATIONS PER SECOND,
FLX - FILE EXCHANGE UTILITY,
FM - FREQUENCY MODULATION,
FMT - FORMAT,
FPLA - FIELD-PROGRAMMABLE LOGIC ARRAY.

FPROM - FIELD-PROGRAMMABLE ROM,
FPU - FLOATING POINT PROCESSING UNIT,
FS - FILE SEPARATOR,
FSK - FREQUENCY SHIFT KEYING.
GE - GREATER OR EQUAL,
GEM - GRAPHICS ENVIRONMENT MANAGER,
GIGO - GARBAGE IN, GARBAGE OUT,
GKS - GRAPHICS KERNEL SYSTEM,
GPS - GENERAL PROBLEM SOLVING,
GSX - GRAPHICS SYSTEM EXTENSION,
GT - GREATER THEN,
HASP - HOUSTON AUTOMATIC SPOOLING PROGRAM,
HDAM - HIERARCHICAL DIRECT ACCESS METHOD,
HDLC - HIGH LEVEL DATA LINK CONTROL,
HDR - HIGH DENSITY RECORDING,
HGC - HERCULES GRAPHICS CARD,
HIC - HYBRID INTEGRATED CIRCUIT,
HISAM - HIERARCHICAL INDEXED SEQUENTIAL ACCESS METHOD,
HLL - HIGH-LEVEL LANGUAGE,
IAD - IMAGE ACQUISITION AND DISPLAY,
IAS - IMMEDIATE ACCESS STORAGE,
IBM - INTERNATIONAL BUSINESS MACHINES,
IC - INTEGRATED CIRCUIT,
ICA - INTRA-APPLICATION COMMUNICATIONS AREA,
ICL - INTERNATIONAL COMPUTERS LTD.,
ID - IDENTIFY MARKER,
IDAS - INDUSTRIAL DATA ACQUISITION SYSTEM,
IDP - INTEGRATED DATA PROCESSING,
IDS - INTEGRATED DATA STORE,
IFF - INTERCHANGE FILE FORMAT,
IFIP - INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING,
ILBM - INTERLEAVE BIT MAP,
IMIS - INTEGRATED MANAGEMENT INFORMATION SYSTEM,
IMP - INTERFACE MESSAGE PROCESSOR,
IMPACT - IMPLEMENTATION PLANNING AND CONTROL TECHNIQUE,
IMR - INTERRUPT-MASK REGISTER,
INFO - INFORMATION FIELD,
INT - INTEGER,
INT - INTERRUPT,
I/O - INPUT/OUTPUT,
IOCS - INPUT/OUTPUT CONTROL SYSTEM,
IOS - INPUT/OUTPUT SYSTEM,
IP - IMAGE POSITIONING,
ISAM - INDEX-SEQUENTIAL ACCESS METHOD,
ISO - INTERNATIONAL STANDARDS ORGANISATION,
ISR - INTERRUPT SERVICE ROUTINE,
ISR - IN SERVICE REGISTER,
IT - INFORMATION TECHNOLOGY,
ITB - INTERMEDIATE TEXT BLOCK,
JCL - JOB CONTROL LANGUAGE,
JMP - JUMP,
KB - KILOBYTE,
KBD - KEYBOARD,
KSAM - KEYED SEQUENTIAL ACCESS METHOD,
LAN - LOCAL AREA NETWORK,
LCD - LIQUID-CRYSTAL DISPLAY,
LCFS - LAST COME-FIRST SERVED,
LF - LINE FEED,
LFU - LEAST FREQUENTLY USED REMOVAL,
LIC - LINEAR INTEGRATED CIRCUIT,
LIFO - LAST-IN FIRST-OUT,
LIOCS - LOGICAL INPUT/OUTPUT CONTROL SYSTEM,
LISP - LIST PROCESSING (LANGUAGE),
LPC - LINEAR PREDICTIVE CODING,
LRU - LEAST RECENTLY USED REMOVAL,
LSB - LOWER-SET SIGNIFICANT BIT,
LSI - LARGE SCALE INTEGRATION,
LT - LESS THAN,
LUT - LOOKUP TABLE,
MAC - MEDIUM ACCESS CONTROL,
MAR - MEMORY ADDRESS REGISTER,
MAU - MEDIUM ATTACHMENT UNIT,
MBR - MEMORY BUFFER REGISTER,
M.C. - MACHINE CYCLE,
MCU - MICROPROCESSOR CONTROL UNIT,
MDA - MONOCHROME DISPLAY/PRINTER ADAPTER,
MDI - MEDIUM DEPENDED INTERFACE,
MDR - MEMORY DATA REGISTER,
MFM - MODIFIED FREQUENCY MODULATION,
MGC - MONOCHROME GRAPHIC/PRINTER CARD,
MIC - MONOLITHIC INTEGRATED CIRCUIT,
MIC - MICROWAVE INTEGRATED CIRCUIT,
MICR - MAGNETIC INK CHARACTER READER,
MICR - MAGNETIC INK CHARACTER RECOGNITION,
MIDI - MUSICAL INSTRUMENT DIGITAL INTERFACE,
MIS - MANAGEMENT INFORMATION SYSTEM,
MMU - MEMORY MANAGEMENT UNIT,
MODEM - MODULATION/DEMODULATION DEVICE,
MOP - MULTIPLE ON-LINE PROGRAMMING,
MS DOS - MICROSOFT DOS,
MSI - MEDIUM-SCALE INTEGRATION,
MTTF - MEAN TIME TO FAILURE,
NAK - NEGATIVE ACKNOWLEDGEMENT,
NC - NUMERICAL CONTROL,
NCP - NETWORK CONTROL PROGRAM,
NDRO - NON-DESTRUCTIVE READOUT,
NIC - NETWORK INTERFACE CONTROLLER,
NIU - NETWORK INTERFACE UNITS,
NLQ - NEAR LETTER QUALITY,
NO-OP - NO-OPERATION INSTRUCTION,
NPR - NON-PROCESSOR REQUEST,
NRFD - NOT READY FOR DATA,
OCR - OPTICAL CHARACTER RECOGNITION,
ODS - OUTPUT DATA STROBE,
OEM - ORIGINAL EQUIPMENT MANUFACTURER,
OPR - OPERATION REGISTER,
PAM - PARTITIONED ACCESS METHOD,
PC - PERSONAL COMPUTER,
PC - PROGRAM COUNTER,
PCM - PULSE CODE MODULATION,
PCN - PERSONAL COMPUTER NETWORK,
PCU - PERIPHERAL CONTROL UNIT,
PDC - PROCESS DIGITAL CONTROLLER,
PDN - PUBLIC DATA NETWORK,
PDS - PROGRAM DEVELOPMENT SYSTEM,
PERT - PROGRAM EVALUATION AND REVIEW TECHNIQUE,
PGC - PROFESSIONAL GRAPHICS CONTROLLER,
PGS - PROFESSIONAL GRAPHICS SYSTEM,
PIC - PROGRAMMABLE INTERRUPT CONTROLLER,
PIC - PERSONAL IDENTIFICATION CODE,
PIF - PROGRAM INFORMATION FILE,
PIO - PROGRAMMABLE INPUT OUTPUT,
PIP - PERIPHERAL INTERCHANGE PROGRAM,
PIPS - PAN INFORMATION PROCESSING SYSTEM,
PL/1 - PROGRAMMING LANGUAGE 1,
PLA - PROGRAMMABLE LOGIC ARRAY,
PLC - PROGRAMMABLE LOGIC CONTROLLER,
PL/M - PROGRAMMING LANGUAGE FOR MICROPROCESSORS,
PL/Z - PROGRAMMING LANGUAGE FOR ZILOG,
PM - PHASE MODULATION,
PMD - POSTMORTEM DUMP,
POST - POWER-ON SELF TEST,
POS TERMINAL - POINT-OF-SALE TERMINAL,
PPI - PROGRAMMABLE PERIPHERAL INTERFACE,
PROM - PROGRAMMABLE READ-ONLY MEMORY,
PSN - PACKED SWITCHING NETWORK,
PSU - POWER SUPPLY UNIT,
PSW - PROCESSOR STATUS WORD,
PSW - PROGRAM STATUS WORD,
QBE - QUERY BY EXAMPLE,
QISAM - QUEUED INDEXED SEQUENTIAL ACCESS METHOD,
QUSAM - QUEUED SEQUENTIAL ACCESS METHOD,
QUTAM - QUEUED TELECOMMUNICATION ACCESS METHOD,
RACS - REMOTE-ACCESS COMPUTING SYSTEM,
RAM - RANDOM ACCESS MEMORY,
RCTL - RESISTOR-CAPACITOR TRANSISTOR LOGIC,
RECOL - RETRIEVAL COMMAND LANGUAGE,
REPROM - REPROGRAMMABLE MEMORY,
RGB - RED-GREEN-BLUE,
RISC - REDUCED INSTRUCTION SET COMPUTER,
RJE - REMOTE JOB ENTRY,
RMM - READ MOSTLY MEMORY,
ROM - READ-ONLY MEMORY,
ROMP - READ-ONLY MEMORY PROGRAMMABLE,
ROS - READ ONLY STORAGE,
RPC - REMOTE PROCEDURE CALL,
RPG - REPORT PROGRAM GENERATOR,
RPN - REVERSE POLISH NOTATION,
RPROM - REPROGRAMMABLE ROM,
RS - RECORD SEPARATOR,
RSX - RESOURCE SHARING EXECUTIVE,
RTC - REAL-TIME CLOCK,
RTL - RESISTOR-TRANSISTOR LOGIC,
RVI - REVERSE INTERRUPT,
RZ - RETURN TO ZERO (CODING),
SAA - SYSTEM APPLICATION ARCHITECTURE,
SAM - SEQUENTIAL ACCESS METHOD,
SATO - SELF-ALIGNED THICK OXIDE (TECHNOLOGY),
SBC - SINGLE-BOARD COMPUTER,
SCCS - SOURCE CODE CONTROL SYSTEM,
SCRAM - STATIC COLUMN RAM,
SDA - SOURCE DATA AUTOMATION,
SDI - SELECTIVE DISSEMINATION OF INFORMATION,
SDLC - SYNCHRONOUS DATA LINK CONTROL,
SGPA - STATIC GRAPH PARTITIONING ALGORITHM,
SI - SHIFT-IN CHARACTER,
SID - SOUND INTERFACE DEVICE,
SIMPP - SIMPLE IMAGE-PROCESSING PACKAGE,
SJN - SHORTEST JOB NEXT.

dokończenie w następnym numerze

EDIT SESSION – sesja edycji,
EDIT WORD – słowo wydawnicze,
EDP – patrz: ELECTRONIC DATA PROCESSING,
EDPC – patrz: ELECTRONIC DATA PROCESSING CENTER,
EDPE – patrz: ELECTRONIC DATA PROCESSING EQUIPMENT,
EDPM – patrz: ELECTRONIC DATA PROCESSING MACHINE,
EDR – patrz: EXTERNAL DATA REPRESENTATION,
EDSAC – patrz: ELECTRONIC DELAY STORAGE AUTOMATIC CALCULATOR,
EDT – patrz: EDITOR,
EDUCABILITY – edukacja, nauczanie,
EDUCATION – patrz: EDUCABILITY,
EDUCATIONAL AIDS – pomoce naukowe,
EDUCATIONAL STATUS – poziom wykształcenia,
EDUCATIONAL SYSTEM – szkolnictwo,
EDULCORATE – usuwać błędną informację,
EDVAC – patrz: ELECTRONIC DISCRETE VARIABLE AUTOMATIC CALCULATOR,
EFFECTIVE – skuteczny, użyteczny, efektywny,
EFFECTIVE ADDRESS – adres efektywny,
EFFECTIVE FOR ... – obowiązujący na okres ...,
EFFECTIVE TIME – czas efektywny, czas użytkowy,
EFFECTIVE TRANSFER RATE – praktyczna szybkość przesyłania (danych),
EFFICIENCY – sprawność, współczynnik sprawności, także: wydajność, efektywność,
EFFICIENT – sprawny, wydajny, efektywny,
EFL – patrz: ERROR FREQUENCY LIMIT,
EGA – patrz: ENHANCED GRAPHIC ADAPTER,
EGOLESS PROGRAMMING – programowanie grupowe (technologia programowania),
EIA – patrz: ELECTRONIC INDUSTRIES ASSOCIATION,
EIGENVALUE – wartość własna,
EIGENVECTOR – wektor własny,
EIGHT'S COMPLEMENT – uzupełnienie do ośmiu, dopełnienie do ośmiu,
EIGHTY-COLUMN CARD – karta 80-kolumnowa,
EIN – patrz: EUROPEAN INFORMATION NETWORK,
EITHER-WAY – półdupleks,
EITHER-WAY CIRCUIT – kanał półdupleksowy,
EJECT – wyrzucać, wypychać, wyprowadzać, wydalać (np. karty z maszyny),
EJECTION – wyrzucanie, wypychanie, wyprowadzanie,
ELABORATE – wypracowany, starannie wykonany (we wszystkich szczegółach), złożony (np. mechanizm), skomplikowany,
ELABORATION – opracowanie, wypracowanie,
ELAPSED TIME CLOCK – licznik zegarowy,
ELECTRIC SENSING – odczytywanie elektryczne,
ELECTRIC SWITCHING SYSTEM – elektroniczny układ przełączania,
ELECTRICAL – elektryczny,
ELECTRICALLY – elektrycznie,
ELECTRICALLY ALTERABLE ROM – pamięć stała reprogramowalna elektrycznie, pamięć EAROM,
ELECTRICALLY ERASABLE DEVICE – urządzenie pamięciowe wymazywalne i programowalne elektrycznie,
ELECTRICALLY ERASABLE ROM – pamięć stała wymazywalna elektrycznie,
ELECTRICITY – elektryczność,
ELECTROMECHANIC – elektromechaniczny,
ELECTROMAGNETIC INTERFERENCE – zakłócenia elektromagnetyczne,
ELECTROMAGNETIC RELAY – przekaźnik elektromagnetyczny,
ELECTRONIC ANALOG COMPUTER – analogowa maszyna elektroniczna,
ELECTRONIC BRAIN – komputer elektroniczny, elektroniczna maszyna matematyczna, "mózg elektronowy",
ELECTRONIC DATA PROCESSING – elektroniczne przetwarzanie danych, EPD,
ELECTRONIC DATA PROCESSING CENTER – centrum elektronicznego przetwarzania danych,
ELECTRONIC DATA PROCESSING EQUIPMENT – elektroniczny sprzęt do przetwarzania danych,
ELECTRONIC DATA PROCESSING MACHINE – elektroniczna maszyna do przetwarzania danych,
ELECTRONIC DELAY STORAGE AUTOMATIC CALCULATOR – elektroniczny licznik automatyczny na liniach opóźniających,

ELECTRONIC DISCRETE VARIABLE AUTOMATIC CALCULATOR – elektroniczny licznik automatyczny zmiennych dyskretnych,
ELECTRONIC DOCUMENT – elektroniczny dokument, elektroniczny tekst,
ELECTRONIC INDUSTRIES ASSOCIATION – Organizacja Standardów Elektronicznych,
ELECTRONIC MAIL – elektroniczna poczta (środki przesyłania i ochrony wiadomości pomiędzy użytkownikami sieci komputerowej),
ELECTRONIC PACKAGING – montaż zespołów z mikroukładów elektronicznych,
ELECTRONIC POCKET-SIZE CALCULATOR – kieszonkowy kalkulator elektroniczny, minikalkulator,
ELECTRONIC SCULPTURING – modelowanie za pomocą komputera analogowego,
ELECTRONIC WORKSHEETS – tabele kalkulacyjne (programy odtwarzające na komputerze zadania planowania i kalkulowania),
ELECTRONICS – elektronika,
ELECTROSTATIC PLOTTER – ploter elektrostatyczny, pisak elektrostatyczny,
ELECTROSTATIC PRINTER – drukarka elektrostatyczna,
ELECTROSTATIC PRINTER-PLOTTER – patrz: ELECTROSTATIC PRINTER,
ELECTROTHERMAL PRINTER – drukarka termiczna,
ELEMENT – element, składnik, część składowa, człon,
ELEMENTARY – elementarny, jednostkowy,
ELEVATE – podnosić,
ELEVATION – podnoszenie, wzniesienie, także: rzut pionowy,
ELEVEN PUNCH – naddziurka w strefie jednostek (w 80-cio kolumnowej karcie dziurkowanej),
ELIGIBLE – gotowy do kontynuacji (np. do dalszej pracy),
ELIMINATION – eliminacja, usuwanie,
ELITE FACE – tryb drukowania z gęstością 12 znaków na cal,
ELONGATE – wydłużać (się),
ELSE – inaczej,
ELSI – patrz: EXTRA-LARGE-SCALE INTEGRATION,
EMBEDDING – wstawka, także: obejmujący, osłaniający,
EMBEDED – włączony, wstawiony, wbudowany (np. podprogram, procedura w ramach programu),
EMBODY – wbudować, włączyć, obejmować,
EMBOSSMENT – zniekształcenie,
EMERGENCY – awaria, stan zagrożenia, niebezpieczeństwo, wypadek,
EMERGENCY CONDITION – sytuacja krytyczna, warunek krytyczny,
EMI – patrz: ELECTROMAGNETIC INTERFERENCE,
EMITTING – emitowanie,
EMOS – patrz: ENHANCEMENT MOS,
EMPHASIZED – wytłuszczone,
EMPIRICAL – empiryczny, doświadczalny,
EMPIRICAL PROBABILITY – prawdopodobieństwo a posteriori,
EMPLOY – zatrudniać,
EMPLOYER – pracownik,
EMPTY – pusty, próżny,
EMPTY LOOP – cykl pusty,
EMPTY SET – zbiór pusty,
EMPTY STRING – łańcuch pusty (o zerowej długości),

EMULATE – imitować, naśladować, także: współzawodniczyć,
EMULATION – emulacja, także: symulacja urządzenia, tworzenie odpowiednika,
EMULATOR – emulator,
ENABLE – umożliwiać,
ENABLE INTERRUPTS – włącz przerwania,
ENABLE PULSE – impuls zezwalający,
ENABLING A LINE – umożliwienie połączenia, nawiązanie połączenia,
ENCAPSULATE – hermetyzować,
ENCAPSULATED ELEMENT – mikroelement w obudowie,
ENCAPSULATION – obudowa, hermetyzacja (mikroelementu), także: przedstawić coś, opisać w sposób ukryty,
ENCIPHER – zaszyfrować, zakodować,
ENCIPHERING – szyfrowanie, kodowanie,
ENCLOSED HEREWITH – w załączeniu,
ENCLOSING – osłanianie, obejmowanie,
ENCLOSURE – załącznik,
ENCODE – kodować, zakodować,
ENCODER – koder, urządzenie kodujące, szyfrator,
ENCODING – kodowanie, szyfrowanie,
ENCOMPASS – obejmować,
ENCRYPTED MESSAGE – wiadomość zaszyfrowana,
ENCRYPTION – szyfrować, utajniać, także: szyfrowanie, kodowanie,
ENCRYPTION OF INFORMATION – utajnianie informacji,
END – koniec,
END ADDRESS – adres końcowy,
END-AROUND BORROW – ujemne przeniesienie cykliczne,
END-AROUND CARRY – przeniesienie cykliczne (z najwyższej pozycji liczby do najniższej),
END DISTORTION – zniekształcenie końcowe,
END-FILE MARK – znacznik końca pliku (zbioru),
ENDLESS FORM – formularz (druk) ciągły,
ENDLESS LOOP – pętla nieskończona,
ENDLESS TAPE – taśma "bez końca",
END MARK – znak końca,
END MODE – tryb zakończenia (rozkazu),
END OF BLOCK – znak końca bloku,
END OF FILE – znacznik końca zbioru,
END-OF-FILE INDICATOR – wskaźnik końca pliku (zbioru) danych,
END-OF-FILE LABEL – etykieta końca zbioru danych,
END OF LINE – koniec wiersza,
END OF MEDIUM – koniec nośnika,

ENGINEERING - technika, inżynieria,
ENGINEERING CYBERNETICS - cybernetyka techniczna,
ENGINEERING TIME - czas obsługi technicznej,
ENHANCE - uwydatniać, uwypuklać, wzmagać, zwiększać, wzmacniać,
ENHANCED - ulepszony, polepszony, udoskonalony,
ENHANCED COLOR DISPLAY - wyświetlacz ulepszonego koloru,
ENHANCED COLOR GRAPHICS BOARD - karta ulepszonej grafiki kolorowej,
ENHANCED GRAPHICS ADAPTER - adapter ulepszonej grafiki,
ENHANCED MODEL - model rozwinięty, model wzbogacony,
ENHANCED VERSION - wersja rozwinięta, wersja wzbogacona,
ENHANCEMENT - rozszerzanie, także: udoskonalanie, wzbogacanie,
ENHANCEMENT MOS - tranzystor MOS z kanałem wzbogaconym,
ENHANCEMENT TRANSISTOR - tranzystor polowy o kanale wzbogaconym, tranzystor ze wzbogaceniem,
ENLARGE - powiększać,
ENLARGEMENT - powiększenie,
ENQ - patrz: ENQUIRY,
ENQUEUE - ustawić w kolejce,
ENQUIRY - zapytanie (np. systemu),
ENQUIRY CHARACTER - symbol zapytania,
ENSAMBLE - zbiór (w matematyce),
ENSURE - zapewniać, upewniać się,
ENTER - umieszczać, wnosić, wprowadzać,
ENTERPRISE - przedsiebiorstwo, także: przedsięwzięcie,
ENTERPRISE ADMINISTRATOR - administrator dziedziny przedmiotowej,
ENTERPRISE DATA BASE - baza danych przedsiębiorstwa,
ENTITY - wyodrębniona całość, jednostka istnienia, istota, sens, sedno, także: objekt,
ENTRAP - zatrzymać, zamknąć,
ENTRY - komunikat wprowadzony do komputera, wejście, zapis (w katalogu), hasło (w słowniku), także: instrukcja, deklaracja w programie,
ENTRY BLOCK - blok wejścia,
ENTRY LABEL - patrz: ENTRY NAME,
ENTRY NAME - nazwa (punktu) wejścia,
ENTRY POINT - punkt wejścia, adres (punktu) wejścia,
ENUMERABLE - policzalny,
ENUMERATE - wyliczać,
ENUMERATED TYPE - typ wyliczeniowy,
ENUMERATION - numeracja, także: przeliczalność,
ENUMERATIVE ALGORITHM - algorytm wyliczeniowy, algorytm przeliczeniowy,
ENVELOPE - koperta, otoczka, powłoka, także: obwiednia,
ENVELOPE CURVE - obwiednia,
ENVELOPE DELAY - opóźnienie grupowe,
ENVELOPE FEEDBACK - sprzężenie zwrotne wewnętrzne,
ENVIRONMENT - zestaw maszynowy, warunki pracy, środowisko, otoczenie,
ENVIRONMENT DIVISION - dział zestawu maszynowego (w Cobolu),
ENVIRONMENTAL - środowiskowy,
ENVIRONMENTAL INPUT - wejście środowiskowe,
EOB - patrz: END OF BLOCK,
EOF - patrz: END OF FILE,
EOI - patrz: NON-SPECIFIC END OF INTERRUPT,
EOR - patrz: END OF RECORD,
EOT - patrz: END OF THE TAPE,
EOT - patrz: END OF TRANSMISSION,
EPROM - patrz: ERASABLE PROGRAMMABLE READ ONLY MEMORY,
E^2PROM - patrz: ELECTRICALLY ERASABLE ROM,
EPSS - sieć komunikacyjna EPSS,
EQUAL - równać się, być równym,
EQUAL ODDS - równe szanse,
EQUALITY - równość,
EQUALITY SIGN - znak równości, (identyczności),
EQUALIZATION - korygowanie,
EQUALIZE - wyrównywać, zrównywać,
EQUALIZER - korektor,
EQUALIZING - wyrównywanie,
EQUALLY - w równym stopniu,
EQUALLY LIKELY CASES - przypadki równie prawdopodobne,
EQUALS SIGN - znak równości, "=",
EQUATE - przyrównywać, także: sprowadzać do wspólnego mianownika,
EQUATES - stałe przyporządkowania,
EQUATION - równanie, także: wzór matematyczny,
EQUATION SOLVER - analizator wielomianów algebraicznych,
EQUIANGULAR - równokątny,
EQUI-BOUNDED - wspólnie ograniczony,
EQUIDISTANT - w równych odstępach,
EQUILATERAL - równoboczny,
EQUILIBRATE - równoważyć,
EQUILIBRIUM - równowaga,
EQUINUMEROUS - równoliczny,
EQUIP - wyposażać,
EQUIPMENT - wyposażenie, sprzęt,
EQUIPMENT COMPATIBILITY - wymienność sprzętu,
EQUIPMENT FAILURE - awaria sprzętu,
EQUIPOLLENT - równoważny,
EQUIPONDERATE - równoważyć,
EQUIPOTENT - równoliczny,
EQUIPROBABLE - jednakowo prawdopodobny, o tym samym stopniu prawdopodobieństwa,
EQUIVALENCE - równoważność, równowartość,
EQUIVALENCE ELEMENT - element równoważności,
EQUIVALENT - równoważny, równowartościowy, równorzędny,
EQUIVALENT CIRCUIT - układ zastępczy,
ERA - patrz: ERASE CHARACTER,
ERASABILITY - wymazywalność,
ERASABILITY OF STORAGE - kasowalność danych,
ERASABLE - dający się wymazać,
ERASABLE PROGRAMMABLE READ ONLY MEMORY - pamięć stała, którą można programować i kasować (np. kasowanie elektryczne),
ERASABLE ROM - pamięć stała kasowalna, pamięć ROM kasowalna,
ERASABLE STORAGE - pamięć kasowalna,
ERASE - wycierać, wymazywać, kasować,
ERASE CHARACTER - znak wymazujący, wycierający, znak błędu,
ERASE KEY - klawisz anulowania, klawisz kasowania,
ERASE ROUTINE - podprogram zerujący, program zerujący, podprogram kasujący,
ERASE TIME - czas wymazywania (w monitorach z lampą pamięciową),
ERASING - wymazywanie, usuwanie,
ERASURE - miejsce wymazane, przekreślenie w tekście, także: wycieranie (informacji gumką),
E-REGISTER - patrz: EXTENSION REGISTER,
EROM - patrz: ERASABLE ROM,
ERR - patrz: ERROR,
ERRATIC - błędny, pomyłkowy,
ERRONEOUS - omyłkowy, błędny,
ERRONEOUSLY - omyłkowo, przez omyłkę,
ERROR - błąd, uchyb,
ERROR BURST - grupa błędów, pakiet błędów występujących z tego samego powodu,
ERROR BYTE - bajt błędu,
ERROR CHECK - kontrola błędu,
ERROR-CHECKING CODE - kod wykrywający błędy, kod sprawdzający,
ERROR CONDITION - przypadek wystąpienia błędu,
ERROR CONTROL - kontrola błędów,
ERROR CORRECTING - korekcja błędów,
ERROR CORRECTING CODE - patrz: ERROR CORRECTION CODE,
ERROR-CORRECTING COMPILER - kompilator z automatyczną korekcją błędów,
ERROR CORRECTION - korekcja błędów,
ERROR CORRECTION CODE - kod korygujący błędy, kod samokorygujący,
ERROR DETECTING CODE - kod detekcyjny (wykrywający błędy),
ERROR DETECTION - wykrywanie błędu,
ERROR DETECTION/CORRECTION - detekcja i korekcja błędów,
ERROR DIAGNOSTIC - komunikaty o błędach, doniesienia o błędach,
ERROR FLAG - flaga błędu, znacznik błędu,
ERROR-FREE - bezbłędny, wolny od błędów,
ERROR FREQUENCY LIMIT - częstość graniczna błędów,
ERROR GAP - odstęp między błędami,
ERROR HANDLING - obsługa błędu,
ERROR HANDLING ROUTINE - podprogram obsługi błędów,
ERROR INDICATE LIGHT - świetlny sygnał błędu,
ERROR INDICATION - wykazywanie błędów,
ERROR INTERRUPT - przerwa w wykonywaniu programu w skutek błędu, który nie może być automatycznie skorygowany,
ERROR MEAN-SQUARE - błąd średniokwadratowy,
ERROR MESSAGE - komunikat błędu,
ERROR PRINTOUT - wykaz błędów,
ERROR PROBABILITY - prawdopodobieństwo popełnienia błędu,
ERROR-PRONE - łatwo ulegający błędom,
ERROR PROPAGATION - powtarzanie (powielanie) błędu,
ERROR PROTECTION - ochrona przed popełnieniem błędu,
ERROR RATE - wskaźnik błędów (np. przy odczycie taśmy),
ERROR RECOVERY - korekcja błędów,
ERROR ROUTINE - program korekcyjny,
ERROR SEARCH PROGRAM - program wyszukiwania błędów,
ERROR TAPE - taśma błędów,
ERROR TRAPPING - wychwytywanie błędów,
ES - patrz: EXTRA SEGMENT,
ESC - patrz: ESCAPE CHARACTER,
ESCALATION - wzrost, wzmaganie się, narastanie,
ESCAPE - unikać, uciekać, także: ucieczka, przejście do innej opcji,
ESCAPE CHARACTER - znak funkcyjny,
ESCAPE CODE - kod kierujący, kod sterujący,
ESCAPE KEY - klawisz wyjścia, klawisz ucieczki (pozwala na przerwanie wykonywanej pracy na komputerze),

Pole minowe

Ta gra przeznaczona jest dla tych, którzy lubią rozwiązywać zagadki logiczne. Jest to adaptacja gry ze „SPECTRUM" do możliwości graficznych „MERITUM". Zasady gry podane są po jej uruchomieniu. Poziom trudności regulowany jest deklarowaną przez gracza ilością min. Ilość ta praktycznie nie powinna przekraczać 100, bo przejście może być niemożliwe, choć na polu zmieścić się może 280 min.

Życzę przyjemnej zabawy.

Bolesław MRUGAŁA

```
10 RANDOM:CLS
20 DIM M(15,20)
30 PRINT CHR$(23)
40 PRINT@460,"P O L E   M I N O W E"
50 FOR D=1 TO 600:NEXT D
60 CLS:PRINT"TWOJE ZADANIE POLEGA NA PRZEJSCIU PRZEZ POLE Z MINAMI
DO PRZECIWLEGLEJ BRAMY. JEZELI STANIESZ NA POLU SASIADUJACYM
Z POLEM ZAMINOWANYM TO USLYSZYSZ SYGNAL. JEZELI STANIESZ NA
POLU Z MINA - SYGNALU JUZ NIE USLYSZYSZ.
80 PRINT@384,"PRZEMIESZCZAJ SIE ZA POMOCA KLAWISZY:
         I - W GORE
         M - W DOL
         J - W LEWO
         K - W PRAWO

100 INPUT"ILOMA MINAMI MA BYC ZAMINOWANE POLE";MN
105 CLS
110 FOR I=4 TO 121
120 SET(I,2):SET(I,45):NEXT I
130 FOR J=2 TO 17
140 SET(4,J):SET(5,J):SET(120,J):SET(121,J):NEXT J
150 FOR J=30 TO 45
160 SET(4,J):SET(5,J):SET(120,J):SET(121,J):NEXT J
170 MI=0
180 MI=MI+1
190 IF MI>MN THEN 230
200 IX=RND(19):IG=RND(14)
210 IF M(IG,IX)=1 THEN 200
220 M(IG,IX)=1:GOTO 180
230 X=20:Y=7:XS=7:YS=20:XN=XS:YN=YS
240 PRINT@Y*64+3*X,CHR$(191);CHR$(179);CHR$(191);CHR$(19);
260 A$=INKEY$
270 IF A$=""THEN 260
280 IF A$="J"THEN YN=YS-1:GOTO 320
290 IF A$="K"THEN YN=YS+1:GOTO 320
300 IF A$="I"THEN XN=XS-1:GOTO 320
310 IF A$="M"THEN XN=XS+1:GOTO 320
315 GOTO 260
320 IF XN<1 OR XN>14 THEN XN=XS:GOTO 260
330 IF XN>5 AND XN<10 THEN 350
340 IF YN<1 OR YN>19 THEN YN=YS:XN=XS:GOTO 260 ELSE 400
350 IF YN>20 THEN YN=YS:XN=XS:GOTO 260
360 IF YN=0 THEN PRINT@6,"  B R A W O  ";ELSE 400
370 PRINT@XS*64+YS*3+1,CHR$(191);CHR$(19);
380 PRINT@XN*64+YN*3,CHR$(191);CHR$(179);CHR$(191);CHR$(19);
390 GOTO 500

400 PRINT@XS*64+YS*3+1,CHR$(191);CHR$(19);
410 PRINT@XN*64+YN*3,CHR$(191);CHR$(179);CHR$(191);CHR$(19);
420 XS=XN:YS=YN
430 IF M(XN,YN)=1 THEN GOTO 1000
435 A=0:IF YN=20 THEN 440 ELSE IF M(XN,YN+1)=1 THEN A=A+1
440 IF M(XN,YN-1)=1 THEN A=A+1
445 IF M(XN+1,YN)=1 THEN A=A+1
447 IF M(XN-1,YN)=1 THEN A=A+1
448 IF A=0 THEN 260
450 FOR J=1 TO A
460 FOR I=1 TO 5:NEXT I
470 FOR I=1 TO 10
480 OUT 254,1:OUT 254,0:NEXT I
490 NEXT J:GOTO 260
500 FOR I=1 TO 14
510 FOR J=1 TO 19
520 IF M(I,J)=1 THEN PRINT@I*64+J*3+1,CHR$(140);CHR$(19);
530 NEXT J: NEXT I
540 PRINT@966," JESZCZE RAZ? - NACISNIJ SPACJE ";
550 IF INKEY$=" " THEN 600 ELSE 550
600 CLS
610 FOR I=1 TO 20
620 FOR J=1 TO 14
630 M(J,I)=0:NEXT J:NEXT I
640 GOTO 100
1000 FOR J=1 TO 10:NEXT J
1010 FOR I=1 TO 11
1020 PRINT@XN*64+YN*3,CHR$(128);CHR$(140);CHR$(128);CHR$(19);
1030 FOR J=1 TO 10:NEXT J
1040 PRINT@XN*64+YN*3,CHR$(166);CHR$(128);CHR$(153);CHR$(19);
1050 FOR J=1 TO 10 :NEXT J
1060 NEXT I:GOTO 500
```

Liga Myślących

Zadanie 1

Znaleźć trzy najmniejsze liczby naturalne n takie, że między n oraz n+10 nie ma żadnej liczby pierwszej, jak również trzy najmniejsze liczby naturalne m takie, że między 10m a 10(m+1) nie ma żadnej liczby pierwszej.

Zadanie 2

Znaleźć wszystkie liczby naturalne a, dla których liczba $a^{10} + 1$ jest podzielna przez 10.

Zadanie 3

Na stole leżą cztery kule o jednakowym promieniu r, stykające się jedna z drugą. We wnękę utworzoną przez nie położono piątą kulę o tym samym promieniu. Znaleźć odległość najwyżej położonego punktu piątej kuli od płaszczyzny stołu.

Zadanie 4

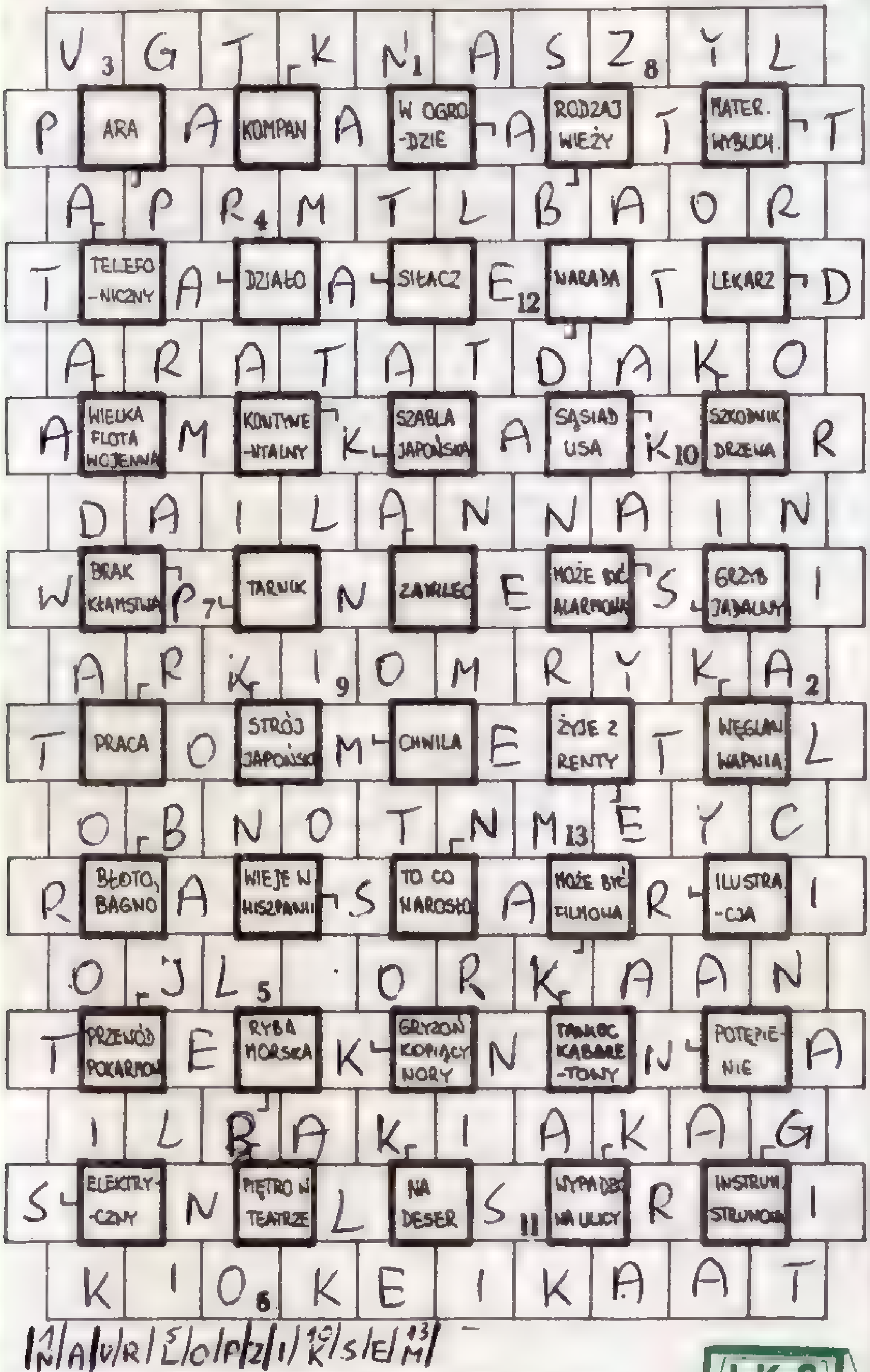
Z miejscowości A w kierunku B wyszedł piechur. Po a godzinach z B wyjechał kolarz na spotkanie piechura i po b godzinach od swego wyjazdu spotkał piechura. Ile czasu potrzeba kolarzowi, a ile piechurowi, aby przebyć całą drogę między A i B, jeżeli kolarz potrzebuje na to o c godzin mniej niż piechur?

Zadanie 5

Wydajność warsztatu A wynosi m procent wydajności warsztatów B i C, a wydajność warsztatu B wynosi n procent wydajności warsztatów A i C. Jaki procent wynosi wydajność warsztatu C w stosunku do wydajności warsztatów A i B?

Rozwiązania zadań prosimy przesyłać do redakcji do końca lipca br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

KRZYŻÓWKA NR 5

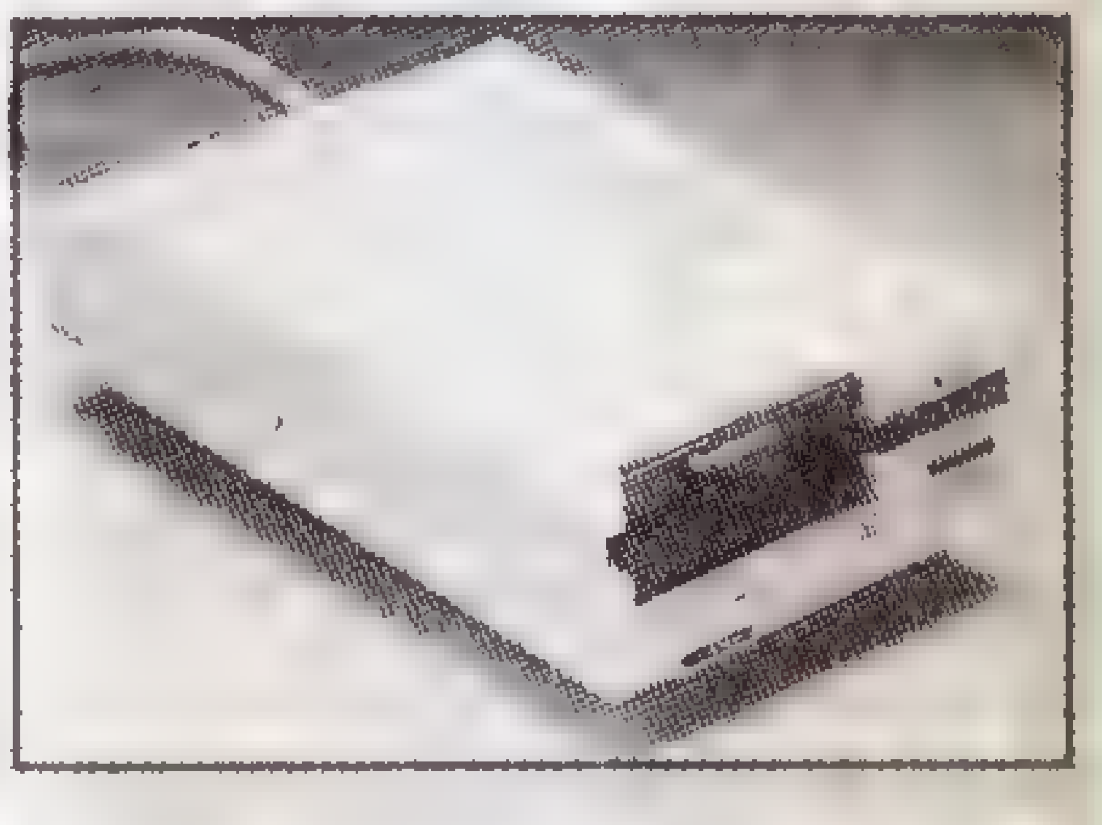


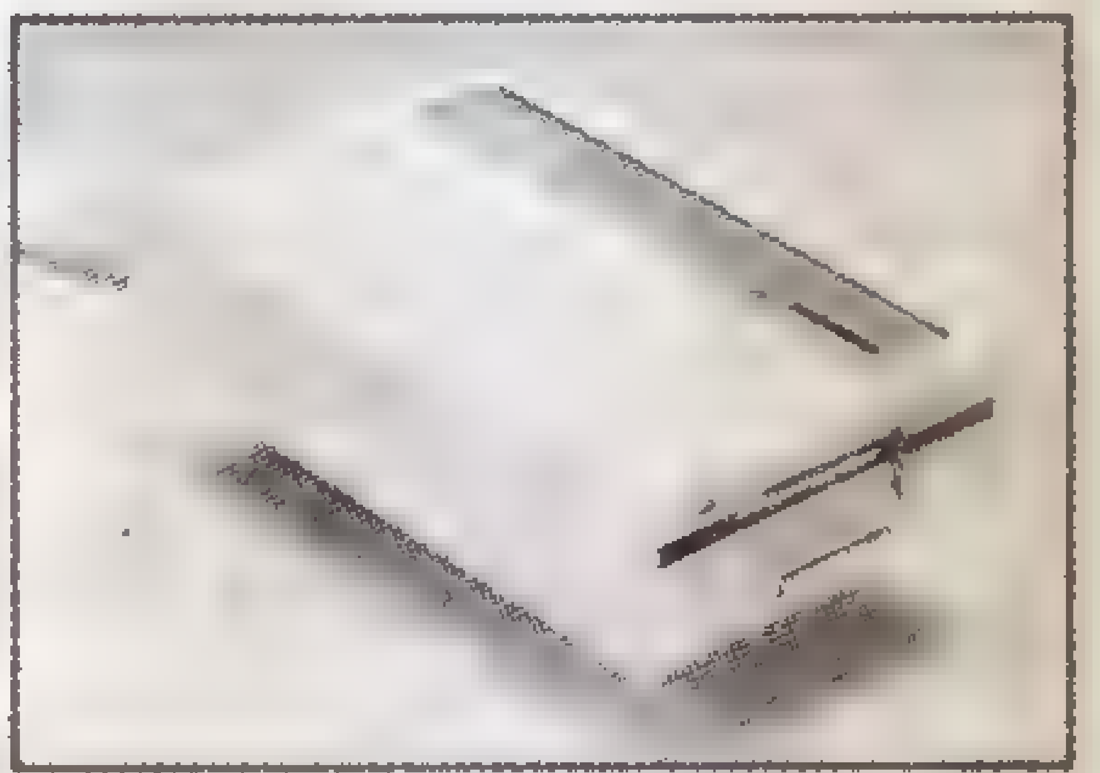


Listy z kratek ponumerowanych od 1 do 13 dadzą hasło, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji na kartach pocztowych w terminie do końca sierpnia naklejając kupon „IKS"-a. Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

MIKROCIEKAWOSTKI

Bondwell









„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1350. Nr ind. 361682. U-23

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 ZŁ

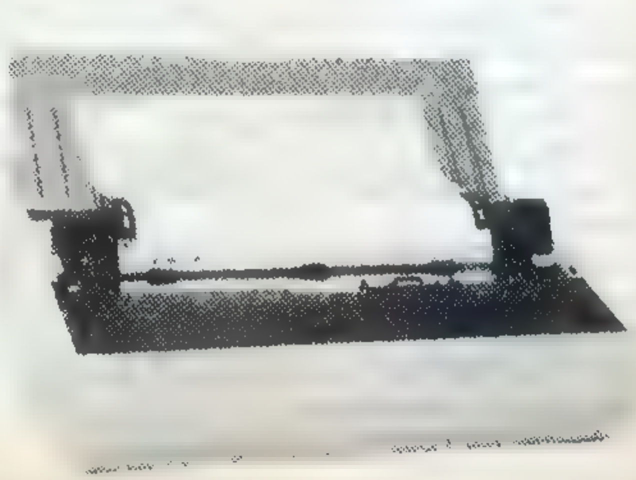
DODATEK DO „ŻOŁNIERZA WOLNOŚCI" Nr 6/1988 ISSN 0860 — 2794

Komputer dla muzyka
— czyli komponować może każdy
„Dodatkową" pamięć Atari
odkrywamy na stronie 20



AMSTRAD 9512
RAM — 512 KB
System operacyjny CP/M+





AMSTRAD 1640
RAM - 640 KB
Mikroprocesor 8086 (8 Mhz)
EGA, Hercules
MDA lub CGA

Zacznijmy nietypowo, bo od cytatu z rządowego „PROGRAMU ROZWOJU NAUKI I TECHNIKI NA LATA 1986—90" będącego załącznikiem do obecnie realizowanego Narodowego Planu Społeczno-Gospodarczego. Na str. 9 tego dokumentu sformułowano następujące zadanie związane bezpośrednio z informatyką:

„Wykorzystując wielkie tradycje polskiej szkoły matematycznej, zostanie w latach 1986—90 stworzony i rozwinięty przemysł oprogramowania, który może i powinien stać się naszą specjalnością eksportową. Za zadanie szczególnej wagi przyjmuje się także opracowanie i wdrożenie do produkcji seryjnej urządzeń komputerowych do wspomagania polskiego naukowca i inżyniera projektanta".

Czytając treść tego zadania można się niewątpliwie zgodzić, że oba wymienione tam cele są ambitne. W sytuacji, gdy na polskim rynku mini- i mikrokomputerowym dominuje oprogramowanie kradzione, z niekompetentną i często na kolanie sporządzoną dokumentacją, a oprogramowanie dużych i średnich komputerów cierpi na uwiąd starczy [1], rzetelna próba osiągnięcia tych celów spotkałaby się z rzeczywistym poparciem wielu ambitnych informatyków. Nic w tym dziwnego, przecież wielu z nas zdaje sobie coraz wyraźniej sprawę z jałowości pobieżnego rozpoznawania i „oswajania" mniej lub bardziej przypadkowo pozyskiwanego [2] z najdziwniejszych źródeł oprogramowania. Każdy może podać przykłady absolwentów najlepszych uczelni, którzy w poszukiwaniu pieniędzy lutują pakiety, w zdobycznych programach wymieniają komunikaty w języku angielskim na polskie, zajmują się sprowadzaniem, składaniem i sprzedażą dalekowschodniego sprzętu informatycznego. W efekcie tracą wiele ze swoich zdolności twórczych, często bezpowrotnie. Oczywiście takie działanie z ich punktu widzenia jest racjonalne i słuszne. Nie oznacza to jednak, że jest obiektywnie korzystne. Stąd wyżej sformułowane zamierzenie rządowe mogło i powinno być odczytane jako chęć stworzenia przez państwo szansy na nowe i nowoczesne wyposażenie miejsca pracy projektantów i programistów, na pracę planową, zorganizowaną i koncepcyjnie trudną, na przyciągającą najlepszych, możliwość uczestniczenia w międzynarodowym podziale informatycznego rynku pracy. Zgadzając się z tym trzeba uczciwie przyznać, że nie jest to zadanie łatwe. Po pierwsze dlatego, że opóźnienie w produkcji elementów i podzespołów elektronicznych oraz rodzimego sprzętu informatycznego tzn. sprzętu wytwarzanego z krajowych elementów, jest bardzo znaczne i wynosi około 10 lat.

W NUMERZE:

Sieci	**— str. 3**
Chińska krzemowa dolina	**— str. 4**
Piraci końca XX wieku	**— str. 4**
Neuronowy komputer	**— str. 5**
W szponach Atari (16)	**— str. 6**
Skalowanie wykresu	**— str. 9**
Inteligentny samochód	**— str. 11**
Przesyłanie kodu maszynowego do instrukcji REM	**— str. 12**
Comodore C-64 — edytor tekstu	**— str. 13**
Można inaczej	**— str. 15**
Nuty z komputera (2)	**— str. 16**
Jak wykorzystać dodatkową pamięć Atari	**— str. 20**
Giełda pomysłów	**— str. 21**
Lilavati (6)	**— str. 22**
Oprogramowanie układów transputerowych	**— str. 23**
Sami tworzymy symbole graficzne	**— str. 26**
Spearmen i Pearson powiedzą Ci prawdę	**— str. 28**
Liga Myślących	**— str. 31**
Też komputery	**— str. 32**

W sytuacji, gdy cały wysoko i średnio rozwinięty świat uporał się z seryjną produkcją komputerów 32-bitowych nasz przemysł państwowy nadal nie jest w stanie wytworzyć elementów do produkcji całkowicie polskiego mikrokomputera 16-bitowego. Nie znaczy to, że domagam się pełnej samowystarczalności, odcięcia się od wyników światowych i namawiam do hołdowania zasadzie „każdy sobie..." Wręcz przeciwnie, jestem za aktywnym udziałem w międzynarodowym rynku produkcji i wymiany. Jednakże uważam, że z ogólnocywilizacyjnych względów kraj tej wielkości co Polska, powinien być w stanie samodzielnie produkować komputery określonych klas. Mikrokomputery 16-bitowe z pewnością do nich należą. Drugą ważną barierą jest to, że realizacja wspomnianego zadania przypadła na lata kryzysowe, których nieodłącznym atrybutem jest ciągły i dotkliwy brak wolnych środków finansowych (złotówkowych i dewizowych). Niezależnie od tych faktów można chyba sądzić, że autorom tego zapisu nie chodziło o spektakularne rozwiązanie. Miejmy nadzieję, że słusznie uznali, korzystając z doświadczeń innych krajów, że rozwój informatyki może być jednym ze sposobów przełamywania tendencji kryzysowych, środki i metody informatyki mogą nie tylko pośrednio wpływać na ogólny postęp gospodarczy, dzięki przyspieszeniu i zracjonalizowaniu procesu podejmowania decyzji, zmniejszaniu energochłonności procesów technologicznych, podniesieniu poziomu niezawodności i nowoczesności tysięcy urządzeń. Mogą być również przedmiotem bezpośredniego eksportu. Dotyczy to przede wszystkim, z uwagi na zły stan bazy elementowej, oprogramowania i usług konsultacyjno-serwisowych. Aby się tak stało trzeba odejść od tradycyjnego pojmowania przemysłu informatycznego i objąć nim również wytwarzanie oprogramowania sprzętu informatycznego. W tym miejscu zgadzam się całkowicie z tezą, że właśnie wytwarzanie oprogramowania jest w polskich warunkach szansą dla informatyki.

Rzeczywiste uznanie tego faktu musi przejawić się w potraktowaniu wytwarzania oprogramowania w kategoriach prawdziwego przemysłu. W szczególności należałoby:

- uznać wytwarzanie oprogramowania za równorzędne produkcji sprzętu informatycznego,
- stworzyć normy chroniące prawa autorskie twórców oprogramowania i regulujące obrót oprogramowaniem komputerów,
- utworzyć ośrodki produkcyjne (np. na zasadzie spółek z kapitałem zagranicznym) produkcyjno-wdrożeniowe, dysponujące nowoczesną (być może unikalną w kraju) techniką komputerową, umożliwiającą wykonywanie oprogramowania na eksport,
- utworzyć agendy handlowe wyspecjalizowane w promocji i handlu oprogramowaniem.

Pobieżna choćby analiza stanu informatyki krajowej prowadzi do wniosku, że mimo już prawie dwuletniej realizacji zacytowanego na wstępie zadania przesłanki powstania polskiego przemysłu wytwarzania oprogramowania nie powstały. Nadal jesteśmy bliżej **chałupnictwa** niż przemysłu. W tej chwili nie ma samodzielnego wyniku oprogramowania komputerów. W dalszym ciągu jest ono dodatkiem do produkowanego sprzętu informatycznego. Nie istnieje ani jedna poważna firma nastawiona na produkcję oprogramowania, nie powstał żaden naprawdę duży produkt oprogramowania podstawowego. [3]. Tymczasem na świecie istnieją i znakomicie prosperują tysiące dużych i małych firm, trudniących się wyłącznie produkcją oprogramowania.

Nie produkujemy obecnie, poza nielicznymi wyjątkami, konkurencyjnego sprzętu. Na szczęście niektóre uczelnie zapewniają dopływ bardzo dobrych programistów i projektantów. Trzeba to wykorzystać jak najprędzej. W tym miejscu chciałbym wreszcie wyjaśnić powody, dla których zdecydowałem się napisać ten odbiegający od ściśle informatycznych rozważań, felieton właśnie do „IKSa". Przede wszystkim chcę upowszechnić informację o tak sformułowanych zamierzeniach rządowych. Niezależnie od osiąganych postępów jest to dla informatyków fakt ważny. Po drugie uważam, że właśnie tysiące specjalistów i hobbystów informatyki stworzą wspólnie masę krytyczną, która doprowadzi do powstania nowoczesnego prawa chroniącego interesy wytwórców programów. Wreszcie po trzecie i najważniejsze: warto zawsze pamiętać, że informatyka jest jedną z najdynamiczniej rozwijających się dziedzin nauki i gospodarki. Dlatego bawiąc się lub profesjonalnie wykorzystując swoje mikro trzeba równolegle rozwijać głębszą wiedzę informatyczną, nadążać za nowymi technologiami organizacji działania i wykorzystania komputerów. Jednym słowem mając ATARI, czy COMMODORE, a w pracy wykorzystując IBM PC dobrze jest wiedzieć, jak działa i w jakich konfiguracjach jest wykorzystywany np. VAX. Wszystko po to, by z chwilą faktycznej budowy przemysłu wytwarzania oprogramowania być gotowym do podjęcia zadań wysokopłatnych i jednocześnie twórczych.

Bolesław SZAFRAŃSKI

---

[1] Potwierdzeniem niech będzie to, że ELWRO wraz z komputerem RtA 34 oferuje jedyny SZBD MADES, którego pierwowzór skonstruowano na początku lat siedemdziesiątych.

[2] Można wg uznania wstawić: ukradzionego, załatwionego, przyniesionego,

[3] Jedynym chwalebnym wyjątkiem jest chyba SZBD RODAN, sprzedany przez ZOWAR do wielu krajów.

SIECI

Gdy stwierdzisz, że gry komputerowe są do siebie podobne i coraz mniej atrakcyjne, że ZX Spectrum jako książka telefoniczna okaże się mniej użyteczny od tradycyjnej książki, a programowanie wyda się jednak nudne i mało ciekawe — wraca pytanie sprzed lat (jak ten czas leci!): jak można ciekawie wykorzystać personalny komputer w domu?

Wydaje się, że odpowiedzią na to pytanie jest podłączenie komputera do sieci telefonicznej.

W warunkach europejskich jest to przedsięwzięcie stosunkowo tanie i nie wymagające specjalnego przygotowania, np. wyposażenie naszego wysłużonego „Spectrum" w port RS 232 oraz modem kosztuje niecałe 100 funtów angielskich. W efekcie stajemy się posiadaczami terminala domowego, który może połączyć nasz komputer z każdym zainstalowanym choćby w Australii. Zasadność takiego połączenia wynika z usług, jakie są dostępne. Na przykład w Wielkiej Brytanii, komputer przez sieć telefoniczną pozwala na szybki dostęp do aktualnych danych o rozrywkach (kina, teatry), pracy, szkole, hobby, sporcie oraz o wielu innych bardzo ciekawych rzeczach.

Aby komputer stał się „oknem na świat" niezbędne jest posiadanie następujących elementów:

1. komputer personalny,
2. szeregowy port RS 232,
3. komunikacyjne oprogramowanie,
4. modem,
5. znajomość numerów telefonicznych, pod które podłączone są komputery udostępniające poszukiwane przez nas informacje.

Jak już wspomniałem, każdy komputer może być podłączony do sieci telefonicznej (nawet Spectrum lub Commodore), a korzystanie z tego typu sieci nie jest drogie — cena nieco mniejsza od rozmowy telefonicznej plus ewentualna opłata, gdy korzystamy z komercyjnych sieci komputerowych (dodatkowy abonament).

Po połączeniu z innym komputerem odnosimy wrażenie, że „komputer rozmówca" jest częścią naszego komputera. Fakt, że druty łączące nasze dwa komputery mają długość np.: 20 000 km, jest bez znaczenia dla procesu przesyłania informacji. Przebiega on podobnie jak wówczas, gdy komputery stoją obok siebie w tym samym budynku. Nie jest ważny także typ naszego komputera, działa on jako terminal współpracujący z dowolnym innym komputerem (tutaj mezaliansów nie ma). W efekcie pozwala to nam korzystać z mocy obliczeniowej innych maszyn, „przenoszenia" tej mocy do naszego komputera

Podstawowym elementem warunkującym realizację tego typu połączenia jest modem — małe urządzenie, które pośredniczy w przesyłaniu informacji między komputerem, a siecią telefoniczną. Modem zamienia wyjściowy sygnał komputera na sygnał akustyczny — postać sygnału, która może być przesyłana drutami telefonicznymi. Należy zatem włączyć modem do gniazdka telefonicznego, wykręcić odpowiedni numer, po usłyszeniu charakterystycznego sygnału adresata (modem przy drugim komputerze) należy nacisnąć odpowiedni klawisz w modemie — połączenie zostało wykonane — komputery mogą sobie „pogadać". Obecnie czynności te, dzięki odpowiedniemu oprogramowaniu, wykonuje sam komputer. „Samodzielność" ta wynika także z coraz doskonalszych konstrukcji modemów. Ma to miejsce na przykład podczas stosowania modemu HAYESa, który uzyskał w Polsce prawo do instalowania w komputerach korzystających z krajowej sieci telefonicznej.

Jak już wspomniałem, komunikacja komputerów połączonych siecią telefoniczną odbywa się za pośrednictwem wysyłanych drutami różnej wysokości tonów. W efekcie pisanie komend na naszej klawiaturze daje efekt w komputerze, z którym „rozmawiamy". Odpowiedzi komputera, do którego dzwoniliśmy, ukazują się na ekranie naszego mikro, tak jak gdyby generowane były przez nasz komputer

W tym miejscu może pojawić się wątpliwość — po co modem, komputer, skoro można zadzwonić i po prostu pogadać z osobą, której chcemy przekazać określoną informację. Odpowiedź jest prosta — mówiąc „przesyłamy" 29—30 liter w czasie jednej sekundy (zakładając, że mamy dobrą dykcję), komputery „gadają" z szybkością 1200 liter na sekundę. Dokonując pewnych rachunków, łatwo dojść do wniosku, że proponowany sposób przesyłania informacji jest znacznie tańszy. Szczególnie wyraźnie jest to widoczne przy porównywaniu kosztów „przesyłania znaków" przez ocean. Wykorzystuje się tutaj tak zwane paczki informacji. Komputer gromadzi określoną liczbę znaków i potem szybko je przesyła na przykład za pośrednictwem sputnika. Usługi takie oferuje Bell System Radio Link, który przesyła informacje z szybkością 1,9 biliona bitów na sekundę. Przesłanie naszej encyklopedii trwałoby około jednej setnej sekundy!

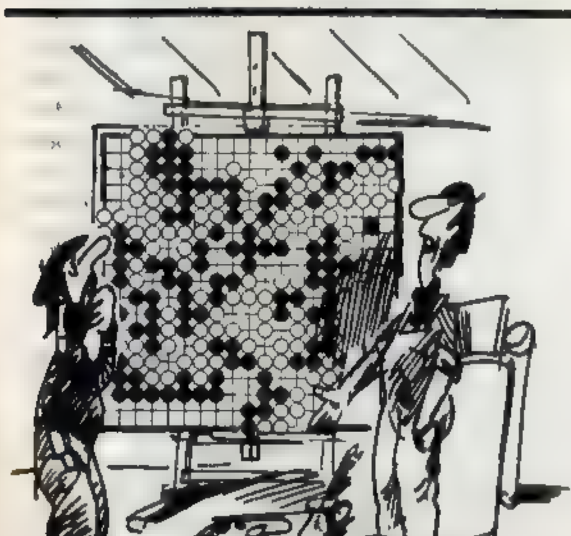
Wykorzystanie możliwości dołączania komputerów do sieci telefonicznej nie wymaga oczywiście umiejętności programowania. Dostępne są bowiem gotowe, tak zwane programy komunikacyjne

Obecnie na świecie dostępne są tysiące baz danych. Przykładami są: The Source w USA i Micronet w Wielkiej Brytanii. Bazy te zawierają setki darmowych programów, gier, informacji, wiadomości i wiele innych użytecznych danych, np. słowniki, encyklopedie.

Brytyjski system biblioteczny BLAISE udostępnia ewidencję wszystkich książek, periodyków, broszur itp. Dane te mogą być wybierane poprzez tytuł, temat, autora, wydawcę lub podając szereg innych parametrów

Korzystanie z komputera w domu staje się wielką przygodą, zabawą o „światowym wymiarze". Sądzę, że nie należy popadać w kompleksy, iż technika ta jest bardzo odległa od naszej domowej rzeczywistości Została bowiem uruchomiona także u nas w kraju sieć komputerowa (poprzez łącza telefoniczne — nie są takie złe!) Obecnie „gromadzi" ona użytkowników (jednym z nich jest piszący te słowa), a jej promotorem jest między innymi nasza konkurencja KOMPUTER, w którym FIDO — owa sieć — wielokrotnie był tematem publikacji

PRAKTYK



— A o sztuce komputerowej chyba słyszałeś?

Tekst

CPC

Ten program wyświetla teksty w specjalnej, wyróżnionej formie. Program, a właściwie podprogram rozpoczynający się od linii 70 należy uruchamiać z następującymi parametrami:

a$ — tekst do wyświetlenia
x, y — współrzędne początku wyświetlanego tekstu
s — numer pióra (PEN) tła tekstu
t — numer pióra znaków tekstu

```
10 MODE 1:INK 0,10:INK 3,0
20 LOCATE 12,8:PRINT "Tekst 1"
30 s=3:t=1:x=8:y=12
40 a$="Tekst 2"
50 GOSUB 70
60 t=0:a$="Tekst 3":x=12:y=18:GOSUB 70:PEN 1:CALL &BB18:STOP
70 REM **WYROZNIONY TEKST**
80 PRINT CHR$(22)CHR$(1)CHR$(23)CHR$(3):x=(x-1)*16:gy=398-(y-1)*16:PLOT -2,-2,s:PEN t
90 TAG:MOVE gx-2,gy:PRINT a$:MOVE gx+2,gy:PRINT a$:MOVE gx,gy+2:PRINT a$:MOVE gx,gy-2:PRINT a$;:TAGOFF
100 LOCATE x,y:PRINT a$CHR$(22)CHR$(0)CHR$(23)CHR$(0):RETURN
```

CHIŃSKA KRZEMOWA DOLINA

Chińskie mikrokomputery nie odbiegają od światowego standardu. Wkrótce mogą się stać konkurentami IBM'a i APPLE'a — pisze J. Maier w ostatnim numerze „BYTE'a" z ubiegłego roku.

Chińska Republika Ludowa już od 1984 roku ma swój własny superkomputer GALAXY (100 milionów operacji na sekundę), a od 1980 roku macierzowy procesor 100—MIPS 757. Chińczycy wytwarzają także własne mikrokomputery, których produkcję szacuje się na około 70 000 sztuk, w tym 25 000 jest zgodne z IBM PC XT. Wytwarza je fabryka na przedmieściach Beijng — CHINA COMPUTER DEVELOPMENT CORP (CCDC). CCDC produkuje także wielodostępny system BCMS 6800 bazujący za Motoroli 68000 oraz UNIX'ie. Maszyna ta jest ewidentnym dowodem dużej popularności UNIX'a w ChRL. Chińska Akademia Nauk rozpoczęła publikowanie tłumaczonych materiałów UNIX'a już w 1984 roku. W tym samym czasie zainstalowano w uniwersytetach i innych chińskich instytucjach ponad 2000 dużych zachodnich komputerów pracujących pod UNIX'em.

CHIŃSKA DOLINA KRZEMOWA

Podczas gdy liczba zainstalowanych w ChRL komputerów nie przekracza miliona (roczna produkcja USA), wzrost chińskiego potencjału informatycznego jest wciąż wykładniczy. Należy pamiętać, że jeszcze kilka lat temu potencjał ten był niewielki.

Minęło zaledwie 10 lat od czasu, gdy pierwszy mikrokomputer został zbudowany w kalifornijskiej Dolinie Krzemowej. Obecnie, za Oceanem, Szanghai i Beijng budują własne Doliny Krzemowe. Przetłumaczono dokładnie nawet ten zwrot na język chiński, brzmi on GUIGOU. Chiński potencjał domowych komputerów ciągle rośnie. Należy się spodziewać, że zaowocuje to pojawieniem się około 1990 roku na rynkach Trzeciego Świata komputerów o światowym standardzie „Made in China".

Szereg ostatnich zdarzeń sygnalizuje ten rozwój. Chińscy przywódcy zdecydowali się nie na import, lecz na własną produkcję. Chińczycy nie są zainteresowani w importowaniu mikroprocesorów, tak jak przed kilku laty, ale kupowaniem technologii. Potwierdzają to ostatnio podpisane umowy z takimi firmami jak: IBM, Hewlett-Packard, Sperry, Wang, Gould, Burroughs, ComputerLand, Solar i inne. Obecnie strategia chińskiego rozwoju informatyki to: *„pierwsza maszyna jest importowana, druga maszyna jest budowana w Chinach, trzecia maszyna jest eksportowana"*.

Opracowanie pt.: „Technology Transfer to China" spowodowało dyskusję nad zaniechaniem ograniczeń w eksporcie nowoczesnych technologii do Chin przez COCOM, Japończyków i większość członków NATO, co może sprawić, że Chiny będą ważnym partnerem państw zachodnich.

CHIŃSKIE POTRZEBY WOBEC WŁASNYCH MOŻLIWOŚCI

W czerwcu 1987 roku w Beijng odbyła się druga międzynarodowa konferencja „Komputery i ich zastosowania", której sponsorami były: amerykańska IEEE Computer Society i Chińska Federacja Komputerowa (CFK), a honory gospodarza konferencji pełniła Chińska Akademia Nauk. W konferencji uczestniczyli reprezentanci 17 krajów z Europy, Azji, USA, Kanady i Australii. Kraje te wydelegowały 300 naukowców informatyków. Zaprezentowali oni 160 referatów, a spośród 100 chińskich naukowców ponad 60 wygłosiło referaty na różne tematy — od lokalnych sieci do mikrokomputerów bazujących na wielojęzykowych systemach przetwarzania słów. Pakiet takiego systemu pracującego na PC XT pod chińską wersją DOSa (z czterema chińskimi dialektami) przedstawił profesor Xinjiang.

Chiny posiadają 170 000 profesjonalnych komputerów i należy oczekiwać przed rokiem 2000 wzrostu tej liczby do miliona. Jednocześnie produkowany będzie milion mikrokomputerów przed rokiem 1990, wobec 300 000 w roku 1987 (połowa importowana). Osiągnięcie tego celu wymaga 500-procentowego wzrostu produkcji w ciągu 3 lat.

Chińska Federacja Komputerowa jest doskonale zorganizowana. Założona w 1985 roku liczy obecnie 25 000 członków ze wszystkich prowincji Chin. CFK jest czynnym organizatorem krajowych i międzynarodowych konferencji, publikuje osiem profesjonalnych periodyków (700 000 egzemplarzy). Komitety Edukacji i Popularyzacji tej federacji skutecznie upowszechniają wiedzę informatyczną wśród miliarda Chińczyków. Komitety te przygotowują 50 różnych samouczków, kompletują książki popularyzujące wiedzę informatyczną oraz wydają specjalne serie dla nastolatków.

Te i wiele innych działań CFK efektywnie zrównuje chińską informatykę ze światowym poziomem produkcji i zastosowań komputerów. Dotyczy to także oprogramowania „Made in China", które w niedługim czasie pojawi się na światowym rynku. Na przykład technologia budowy systemów typu ekspert jest obecnie dobrze znana w Chinach. Ilustruje to między innymi MYCIN — komputerowy system chińskiej tradycyjnej diagnostyki medycznej (jest on powszechny w Chinach od kilku lat).

Chiny ulegają metamorfozie w stylu Japonii — pod względem ekonomicznym i technologicznym. ChRL jest jedną z 10 potęg ekonomicznych świata, których bogactwo rośnie od 5 do 7 proc. rocznie, w 2000 roku oczekuje się, że dochód narodowy Chin wyniesie 1,5 tryliona dolarów.

Wielu ekonomistów uważa, że XXI wiek będzie stuleciem KRAJÓW PACYFIKU.

W.G.

Opracowane na podstawie: The State of Chinese Computing, John H Maier, BYTE vol. 12 No. 14, Dec. 1987.

Piraci końca XX wieku

Świat już jakby nieco ochłonął po wyścigu o najnowocześniejsze i najsprawniejsze komputery. Coraz częściej za to obserwujemy rywalizację w dziedzinie oprogramowania. I jest to zupełnie naturalna kolej rzeczy — po udoskonaleniu maszyn przyszła pora na znalezienie sposobu optymalnego ich wykorzystania. Szybko jednak okazało się, że rywalizacja ta może posłużyć do celów związanych nie tylko z rozwojem komputeryzacji.

Powszechnym zjawiskiem stało się podrabianie wyrobów znanych i co ważniejsze — uznanych firm. Małe wytwórnie, przeważnie z Dalekiego Wschodu, produkują „oryginalne" zegarki, perfumy, aparaty fotograficzne i inne towary, bezkarnie na ogół wykorzystując szyld renomowanych marek. Czerpią z tego nielegalnego procederu ogromne zyski, narażając prawdziwego producenta nie tylko na straty finansowe, ale także utratę zaufania klientów, jako że falsyfikaty są zazwyczaj bardzo niskiej jakości. Ostatnio, oprócz tradycyjnie już fałszowanych towarów, rozkwita handel nielegalnie nagranych kaset wideo i oprogramowania komputerowego. No cóż, piraci nie próżnują, zmieniają się tylko obiekty ich zainteresowań.

Dostrzegając ten problem, a także odczuwając realne już straty finansowe, największe amerykańskie firmy zajmujące się programami komputerowymi postanowiły połączyć siły w walce z piratami na obcych rynkach. Rynki te stanowią ważną pozycję na liście klientów, popyt na amerykańskie oprogramowanie jest bowiem często większy właśnie za granicą niż w kraju. Składa się na to wiele przyczyn, wśród których należy wymienić duże już nasycenie rynku wewnętrznego. Spadek wartości dolara obserwowany w ostatnim czasie, także sprzyja zwiększeniu zysków i rozwijaniu współpracy na nowych rynkach.

Współpraca ta pociąga jednak za sobą niebezpieczeństwo dokonywania nielegalnych kopii. Aby temu przeciwdziałać pięć liczących się na rynku amerykańskim firm — Microsoft, Lotus, Ashton-Tate, Autodesk i Wordperfect-Apple utworzyło stowarzyszenie chroniące ich interesy — Business Software Association (BSA). Większość z tych firm podejmowała już próby walki z piractwem, jednakże teraz po raz pierwszy przystąpiły do wspólnego działania mając nadzieję na osiągnięcie skuteczniejszych rezultatów.

Neuronowy komputer

Czy możliwe jest skonstruowanie komputera działającego na podobnej zasadzie, jak mózg człowieka? Na obecnym etapie rozwoju techniki jest to niemożliwe, ale wiele firm prowadzi prace badawczo-konstrukcyjne nad komputerem, którego działanie oparte byłoby na podobnej zasadzie, jak funkcjonowanie mózgów tak nieskomplikowanych zwierząt, jak ślimaki lub żaby.

Na czym polega różnica w działaniu tradycyjnego komputera i komputera nietypowego, tzw. sieci neuronowej? Komputery tradycyjne pracujące zgodnie z zasadą von Neumanna mają zwykle jeden procesor, jeżeli posiadają więcej niż jeden, to są one połączone liniowo za pomocą odpowiedniego złącza (bus'a). W tego typu pracujących równolegle komputerach nie może być połączonych więcej niż 4 procesory. Mózg zawiera zdecydowanie więcej połączeń równoległych. W przeciętnym mózgu znajduje się około 4 milionów komórek nerwowych — neuronów, z których każda pracuje jak oddzielny procesor lub układ logiczny. Są one połączone ze sobą za pomocą tysięcy łączy zwanych synapsami. Gdy komputery pracujące według zasady von Neumanna wykonują działania ściśle określone za pomocą instrukcji, to sieć neuronowa mózgu obrabia informację poprzez zestawienie odpowiednich połączeń między neuronami. Konstruktorzy sieci neuronowej starają się skopiować tę zasadę działania za pomocą elementów symulujących. Do tego celu nie jest potrzebne specjalne oprzyrządowanie. Większość procesów symulacyjnych przeprowadzana jest za pomocą konwencjonalnych procesorów i pamięci RAM na specjalnie wykonanych tablicach, które następnie umieszczane są w obudowach zwykłych komputerów, takich jak np. IBM PC.

Kluczem do sieci neuronowej jest zastosowane oprogramowanie symulacyjne, które wykorzystuje standardowe elementy sprzętu, jako ciąg dyskretnych procesorów lub neuronów, ze zmienną liczbą połączeń — synaps między nimi.

Oprogramowanie określa rozmiar sieci (symulacja neuronu może być wykonana za pomocą kilku bitów pamięci) oraz ilość połączeń pomiędzy neuronami. Rozmiar sieci, a w szczególności ilość połączeń pomiędzy neuronami, decyduje o mocy i możliwościach całego systemu. Jest to zasadnicza różnica między siecią neuronową a tradycyjnym komputerem.

Zasady działania sieci neuronowej podobne są do zasad funkcjonowania mózgu. Sieć uruchamia największą liczbę neuronów i zestawia struktury różnych jednoczesnych połączeń między nimi. Każdej informacji odpowiada określona struktura połączeń między neuronami. Informacja jest przetwarzana jako ciągi takich struktur.

Jest to bardziej elastyczny sposób analizowania niektórych typów informacji, w szczególności, gdy informacja jest zamazana lub nieczytelna, jak w wypadkach, systemów sztucznej inteligencji.

Gdy sieć ma dużą liczbę połączeń między neuronami, komputer może samoczynnie „nauczyć się" danych początkowych przez kilkakrotne powtarzanie ciągów struktur możliwych połączeń między neuronami. Dane nie muszą być ściśle precyzowane, zanim zostaną „wyuczone", jak jest to wymagane w tradycyjnych komputerach. W ten sposób mogą być tworzone łączenia różnych typów informacji w sposób o wiele bardziej elastyczny bez konieczności ograniczenia ich do kilku kluczowych cech.

W obliczeniach komputery sieci neuronowej są wolniejsze od tradycyjnych. Można także wykorzystać w ich pracy oprogramowanie standardowe. Jest jednak mało prawdopodobne, aby komputery te zastąpiły już teraz tradycyjne konstrukcje. Dzięki możliwości szybkiego przetwarzania danych nieczytelnych, zamazanych lub niejednoznacznych, posiadaniu pamięci skojarzeniowej i umiejętności uczenia się, a nie korzystania ze zbiorów danych, komputery te znajdą zastosowanie w systemach sztucznej inteligencji do rozpoznawania mowy i języków oraz analizowania obrazów dynamicznych np. zarejestrowanych na taśmach wideo. Wykorzystanie do takich celów komputerów tradycyjnych jest przeważnie bardzo trudne lub niemożliwe ze względu na ich małe możliwości „myślenia" abstrakcyjnego.

W najbliższych latach przewiduje się gwałtowny rozwój komputerów sieci neuronowej, dzięki którym znacznie powiększą się możliwości systemów sztucznej inteligencji. Najbardziej zaawansowane prace w tym kierunku prowadzone są w Stanach Zjednoczonych. Jak do tej pory wyniki tych prac objęte są ścisłą tajemnicą i niewiele firm zdecydowało się na zaprezentowanie swoich osiągnięć. Według nieoficjalnych informacji amerykańskie firmy TRW i IBM zbudowały procesory dla tych komputerów mające możliwość zestawienia ponad 500 000 połączeń. Firma Synaptics z San Diego w Kalifornii konstruuje układ elektroniczny do sieci neuronowej symulującej system wzrokowy żaby dla przetworzenia danych wizyjnych z oka umieszczonego na końcu ramienia robota. Firma Neuralware z Pensylwanii wypuściła zestaw oprogramowań różnych konfiguracji sieci neuronowej, możliwych do stosowania na IBM PC XT z pamięcią co najmniej 512 KB RAM pozwalających użytkownikowi dowolnie eksperymentować z dziewięcioma rodzajami sieci neuronowej. W firmie HNC skonstruowano przystawkę sieci neuronowej ze specjalną pamięcią 4 Mbajtową, procesorem centralnym MC 68020 i wspomagającym MC 68881, mogącą współpracować z komputerem IBM PC lub kompatybilnym. Jej koszt — 10 000 $.

Firma zbudowała także urządzenie — Nestor Writer, które na podstawie sieci neuronowej weryfikuje prawidłowość podpisów na czekach. Koszt tego urządzenia — 1535 $.

Komputery sieci neuronowej dopiero zaczynają swoją karierę. Ale nawet te największe obecnie istniejące zawierają zaledwie drobny fragment połączeń możliwych w mózgu organicznym. Do skonstruowania sztucznego mózgu w postaci kości elektronicznej jest jeszcze daleka droga. Nadal zbyt mało wiemy o mózgu, a kolejne próby zbudowania coraz doskonalszej sieci neuronowej stwarzają więcej problemów, niż rozwiązują.

MBO

Pierwszym udanym osiągnięciem BSA było wykrycie działalności fałszerzy w Hongkongu. Akcja doprowadziła do aresztowania 10 osób zajmujących się tym procederem oraz konfiskaty ponad 2 milionów dolarów uzyskanych ze sprzedaży nielegalnie skopiowanych programów. Sukces ten zwrócił uwagę innych firm także myślących o wspólnym zwalczaniu piractwa.

Mimo że stowarzyszenie działa już od roku, dopiero niedawno zostało formalnie zarejestrowane. Podczas pierwszego walnego zebrania, które odbyło się w San Francisco, ustanowiono statut oraz wytyczono główne kierunki działań. Zapadły także decyzje w sprawie liczebności członków — uznano, że BSA będzie bardziej aktywne i efektywne w działaniu, jeśli pozostanie organizacją niedużą. Dlatego postanowiono przyjąć tylko trzech nowych kandydatów, spośród takich firm jak Aldus, Adobe, Micropo lub Microrim. Ograniczenie liczby członków ma także na celu skupienie uwagi na przeciwdziałaniu piractwu na wielką skalę, bez angażowania się w drobniejsze akcje, które mogą prowadzić na własną rękę inne, mniejsze firmy.

Działalność BSA polega na zwalczaniu nielegalnego handlu oprogramowaniem. Podejmowane są więc konkretne działania. Chodzi tu przede wszystkim o zebranie dostatecznej liczby dowodów obciążających „podejrzaną" firmę zagraniczną i dostarczenie tych dowodów organom miejscowej policji bądź przedstawicielom władz celnych. Kompletowaniem dowodów zajmują się wynajęci przez stowarzyszenie prywatni detektywi. Oczywiście tego typu akcje kosztują. Ale BSA nie szczędzi środków, ponieważ straty powstałe w wyniku działalności piratów wielokrotnie przewyższają nakłady przeznaczone na ich zwalczanie.

Dlaczego jednak wielkie firmy amerykańskie same musiały zająć się ściganiem fałszerzy? W wielu krajach brak uregulowań prawnych problemów związanych z informatyką, w tym kwestii praw autorskich i ochrony prawnej ich dzieła — programu komputerowego. W Hongkongu, Kanadzie i krajach Commonwealthu zainteresowane firmy musiały same podjąć działania w celu wytropienia piratów, w przeciwnym bowiem wypadku nie mogłyby dochodzić swych praw i wyrównać strat. Mimo deklaracji poparcia ze strony rządów krajów o „podejrzanych" rynkach, tak jak w przypadku Hongkongu, zbieranie dowodów przez wynajętych detektywów trwało całe miesiące, zanim władze zdecydowały się także przystąpić do akcji.

Determinację firm stowarzyszonych w BSA najlepiej uzasadnia wysokość strat. W czasie śledztwa w Hongkongu wyszło na jaw, że piraci sprzedawali niektóre programy nawet po 2 dolary, podczas gdy ich oficjalna cena wynosi 500 dolarów! Warto więc ponosić nawet wysokie koszty i angażować się w zakrojone na szeroką skalę akcje, zwłaszcza że pirackie firmy są na swoim terenie niemal zupełnie bezkarne.

Małgorzata NOWOTNY

GRAFIKA — wycinanie i zakrywanie

Tomasz MROWIEC, Ludwik PIELA

Dowolnie skomplikowany rysunek możemy utworzyć za pomocą dwóch podstawowych instrukcji: **PLOT X,Y** i **DRAWTO X,Y**. Liczba wyświetlanych punktów i linii zależy od tego, ile razy użyjemy tych instrukcji. Kształt uzyskanego rysunku zależy od wartości X i Y, których będziemy używać podczas kreślenia. Najczęściej stosowane są dwa podstawowe sposoby otrzymania wartości współrzędnych. Pierwszy z nich polega na wyliczaniu wartości współrzędnych na podstawie różnorodnych wzorów matematycznych. Używany jest głównie wtedy, gdy możemy dobrać odpowiedni wzór dla uzyskania żądanego kształtu lub kreślimy wykresy funkcji zadanych w postaci wzoru. Drugi sposób polega na wykorzystaniu do kreślenia wartości współrzędnych zadanych w postaci jawnej. Wymaga to zwykle wcześniejszego wprowadzenia ich do odpowiedniej tablicy lub tablic. Po bliższym przyjrzeniu okaże się, że w momencie kreślenia oba sposoby są identyczne. Punkt lub odcinek prostej tworzymy na podstawie wartości współrzędnych istniejących w postaci jawnej. Różnice polegają na sposobach przygotowania tych wartości. W pierwszym zadane są w sposób niejawny, w drugim — jawny. Rezultat jest podobny — w obszarze pamięci komputera, zwanym pamięcią ekranu, zostają ustawione odpowiednie bity. Na podstawie zawartości tego obszaru pamięci ANTIC wyświetla utworzony rysunek na ekranie monitora lub telewizora (patrz „IKS" nr 7—8 z 1987 roku).

Każdy z przedstawionych sposobów ma swoje zalety i wady. W pierwszym z nich cały ciężar wyznaczenia wartości współrzędnych punktów przerzucamy na komputer, który na podstawie dosyć lakonicznego zapisu podanego wzoru wyznacza setki, a nawet tysiące wartości, a następnie wykreśla je na ekranie. Program realizujący to zadanie wymaga zwykle mało pamięci.

Drugi sposób wymaga wcześniejszego przygotowania wartości współrzędnych kreślonych punktów lub linii. Najprostszym sposobem ich uzyskania jest wykreślenie rysunku na papierze milimetrowym, a następnie określenie współrzędnych charakterystycznych punktów i wprowadzenie ich do programu. Sposób jest dosyć żmudny, umożliwia jednak uzyskanie dowolnych kształtów. Dla programów w języku BASIC dane o rysunku umieszczane są zwykle w liniach DATA. Zaletą takiego sposobu przygotowania danych jest zachowanie wartości, na których podstawie wykreślono rysunek. Umożliwia to nie tylko szybkie odtworzenie rysunku, lecz także wykonanie pewnych manipulacji całym rysunkiem lub jego częścią.

Ważny jest sposób przechowywania danych o rysunku. Dla prostych rysunków można przechowywać współrzędne punktów w tablicach jednowymiarowych w takiej kolejności, w jakiej będą później połączone. W wypadkach bardziej złożonych przez jeden punkt może przechodzić kilka linii lub mogą istnieć fragmenty rysunku nie połączone liniami z resztą. Dlatego dane o rysunku wygodnie jest przechowywać w tablicach dwuwymiarowych. Pierwszy indeks elementu tablicy oznacza numer fragmentu rysunku, a drugi wskazuje numer punktu w tym fragmencie. W ten sposób para wartości (X (2,1), Y(2,1)) określa współrzędne pierwszego punktu drugiego fragmentu rysunku.

Wprawdzie rysunek zapamiętany jest również, bez względu na sposób tworzenia, w pamięci ekranu, jednak manipulowanie nim jest dosyć trudne. Przyczyną tego jest sposób odwzorowania w pamięci.

Przekształcenia mogą być wykonywane na całym rysunku lub na określonych jego fragmentach (szerzej napiszemy o tym w jednym z tegorocznych zeszytów programów komputerowych). Fragmenty te mogą być wydzielone

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Program nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
BK 20 GRAPHICS 0:POKE 752,1
LB 30 DIM X(15),Y(15),M$(1)
GK 40 XM=319:YM=159
BE 60 REM ** ODCZYT DANYCH WYKRES
   U **
ID 70 FOR I=1 TO 13
HQ 80 READ X,Y
WX 90 X(I)=X:Y(I)=Y
FR 100 NEXT I
TK 120 GOSUB 160
TX 130 GOSUB 260
QV 140 GOTO 770
XA 160 REM ** WYKRESLANIE **
AG 180 GRAPHICS 8:COLOR 1
XV 190 PLOT X(1),Y(1)
SW 200 FOR I=2 TO 13
ZQ 210 DRAWTO 2*X(I),Y(I)
FW 220 NEXT I
ZG 240 RETURN
LS 260 REM ** DODANIE OKNA **
KE 280 ? CHR$(125);"SRODEK OKNA (
   X,Y)";
KE 290 TRAP 280:INPUT XO,YO
EO 330 ? CHR$(125):? "WYSOKOSC I
   SZEROKOSC OKNA";
CM 340 TRAP 330:INPUT OW,OS
JC 350 TRAP 40000
NB 360 L=XO-OS/2
NF 370 P=XO+OS/2
OF 380 G=YO-OW/2
NA 390 D=YO+OW/2
YJ 400 IF L>P OR L<0 OR P>XM OR G
   >D OR G<0 OR D>YM THEN 330
GH 450 PLOT L,G
DB 460 DRAWTO P,G
BN 470 DRAWTO P,D
WW 480 DRAWTO L,D:DRAWTO L,G
TV 500 ? CHR$(125);"K--KONIEC","Z
   --ZMIANA OKNA"
EP 510 INPUT M$
HJ 520 IF M$="K" OR M$="k" THEN 7
   70
PQ 590 REM ** KASOWANIE OKNA **
XD 600 COLOR 0
GB 610 PLOT L,G
CV 620 DRAWTO P,G
BH 630 DRAWTO P,D
WQ 640 DRAWTO L,D:DRAWTO L,G
XY 650 COLOR 1
SI 680 ? CHR$(125);"CZY ODTWORZYC
    RYSUNEK (T/N)"
FG 690 INPUT M$
EK 720 IF M$="N" OR M$="n" THEN 2
   60
VC 730 GOSUB 190
OS 740 GOTO 260
ZN 750 RETURN
OK 770 END
JV 1000 DATA 0,159,10,120,20,60,3
   0,50,40,100,50,120,60,100,70,3
   0,80,40,90,100,100,120,110,110
   ,120,159
```

w specjalny sposób, mogą być usunięte z ekranu lub zachowane i przekształcone, a jednocześnie reszta rysunku będzie usunięta.

Często wymagane jest wyróżnienie pewnych elementów rysunku w celu skupienia na nich uwagi. Elementy te mogą być narysowane innym kolorem lub wykreślone liniami o zwiększonej jaskrawości. Taki sam efekt można uzyskać poprzez wykreślenie w wydzielonym obszarze linii podwójnej grubości: zamiast linii wykreśla się dwie linie w odległości jednego kroku rastra. Innym sposobem zwracania uwagi na wybrany obszar jest otoczenie go ramką, okrągłą lub prostokątną.

W celu nałożenia na rysunek okręgu wystarczy określić współrzędne jego środka (XC, YC) i promień R. Do programu należy dodać podprogram kreślenia okręgu, do którego zwracamy się za każdym razem, kiedy potrzebujemy wydzielić żądany obszar. Można napisać program, który pozwoli eksperymentować z położeniem i rozmiarami okręgu dla uzyskania najlepszego efektu.

Podczas używania, do wydzielania obszaru ekranu, prostokątnej ramki można określać wszystkie cztery lub tylko dwa przeciwległe rogi prostokąta. Można także napisać program, który prosi o podanie współrzędnych środka prostokąta i jego rozmiarów. Przykładem może być program nr 1, który wykreśla prostokąt o środku w punkcie *(XB, YB)* szerokości *W* i wysokości *H*.

Oprócz wyróżniania określonych obszarów ekranu, niekiedy trzeba usunąć części utworzonych rysunków. W takich wypadkach można wykorzystać metody wydzielania obszarów dla identyfikacji fragmentów rysunku, które należy zakryć (usunąć) lub wyciąć. W pierwszym wypadku usuwamy rysunek ze wskazanego obszaru ekranu, w drugim — zachowujemy rysunek tylko w tym obszarze, a usuwamy resztę.

Sposoby wydzielania obszaru ekranu, za pomocą okręgu lub prostokąta, można wykorzystać do określenia ścieranego obszaru. Potrzeba usuwania części rysunku może wynikać z wielu powodów. Na przykład chcemy uprościć złożony rysunek lub uwolnić miejsce na wyświetlenie dodatkowej informacji. Operacje te wygodnie jest wykonać podczas dobierania formatu wyświetlania rysunków w celu uwolnienia miejsca na ekranie podczas powiększenia lub innych przekształceń elementów rysunku.

Wydzielone fragmenty rysunku można oczywiście usunąć, zmieniając program tak, aby rysunek tworzony był bez nich. Prowadzi to do istotnych zmian w programie graficznym. Jeśli później chcemy odtworzyć usunięte fragmenty, program znowu trzeba będzie przerabiać. Jeśli trzeba często usuwać fragmenty rysunku, można specjalnie opracować uniwersalny program do kasowania całej informacji wewnątrz zadanego obszaru.

Wykorzystanie prostokąta do identyfikacji oczyszczonego obszaru umożliwia łatwe wycieranie znaków, punktów i części linii, znajdujących się wewnątrz prostokąta. Trzeba tylko zapełnić ten obszar liniami w kolorze tła.

Można czyścić także wewnątrz granic okręgu. Jednak w tym wypadku program będzie działać wolniej, ponieważ należy obliczać współrzędne punktów okręgu, które następnie wykorzystuje się do wskazania końców odcinków linii zapełniających wewnętrzny obszar okręgu.

W pewnych wypadkach należy przechować część rysunku i wykasować resztę. W tym celu wybrany obszar ekranu zamyka się w prostokątną ramkę lub okno. Kasowanie elementów rysunku na zewnątrz okna nazywamy wycinaniem.

Program, powodujący wycinanie dowolnego zobrazowania, powinien znajdować i przechowywać wszystkie napisy, oddzielne punkty i części linii mieszczące się w obszarze okna. Analizuje on współrzędne końców odcinków i położenie napisów w celu określenia przechowywanych elementów rysunku.

Przede wszystkim rozważmy działanie programu wycinania dla punktów i odcinków linii prostych. Będzie on wprowadzał współrzędne lewego górnego rogu okna *(XW, YW)* szerokość *WW* i wysokość okna *HW*. Współrzędne elementów rysunku przechowujemy w tablicach *X* i *Y*.

Oddzielny punkt (nie będący częścią linii) przechowywany jest przez program wycinania, jeśli znajduje się wewnątrz okna, tj. jeśli jego współrzędna *X* leży w przedziale od *XW* do *XW + WW* a współrzędną *Y* — w przedziale od *YW* do *YW+ HW*. Odcinek prostej jest przechowywany, jeśli oba jego końce znajdują się wewnątrz obszaru okna. Linia usuwana jest w całości. Jeśli wszystkie jej punkty leżą na zewnątrz okna. Jeśli część linii znajduje się na zewnątrz okna, jest odcinana. Przechowywana jest tylko ta część linii, która znajduje się wewnątrz.

Program nr 2 rozpoczyna działanie od analizy współrzędnych względem lewej granicy okna. Punkty, których współrzędne poziome przewyższają wartość *X* lewej granicy okna, zapamiętywane są w tablicach *X1* i *Y1*. Punkty przecięcia linii z lewą granicą ekranu także zapamiętywane są w tablicach *X1* i *Y1*. Następnie współrzędne z tablic *X1* i *Y1* porównywane są z położeniem górnej granicy okna.

Punkty przecięcia i wszystkie punkty izolowane, których współrzędna pionowa przewyższa wartość *Y* górnej granicy okna, zapamiętywane są w tablicach *X2* i *Y2*. Pozostałe punkty odnoszone są do prawej granicy okna. Tablice *X1* i *Y1* wykorzystywane są powtórnie do przechowywania współrzędnych punktów przecięcia i punktów, których współrzędna pozioma nie przewyższa współrzędnej *X* prawej granicy. W końcu, współrzędne punktów z tablic *X1* i *Y1* porównywane są z położeniem dolnej granicy okna. Tablice *X2* i *Y2* wykorzystuje się do przechowywania końcowej informacji o punktach i odcinkach linii, które powinny być zobrazowane po oczyszczeniu ekranu.

Rozszerzymy teraz program nr 2, aby zapewnić wykonanie operacji wycinania przy obecności napisów. Można przetwarzać napisy tak jak linie, to jest przechowywać tę część napisu, która znajduje się wewnątrz okna. Jednak dla uproszczenia programu będziemy przechowywać tylko te opisy, które całkowicie

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Prosram nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
AC 160 GRAPHICS 8:COLOR 1
OU 170 DIM X(8,20),Y(8,20),X1(8,2
   0),Y1(8,20),X2(8,20),Y2(8,20),
   NE(8),M$(1),ME(20)
SJ 180 XM=319
TK 190 YM=159
BH 195 REM *******************
WQ 200 REM ODCZYT WSPOLRZEDNYCH
AQ 205 REM *******************
XJ 210 READ N
FN 220 FOR P=1 TO N
RB 230 READ NE:NE(P)=NE
UZ 240 FOR E=1 TO NE
SP 250 READ X,Y:X(P,E)=X:Y(P,E)=Y
EQ 260 NEXT E
IY 270 NEXT P
EZ 280 GRAPHICS 8:COLOR 1:POKE 75
   2,1
RN 290 GOSUB 340:REM KRESLENIE RY
   SUNKU
MM 300 GOSUB 450:REM OKRESLENIE O
   KNA
VP 305 IF M$="Z" THEN 330
YC 310 GOSUB 850:REM WYCINANIE OK
   NA
ZH 320 GOSUB 2480:REM KRESLENIE W
   YCIETYCH PUNKTOW
CY 330 IF PEEK(764)=255 THEN 330
XO 335 POKE 764,255:GOTO 280
BG 338 REM *******************
UR 340 REM PODPROGRAM KRESLENIA
AZ 345 REM *******************
RH 350 ? #6;CHR$(125);
FW 360 FOR P=1 TO N
CP 370 IF NE(P)>1 THEN 400
BD 380 PLOT X(P,1),Y(P,1)
OL 390 GOTO 430
IG 400 FOR E=1 TO NE(P)-1
EQ 410 PLOT X(P,E),Y(P,E):DRAWTO
   X(P,E+1),Y(P,E+1)
EK 420 NEXT E
IS 430 NEXT P
ZI 440 RETURN
ZA 445 REM ***************
HN 450 REM OKRESLENIE OKNA
ZC 455 REM ***************
LP 460 ? CHR$(125);
PL 465 ? "PARAMETRY OKNA:"
IX 470 ? "Lewy gorny rog (x,y):";
XY 480 TRAP 460:INPUT XW,YW
ZE 520 ? "Wysokosc i szerokosc:";
KU 530 TRAP 490:INPUT HW,WW
LM 540 ? CHR$(125);
JE 550 TRAP 40000
RV 560 L=XW
AF 570 R=XW+WW
TU 580 T=YW
RU 590 B=YW+HW
WS 600 IF L>=R OR L<0 OR R>XM OR
   T>=B OR T<0 OR B>YM THEN 460
ZT 660 PLOT L,T:DRAWTO L,B
BL 670 DRAWTO R,B
LF 680 DRAWTO R,T
IN 690 DRAWTO L,T
SD 700 ? "K - kasowanie ramki W -
    wycinanie              Z - zakr
   ywanie";
ER 710 INPUT M$
XA 720 IF M$="Z" OR M$="z" THEN G
   OSUB 5000:RETURN
TM 730 IF M$="W" OR M$="w" THEN R
   ETURN
ZB 735 REM ***************
JG 740 REM KASOWANIE RAMKI
ZD 745 REM ***************
LQ 750 ? CHR$(125);
XQ 760 COLOR 0
ZY 780 PLOT L,T:DRAWTO L,B
BQ 790 DRAWTO R,B
KR 800 DRAWTO R,T
HZ 810 DRAWTO L,T
XU 820 COLOR 1
PA 830 GOTO 450
VL 840 REM **********************
WE 850 REM WYCINANIE WZGLEDEM
DH 860 REM LEWEGO BRZEGU
WE 865 REM **********************
DZ 870 REM W X1 I Y1
XD 880 P1=1
GH 890 FOR P=1 TO N
UD 900 E1=0
IN 910 FOR E=1 TO NE(P)-1
JE 930 IF X(P,E)<L THEN 1020
QB 940 REM PIERWSZY PUNKT JEST W
   SRODKU
DF 950 E1=E1+1
RP 960 X1(P1,E1)=X(P,E)
BL 970 Y1(P1,E1)=Y(P,E)
YI 980 REM CO Z DRUGIM PUNKTEM?
DF 990 IF X(P,E+1)>=L THEN 1070
BS 1000 GOSUB 1190
QE 1010 GOTO 1070
OO 1020 REM PIERWSZY PUNKT JEST P
   OZA
SE 1030 REM CO Z DRUGIM PUNKTEM?
UQ 1050 IF X(P,E+1)>=L THEN GOSUB
    1190
DS 1070 NEXT E
IU 1080 REM
KL 1090 IF X(P,NE(P))<L THEN 1130
AE 1100 E1=E1+1
MX 1110 X1(P1,E1)=X(P,NE(P))
NW 1120 Y1(P1,E1)=Y(P,NE(P))
RL 1130 IF E1=0 THEN 1160
SU 1140 ME(P1)=E1
HC 1150 P1=P1+1
II 1160 NEXT P
PH 1170 MN=P1-1
RM 1180 GOTO 1260
IZ 1190 REM
AG 1200 E1=E1+1
ET 1210 M=(Y(P,E+1)-Y(P,E))/(X(P,
   E+1)-X(P,E))
RG 1220 Y1(P1,E1)=M*(L-X(P,E))+Y(
   P,E)
KN 1230 X1(P1,E1)=L
AP 1240 RETURN
EV 1250 REM *******************
OI 1255 REM WYCINANIE WZGLEDEM
UW 1260 REM GORNEGO BRZEGU
AO 1265 REM *******************
PA 1270 P1=1
TN 1280 FOR P=1 TO MN
MJ 1290 E1=0
FN 1300 FOR E=1 TO ME(P)-1
OX 1310 IF Y1(P,E)>=T THEN 1330
AW 1320 IF Y1(P,E)<T THEN 1420
HT 1330 REM PIERWSZY PUNKT LINII
AY 1335 REM JEST W SRODKU
AU 1340 E1=E1+1
BU 1350 Y2(P1,E1)=Y1(P,E)
BB 1360 X2(P1,E1)=X1(P,E)
LJ 1370 REM CO ZDRUGIM PUNKTEM?
WW 1380 IF Y1(P,E+1)<T THEN GOSUB
    1580
JD 1390 REM
RS 1400 GOTO 1460
IK 1420 REM
FS 1440 IF Y1(P,E+1)>=T THEN GOSU
   B 1580
IT 1450 REM
DX 1460 NEXT E
EX 1470 REM BADANIE PUNKTU KONCOW
   EGO
HS 1480 IF Y1(P,ME(P))<T THEN 152
   0
BL 1490 E1=E1+1
NE 1500 Y2(P1,E1)=Y1(P,ME(P))
ML 1510 X2(P1,E1)=X1(P,ME(P))
TX 1520 IF E1=0 THEN 1550
SZ 1530 ME(P1)=E1
HH 1540 P1=P1+1
IN 1550 NEXT P
PM 1560 MN=P1-1
UN 1570 GOTO 1680
JE 1580 REM
BN 1590 E1=E1+1
PJ 1600 IF X1(P,E+1)<>X1(P,E) THE
   N 1630
AS 1610 X2(P1,E1)=X1(P,E)
SN 1620 GOTO 1650
SS 1630 M=(Y1(P,E+1)-Y1(P,E))/(X1
   (P,E+1)-X1(P,E))
ON 1640 X2(P1,E1)=(T-Y1(P,E))/M+X
   1(P,E)
QM 1650 Y2(P1,E1)=T
BD 1660 RETURN
```

```
FJ 1670 REM ******************
OF 1680 REM WYCINANIE WZGLEDEM
XO 1685 REM PRAWEGO BRZEGU
GO 1687 REM ******************
PO 1690 P1=1
SZ 1700 FOR P=1 TO MN
LV 1710 E1=0
GB 1720 FOR E=1 TO ME(P)-1
SY 1730 IF X2(P,E)<=R THEN 1750
HF 1740 IF X2(P,E)>R THEN 1840
IZ 1750 REM
BI 1760 E1=E1+1
BW 1770 X1(P1,E1)=X2(P,E)
CV 1780 Y1(P1,E1)=Y2(P,E)
JL 1790 REM
GS 1800 IF X2(P,E+1)>R THEN GOSUB
   2000
IP 1810 REM
VC 1820 GOTO 1880
IY 1840 REM
NI 1860 IF X2(P,E+1)<=R THEN GOSU
   B 2000
JH 1870 REM
EL 1880 NEXT E
JN 1890 REM
NX 1900 IF X2(P,ME(P))>R THEN 194
   0
AX 1910 E1=E1+1
NG 1920 X1(P1,E1)=X2(P,ME(P))
OF 1930 Y1(P1,E1)=Y2(P,ME(P))
ZD 1940 IF E1=0 THEN 1970
TN 1950 ME(P1)=E1
HV 1960 P1=P1+1
JB 1970 NEXT P
QA 1980 MN=P1-1
SF 1990 GOTO 2070
HX 2000 REM
AG 2010 E1=E1+1
WA 2020 M=(Y2(P,E+1)-Y2(P,E))/(X2
   (P,E+1)-X2(P,E))
FV 2030 Y1(P1,E1)=M*(R-X2(P,E))+Y
   2(P,E)
OF 2040 X1(P1,E1)=R
AP 2050 RETURN
ZR 2060 REM ******************
OI 2065 REM WYCINANIE WZGLEDEM
QY 2070 REM DOLNEGO BRZEGU
AO 2075 REM ******************
PA 2080 P1=1
TN 2090 FOR P=1 TO MN
LH 2100 E1=0
FN 2110 FOR E=1 TO ME(P)-1
BF 2120 IF Y1(P,E)<=B THEN 2140
SI 2130 IF Y1(P,E)>B THEN 2250
IL 2140 REM
AU 2150 E1=E1+1
BU 2160 Y2(P1,E1)=Y1(P,E)
BB 2170 X2(P1,E1)=X1(P,E)
IX 2180 REM
LC 2190 IF Y1(P,E+1)>B THEN GOSUB
   2390
IB 2200 REM
RS 2210 GOTO 2270
RZ 2230 REM PIERWSZY PUNKT POZA
GM 2250 IF Y1(P,E+1)<=B THEN GOSU
   B 2390
IT 2260 REM
DX 2270 NEXT E
IZ 2280 REM
US 2290 IF Y1(P,ME(P))>B THEN 233
   0
AJ 2300 E1=E1+1
NE 2310 Y2(P1,E1)=Y1(P,ME(P))
ML 2320 X2(P1,E1)=X1(P,ME(P))
TX 2330 IF E1=0 THEN 2360
SZ 2340 ME(P1)=E1
HH 2350 P1=P1+1
IN 2360 NEXT P
PM 2370 MN=P1-1
BE 2380 RETURN
JE 2390 REM
AL 2400 E1=E1+1
PJ 2410 IF X1(P,E+1)<>X1(P,E) THE
   N 2440
AS 2420 X2(P1,E1)=X1(P,E)
SN 2430 GOTO 2460
SS 2440 M=(Y1(P,E+1)-Y1(P,E))/(X1
   (P,E+1)-X1(P,E))
CT 2450 X2(P1,E1)=(B-Y1(P,E))/M+X
   1(P,E)
FK 2460 Y2(P1,E1)=B
BD 2470 RETURN
AW 2475 REM ******************
FJ 2480 REM KRESLENIE WYCIETYCH
TO 2485 REM PUNKTOW
BH 2487 REM ******************
FH 2490 ? #6;CHR$(125);
FZ 2500 PLOT L,T:DRAWTO L,B
KJ 2510 DRAWTO R,B
UW 2520 DRAWTO R,T
RZ 2530 DRAWTO L,T
TI 2540 FOR P=1 TO MN
GH 2550 FOR E=1 TO ME(P)-1
KB 2560 PLOT X2(P,E),Y2(P,E):DRAW
   TO X2(P,E+1),Y2(P,E+1)
ED 2570 NEXT E
IX 2580 NEXT P
BL 2590 RETURN
ZL 2600 REM ******************
MV 2610 DATA 8
EK 2630 DATA 12,195,87,210,90,232
   ,65,232,60,213,50,200,35,165,3
   7
HU 2640 DATA 145,50,85,55,70,30,6
   0,30,60,57
EH 2650 DATA 3,69,66,85,80,152,85
VZ 2670 DATA 6,160,70,130,120,145
   ,135,170,135,170,128,200,75
FO 2680 DATA 6,203,40,190,5,195,0
   ,170,0,165,5,167,36
RY 2700 DATA 3,73,35,70,35,80,45
UB 2710 DATA 4,65,50,53,70,65,70,
   75,60
IM 2730 DATA 7,230,63,235,63,230,
   25,240,25,230,100,240,100,235,
   63
UH 2731 DATA 6,208,80,210,80,210,
   82,208,82,208,80,210,82
FO 2740 END
XN 5000 COLOR 0
JN 5010 FOR X=L TO R
VE 5020 PLOT X,T:DRAWTO X,B
MC 5025 NEXT X
GI 5030 ? CHR$(125)
AP 5040 RETURN
```

mieszczą się w granicach ramki. Analiza położenia napisów polega na sprawdzeniu, czy początek tekstu znajduje się na prawo od lewej granicy okna, a koniec na lewo od prawej granicy. Oprócz tego napis powinien znajdować się w wierszu położonym między górną i dolną granicą ramki.

Zmienna *T$* wykorzystywana jest do przechowywania napisów — ciągów znaków. Tablice *XL* i *XR* używane są do przechowywania współrzędnych *X* początków i końców wszystkich ciągów. W tablicach **YT** i **YB** przechowywane są współrzędne górnej i dolnej granicy każdego ciągu znaków.

Na jeden symbol wydziela się 8×8 elementów rastra. Faktycznie zajmuje on obszar 7×7, to znaczy, że odstęp między literami w wierszu równy jest jednemu elementowi rastra. Taki sam jest odstęp między wierszami. Symbole zaczynają się w punktach o współrzędnych *X*: 0, 8, 16, 24, ... i kończą się w punktach 7, 15, 23, 31, ... Górna granica symboli w wierszach ma współrzędne *Y* 0, 8, 16, ... a dolna — 7, 15, 23, ... W celu określenia początku obszaru wyświetlania pewnego znaku musimy odjąć 1 od numeru pozycji w wierszu i pomnożyć przez 8. W podobny sposób określa się koniec obszaru wyświetlania, górną i dolną jego granicę. Zresztą i tak będzie to za nas robił komputer. Odpowiednie obliczenia zawarte są w programie nr 3, który jest uzupełnieniem programu nr 2.

Program nr 3 może być tak zmodyfikowany, aby przechowywać dowolną część tekstu mieszczącego się wewnątrz granic okna. Można też ulepszyć algorytm wycinania linii, w celu zmniejszenia liczby wykonywanych obliczeń. Będzie to ważne przy dużej liczbie linii.

Po wykonaniu wycinania można w dowolny sposób przekształcić zobrazowanie wewnątrz okna:

```
FK 1 REM ***************
FJ 2 REM *             *
YO 3 REM * Program nr 3 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
VI 171 DIM TX$(100),RO(10),CO(10)
   ,TL(10),TR(10),TT(10),TB(10),T
   $(10),LT(10),PT(10)
ZS 271 REM ODCZYTANIE TEKSTOW I W
   SPOLRZEDNYCH
AG 272 READ TN:F=1
HF 273 FOR K=1 TO TN
RR 274 READ T$,RO,CO:TX$(F)=T$:RO
   (K)=RO:CO(K)=CO:LT(K)=LEN(T$):
   PT(K)=F
CQ 275 TL(K)=(CO(K)-1)*8
QT 276 TR(K)=(TL(K)-1)+LEN(T$)*8
LQ 277 TT(K)=(RO(K)-1)*8
DF 278 TB(K)=TT(K)+7
VI 279 F=F+LEN(T$):NEXT K
MV 431 REM UMIESZCZENIE TEKSTOW
GW 432 FOR K=1 TO TN
NY 433 Y=RO(K):X=CO(K):T$=TX$(PT(
   K),PT(K)+LT(K)-1)
SV 434 GOSUB 10000
HJ 435 NEXT K
WP 2581 FOR K=1 TO TN
GI 2582 IF TL(K)<L OR TR(K)>R OR
   TT(K)<T OR TB(K)>B THEN 2589
VG 2583 Y=RO(K):X=CO(K):T$=TX$(PT
   (K),PT(K)+LT(K)-1)
BW 2584 GOSUB 10000
IE 2589 NEXT K
LT 2732 DATA 4
NM 2733 DATA KADLUB,12,10
BB 2734 DATA OGON,7,3
UM 2735 DATA SKRZYDLO,18,16
PV 2736 DATA SMIGLO,2,26
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
   TURN
```

— przenieść go w inne miejsce ekranu, powiększyć lub pomniejszyć lub też obrócić;

— zachować początkowy rysunek, nałożywszy na niego (np.: w kącie ekranu) wydzielony za pomocą okna fragment rysunku (być może powiększony).

Miejsce umieszczenia zawartości okna, po ewentualnych przekształceniach, nazywamy polem wprowadzania. Jego położenie, podobnie jak dla okna, można opisać podając współrzędne lewego, górnego rogu i rozmiary. Okno określa „co" chcemy zobaczyć, pole wyprowadzania wskazuje, „gdzie" chcemy zobaczyć to na ekranie. Rozmiary pola wyprowadzania mogą być większe lub mniejsze od rozmiarów okna lub takie same. Pole wyprowadzania może zajmować cały ekran lub być niewielką wstawką na ekranie. Okno i pole wyprowadzania można rozmieścić w różnych obszarach ekranu lub uczynić je przecinającymi się. Za pomocą pola wyprowadzania, którego rozmiary są znacznie większe od rozmiarów okna, można powiększyć zobrazowanie oddzielnych elementów i obejrzeć szczegóły, które są początkowo słabo rozróżniane.

Program przekształcający obszar okna w pole wyprowadzania powinien przekształcać współrzędne wszystkich punktów wewnątrz okna na współrzędne punktów w polu wyprowadzania.

Nowe współrzędne punktów w polu wyprowadzania związane są ze współrzędnymi

odpowiadających punktów w obszarze okna następującymi zależnościami:

$$XN = (X - XW) * (WY/WW) + XV$$

$$YN = (Y - YW) * (HV/HW) + YV$$

Parametry *(WV/WW)* i *(HV/HW)* są współczynnikami skalowania. Jeśli są one równe 1, rozmiary elementów zobrazowania podczas przekształcania zmieniają się. Wartości większe od 1 powodują zwiększenie rozmiarów, a wartości mniejsze od 1 — zmniejszenie. Jeśli stosunek *WV/HV* nie jest równy *WW/HW* obserwuje się naruszenie proporcji zobrazowania, co równoważne jest nierównomiernemu skalowaniu wzdłuż osi *X* i *Y*. Człony *XV* i *YV* w powyższych równaniach określają wielkość przemieszczenia.

Przekształcanie zobrazowania z obszaru okna w pole wyprowadzania wykonuje **program nr 4.** Aby zobaczyć jego działanie, należy dołączyć go do programu nr 2.

```
FK 1 REM ****************
FJ 2 REM *              *
ZI 3 REM * Program nr 4 *
FC 4 REM *              *
FO 5 REM ****************
NL 6 REM
VI 175 DIM XN(8,20),YN(8,20)
XL 335 POKE 764,255:GOTO 2750
IX 2740 REM
JA 2750 REM
EH 2810 GOSUB 2850
BY 2820 GOSUB 3070
BM 2830 GOSUB 3150
EG 2840 IF PEEK(764)=255 THEN 284
   0
EI 2845 POKE 764,255:GOTO 290
JC 2850 REM
TT 2860 ? CHR$(125);
SJ 2870 ? "Lewy gorny rog x,y ";
RQ 2880 TRAP 2860:INPUT XV,YV
UC 2890 ? CHR$(125);
FT 2920 ? "Podaj wysokosc i szero
   kosc ";
DE 2930 TRAP 2890:INPUT HV,WV
PA 2940 TRAP 40000
TS 2950 ? CHR$(125);
MX 2960 L=XV
DL 2970 R=XV+WV
PH 2980 T=YV
TY 2990 B=YV+HV
YE 3000 IF L>=R OR R>XM OR T>=B O
   R T<0 OR B>YM THEN 2860
AT 3060 RETURN
JH 3070 REM PRZELICZENIE OKNA
TL 3080 FOR P=1 TO MN
CZ 3090 FOR E=1 TO ME(P)
MH 3100 XN(P,E)=(X2(P,E)-XW)*(WV/
   WW)+XV
HK 3110 YN(P,E)=(Y2(P,E)-YW)*(HV/
   HW)+YV
DH 3120 NEXT E
IB 3130 NEXT P
AP 3140 RETURN
ZF 3150 REM WYKRESLENIE WYCIETYCH
    PUNKTOW
LJ 3160 ? #6;CHR$(125)
GN 3170 PLOT L,T:DRAWTO L,B
KX 3180 DRAWTO R,B
VK 3190 DRAWTO R,T
RL 3200 DRAWTO L,T
SU 3210 FOR P=1 TO MN
FT 3220 FOR E=1 TO ME(P)-1
AU 3230 PLOT XN(P,E),YN(P,E):DRAW
   TO XN(P,E+1),YN(P,E+1)
DP 3240 NEXT E
IJ 3250 NEXT P
AX 3260 RETURN
CZ 3270 REM *********************
   *****
FR 3280 END
```

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Program nr 2 *
TG 4 REM * uzupelnienie *
FM 5 REM *              *
FP 6 REM ****************
NM 7 REM
NN 8 REM
PV 10 REM W programie nr 2 z IKS
QJ 12 REM nr 2 z br. omylkowo nie
PJ 14 REM zamieszczono podpro-
OA 16 REM gramu, bedacego zasad-
BZ 18 REM nicza czescia przykla-
VF 19 REM du.
DT 20 REM Przepraszamy za zwia-
LA 22 REM zane z tym klopoty.
HE 24 REM Ponizszy program nale-
NK 26 REM zy wprowadzic do kom-
FN 28 REM putera i zapisac na
EK 30 REM tasmie magnetycznej.
JQ 32 REM Nastepnie nalezy dola-
IJ 34 REM czyc go do programu
IC 36 REM nr 2.
NK 70 GOSUB 31000
XV 31000 W=0:V=1:RESTORE 31050
RS 31010 READ X,Y:FOR I=X TO Y:RE
   AD Z:POKE I,Z:NEXT I
KY 31020 READ Z:IF Z<>-1 THEN ? "
   BLAD W DANYCH! SPRAWDZ LINIE D
   ATA !!!":END
MI 31030 W=W+1:IF W<V THEN 31010
DR 31040 RETURN
CX 31050 DATA 1536,1700
NV 31060 DATA 104,141,185,6,141,1
   86,6,162,0,104,157,173,6,157,1
   81,6,104,157,169,6
XO 31070 DATA 157,177,6,169,0,157
   ,165,6,232,236,186,6,208,231,1
   69,7,162,6,160,43
FV 31080 DATA 76,92,228,173,185,6
   ,240,88,169,3,141,15,210,169,0
   ,141,8,210,141,185
US 31090 DATA 6,170,189,165,6,208
   ,52,168,189,173,6,133,1,189,16
   9,6,133,0,177,0
JU 31100 DATA 72,138,10,168,104,1
   53,0,210,169,164,153,1,210,160
   ,1,177,0,157,165,6
ZM 31110 DATA 136,17,0,240,22,189
   ,169,6,24,105,2,157,169,6,144,
   3,254,173,6,169
GG 31120 DATA 127,238,185,6,222,1
   65,6,232,236,186,6,208,185,76,
   98,228,162,0,189,181
EA 31130 DATA 6,157,173,6,189,177
   ,6,157,169,6,232,236,186,6,208
   ,238,173,186,6,141
DT 31140 DATA 185,6,76,133,6,-1
```

Opracowując powyższy artykuł korzystaliśmy z książki wydanej w 1987 roku w Moskwie przez Wydawnictwo „Mir" pod tytułem: „Microcomputer graphics. Techniques and Applications" napisanej przez D. Hearn i M. P. Baker.

Tomasz MROWIEC
Ludwik PIELA

Skalowanie wykresu

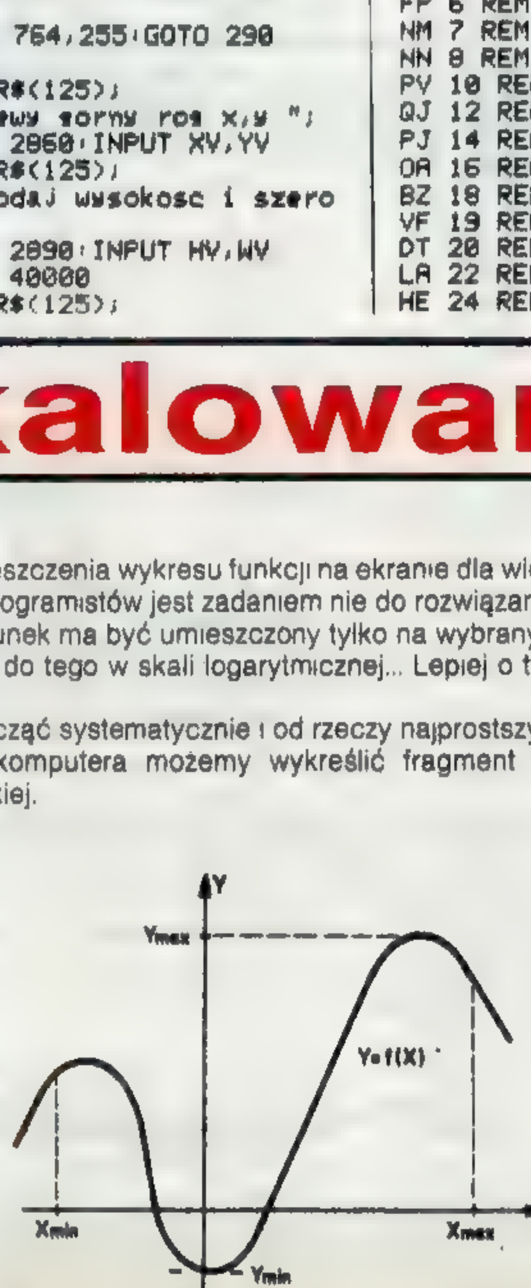
Krzysztof POŹNIAK

Problem umieszczenia wykresu funkcji na ekranie dla wielu początkujących programistów jest zadaniem nie do rozwiązania. Jeżeli jeszcze rysunek ma być umieszczony tylko na wybranym obszarze ekranu i do tego w skali logarytmicznej... Lepiej o tym nie myśleć!

Proponuję zacząć systematycznie i od rzeczy najprostszych.

Na ekranie komputera możemy wykreślić fragment funkcji $Y = f(X)$, np. takiej.

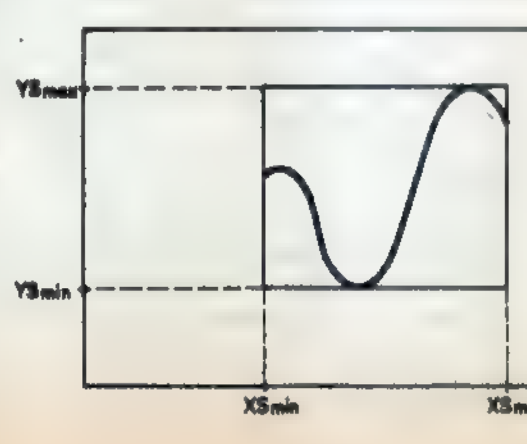
rysunek 1



gdzie: X_{min} — początkowa wartość argumentu funkcji,
X_{max} — końcowa wartość argumentu funkcji,
Y_{min} — minimalna wartość funkcji dla przedziału $X \in <X_{min}, X_{max}>$,
Y_{max} — maksymalna wartość funkcji dla przedziału $X \in <X_{min}, X_{max}>$,
uwaga: wartości Y_{min} i Y_{max} mogą być podawane z odpowiednimi zapasami, odpowiednio w dół i w górę, ale wtedy funkcja nie obejmie całego obszaru, na którym będzie rysowana.

Prezentowany fragment wykresu chcemy umieścić na zadanym obszarze ekranu, np:

rysunek 2

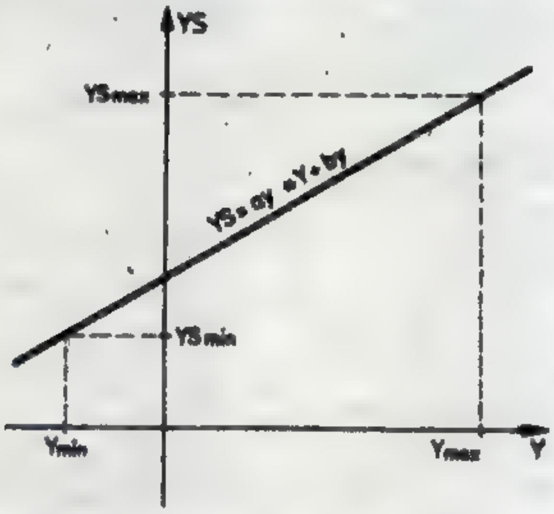


gdzie: XS_{min} — początkowy punkt na osi poziomej XS ekranu,
XS_{max} — końcowy punkt na osi poziomej XS ekranu,
YS_{min} — najniższy punkt na osi pionowej YS ekranu,
YS_{max} — najwyższy punkt na osi pionowej YS ekranu.

Tak więc naszym zadaniem jest znaleźć takie przekształcenie, aby przejść z poziomej osi wykresu *X* na poziomą oś ekranu *XS* i z pionowej osi wykresu *Y* na pionową oś ekranu *YS*.

Zajmijmy się na początek osiami pionowymi. Graficzne przejście pomiędzy osią *Y*, a *YS* można przedstawić następująco:

rysunek 3



Jak widać funkcja przekształcająca jest linią prostą, czyli przekształcenie można zapisać w ten sposób: $YS = a_y * Y + b_y$. Parametry oblicza się następująco:

$$a_y = \frac{YS_{max} - YS_{min}}{Y_{max} - Y_{min}}$$

$$b_y = YS_{min} - a_y * Y_{min} = YS_{max} - a_y * Y_{max} = \frac{YS_{min} * Y_{max} - YS_{max} * Y_{min}}{Y_{max} - Y_{min}}$$

Sprawdźmy nasze przekształcenia na prostym przykładzie: niech wartość funkcji zmienia się w zakresie od —3 do 2, a chcemy aby na ekranie funkcja rysowała się w pionie między 100-nym, a 200-nym punktem na ekranie.

Rozwiązanie: dane: $Y_{min} = -3$
$Y_{max} = 2$
$YS_{min} = 100$
$YS_{max} = 200$

stąd

$$a_y = \frac{200 - 100}{2 - (-3)} = 20; \quad b_y = \frac{100 * 2 - 200 * (-3)}{2 - (-3)} = 160$$

otrzymujemy funkcję przekształcającą: $YS = 20 * Y + 160$

Problem przekształcenia osi poziomej funkcji *X* na oś poziomą ekranu *XS* przeprowadza się identycznie, ale odwrotnie.

I tutaj chciałbym wtrącić ważną uwagę: aby zachować jak największą dokładność rysunku na ekranie, argument funkcji należy obliczać na podstawie współrzędnej punktu na ekranie. Dzięki temu w sposób jednoznaczny określamy współrzędną poziomą punktu na ekranie.

Graficznie funkcję przekształcającą można przedstawić następująco:

rysunek 4



Jest to również linia prosta, stąd $X = a_x * XS + b_x$, gdzie:

$$a_x = \frac{X_{max} - X_{min}}{XS_{max} - XS_{min}}$$

$$b_x = X_{min} - a_x * XS_{min} = X_{max} - a_x * XS_{max} = \frac{X_{min} * XS_{max} - X_{max} * XS_{min}}{XS_{max} - XS_{min}}$$

Zajmijmy się teraz innym problemem: skalą logarytmiczną.

Dla osi pionowej sprawa jest prosta — należy wartość funkcji zlogarytmować. Pozostałe przekształcenia są identyczne, jak dla poprzednio obliczanej funkcji przeliczającej oś *Y* na oś *YS*, z tą różnicą, że zamiast Y_{min} podstawiamy $m * log_a(Y_{min})$, a zamiast Y_{max} podstawiamy $m * log_a(Y_{max})$, gdzie:
m — pewien współczynnik liniowy
a — podstawa logarytmu
(np. dla skali w decybelach [*dB*] $m = 20$, $a = 10$)

Otrzymujemy następującą funkcję przekształcającą:

$$YS = a_w * m * log_a(Y) + b_w$$

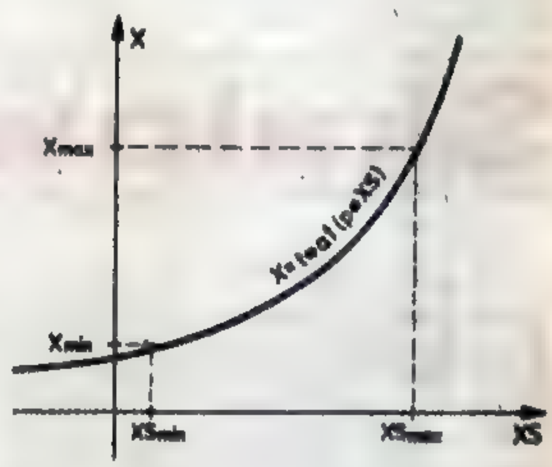
gdzie:

$$a_w = \frac{YS_{max} - YS_{min}}{m * log_a\left(\frac{Y_{max}}{Y_{min}}\right)}$$

$$b_w = YS_{min} - m * a_w * log_a(Y_{min}) = YS_{max} - m * a_w * log_a(Y_{max}) = \frac{YS_{min} * log_a(Y_{max}) - YS_{max} * log_a(Y_{min})}{log_a\left(\frac{Y_{max}}{Y_{min}}\right)}$$

Bardziej złożone przekształcenie zachodzi dla osi poziomych, dla której argument obliczamy na podstawie współrzędnej punktu na ekranie. Funkcją odwrotną do logarytmicznej jest funkcja wykładnicza, stąd graficzne przekształcenie osi poziomych wygląda następująco:

rysunek 5



Otrzymujemy następującą funkcję przekształcającą:
$X = t * a \uparrow (p * XS)$ gdzie:
↑ — znak potęgowania
a — podstawa funkcji wykładniczej

$$p = \frac{log_a\left(\frac{X_{max}}{X_{min}}\right)}{XS_{max} - XS_{min}}$$

$$t = a \uparrow \left(\frac{XS_{max} * log_a(X_{min}) - XS_{min} * log_a(X_{max})}{(XS_{max} - XS_{min})}\right)$$

Po tych przerażających obliczeniach pora przejść do części praktycznej. Program pozwala wyrysować na dowolnym obszarze ekranu obrysowanego ramką funkcję wprowadzoną przez użytkownika. Tam, gdzie to było możliwe, zachowałem nazwy zmiennych używanych w części teoretycznej artykułu, aby łatwo można było prześledzić proces obliczeniowy.

W skali logarytmicznej oś pionowa przedstawiana jest w [*dB*].
Jeśli funkcja wychodzi poza ramkę, to program kontynuuje obliczanie, ale nie stawia punktu

```
   10 REM ************************
   11 REM
   12 REM
   13 REM    Krzysztof Poźniak
   14 REM
   15 REM   Skalowanie  wykresu
   16 REM
   17 REM                  ©1988
   18 REM
   19 REM ************************

   80 LET XSkon=255: LET YSkon=17
5
   90 REM wprowadzanie danych
  100 INPUT "podaj wzor funkcji "
'"y=f(x)="; LINE x$
  110 INPUT "podaj minimalna wart
osc argumentu Xmin=";Xmin'"podaj
 maksymalna wartosc argumentu Xm
ax=";Xmax
  111 IF Xmin>=Xmax THEN GO TO 11
0
  120 INPUT "podaj minimalna wart
osc na osi Y: Ymin=";Ymin'"podaj
 maksymalna wartosc na osi Y: Ym
ax=";Ymax
  121 IF Ymin>=Ymax THEN GO TO 12
0
  130 INPUT "podaj minimalna wspo
lrzedna ekranu na osi X: XSmin="
;XSmin'"podaj maksymalna wspolrz
edna ekranu na osi X: XSmax=";XS
max
  131 IF XSmin>=XSmax OR XSmin<0
OR XSmax>XSkon THEN GO TO 130
  132 LET XSmin=INT XSmin: LET XS
max=INT XSmax
  140 INPUT "podaj minimalna wspo
lrzedna ekranu na osi Y: YSmin="
;YSmin'"podaj maksymalna wspolrz
edna ekranu na osi Y: YSmax=";YS
max
  141 IF YSmin>=YSmax OR YSmin<0
OR YSmax>YSkon THEN GO TO 140
  142 LET YSmin=INT YSmin: LET YS
max=INT ySmax
  200 CLS : PRINT "funkcja y=f(x)
=";x$'"Xmin=";Xmin,"Xmax=";Xmax'
"Ymin=";Ymin,"Ymax=";Ymax'"XSmin
=";XSmin,"XSmax=";XSmax'"YSmin="
;YSmin,"YSmax=";YSmax
  205 PRINT '"opcje:"''"0-nowe da
ne"'"1-skala liniowa"'"2-skala l
ogarytmiczna"
  210 LET i$=INKEY$:
  211 IF i$="0" THEN GO TO 100
  212 IF i$="1" THEN GO TO 300
  213 IF i$="2" THEN GO TO 400
  215 GO TO 210
  300 REM skala liniowa
  310 LET ax=(Xmax-Xmin)/(XSmax-X
Smin): LET bx=(Xmin*XSmax-Xmax*X
Smin)/(XSmax-XSmin)
  315 LET ay=(YSmax-YSmin)/(Ymax-
Ymin): LET by=(YSmin*Ymax-YSmax*
Ymin)/(Ymax-Ymin)
  320 GO TO 500
  400 REM skala logarytmiczna-dec
ybelowa
  410 IF Xmin<=0 THEN PRINT "skal
a rzednych X ma wartosci ujemne"
  411 IF Ymin<=0 THEN PRINT "skal
a odcietych y ma wartosci ujemne
"

  412 IF Xmin<=0 OR Xmax<=0 THEN
BEEP .1,20: PAUSE 1: PAUSE 0: GO
 TO 200
  415 LET m=20
  420 LET aw=(YSmax-YSmin)/(m*LN
(Ymax/Ymin)/LN 10)
  421 LET bw=(YSmin*LN Ymax-YSmax
*LN Ymin)/LN (Ymax/Ymin)
  425 LET bw=(YSmin*LN Ymax-YSmax
*LN Ymin)/LN (Ymax/Ymin)
  430 LET p=LN (Xmax/Xmin)/(XSmax
-XSmin)
  431 LET t=EXP ((XSmax*LN Xmin-X
Smin*LN Xmax)/(XSmax-XSmin))
  500 REM ramka
  505 CLS : PLOT XSmin,YSmin: DRA
W XSmax-XSmin,0: DRAW 0,YSmax-YS
min: DRAW XSmin-XSmax,0: DRAW 0,
YSmin-YSmax
  520 REM petla rysowania funkcji
  530 FOR k=XSmin TO XSmax
  540 IF i$="1" THEN LET x=ax*k+b
x
  541 IF i$="2" THEN LET x=t*EXP
(k*p)
  550 IF i$="1" THEN LET YS=ay*VA
L x$+by
  551 IF i$="2" THEN LET YS=m*aw*
LN (VAL x$)/LN 10+bw
  560 IF YS>=YSmin AND YS<=YSmax
THEN PLOT k,YS
  570 NEXT k
  580 PRINT #1;"wcisnij dowolny k
lawisz": PAUSE 1: BEEP .1,20: PA
USE 0: GO TO 200
```

K. POŹNIAK

Inteligentny samochód (?)

W ponad stuletniej historii motoryzacji przyzwyczailiśmy się, że charakterystycznymi elementami każdego samochodu są cztery koła, kierownica i silnik. Tymczasem niepostrzeżenie do elementów tych dołączył komputer.

W czołowych firmach samochodowych świata przeprowadzane są zaawansowane prace nad pojazdami XXI wieku, w których komputer będzie odgrywał niezwykle istotną rolę. Jego podstawowym zadaniem ma być sterowanie pracą silnika oraz wszystkich podzespołów mechanicznych, a także ich kontrolowanie. W wypadku wykrycia nieprawidłowości w pracy pojazdu kierowca będzie o tym natychmiast informowany wraz ze wskazaniem sposobu naprawienia samochodu. Jeżeli kierowca nie będzie w stanie poradzić sobie z naprawą, komputer poinformuje go, gdzie znajduje się najbliższy serwis.

Komputer będzie także wspomagał kierowcę w prowadzeniu pojazdu. Każdy samochód będzie wyposażony w system automatycznego utrzymywania bezpiecznej odległości od innych użytkowników drogi. Radarowe czujniki będą śledzić otoczenie pojazdu i w razie wykrycia zagrożenia komputer automatycznie uruchomi hamulec lub przyspieszy w razie potrzeby. Proces hamowania ma być oczywiście kontrolowany, przez co uniknie się blokowania hamulców.

Znacznie rozbudowany będzie także system komputerowego radia kierowców. Kierowca wsiadając do samochodu wprowadzi do pamięci komputera informację o docelowym miejscu podróży. W miastach oraz wzdłuż wszystkich dróg publicznych rozmieszczone będą nadajniki przekazujące drogą radiową komunikaty o aktualnych warunkach drogowych i atmosferycznych. Komputer automatycznie śledząc wszystkie komunikaty na bieżąco naniesie poprawki dotyczące trasy przejazdu.

W każdym samochodzie znajdzie się także radiotelefon, za którego pomocą będzie można uzyskać połączenie z dowolnym abonentem, w dowolnym miejscu.

Elektroniczny numer rejestracyjny umożliwi w razie potrzeby znajdowanie skradzionych samochodów oraz identyfikację pojazdów, które naruszyły przepisy drogowe.

Komputery wyręczą także policję drogową w kontrolowaniu ruchu pojazdów. Wszelkie przewinienia drogowe za sprawą odpowiednio rozmieszczonych kamer będą automatycznie wychwytywane i każdy użytkownik drogi, który dopuścił się wykroczenia, będzie natychmiast identyfikowany i drogą pocztową otrzyma do zapłacenia stosowny przekaz.

MBO

Przesyłanie kodu maszynowego do instrukcji REM

Prezentowany program jest wygodnym narzędziem dla tych, którzy chcą przechowywać ciągi bajtów (m. in. kody maszynowe) w programie Basic'a. Program wprowadza dowolnie długi ciąg bajtów z każdej części pamięci komputera do linii komentarza **REM**, którą sam tworzy.

Procedurę wywołujemy przez **FN**f(*a*, *b*, *c*), gdzie:

- *a* — adres początku bajtów w pamięci, 0≤a≤65535
- *b* — ilość bajtów, 0≤b≤65535
- *c* — numer linii **REM**, w której umieścimy bajty, 1≤c≤9999

Po wykonaniu procedury funkcja użytkownika (**FN**) przekazuje adres pierwszego elementu ciągu bajtów (można go użyć jako adresu startu kodu maszynowego)

Program jest zabezpieczony przed błędnymi danymi. W momencie wykrycia błędu system przerywa pracę i wypisuje odpowiedni komunikat:

- *Q Parrametr error* — gdy nie ma zgodności ilości parametrów lub gdy parametry nie zawierają się w dopuszczalnych granicach.
- *B Integer out of range* — gdy numer linii wynosi 0 lub przekracza 9999
- *4 Out of memory* — gdy zabraknie miejsca na utworzenie linii
- *G No room for line* — gdy próbujemy utworzyć linię o numerze linii już istniejącej w programie.

Dodatkowa funkcja: *jeżeli ustalimy długość ciągu bajtów równą zero, to procedura nie utworzy nowej linii tylko poda przez FN adres początku bajtów w tej linii.*

Ważna uwaga: *ponieważ tworzenie linii przesuwa obszary pamięci, to aby uniknąć błędów systemu, należy utworzyć linię* **REM** *za linią zawierającą definicję funkcji (***DEF FN***) i linią wywołującą funkcję (***FN***). W wypadku prezentowanego programu za linią 300.*

Program jest relokowalny i można go umieścić w dowolnym miejscu za **RAMTOP-em**, tak aby nie kolidował z innymi procedurami maszynowymi.

```
10 REM ************************
11 REM
12 REM    Krzysztof Poźniak
13 REM
14 REM   'przesylanie kodu'
15 REM   ' maszynowego do '
16 REM   ' instrukcji REM '
17 REM
18 REM                 ©1988
19 REM
20 REM ************************

  25 CLEAR 65299
  30 DEF FN f(a,b,c)=USR 65300
 100 REM wczytywanie kodu
 110 RESTORE 9000
 120 FOR k=0 TO 9
 130 LET s1=0: LET s2=0: LET z=1
 140 FOR l=1 TO 15
 150 READ a: POKE 65299+k*15+l,a
: LET s1=s1+a: LET s2=s2+z*a: LE
T z=-z
 160 NEXT l: READ sk1,sk2: IF sk
1<>s1 OR sk2<>s2 THEN BEEP .1,20
: PRINT "blad w linii ";9000+10*
k: STOP
 170 NEXT k
 200 CLS : BEEP .1,20: PRINT "pr
ogram zostal wczytany"
 210 INPUT "padaj adres poczatku
 kodu maszynowego: ";adres
 220 INPUT "podaj dlugosc ciagu
bajtow: ";dlugosc
 230 INPUT "podaj nr linii przez
naczonej na zapis bajtow: ",nrli
nii
 300 CLS : PRINT "adres poczatku
 ciagu bajtow :",FN f(adres,dlug
osc,nrlinii)
 310 LIST nrlinii

9000 DATA 221,42,11,92,6,3,221,2
29,225,35,35,175,17,4,0,1316,156
9010 DATA 182,35,182,35,35,35,18
2,25,16,246,254,0,32,39,221,1519
,689
9020 DATA 78,12,221,70,13,3,3,19
7,3,3,3,3,197,205,5,1016,30
9030 DATA 31,221,110,20,221,102,
21,125,180,40,7,229,1,240,216,17
64,-190
9040 DATA 9,48,15,46,10,195,85,0
,46,25,195,85,0,46,15,820,-70
9050 DATA 195,85,0,225,205,110,2
5,8,221,126,12,221,182,13,40,166
8,92
9060 DATA 50,8,40,235,43,193,205
,85,22,35,35,221,126,21,119,1438
,-158
9070 DATA 221,126,20,35,119,35,1
93,113,35,112,35,54,234,19,62,14
13,425
9080 DATA 13,18,35,235,221,110,4
,221,102,5,221,78,12,221,70,1566
,-210
9090 DATA 13,213,237,176,193,201
,1,5,0,9,193,193,229,24,245,1932
,260
```

```
00001 ;********************************
00002 ;*                              *
00003 ;*      Krzysztof Pozniak       *
00004 ;*                              *
00005 ;*  program przesyla kod        *
00006 ;*  maszynowy do insrukcji      *
00007 ;*     jezyka Basic: REM        *
00008 ;*                              *
00009 ;*                     @1988    *
00010 ;********************************
00011 ;
00012 ; program jest relokowalny
00013 ; i calkowicie zabezpieczony
00014 ; przed blednymi danymi przesy-
00015 ; lanymi z poziomu Basic'a przy
00016 ; pomocy funkcji uzytkownika FN
00017 ;
00018 ; adresy procedur ROM-u
00019 linead equ  196eh
00020 makero equ  1655h
00021 testro equ  1f05h
00022 error  equ  55h
00023 ; adres zmiennej systemowej
00024 defadd equ  5c0bh
00025 ; wartosci stale programu
00026 nrmax  equ  9999
00027 illicz equ  3
00028 skok   equ  4
00029 ENTER  equ  13
00030 REM    equ  234
00031 komunB equ  10
00032 komunG equ  15
00033 komunQ equ  25
00034 ;
00035 ;       P R O G R A M
00036 ;
00037 start  ld   ix,(defadd)
00038 ;kontrola poprawnosci danych
00039        ld   b,illicz
00040        push ix
00041        pop  hl
00042        inc  hl
00043        inc  hl
00044        xor  a
00045        ld   de,skok
00046 petla  or   (hl)
00047        inc  hl
00048        or   (hl)
00049        inc  hl
00050        inc  hl
00051        inc  hl
00052        or   (hl)
00053        add  hl,de
00054        djnz petla
00055        cp   0
00056        jr   nz,blad_Q
00057        ld   c,(ix+12)
00058        ld   b,(ix+13)
00059        inc  bc
00060        inc  bc
00061        push bc
00062        inc  bc
00063        inc  bc
00064        inc  bc
00065        inc  bc
00066        push bc
00067        call testro
00068        ld   l,(ix+20)
00069        ld   h,(ix+21)
00070        ld   a,l
00071        or   h
00072        jr   z,blad_B
00073        push hl
00074        ld   bc,-nrmax-1
00075        add  hl,bc
00076        jr   nc,dalej
00077 ; obsluga bledow
00078 blad_B ld   l,komunB
00079        jp   error
00080 blad_Q ld   l,komunQ
00081        jp   error
00082 blad_G ld   l,komunG
00083        jp   error
00084 dalej  pop  hl
00085        call linead
00086        ex   af,af'
00087        ld   a,(ix+12)
00088        or   (ix+13)
00089        jr   z,adres
00090        ex   af,af'
00091        jr   z,blad_G
00092 ; dane poprawne
00093 ; czesc wykonawcza
00094        dec  hl
00095        pop  bc
00096        call makero
00097        inc  hl
00098        inc  hl
00099        ld   a,(ix+21)
00100        ld   (hl),a
00101        ld   a,(ix+20)
00102        inc  hl
00103        ld   (hl),a
00104        inc  hl
00105        pop  bc
00106        ld   (hl),c
00107        inc  hl
00108        ld   (hl),b
00109        inc  hl
00110        ld   (hl),REM
00111        inc  de
00112        ld   a,ENTER
00113        ld   (de),a
00114        inc  hl
00115        ex   de,hl
00116        ld   l,(ix+4)
00117        ld   h,(ix+5)
00118        ld   c,(ix+12)
00119        ld   b,(ix+13)
00120        push de
00121        ldir
00122 koniec pop  bc
00123        ret
00124 adres  ld   bc,5
00125        add  hl,bc
00126        pop  bc
00127        pop  bc
00128        push hl
00129        jr   koniec
00130        end
```

K. POŹNIAK

COMMODORE C-64 — edytor tekstu

T. CISEK

W cyklu artykułów pod tym tytułem autor pragnie podzielić się z czytelnikami „IKS-a" swoimi doświadczeniami w stosowaniu różnych edytorów tekstu dla komputera Commodore 64. Omówiony zostanie sposób ich działania (instrukcje), możliwości, wady i zalety.

Autor „wypróbował" wszystkie omówione edytory, dysponując następującym wyposażeniem: drukarką OKIDATA 120 (120 znaków/s w trybie draft i 20 w trybie NLQ), stacją dysków FSD-1 „EXCELERATOR" +", kompatybilną ze stacją 1541 i oczywiście mikrokomputerem C-64.

GEOWRITE

Edytor tekstu Geowrite stanowi część programu GEOS (Graphic Environment Operating System), który został opracowany w 1985 r. przez firmę Berkeley Softworks z Kalifornii. Poniższy opis dotyczy edytora zawartego w programie GEOS, wersja 1.2.

1. Uruchamianie edytora

Uruchamianie programu Geowrite odbywa się w sposób taki jak każdego innego zawartego w Geosie, to znaczy można dokonać tego przez:

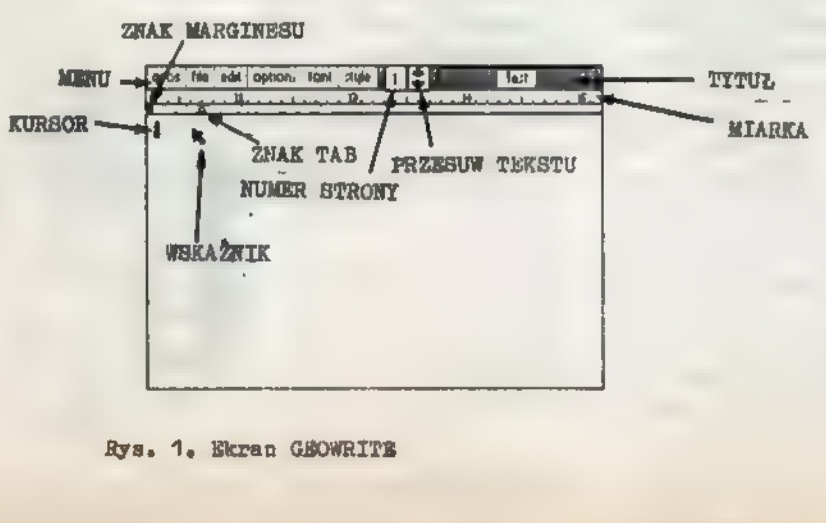
a) jednokrotne naciśnięcie przycisku Fire joysticka po ustawieniu wskaźnika (strzałki) na ikonie Geowrite i otwarcie pliku przez wybranie „OPEN" z menu „FILE".

b) dwukrotne, szybkie naciśnięcie przycisku Fire na ikonie Geowrite. Po załadowaniu Geowrite należy dokonać wyboru jednej z trzech możliwości: CREATE — rozpoczęcie nowego dokumentu, OPEN — otwarcie dokumentu istniejącego, QUIT — zakończenie pracy edytora. Należy zaznaczyć, że wszystkie rozkazy i funkcje edytora wybierane są za pomocą strzałki kierowanej joystickiem i naciśnięciu przycisku *FIRE*.

Po wybraniu CREATE należy podać nazwę dokumentu. Po wybraniu OPEN program wyświetla nazwy plików (dokumentów) znajdujących się na dyskietce, z których jeden może zostać wprowadzony do pamięci komputera. Można również otworzyć istniejący dokument w taki sam sposób, w jaki został załadowany Geowrite (dwie możliwości). Przy takim postępowaniu Geowrite zostanie załadowany automatycznie.

2. Zawartość ekranu (okna) Geowrite

Po uruchomieniu edytora i wybraniu CREATE ekran Geowrite wygląda tak jak na rys. 1, przy założeniu, że nowo tworzony dokument został zatytułowany „Test".



Rys. 1. Ekran GEOWRITE

Rozwinięcie poszczególnych funkcji zawartych w menu wygląda następująco:

GEOS	FILE	EDIT
GEOWRITE INFO	CLOSE	CUT
CALCULATOR	UPDATE	COPY
PREFERENCE MGR	PREVIEW	PASTE
NOTE PAD	RECOVER	
PHOTO MANAGER	RENAME	
TEXT MANAGER	PRINT	
ALARM CLOK	QUIT	

OPTIONS	FONT	STYLE
PREVIOUS PAGE	BSW	PLAIN TEXT
NEXT PAGE	EWANS	BOLD
GOTO PAGE	HEARST	ITALIC
HIDE PICTURES	CORY	OUTLINE
PAGE BREAK	ROMA	UNDERLINE
LAST PAGE	ORMAND	
	DWINELLE	

3. Pisanie tekstu

Miejsce, w którym tekst jest pisany wskazuje kursor. Za pomocą wskaźnika kursor można ustawić w dowolnym miejscu (naciśnięcie przycisku *FIRE*). Poszczególne znaki mogą być usuwane za pomocą klawisza DEL lub SPACE. Piszący widzi tylko ok. 2/3 szerokości okna, stąd aby uniknąć męczącego oczy przechodzenia z prawej strony na lewą stronę ekranu i odwrotnie należy prawy margines przesunąć w lewo tak aby szerokość tekstu nie była większa niż 5 cali (patrz: formatowanie tekstu). Podczas pisania Geowrite przenosi oczywiście cały wyraz do następnego wiersza, jeżeli on nie mieści się w wierszu poprzednim. Naciśnięcie RETURN powoduje utworzenie akapitu. Tablicowanie odbywa się za pomocą klawiszy *CTRLi-I*. Pozostawienie części strony pustej (nie zapisanej) i przejście do następnej strony dokonuje się przez wybranie PAGE BREAK z menu OPTIONS. Przeglądanie zawartości strony — wybranie PREVIEW z menu FILE.

5 Poruszanie się po tekście (ekranie)

Poruszanie się po ekranie (dokumencie) w prawo — lewo odbywa się przez:

— ustawienie kursora za pomocą wskaźnika w dowolnym miejscu dokumentu,

— zaczepienie wskaźnika (naciśnięcie FIRE) na okienku reprezentującym ekran w SCROLL BOX

Poruszanie się po ekranie w górę — w dół

— przesuwanie o jeden wiersz — naciskanie *FIRE* na strzałkach przesuwu tekstu (okna),

— zaczepienie wskaźnika na okienku reprezentującym ekran w SCROLL BOX

Przejście do przodu lub cofnięcie się o jedną stronę

— wybranie *NEXT PAGE* lub *PREVIOUS PAGE* z menu OPTIONS

Przejście do dowolnej strony:

— wybranie GOTO PAGE z menu OPTIONS, podanie numeru strony i naciśnięcie RETURN.

6 Edycja tekstu

Wyróżnić tutaj można następujące funkcje

— wprowadzenie tekstu do tekstu istniejącego (napisanego wcześniej) — ustawienie kursora w żądanym miejscu w tekście istniejącym. Tekst wpisywany będzie powodował „rozsuwanie" tekstu istniejącego. Jeżeli istniejący tekst (dokument) jest wielostronicowy, to również nastąpią przesunięcia w kolejnych stronach,

— określenie (wybieranie) tekstu przeznaczonego do przeniesienia, kopiowania lub usunięcia — żądany tekst należy najpierw „podświetlić przesuwając wskaźnik przy wciśniętym przycisku *FIRE*. Przesuwanie poziome powoduje podświetlanie liter i słów. Przesuwanie pionowe podświetla kolejne wiersze.

Po podświetleniu tekstu (fragmentu) możliwe są następujące operacje:

— zastąpienie go innym tekstem wprowadzonym z klawiatury,

— usunięcie — naciśnięcie *DEL*,

— zmiana rodzaju lub stylu tekstu (*FONT i STYLE*),

— *CUT* — wycięcie fragmentu z jednoczesnym umieszczeniem go w TEXT SCRAP (wycinek tekstu),

— *COPY* — skopiowanie fragmentu do TEXT SCRAP bez usunięcia z miejsca podświetlonego,

— PASTE — przeniesienie zawartości TEXT SCRAP do tego miejsca w tekście, gdzie aktualnie znajduje się kursor.

7. **Ustawienie marginesów**

Możliwe jest dowolne ustawienie marginesów przez zaczepienie wskaźnika na symbolu marginesu (M) i przesunięcie w dowolne miejsce.

8. **Używanie różnych rodzajów i stylów pisma**

Do ustalania i zmiany rodzaju i stylu pisma służą funkcje FONT i STYLE. Dokonać tego można w dowolnej chwili, a zmian w tekście już istniejącym dokonuje się po uprzednim jego podświetleniu.

9. **Inne możliwości Geowrite**

— tablicowanie tekstu — Geowrite umożliwia zakładanie tablicy o ośmiu pozycjach. Dokonuje się tego po ustawieniu wskaźnika tablicy w żądanym miejscu w taki sam sposób, jak ustawia się margines,

— usuwanie pustego miejsca na stronie (miejsca powstałego po użyciu funkcji *PAGE BRAKE*) — ustawienie kursora w górnym lewym rogu strony następnej, naciśnięcie *DEL* i wybranie *OK* po ukazaniu się pytania, czy chcemy usunąć ostatni znak na poprzedniej stronie.

10. **Uwagi końcowe**

Maksymalna długość jednego tekstu (dokumentu) pisana za pomocą Geowrite wynosi 64 strony. Przy pisaniu długich tekstów korzystne jest posługiwanie się pismem normalnym (BSW) o wysokości 9 punktów i częste używanie funkcji *PAGE BREAK*.

Jeżeli podczas pracy w Geowrite obawiamy się przerwy w zasilaniu należy używać funkcji *UPDATE* w celu zapisania napisanych już fragmentów tekstu. Po zakończeniu pisania strony następuje jej automatyczne zapisanie na dyskietce.

W tekście pisanym za pomocą omawianego edytora można umieszczać rysunki wykonane za pomocą programu GEOPAINT zawartego również w GEOSIE.

Drukowanie dokumentu odbywa się przez wybranie *PRINT* z menu FILE. Po zakończeniu dokumentu należy go zamknąć wybierając *CLOSE* z menu FILE.

Proba pisma: font BSW, 9 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma: font ROMA , 9 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font ROMA, 12 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font ROMA, 18 pkt: plain, bold, *italic*, outline, underline
Proba pisma font ROMA, 24 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font University 6 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font University 10 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font University 12 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font University 14 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font University 18 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font University 24 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font Dwinelle 18 pkt : plain, **bold**, *italic* outline, underline
Proba pisma font Cory 12 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font Cory 24 pkt : plain, **bold**, *italic*, outline, underline
Proba pisma font California 10 pkt. plain, **bold**, *italic*, outline, underline
Proba pisma font California 12 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font California 14 pkt: plain, **bold**, *italic*, outline, underline
Proba pisma font California 18 pkt: plain, bold, *italic*, outline, underline

Rys. 2. Niektóre rodzaje i style pisma GEOWRITE.

11. **Ocena edytora**

Wady:

— brak numeracji stron,

— brak możliwości ustawienia odstępu między wierszami (liczby wierszy na stronie),

— konieczność posługiwania się joystickiem lub myszką,

— brak polskich liter.

Zalety:

— szybkie wybieranie poszczególnych funkcji i rozkazów,

— bardzo bogaty zestaw rodzajów i stylów pisma (rys. 2),

— możliwość łączenia tekstu z rysunkami wykonanymi za pomocą GEOPAINT.

T. CISEK

CPC • CPC • CPC • CPC • CPC • CPC • CPC • CPC

Ten program pozwala na uzyskanie ciekawego efektu animacji grafiki trójwymiarowej.

```
10   INK   0,0:INK   1,26:MODE
0:st=4:loop=14
20 FOR n=0 TO 640 STEP st
30 c=c+0.5:IF c=15.5 THEN c=1
40 PLOT 1000,1000,c:z=2*n-320
50  MOVE  z,200:DRAW   n,150:DRAW
n,40
60 MOVE z,0:DRAW n,40
70  MOVE  z,400:DRAW   n,350:DRAW
n,240
80 MOVE z,200:DRAW n,240
90 NEXT
100 INK 1,7:INK 15,0:FOR  n=2  TO
15:INK n,15:INK n-1,0:FOR t=0  TO
loop:NEXT:NEXT:GOTO 100
```

Jest to ciekawy program graficzny imitujący ruch ważki. Parametrem jest długość boku kwadratu (w punktach) będącego podstawowym elementem rysowanego obrazu.

```
10   DEFINT   a-z:MODE   1:INPUT
  "Dlugosc boku";x:CLS
20 DIM c(20):y=0
30 PLOT 320,200
40 DRAWR x,y:k=1
50 IF c(k)=0 THEN 70
60 c(k)=1:k=k+1:GOTO 50
70 c(k)=1:k=k+1
80 IF c(k)=0 THEN 100
90 z=x:x=y:y=-z:GOTO 40
100 z=x:x=-y:y=z:GOTO 40
```

Cezary SOBCZAK



— Czy mogę wypożyczyć Józia do ustawienia oprogramowania naszego komputera?

PROGRAM 55

Zmiana nazwy dyskietki

Program pozwala na dowolną zmianę nazwy dyskietki. Wykonywanie tej czynności jest czasem konieczne i pomaga utrzymać ład w naszej „dyskotece".

```
10 PRINT '■■ ZMIANA NAZWY DYSKIETKI'
20 PRINT '■ NR STACJI DYSKOW NR (8-15) ?
 8 ■■■■';
30 INPUT U: IF U<8 OR U>15 THEN PRINT '■
■■': GOTO 20
40 PRINT '■ DYSK DRIVE NR (0/1) ? 0■■■';

50 INPUT D: IF D<0 OR D>1 THEN PRINT '■■
■': GOTO 40
60 PRINT '■■ UMIESC DYSKIETKE W STACJI N
R';U;' DRIVE';D
70 PRINT ' I NACISNIJ DOWOLY KLAWISZ'
80 GET X$: IF X$=''THEN 80
90 A=1: V=16: Q$=CHR$(34): P$=CHR$(160)
100 T=18: S=0: I$='I'+RIGHT$(STR$(D),1)
110 OPEN 1,U,15,I$: GOSUB 350
120 GOSUB 310: F$=H$: GET#2,X$,X$,A$,B$
130 PRINT '■■■ AKTUALNA NAZWA ';Q$;F$;Q$

140 PRINT ' SYMBOL DYSKIETKI:  ';A$;B$

150 PRINT'■ NOWA NAZWA (MAX 16 ZNAKOW) L
UB          'STOP'-UTRZYMANIE STAREJ NAZ
WY'
160 INPUT ' ?STOP■■■■■■■';N$: L=LEN(N$):
IF L>V THEN 130
170 IF N$='STOP'   THEN PRINT '■';: GOTO
 370
180 PRINT '■ WPISANIE NOWEJ NAZWY (T/N)
? T■■■';
190 INPUT X$: IF X$<>'T' THEN 130
200 IF L=V THEN 220
210 FOR X=L+A TO V: N$=N$+P$: NEXT
220 PRINT#1,'B-P:';2;144: PRINT#2,N$;
230 PRINT#1,'U2:';2;D;T;S: GOSUB 350
240 PRINT#1,I$: GOSUB 350: CLOSE 2
250 PRINT '■ STARA NAZWA ';Q$;F$;Q$
260 GOSUB 310: CLOSE 2: CLOSE 1
270 PRINT '■ NOWA NAZWA ';Q$;H$;Q$
280 PRINT '■ NASTEPNA DYSKIETKA (T/N) ?
T■■■';
290 INPUT X$: IF X$='T' THEN 60

300 PRINT '■';: END
310 OPEN 2,U,2,'#': GOSUB 350: H$=''
320 PRINT#1,'U1:';2;D;T;S: GOSUB 350
330 PRINT#1,'B-P:';2;144: FOR X=A TO V
340 GET#2,T$: H$=H$+T$: NEXT: RETURN
350 INPUT#1,E,M$,J,K:IF E=0 THEN RETURN
360 PRINT '■ BLAD: ';E;M$;J;K
370 CLOSE 2: CLOSE 1: END
READY.
```

Opr. Tadeusz CISEK

Można inaczej

Poznań, dnia 17 marca 1988 r.

Po zapoznaniu się z artykułem p. Janusza Morbitzera „Generowanie liczb pierwszych — sito Eratostenesa" w zeszycie 1/1988 „IKS"-a nasunęło mi się kilka uwag, o których chciałbym poinformować Redakcję i czytelników.

We wspomnianym artykule liczba $2^{11213}-1$ została podana jako największa znana liczba pierwsza. Tymczasem jest ona zaledwie dwudziestą trzecią kolejną liczbą pierwszą Mersenne'a, a w roku 1985 odkryta została już trzydziesta kolejna liczba pierwsza Mersenne'a, mianowicie liczba $2^{216091}-1$. Aktualne informacje o dużych liczbach pierwszych podawał niedawno „Komputer", publikując list do redakcji profesora dr. hab. Władysława Narkiewicza (zeszyt 1/88) oraz mój list do redakcji (zeszyt 9/87).

Program podany przez p. Janusza Morbitzera ma dwie wady, mianowicie działa stosunkowo wolno oraz oblicza liczby pierwsze nie większe od 7000. Napisałem krótki program w Basicu przeznaczony dla **Commodore 64**, który wykorzystuje przyspieszoną wersję sita Eratostenesa. Po uruchomieniu programu za pomocą **RUN1** należy wprowadzić dowolną liczbę naturalną *N* nie przekraczającą 58127. Po naciśnięciu klawisza **RETURN** komputer podaje na ekranie wszystkie kolejne liczby pierwsze do *N*, a czasem i dalsze.

Przypuszczam, że załączony program zainteresuje młodych czytelników „IKS"-a.

Z poważaniem

Andrzej WIĘCKOWSKI

```
0 FORJ=JTONSTEPP+P:A%(J)=N:NEXT:RETURN
1 INPUTN:PRINT2:PRINT3:N=N/3:DIMA%(N):FORI=.TON:IFA%(I)GOTO3
2 P=I*3+5-(IAND1):PRINTP:J=I:GOSUB:J=P+P-I-3:IFJ<=NTHENGOSUB
3 NEXT

READY.
```

W kolejnych częściach artykułu chcemy przedstawić czytelnikom podstawowe funkcje sekwencera, zaproponować organizację programu i współpracę komputera z urządzeniami muzycznymi. Jest to bardzo rozległy problem i dlatego całość podzieliliśmy na odrębne części tematyczne. Ponieważ tylko pewna część zainteresowanych posiada ZX Spectrum (a na tym komputerze został napisany prezentowany sekwencer) postaramy się dokładnie opisać program, aby łatwo go było zaimplementować na inne komputery.

NUTY Z KOMPUTERA (2)

Krzysztof POŹNIAK, Michał ROLLINGER, Jarosław ZIEMBICKI

EDYTOR TAKTU

I. Informacje podstawowe:

a) Krok w takcie jest najmniejszą możliwą do wykonania jednostką rytmiczną. W zależności od stosowanego metrum i wartości rytmicznych uzyskujemy następujące ilości kroków w takcie:



rys. 1

b) Edytor dopuszcza takt składający się maksymalnie z 256 kroków i praktycznie nieograniczonej ilości ścieżek. Taka organizacja zapisu ukazuje pole edytora jako tablicę. W jego każdej komórce możemy zapisać informacje o wysokości jednego dźwięku.

c) Formy zapisu wysokości dźwięku:

1) **zapis literowy** — obejmuje dźwięki od C2 (C subkontry) do g7 (g siedmiokreślnego). Jest to zakres z nadmiarem, w praktyce wykorzystujemy dźwięki od A2 do c5 (klawiatura fortepianu). Ponieważ w pamięci komputera wysokość dźwięku jest zapisywana jako liczba, stąd nie możemy stosować różnych nazw dla tego samego dźwięku. Wykorzystujemy następującą skalę:

najmniejsza wartość rytmiczna:	♩	♪	𝅘𝅥𝅯	𝅘𝅥𝅰
metrum 2/4	2	4	8	16
metrum 3/4	3	6	12	24
metrum 4/4	4	8	16	32
metrum 5/4	5	10	20	40

rys. 2

Jeżeli w danym materiale nutowym występują dźwięki obniżone, to musimy zmienić je enharmonicznie na dźwięki podwyższone:

des = cis
es = dis
ges = fis
as = gis
b = dis

uwaga: ze względów graficznych dźwięki podwyższone są zapisywane jako dźwięki podstawowe z kreską nad literą.

2) **zapis graficzny** — jest używany do sterowania urządzeniami perkusyjnymi. Znak [ ] oznacza, że w danym kroku nastąpi wyzwolenie instrumentu

3) **zapis cyfrowy** — pozwala na wprowadzenie liczb z zakresu <0÷127>. Ta forma zapisu jest wygodna do pewnych specyficznych zastosowań, a jej wykorzystanie opiszemy w kolejnych częściach artykułu

uwaga: zapis literowy i cyfrowy dopuszczają dźwięki przedłużone (zapisywane jako —) oraz pauzy (oznaczane **)

d) struktura pola ekranowego edytora taktu:

1) po polu ekranowym poruszamy się kursorem za pomocą klawiszy:

klawisz <5> — lewo
klawisz <6> — dół
klawisz <7> — góra
klawisz <8> — prawo

2) ponieważ ilość kroków taktu może przekroczyć pojemność ekranu (maksymalna liczba wyświetlanych kroków wynosi 10), stąd takt dzieli się na strony. Stronicowanie pozwala nam (w zależności od długości taktu) na jego logiczny podział. Takt przegląda się stronami za pomocą klawiszy:

klawisz <4> — cofnięcie się o stronę
klawisz <9> — przejście na następną stronę

3) informacje dodatkowe:

— najwyższa linia ekranowa podaje nazwę taktu i jego stan ilościowy:

is = aktualna ilość ścieżek w takcie
ik = aktualna ilość kroków w takcie
s = numer wyświetlanej strony

— najniższa linia opisuje ścieżkę, na której znajduje się kursor

e) ścieżkę opisują następujące parametry:

1) nazwa instrumentu — ścieżka jest przypisana do jednego z obsługiwanych przez sekwencer instrumentów

2) nazwa źródła — ścieżka może obsługiwać tylko jedno źródło dźwięku w przyporządkowanym jej instrumencie

3) tryb zapisu — w zależności od rodzaju urządzenia muzycznego posługujemy się zapisem literowym, graficznym lub cyfrowym

II. Funkcje edytora:

uwaga: każdą funkcję edytora możemy przerwać naciskając klawisz <P>

klawisz <z> — umożliwia zapis dźwięku w miejscu położenia kursora, wg zadeklarowanego trybu:

a) tryb literowy — obejmuje dźwięki od C2 do g7. Najpierw wprowadza się literę (dużą lub małą), jeśli ma to być dźwięk podwyższony, to kolejnym krokiem jest wciśnięcie klawisza <I> (od przyrostka „is"); zapis kończymy wprowadzeniem cyfry

b) tryb graficzny — dopuszcza zapis uderzenia (klawisz <i>) lub pauzy (klawisz <B>)

c) tryb cyfrowy — pozwala na wprowadzenie liczby z zakresu <0÷127>. Ponieważ w polu pojedynczego dźwięku mieszczą się jedynie dwa znaki, stąd liczby od 100 wzwyż są wyświetlane w postaci dwóch ostatnich cyfr z kreską nad pierwszą cyfrą. Zakończenie wpisywania liczby uzyskujemy naciskając klawisz <SPACE>.

uwaga: w trybie literowym i cyfrowym pauzę uzyskujemy klawiszem <SPACE>, a przedłużenie dźwięku klawiszem <k>

uwaga: każda poniżej opisywana funkcja wymaga podania odpowiednich parametrów w postaci liczb. Po wprowadzeniu liczb należy wcisnąć klawisz <SPACE>. Ukaże się wtedy napis (t/n); wciśnięcie klawisza <t> wprowadzi daną informację do pamięci

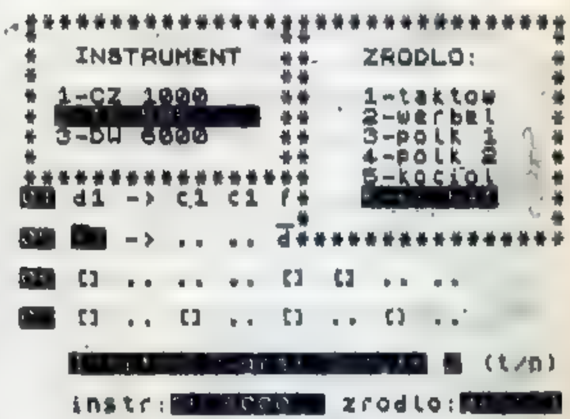
klawisz <c> — umożliwia czyszczenie (wymazanie zawartości) podanej ścieżki
parametry: 1) numer ścieżki

klawisz <s> — skasowanie ścieżki o podanym numerze. Ilość ścieżek zmniejsza się o jedną
parametry: 1) numer ścieżki

klawisz <d> — dołączenie (wsunięcie) nowej ścieżki na pozycję o podanym numerze. Wszystkie ścieżki, począwszy od podanego numeru przesuną się o jedną pozycję w dół. Ich ilość wzrośnie o jedną.

uwaga: jeśli podany numer jest o jeden większy od ilości ścieżek, to nowa ścieżka zostanie dopisana na końcu

parametry: 1) numer ścieżki
2) numer instrumentu (wg tabeli)
3) numer źródła wybranego instr. (wg tabeli)
4 tryb zapisu



rys. 3

na rys. 3 przedstawiamy stan ekranu podczas wprowadzania czwartego parametru — trybu zapisu (wybraliśmy tryb graficzny — cyfra 2) po uprzednim wyborze instrumentu: TR 505 i źródła: hi-hat (nazwy w trybie inwersyjnym)

klawisz <w> — wymiana dwóch ścieżek miejscami. Funkcja ma charakter wyłącznie porządkowy
parametry: 1 numer pierwszej ścieżki (mniejszy)

2) numer drugiej ścieżki (większy)

klawisz <p> — powielanie treści jednej ścieżki na drugą. Warunkiem koniecznym wykonania funkcji jest zgodność typów zapisów

parametry: 1) numer ścieżki-źródła
2) numer ścieżki-odbiorcy

klawisz <r> — rozszerzenie taktu na zasadzie „mnożenia" pozwala, po jego napisaniu, na wprowadzenie mniejszych wartości rytmicznych niż początkowo zamierzaliśmy. Funkcja dopisuje pauzy i zmienia długości dźwięków. Przykładowe rozszerzenie taktu razy 2 przedstawiliśmy na rysunkach: nr 4 — stan wyjściowy i nr 5 — po wykonaniu funkcji

parametry: 1) mnożnik taktu

klawisz <R> — pozwala na wstawienie zadanej ilości dodatkowych kroków w określone miejsce taktu

uwaga: jeżeli numer kroku, od którego będziemy dołączać nowe kroki będzie o jeden większy od ich aktualnej liczby, to nowe kroki dołączą się na końcu taktu

parametry: 1) numer kroku, od którego nastąpi wstawienie
2) ilość dołączanych kroków

klawisz <k> — kompresja taktu na zasadzie „dzielenia" — działa odwrotnie do funkcji <r>

parametry: 1) podzielnik taktu

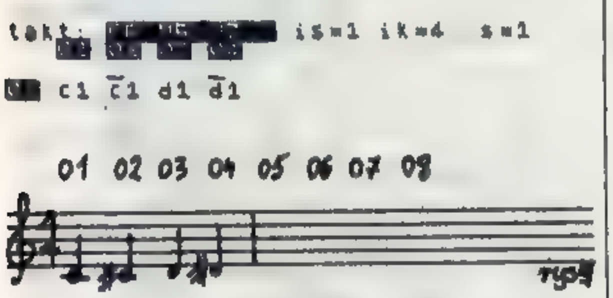
klawisz <K> — kasowanie grupy kroków — działa odwrotnie do funkcji <R>

parametry: 1) numer kroku, od którego nastąpi kasowanie
2) ilość kasowanych kroków

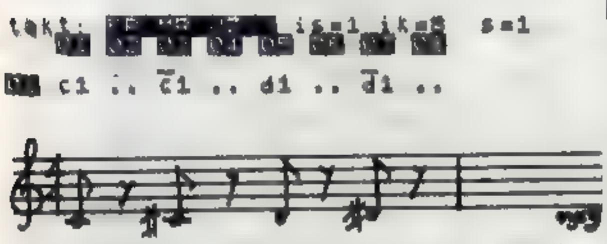
klawisz <i> — określa ile kroków zawiera jedna strona taktu (w ZX Spectrum maksymalna liczba kroków na stronie wynosi 10)

parametry: 1) ilość kroków na jednej stronie
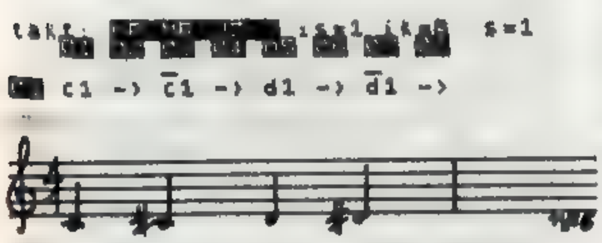
klawisz <t> lub <T> — transpozycja ścieżki (w górę — klawisz <t>, dół — klawisz <T>) o zadany interwał (ilość półtonów)



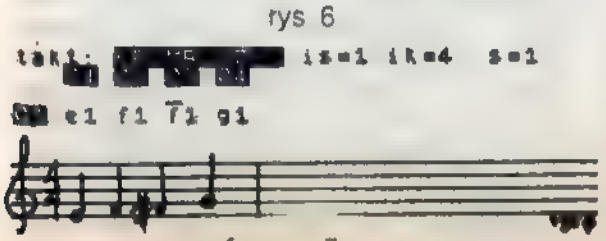
rys 4



rys 5



rys 6



rys 7

parametry: 1) numer transponowanej ścieżki
2) ilość półtonów

uwaga: transpozycja pozwala na zmianę tonacji bez żmudnego poprawiania wszystkich dźwięków

uwaga: jeżeli podczas transpozycji dźwięk przekroczy zakres C2÷g7, to na jego miejsce zostanie wprowadzona pauza

Przykładową zmianę tonacji o cztery półtony w górę przedstawiliśmy na rysunkach: nr 4 — stan wyjściowy i nr 7 — po wykonaniu transpozycji

klawisz <j> — rozszerzenie jednostki rytmicznej — funkcja podobna do <r>, jednak z tą różnicą, że zachowuje długości dźwięków

parametry: 1) mnożnik jednostki rytmicznej

Działanie tej funkcji jest uwidocznione na rysunkach: nr 4 — stan początkowy i nr 6 — sytuacja po przypisaniu wyjściowej jednostce rytmicznej dwóch kroków

uwaga: jeśli wywołujemy funkcje za pomocą dużych liter: <C>, <S>, <D>, <W>, <P>, to pierwszy parametr jest określony przez aktualną pozycję kursora

Na rysunku numer 8 został przedstawiony fragment utworu muzycznego w zapisie sekwencerowym i nutowym. Warto zwrócić uwagę na to, że „takt sekwencerowy" nie musi się pokrywać z taktem muzycznym.

3) **i$** — posiada nazwy dostępnych instrumentów muzycznych i informacje o niezależnych źródłach wchodzących w skład instrumentów

4) **s$** — przechowuje nazwy źródeł wszystkich instrumentów

b) najważniejsze zmienne proste:

ii — ilość dostępnych instrumentów (w programie przykładowo przyjęliśmy: ii = 3)

cike — całkowita ilość kroków na ekranie (dla ZX Spectrum cike = 10)

cise — całkowita ilość ścieżek na ekranie (dla ZX Spectrum cise = 8)

is — aktualna ilość ścieżek

ik — aktualna ilość kroków
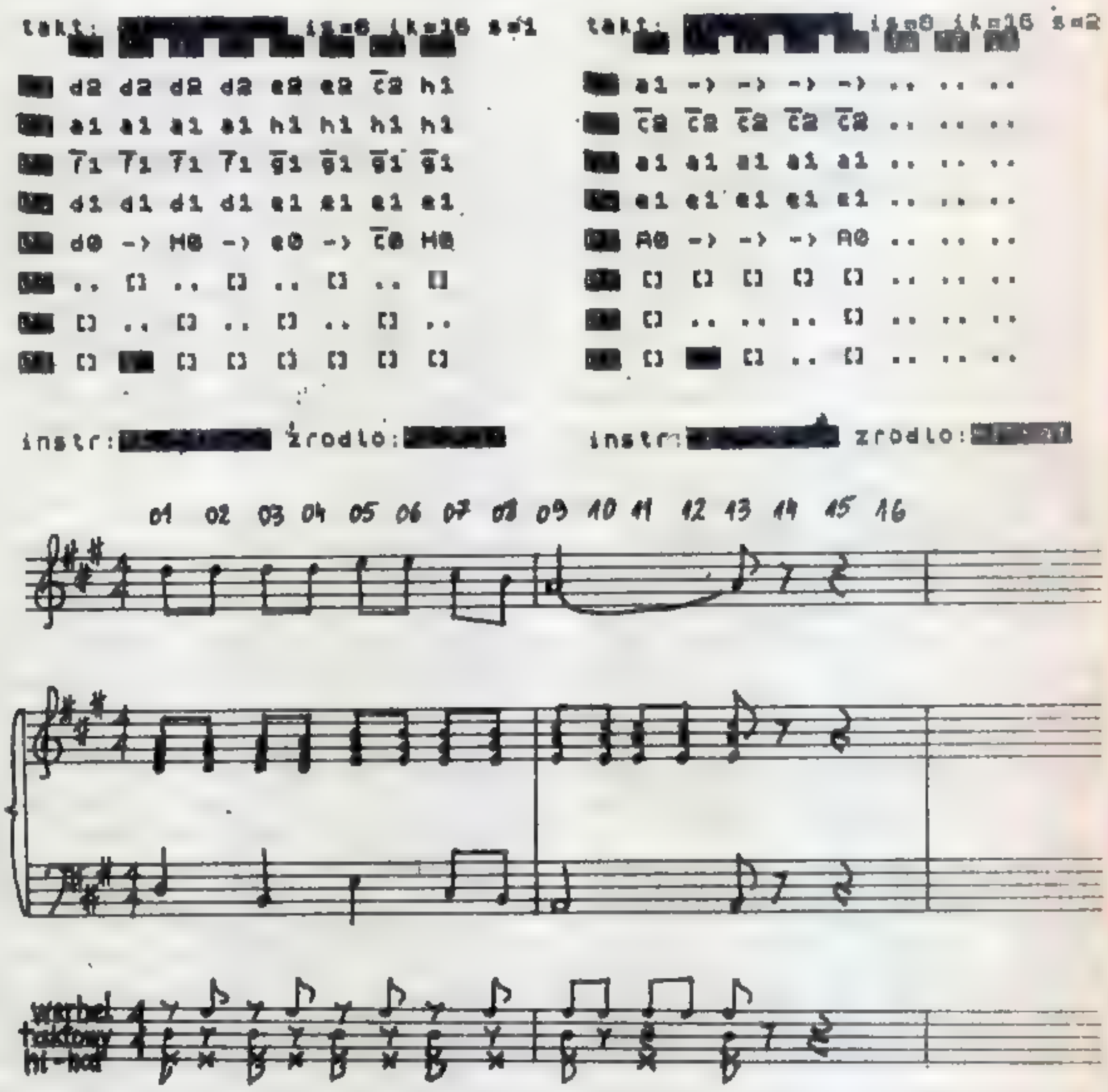
pns — najmniejszy (początkowy) nr ścieżki wyświetlany na ekranie

kns — największy (końcowy) nr ścieżki wyświetlany na ekranie

pnk — najmniejszy (początkowy) nr kroku wyświetlany na ekranie

kns — największy (końcowy) nr kroku wyświetlany na ekranie

mik — maksymalna ilość kroków (ustalana funkcją <i>)



rys 8

III. Zmienne programu:

a) dane edytora taktu są zapisywane w zmiennych tekstowych (wg schematów z rysunku numer 9)

1) **e$** — przechowuje dźwięki taktu (kolejno ścieżka po ścieżce)

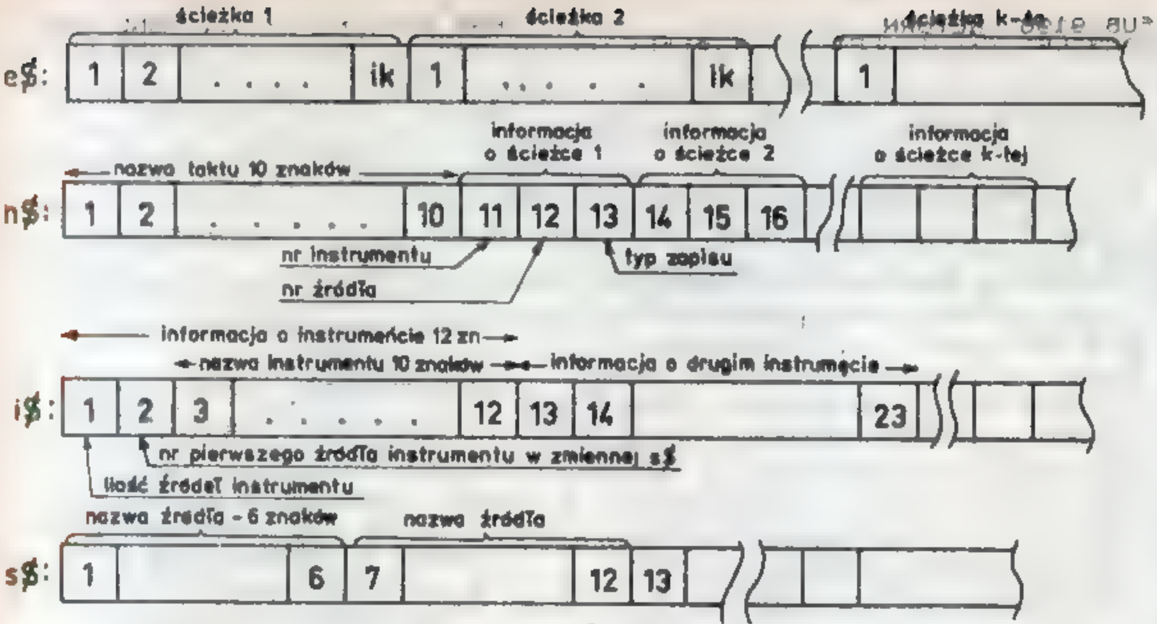
2) **n$** — zawiera nazwę taktu i informacje o każdej ścieżce

eik — ilość kroków aktualnie wyświetlanych na ekranie

eis — ilość ścieżek aktualnie wyświetlanych na ekranie

str — numer aktualnie wyświetlanej strony

kx, ky — współrzędne kursora



rys. 9

Ponieważ rozbudowa programu spowoduje pewne zmiany w prezentowanym listingu zachęcamy czytelników, którzy będą wprowadzać program do dokładnego jego odwzorowania.

```
   1 REM
*******************************
       EDYTOR    TAKTU
*******************************
   2 DEF FN r(x)=CODE n$(10+3*(pns+x-1))
   3 DEF FN m(y,x)=x*(x<y)+y*(x>=y)
   4 DEF FN o(q)=ik*(q-1)+1
   5 DEF FN t(q)=11+(q-1)*3
   6 DEF FN a(y,x)=ik*(pns+y-2)+mik*(str-1)+x
  10 INK 7: PAPER 0: CLS : BORDER 0
  15 REM
  *******************************
**
  20 LET ii=3: LET i$=CHR$ 4+CHR$ 1+"CZ 1000   "+CHR$ 6+CHR$ 5+"TR 505    "+CHR$ 4+CHR$ 11+"DW 6000   "
  21 LET s$="gener1gener2gener3gener4taktowwerbelpolk 1polk 2kociolhi hatgener1gener2gener3gener4"
  30 LET n$=CHR$ 1+CHR$ 1+CHR$ 0
  31 LET n$="KP MR JZ   "+n$
  40 LET e$=CHR$ 128: LET cike=10: LET cise=8: LET is=1: LET ik=1
  50 LET t$="c0c1d0d1e0f0f1g0g1a0a1h021001234567"
  60 REM
*******************************
  99 GO TO 1000
 100 REM
 102 CLS : PRINT AT 0,0;"takt: "; INVERSE 1;n$( TO 10); INVERSE 0;TAB 17;"is=";is;TAB 22;"ik=";ik;TAB 28;"s=";str;TAB 0;
 103 LET pnk=(str-1)*mik+1: PRINT "  ";: LET knk=pnk+eik-1: FOR l=pnk TO knk: PRINT " "; INVERSE 1;"0" AND (l<10);l; INVERSE 0;: NEXT l
 104 LET da=ik-eik: LET flash=0: LET a=FN a(1,1)
 110 LET kns=pns+eis-1: FOR y=1 TO eis: LET rodz=FN r(y): PRINT AT 2*y+1,0; INVERSE 1;"0" AND ((y+pns-1)<10);pns+y-1;: FOR x=1 TO eik
 112 GO SUB 130: LET a=a+1: NEXT x: LET a=a+da: NEXT y: RETURN
 120 REM
 121 LET a=FN a(ky,kx)
 125 LET rodz=FN r(ky)
 130 LET q=CODE e$(a): IF rodz=0 AND q<128 THEN LET m=INT (q/12): LET n=q-12*m: LET y$=(" " AND t$(2*n+2)="0")+("_" AND t$(2*n+2)="1"): LET x$=CHR$ (CODE t$(2*n+1)-32*(m<3))+t$(26+m): GO TO 140
 131 IF rodz=0 THEN LET x$=("->" AND q=129)+(".." AND q<>129): LET y$=" ": GO TO 140
 135 IF rodz=2 THEN LET n=INT (q/100): LET m=q-100*n: LET y$=(" " AND NOT n)+("_" AND n): LET x$=("0" AND m<10)+STR$ m: LET y$=(y$ AND q<=127)+(" " AND q>127): LET x$=(x$ AND q<=127)+(".." AND q=128)+("->" AND q=129): GO TO 140
 137 LET y$=" ": LET x$=(".." AND q<>1)+("[]" AND q=1)
 140 PRINT AT 2*y,3*x;y$;AT 2*y+1,3*x; FLASH flash;x$;: RETURN
 150 REM
 151 LET by=pnk+ky-1: LET n=5+3*(pns+ky): LET aki=CODE n$(n): LET aks=CODE n$(n+1): PRINT AT 21,0;"instr:"; INVERSE 1;i$(3+12*(aki-1) TO 12*aki-1); INVERSE 0;" zrodlo:"; INVERSE 1;s$(1+(aks-1)*6 TO aks*6)
 153 LET x=kx: LET y=ky: LET inv=0: LET flash=1: GO TO 120
 160 REM
 162 LET x=kx: LET y=ky: LET flash=0: GO TO 120
 170 REM
 172 LET str=FN m(str,INT (ik/mik)+1): LET pnk=mik*(str-1)+1
 174 LET eik=FN m(ik-mik*(str-1),mik)
 176 LET kx=FN m(kx,eik)
 178 GO TO 100
9000 REM
*******************************
9010 CLS
9020 LET pns=1: LET kx=1: LET ky=1: LET mik=FN m(ik,cike): LET eik=mik: LET eis=FN m(is,cise): LET str=1: GO SUB 100
9024 REM
9025 GO SUB 9198: GO SUB 150
9026 LET x$=INKEY$: IF x$="" THEN GO TO 9026
9030 IF x$="5" THEN GO SUB 160: GO SUB 9100: GO TO 9025
9031 IF x$="4" THEN GO SUB 160: GO SUB 9105: GO TO 9025
9032 IF x$="8" THEN GO SUB 160: GO SUB 9110: GO TO 9025
9033 IF x$="9" THEN GO SUB 160: GO SUB 9115: GO TO 9025
9034 IF x$="7" THEN GO SUB 160: GO SUB 9120: GO TO 9025
9036 IF x$="6" THEN GO SUB 160: GO SUB 9130: GO TO 9025
9040 IF x$="z" THEN GO SUB 9300: GO TO 9025
9041 IF x$="c" THEN LET q=0: GO SUB 9140: GO TO 9025
9042 IF x$="C" THEN LET q=by: GO SUB 9140: GO TO 9025
9043 IF x$="s" THEN LET q=0: GO SUB 9145: GO TO 9025
9044 IF x$="S" THEN LET q=by: GO SUB 9145: GO TO 9025
9045 IF x$="d" THEN LET q=0: GO SUB 9150: GO TO 9025
9046 IF x$="D" THEN LET q=by: GO SUB 9150: GO TO 9025
9047 IF x$="w" THEN LET q=0: GO SUB 9155: GO TO 9025
9048 IF x$="W" THEN LET q=by: GO SUB 9155: GO TO 9025
9049 IF x$="p" THEN LET q=0: GO SUB 9160: GO TO 9025
9050 IF x$="P" THEN LET q=by: GO SUB 9161: GO TO 9025
9051 IF x$="r" THEN LET q=0: GO SUB 9165: GO TO 9025
9052 IF x$="R" THEN LET q=0: GO SUB 9170: GO TO 9025
9053 IF x$="k" THEN LET q=0: GO SUB 9175: GO TO 9025
9054 IF x$="K" THEN LET q=0: GO SUB 9200: GO TO 9025
9055 IF x$="i" THEN LET q=0: GO SUB 9205: GO TO 9025
9056 IF x$="t" THEN LET q=0: GO SUB 9210: GO TO 9025
9057 IF x$="T" THEN LET q=0: GO SUB 9215: GO TO 9025
9058 IF x$="j" THEN LET q=0: GO SUB 9220: GO TO 9025
9060 GO TO 9025
9099 REM
9100 IF kx>1 THEN LET kx=kx-1: GO TO 153
9102 IF pnk>1 THEN LET kx=mik: GO TO 9107
9103 BEEP .1,20: GO TO 153
9105 IF pnk=1 THEN BEEP .1,20: GO TO 150
9107 LET str=str-1: LET eik=mik: GO TO 100
9110 IF kx<eik THEN LET kx=kx+1: GO TO 153
9112 IF knk<ik THEN LET kx=1: GO TO 9117
9114 BEEP .1,20: GO TO 153
9115 IF knk>=ik THEN BEEP .1,20: GO TO 150
9117 LET str=str+1: LET eik=FN m(mik,ik-knk): LET kx=FN m(kx,eik): GO TO 100
9120 IF ky>1 THEN LET ky=ky-1: GO SUB 150: RETURN
9122 IF pns>1 THEN LET pns=pns-1: GO TO 100
9123 BEEP .1,20: GO TO 150
9130 IF ky<eis THEN LET ky=ky+1: GO TO 150
9132 IF kns<is THEN LET pns=pns+1: GO TO 100
9133 BEEP .1,20: GO TO 150
9140 LET md=is: LET z$="czyszczenie sciezki nr:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9141 LET k=FN o(q): LET m=k+ik-1: FOR l=k TO m: LET e$(l)=CHR$ 128: NEXT l: GO TO 100
9145 LET md=is: LET z$="skasowanie sciezki nr:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9146 IF is=1 THEN GO TO 9141
9147 LET ky=ky-(is<=cise)*(ky>1): LET pns=pns-(is>cise)*(is=pns+cise-1 OR q<pns)
9148 LET k=FN o(q): LET e$=e$( TO k-1)+e$(k+ik TO ): LET k=FN t(q): LET n$=n$( TO k-1)+n$(k+3 TO ): LET is=is-1: LET eis=FN m(is,eis): GO SUB 100: IF is<cise THEN PRINT AT 2*is+2,0;TAB 31;" ";TAB 31;" "
9149 GO TO 9198
9150 LET md=is+1: LET z$="dolacz. sciez. nr:": GO SUB 9190: IF q=0 THEN GO TO 9198
9151 LET ans=q: GO SUB 9180: LET md=ii: LET q=0: LET z$="podaj nr instumentu:": GO SUB 9190: IF q=0 THEN GO TO 100
9152 LET ani=q: PRINT AT 3+q,2; INVERSE 1;q;"-";i$(12*(q-1)+3 TO 12*q): LET aai=1+12*(q-1): LET ais=CODE i$(aai TO aai): LET asp=CODE i$(aai+1 TO aai+1): GO SUB 9185: LET md=ais: LET q=0: LET z$="podaj nr zrodla:": GO SUB 9190: IF q=0 THEN GO TO 100
9153 PRINT AT 3+q,20; INVERSE 1;q;"-";s$(6*(asp+q-2)+1 TO 6*(asp+q-1)): LET ansi=q: LET md=3: LET q=0: LET z$="1-liter 2-graf 3-cyfr": GO SUB 9190: IF q=0 THEN GO TO 100
9154 LET k=FN o(ans): LET z$="": FOR n=1 TO ik: LET z$=z$+CHR$ 128: NEXT n: LET e$=e$( TO k-1)+z$+e$(k TO ): LET k=FN t(ans): LET n$=n$( TO k-1)+CHR$ ani+CHR$ (asp+ansi-1)+CHR$ (q-1)+n$(k TO ): LET eis=eis+(is<cise): LET is=is+1: GO TO 100
9155 LET md=is: LET z$="wymiana sciez. nr:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9156 LET ans=q: LET md=is: LET q=0: LET z$="wym sc. "+STR$ ans+" ze sc.:": GO SUB 9190: IF q=0 OR q<ans THEN GO TO 9198
9157 LET k=FN o(ans): LET z$=e$(k TO k+ik-1): LET n=FN o(q): LET e$=e$( TO k-1)+e$(n TO n+ik-1)+e$(k+ik TO ): LET e$=e$( TO n-1)+z$+e$(n+ik TO )
9158 LET k=FN t(ans): LET z$=n$(k TO k+2): LET n=FN t(q): LET n$=n$( TO k-1)+n$(n TO n+2)+n$(k+3 TO ): LET n$=n$( TO n-1)+z$+n$(n+3 TO ): GO TO 100
9160 LET md=is: LET z$="powiel. sciez. nr:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9161 LET ans=q: LET md=is: LET q=0: LET z$="powiel. sc. "+STR$ ans+" na sc.:": GO SUB 9190: IF q=0 THEN GO SUB 100: GO SUB 9198: RETURN
9162 LET k=FN t(ans): LET n=FN t(q): IF n$(k+2)<>n$(n+2) THEN PRINT AT 19,0; FLASH 1;"NIEZGODNOSC TYPOW ZAPISU - Enter": PAUSE 1: PAUSE 0: GO TO 9198
9163 LET k=FN o(ans): LET n=FN o(q): LET e$=e$( TO n-1)+e$(k TO k+ik-1)+e$(n+ik TO ): GO TO 100
9165 LET md=INT (256/ik): LET z$="rozszerzenie t. razy:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9166 LET z$="": FOR k=1 TO q-1: LET z$=z$+CHR$ 128: NEXT k: LET l=LEN e$*q: FOR k=1 TO l STEP q: LET e$=e$( TO k)+z$+e$(k+1 TO ): NEXT k: LET ik=ik*q: GO TO 170
9170 LET md=ik+1: LET z$="rozszerzenie na kroku:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9171 LET k=q: LET q=0: LET md=128-ik: LET z$="ile krokow:": GO SUB 9190: IF q=0 THEN GO SUB 9198: RETURN
9172 LET z$="": FOR m=1 TO q: LE
```

```
T z$=z$+CHR$ 128: NEXT m: LET l=
(ik+q)*is: FOR k=k-1 TO l STEP i
k+q: LET e$=e$( TO k)+z$+e$(k+1
TO ): NEXT k: LET ik=ik+q: GO TO
 170
9175 LET md=ik: LET z$="kompresj
a taktu razy:": GO SUB 9190: IF
q=0 THEN GO SUB 9198: RETURN
9176 LET k=INT (ik/q): IF ik/q<>
k THEN PRINT AT 19,0; FLASH 1;"
PODZIAL  NIEPRAWIDLOWY - Enter "
: PAUSE 1: PAUSE 0: GO TO 9198
9178 LET ik=k: FOR k=1 TO ik*is:
 LET e$=e$( TO k)+e$(q+k TO ): N
EXT k: GO TO 170
9179 REM
9180 LET ax=0: LET k=0. GO SUB 9
183: GO SUB 9184: GO SUB 9184: P
RINT AT 2,3;"INSTRUMENT": GO SUB
 9184: FOR k=4 TO ii+3: PRINT AT
 k,0;"* ";k-3;"-";i$(12*(k-4)+3
TO 12*(k-3));TAB 15;"*": NEXT k:
 GO SUB 9184: GO SUB 9183
9182 RETURN
9183 PRINT AT k,ax;"************
****": LET k=k+1: RETURN
9184 PRINT AT k,ax;"*",TAB ax+15
;"*": LET k=k+1: RETURN
9185 LET ax=16: LET k=0: GO SUB
9183: GO SUB 9184: GO SUB 9184:
PRINT AT 2,20;"ZRODLO:": GO SUB
9184: FOR k=4 TO ais+3: PRINT AT
 k,16;"*   ";k-3;"-";s$(6*(asp+k
-5)+1 TO 6*(asp+k-4));TAB 31;"*"
: NEXT k: GO SUB 9184: GO SUB 91

83
9188 RETURN
9189 REM
9190 GO SUB 9198: PRINT AT 19,0;
 INVERSE 1;z$;: PRINT ;" ";: IF
q<>0 THEN PRINT ;q;: LET x$=" ":
 GO TO 9195
9191 LET q=0
9192 LET x$=INKEY$: IF x$="" THE
N GO TO 9192
9193 IF x$="p" THEN LET q=0: RET
URN
9194 IF NOT (x$>="1" AND x$<="9"
 OR x$=" " AND q<>0 OR x$="0" AN
D q<>0) THEN GO SUB 9326: GO TO
9192
9195 IF x$=" " THEN PRINT ;" ";
FLASH 1;"(t/n)": PAUSE 1: PAUSE
0:: LET x$=INKEY$: LET q=q*(x$="
t"): BEEP .1,30: RETURN
9196 LET q=q*10+VAL x$: IF q>md
THEN LET q=0: GO SUB 9326: GO TO
 9190
9197 PRINT INVERSE 1;x$;: GO TO
9192
9198 REM
9199 PRINT AT 19,0,TAB 31;" ": R
ETURN
9200 IF ik=1 THEN RETURN
9201 LET md=ik: LET z$="kasowani
e od kroku:": GO SUB 9190: IF q=
0 THEN GO TO 9198
9202 LET k=q: LET md=ik-q+1: LET
 q=0: LET z$="ile krokow kasowac
:": GO SUB 9190: IF q=0 THEN GO
SUB 9198: RETURN
9203 LET l=k-1: LET n=(ik-q)*is:
 FOR m=l TO n STEP ik-q: LET e$=
e$( TO m)+e$(q+m+1 TO ): NEXT m:
 LET ik=ik-q: GO TO 170
9205 LET md=cike: LET z$="ilosc
krok. na stronie:": GO SUB 9190:
 IF q=0 THEN GO TO 9198
9207 LET mik=q: GO TO 170
9210 LET md=is: LET z$="nr sciez
ki transpon.:": GO SUB 9190: IF
q=0 THEN GO TO 9198
9211 LET k=q: LET q=0: LET md=12
7: LET z$="transpozycja w gore o
:": GO SUB 9190: IF q=0 THEN GO
TO 9198
9212 LET l=FN o(k): FOR m=l TO i
k+l-1: LET n=CODE e$(m): IF n>12
7 THEN GO TO 9214
9213 LET n=n+q  IF n>127 OR n<0
THEN LET n=128
9214 LET e$(m)=CHR$ n: NEXT m: G
O TO 100
9215 LET md=is: LET z$="nr sciez
ki transpon.:": GO SUB 9190: IF
q=0 THEN GO TO 9198
9216 LET k=q: LET q=0: LET md=12
7: LET z$="transpozycja w dol o:
": GO SUB 9190: IF q=0 THEN GO T
O 9198
9217 LET q=-q: GO TO 9212

9220 LET md=INT (256/ik): LET z$
="rozsz. j. rytm. razy:": GO SUB
 9190: IF q=0 THEN GO SUB 9198:
RETURN
9222 LET x$="": LET z$="": FOR k
=1 TO q-1: LET x$=x$+CHR$ 129: L
ET z$=z$+CHR$ 128: NEXT k: LET l
=LEN e$*q: FOR k=1 TO l STEP q:
LET e$=e$( TO k)+(z$ AND e$(k)=C
HR$ 128)+(x$ AND e$(k)<>CHR$ 128
)+e$(k+1 TO ): NEXT k: LET ik=ik
*q: GO TO 170
9300 REM
9310 PRINT AT 19,0; INVERSE 1;"z
apis ";"literowy" AND NOT rodz;"
graficzny" AND rodz=1;"liczbowy"
 AND rodz=2
9320 LET m=CODE e$(FN a(ky,kx)):
 LET y$=" ": LET cis=0: GO SUB
9325
9321 GO SUB 9330+(10*rodz)
9322 LET e$(FN a(ky,kx))=CHR$ m:
 LET powr=(x$="p"): GO SUB 160:
IF powr THEN RETURN
9323 GO SUB 9110: IF ky=1 THEN G
O SUB 150
9324 GO TO 9320
9325 PRINT AT ky*2,3*kx;" " AND
NOT cis,"_" AND cis;AT ky*2+1,3*
kx; FLASH 1;y$: RETURN
9326 BEEP .1,20: BEEP .1,30: RET
URN
9330 GO SUB 9365: IF x$="p" THEN
 RETURN
9331 IF NOT (x$=" " OR x$="k" OR
 x$="a" OR x$="A" OR x$>="c" AND
 x$<="h" OR x$>="C" AND x$<="H")
 THEN GO SUB 9326: GO TO 9330
9332 LET y$=((x$+" ") AND (x$<>"
" AND x$<>"k"))+("._" AND (x$="
"))+("->" AND x$="k"): GO SUB 9
325: IF x$=" " OR x$="k" THEN LE
T m=128+(x$="k"): GO TO 9338
9333 GO SUB 9365: IF x$="p" THEN
 RETURN
9334 LET n=(y$(1)<="H"): IF NOT
(x$>="0" AND x$<="2" OR NOT n AN
D x$>="3" AND x$<="7" OR x$="i"
AND (y$(1)<>CHR$ (CODE "e"-n*32)
) AND (y$(1)<>CHR$ (CODE "h"-n*3
2))) THEN GO SUB 9326: GO TO 933
3
9335 IF x$="i" THEN LET cis=NOT
cis: GO SUB 9325: GO TO 9333
9336 FOR m=0 TO 11. IF y$(1)<>CH
R$ (CODE t$(2*m+1)-n*32) THEN NE
XT m: STOP
9337 LET m=m+cis+12*((2-VAL x$)*
n+(3+VAL x$)*NOT n): IF m>=128 T
HEN GO SUB 9326: LET m=128
9338 RETURN
9340 GO SUB 9365· IF x$="p" THEN
 RETURN
9341 IF x$<>"1" AND x$<>"0" THEN
 GO SUB 9326· GO TO 9340
9342 LET m=(x$="1"): RETURN

9350 GO SUB 9365: IF x$="p" THEN
 RETURN
9351 IF (x$<"0" OR x$>"9") AND x
$<>"p" AND x$<>"k" AND x$<>" " T
HEN GO SUB 9326. GO TO 9350
9352 IF x$=" " OR x$="k" OR x$="
0" THEN LET m=128*(x$=" ")+129*(
x$="k"): RETURN
9353 LET y$=" "+x$: GO SUB 9325
9354 GO SUB 9365: IF x$="p" THEN
 RETURN
9355 IF (x$<"0" OR x$>"9") AND x
$<>"p" AND x$<>" " THEN GO SUB 9
326: GO TO 9354
9356 IF x$=" " THEN LET m=VAL y$
(2): RETURN
9357 LET y$=y$(2)+x$: GO SUB 932
5: IF VAL y$*10>127 THEN LET m=V
AL y$: RETURN
9358 GO SUB 9365: IF x$="p" THEN
 RETURN
9359 IF (x$<"0" OR x$>"9") AND x
$<>" " THEN GO SUB 9326: GO TO 9
358
9360 IF x$="p" THEN RETURN
9362 IF x$=" " THEN LET m=VAL y$
: RETURN
9363 LET cis=1: LET y$=y$(2 TO 2
)+x$: LET m=VAL y$+100*cis: IF m
<128 THEN RETURN
9364 IF m>127 THEN LET m=CODE e$
(m): LET y$=" ": LET cis=0: GO
SUB 9325: GO TO 9350
9365 REM
9366 LET x$=INKEY$· IF x$="" THE
N GO TO 9366
9367 RETURN
```

Krzysztof POŹNIAK
Michał ROLLINGER

...ze świata

Komputery przenośne zyskują w świecie coraz większą popularność. Wprawdzie ich moc obliczeniowa zazwyczaj jest mniejsza od mocy komputerów stacjonarnych, ale za to można z nich korzystać na przykład w czasie podróży. W produkcji takich komputerów prym wiedzie japońska firma Toshiba, która swymi ostatnimi produktami udowadnia, że komputer przenośny wcale nie musi być gorszy od stacjonarnego.

Weźmy Toshibę T 1000. Jego wymiary 32×5×28 cm są niezwykłe nawet w klasie mikrokomputerów przenośnych. Pamięć RAM ma pojemność 512 KB, a wewnątrz obudowy można umieścić dodatkowe 768 KB pamięci. W komputerze zastosowany został monochromatyczny ekran ciekłokrystaliczny o rozdzielczości 640 na 200 punktów. Odpowiada to 25 liniom tekstu po 80 znaków. W 82-klawiszowej klawiaturze 10 klawiszy wydzielonych zostało jako funkcyjne i do sterowania kursorem. Standardowe wyposażenie, to łącze szeregowe RS-232C, łącze równoległe i gniazdo RGB dla dodatkowego monitora kolorowego. T 1000 może być zasilany z baterii Ni-Cd pozwalających na 4-godzinną pracę bez ładowania. Nowsza wersja tego komputera, T 1200 uzupełniona została o dysk twardy o pojemności 20 MB.

Taki sam dysk wraz z kontrolerem RLL wprowadziła Toshiba do modelu T 3100/20. Wewnątrz tego komputera umieszczono procesor 80 286, 640 KB pamięci RAM (z możliwością rozszerzenia jej do 2,6 MB), stację dysków elastycznych 3,5 cala o pojemności 720 KB i wspomniany już dysk twardy. Poza tym można tu umieścić modem o prędkości przesyłania 1200 bodów. Ekran ciekłokrystaliczny o rozdzielczości 640 na 400 punktów, pozwala na uzyskanie monochromatycznej grafiki średniej rozdzielczości. Fachowcy twierdzą, że właśnie ten komputer, jak żaden inny, łączy w sobie szybkość i względnie dużą moc przetwarzania z wygodą, łatwością posługiwania się i przenoszenia.

Nowszy T 5100 jest komputerem kompatybilnym z rodziną IBM PC/XT/AT. Wyposażony jest w procesor 80386 z przełączanym zegarem 8/16 MHz, 2 MB pamięć RAM (z możliwością rozszerzenia do 4 MB), gniazdo do zamontowania koprocesora matematycznego 80387, stację dysków 3,5 cala i 1,44 MB oraz twardy dysk 40 MB. Model dodatkowo wyposażony jest w wyjście szeregowe i równoległe, gniazdo do podłączenia zewnętrznego monitora kolorowego oraz zewnętrznej stacji dysków 5,25 cala. Dzięki temu wyposażeniu T 5100 kosztuje jedynie nieco ponad... 6 tys. dolarów amerykańskich.

PC, to skrót oznaczający komputer przenośny — portable computer. PC może też oznaczać komputer kieszonkowy — pocket computer. I w wypadku jednego z najnowszych wytworów firmy Sharp — oznacza. Takim bowiem komputerem jest PC-1600. Posiada on pamięć RAM wielkości 16 KB z możliwością rozszerzenia za pomocą specjalnych modułów do 80 KB. Zawartość pamięci podtrzymywana jest bateriami przez 5 lat, jeśli moduł znajduje się w komputerze, a przez 2 lata — gdy poza nim. PC-1600 posiada również 96 KB pamięć ROM, która poza systemem operacyjnym zawiera dialekt języka Basic. Standardowy interfejs, RS-232 umożliwia bezpośrednie połączenie z innymi typami mikrokomputerów. Dodatkowy optyczny interfejs przeznaczony jest do szybkiej komunikacji optycznej. Zastosowany w komputerze wyświetlacz ciekłokrystaliczny może pracować w systemie alfanumerycznym (4 wiersze po 26 znaków) lub graficznym (156×32 punkty). Zasilanie — z baterii umożliwiających 25-godzinną pracę bez przerwy.

Do tego komputera firma Sharp produkuje sprzęt peryferyjny. Są to czterokolorowy printer (ploter drukujący na papierze formatu A-4, stacja 2,5-calowych dysków elastycznych o pojemności 64 KB i interfejs do podłączenia standardowego magnetofonu kasetowego.

Jerzy RAJCH

JAK WYKORZYSTAĆ DODATKOWĄ PAMIĘĆ ATARI



— *Przydałby się naszemu panu komputer...*

Od dłuższego czasu wśród użytkowników komputerów Atari krąży program RAMdysk (przedstawiany również na łamach „IKS-a"). Symuluje on pracę bardzo szybkiej pamięci zewnętrznej, wykorzystującej dodatkową pamięć RAM. Poniższy program, który zaczerpnięty został z miesięcznika „COMPUTE!", przeznaczony jest dla Atarii 130XE oraz innych komputerów serii XE i XL mających pamięć rozszerzoną do 128KB. Oczywiście program będzie pracował poprawnie również na komputerach mających jedynie (a może aż) 64KB, ale nie będzie pozwalał na całkowite wykorzystanie jego możliwości.

Dlaczego, aby wykorzystać całe 128KB komputera (np. w Atari 130XE) należy użyć specjalnego programu — na przykład takiego, jak ten? Dzieje się tak ze względu na strukturę wewnętrzną komputera. Maszyny Atari serii XE oraz XL są urządzeniami mającymi procesory ośmiobitowe z szesnastobitową szyną danych. Mogą więc adresować jednocześnie 64KB (adresy od 0 do 65535). Dlatego, aby wykorzystać dodatkowy bank pamięci, należy napisać program (w języku maszynowym, gdyż tylko on zapewnia odpowiednią szybkość) przełączający owe banki. W podobny sposób działa właśnie poniższy program.

Teraz parę słów o inicjacji programu. Jest on napisany w języku AtariBasic. Uruchamiamy go rozkazem **RUN** — komputer ładuje procedurę maszynową; umieszcza ją na szóstej stronie pamięci (adresy od 1536 do 1780). Po zgłoszeniu się interpretera Basic'a komunikatem **READY**, możemy przystąpić do właściwego wykorzystania programu. W wypadku pojawienia się komunikatu o błędzie, należy program wydrukować oraz wyszukać i poprawić błędne linie (lub częściej dane w liniach DATA). Mam nadzieję, że sumy kontrolne w znacznym stopniu zmniejszą prawdopodobieństwo pojawienia się błędu w wydruku.

Program możemy wykorzystać do umieszczenia kilkunastu, a nawet kilkudziesięciu np. pamięci ekranów (tj. obszarów pamięci, w których komputer przechowuje dane dotyczące obrazu) w dodatkowych 64 kilobajtach pamięci. **Dokładniej, mając do dyspozycji komputer Atarii 130XE oraz poniższy program, możemy zgromadzić (jednocześnie) osiem obrazów w trybie graficznym 8 lub np. sześćdziesiąt osiem obrazów w trybie tekstowym 0. Szybkość przesyłania danych z dodatkowej pamięci wynosi 9000 bajtów w ciągu sekundy.**

Wiedząc, iż jeden ekran w trybie 0 zajmuje 992 bajty, dokonujemy zdumiewającego odkrycia — będzie on transmitowany ok. 0,1 sekundy. Wydaje mi się, że jest to zadowalający wynik.

Program realizuje cztery następujące funkcje

a) odczytuje zawartość dowolnej komórki pamięci dodatkowej (ang. PEEK),

b) zapisuje dowolną komórkę pamięci dodatkowej wartości od 0 do 255 (ang. POKE),

c) przepisuje dowolny obszar pamięci do obszaru rozszerzenia,

d) transmituje w kierunku odwrotnym niż określony w podpunkcie c

Poszczególne funkcje wywołujemy za pomocą instrukcji USR. Dla tych czytelników, którzy jeszcze nie wiedzą do czego służy ta komenda, przypominam, iż przekazuje ona sterowanie do procedury maszynowej umieszczonej pod określonym adresem.

Czas na szczegółowe omówienie wszystkich możliwości programu

ad. a) Aby odczytać wartości którejkolwiek komórki pamięci dodatkowej należy wpisać w trybie bezpośrednim bądź wykorzystać we własnym programie następujące linie:

X = USR (AUXBYTE, *ad*)

gdzie zmienna *'ad'*, mogąca przyjmować wartość w zakresie od 0 do 65535, określa adres komórki, której zawartość zamierzaliśmy zbadać. Po wykonaniu tej sekwencji otrzymamy liczbę od 0 do 255, określającą stan komórki pamięci.

ad. b) Wpisanie pewnej wartości liczbowej do komórki dodatkowego banku pamięci również nie nastręcza naszemu programowi żadnych trudności, linia:

X = USR (AUXBYTE, *ad, wart*)

zapewnia wprowadzenie do komórki o adresie określonym zmienną *'ad'* wartości, którą zawiera zmienna *'wart'*. UWAGA! Zmienna numeryczna *'ad'* musi należeć tak jak i w poprzednim wypadku do przedziału 0—65535, a zmienna *'wart'* od 0—255.

ad. c) Znacznie bardziej skomplikowana jest operacja przesłania bloku danych. Wygląda ona następująco:

X = USR (AUXDUMP, *źródło, cel, dług*).

A oto znaczenie poszczególnych zmiennych:

'źródło' — określa początek bloku w pamięci podstawowej, przygotowanego do przesłania; możemy nadać tej zmiennej wartość od 0 do 6553.

'cel' — określa adres w dodatkowym bloku pamięci, tj. adres, pod którym znajdą się dane po przesłaniu. Może przyjmować wartości takie, jak poprzednia zmienna.

'dług' — „mówi" komputerowi, jak długi jest przesyłany blok.

ad. d) jest to ostatnie (niestety) udogodnienie wnoszone przez ten program, umożliwia ono powrót danych z pamięci dodatkowej do podstawowego bloku 64KB. Rozkaz, który przeprowadza tę transmisję ma postać bardzo podobną do postaci poprzedniego rozkazu, jedyna różnica polega na użyciu innej stałej programowej — zamiast AUXDUMP należy użyć AUXLOAD. Linia ta wygląda następująco:

X = USR (AUXLOAD, *źródło, cel, dług*).

Należy także zmienić kolejność występowania zmiennych *'źródło'* i *'cel'*, w tym wypadku pierwsza oznacza adres bloku w pamięci dodatkowej, a druga adres pamięci, pod którym dane zostaną umieszczone.

Na koniec pragnę zaprezentować jeden z praktycznych przykładów programu, o którym była mowa. Będzie on przesyłał dane obrazu w grafice 8 do pamięci dodatkowej i z powrotem (tj. do podstawowego bloku 64KB).

Przede wszystkim należy określić adres początku pamięci obrazu; jest on przechowywany w komórkach 88 (LSB — młodszy bajt) i 89 (MSB — starszy bajt). Znając zawartości tych komórek, możemy w prosty sposób odnaleźć poszukiwany początek pamięci obrazu.

pocz = REEK (88) + 256 x PEEK (89)

Będzie nam jeszcze potrzebna informacja, jak duży obszar pamięci wykorzystany jest w celu zapamiętania obrazu w trybie

graficznym bądź tekstowym. W wypadku trybu numer 8 wynosi on 8138 bajtów. Informacje o pozostałych trybach znajdują się poniżej:

tryb 0	(tekstowy)	—	992	bajty
tryb 1	(tekstowy)	—	672	bajty
tryb 2	(tekstowy)	—	420	bajtów
tryb 3	(graficzny)	—	432	bajty
tryb 4	(graficzny)	—	696	bajtów
tryb 5	(graficzny)	—	1176	bajtów
tryb 6	(graficzny)	—	2184	bajty
tryb 7	(graficzny)	—	4200	bajtów
tryb 8, 9, 10, 11	(graficzne)	—	8138	bajtów
tryb 12	(graficzny)	—	1152	bajty
tryb 13	(graficzny)	—	660	bajtów
tryb 14	(graficzny)	—	4296	bajtów
tryb 15	(graficzny)	—	8138	bajtów

Te wiadomości w zupełności wystarczą, by dokonać transmisji danych obrazu:

X = USR (AUXDUMP, *pocz.*, 0,8138)

Liczba 0 określa, pod jakim adresem bloku rozszerzenia pamięci znajdzie się początek przesłanego obszaru danych. Aby dane te otrzymać z powrotem wpisujemy następującą linię:

X = USR (AUXLOAD,0 *pocz.*, 0,8112)

Dzięki niej zapamiętany obrazek wraca do podstawowego bloku pamięci

```
DB 10 AUXBYTE=1624
GK 20 AUXDUMP=1655
YB 30 AUXLOAD=1718
IX 40 FOR I=1536 TO 1780:READ A:P
   OKE I,A:X=X+A:NEXT I:IF X<>353
   14 THEN PRINT CHR$(125);"BLAD
   W LINIACH 50-220"
DR 50 DATA 160,0,173,1,211,41,195
   ,133,217,165,216,41,192,74
VN 60 DATA 74,74,74,9,32,5,217,14
   1,1,211,165,216,41,63
VK 70 DATA 9,64,133,215,165,212,1
   45,214,165,217,9,48,141,1
OS 80 DATA 211,96,160,0,173,1,211
   ,41,195,133,217,165,216,41
VJ 90 DATA 192,74,74,74,74,9,32,5
   ,217,141,1,211,165,216
LK 100 DATA 41,63,9,64,133,215,17
   7,214,133,212,165,217
NM 110 DATA 9,48,141,1,211,96,104
   ,133,245,198,245,104,133,216
XK 120 DATA 104,133,214,165,245,2
   08,6,32,44,6,132,213,96,104
UY 130 DATA 104,133,212,32,0,6,24
   ,144,243,104,104,133,225,104
CK 140 DATA 133,224,104,133,216,1
   04,133,214,104,133,227,104,133
   ,226
X8 150 DATA 160,0,132,229,132,228
   ,177,224,133,212,32,0,6,230
FB 160 DATA 224,208,2,230,225,230
   ,214,208,2,230,216,230,228,208
DU 170 DATA 2,230,229,165,228,197
   ,226,208,225,165,229,197,227,2
   08
FY 180 DATA 219,96,104,104,133,21
   6,104,133,214,104,133,225,104,
   133
RB 190 DATA 224,104,133,227,104,1
   33,226,160,0,132,229,132,228,3
   2
LL 200 DATA 44,6,165,212,145,224,
   230,224,208,2,230,225,230,214
YX 210 DATA 208,2,230,216,230,228
   ,208,2,230,229,165,228,197,226
AG 220 DATA 208,225,165,229,197,2
   27,208,219,96
```

Mam nadzieję, że program ten stanie się użytecznym narzędziem dla wszystkich, którzy chcieliby wykorzystać dotychczas niedostępne i „marnujące się" kilobajty pamięci w swoich komputerach

Na podst. miesięcznika COMPUTE!
opracował: **Piotr JASINIEWSKI**

GIEŁDA POMYSŁÓW

Rafał FOLTYNIEWICZ

Drukowanie minikatalogu dyskietki

```
10 'DRUKOWANIE miniKATALOGU DYSKIETKI
20 MODE 1
30 INPUT "Wlasciciel ";w$:d=LEN(w$)
40 IF d>11 THEN 30
50 w$=w$+SPACE$(11-d)+" *"
60 'inicjalizacja drukarki
70 RESTORE 600
80 FOR i=1 TO 8
90 READ zn:PRINT #8,CHR$(zn);:NEXT i
100 INPUT "Numer i strona dyskietki ";o$:d=LEN(o$)
110 IF d>10 THEN 100
120 o$="* "+o$+SPACE$(10-d)
130 CLS:CAT
140 'kopiowanie linii z ekranu
150 l=3:z$=""
160 WHILE z$<>" "
170 l=l+1:LOCATE 1,l:GOSUB 400
180 WEND
190 f$=""
200 FOR i=1 TO 9
210 LOCATE i,l+1:GOSUB 400
220 f$=f$+z$
230 NEXT i
240 'drukowanie naglowka
250 GOSUB 500
260 PRINT #8,o$;"  *  ";f$;"  *  ";w$:GOSUB 500
270 'drkowanie katalogu
280 FOR k=4 TO l-1:l$="* "
290 FOR i=1 TO 40
300 LOCATE i,k:GOSUB 400
310 l$=l$+z$
320 NEXT i
325 l$=l$+" *"
330 PRINT #8,l$:l$="* "
340 NEXT k:GOSUB 500
350 LOCATE 5,l+3:PRINT "Kolejna dyskietka ? (t/n)"
360 c$=INKEY$:IF c$="" THEN 360
370 IF UPPER$(c$)="T" THEN 60
380 END
400 'kopiowanie znaku z ekranu
410 z$=COPYCHR$(#0):RETURN
500 'drukowanie ramki
510 PRINT #8,STRING$(44,"*")
520 RETURN
600 'znaki dla drukarki
610 DATA 27,64,27,49,15,27,83,0
```

Obracanie ekranu

```
5 'OBRACANIE EKRANU  1987.12.30
10 'kod maszynowy
20 MEMORY &9FFF:adres=&A000
30 RESTORE 210
40 FOR i=1 TO 14
50 suma=0:READ a$,b$
60 FOR k=1 TO 31 STEP 2
70 wartosc=VAL("&" + MID$(a$,k,2))
80 POKE adres,wartosc
90 suma=suma+wartosc:adres=adres+1
100 NEXT k
110 IF suma<>VAL("&"+b$) THEN PRINT "Zle dane w
linii nr: ";200+i*10:STOP
120 NEXT i
130 SAVE "obroc",b,&A000,&D7
140 END
200 '
210 DATA CD11BC38072B0A21A2A0180821ADA018,514
220 DATA 03219BA02256A0DD21C9A0FD21CEA0CD,837
230 DATA 0BBCE5DD46000E000922D6A0D121FF0F,67E
240 DATA 19113000A7ED527CE60F67D608300411,53B
250 DATA 0008191100F01922D8A02AD6A0ED5BD8,695
260 DATA A001D007C5CDADA0C10B78B12806C5CD,80C
270 DATA 85A018F1DD23FD23FD230100082AD6A0,717
280 DATA 0922D6A07CE6F0FEE0C82AD8A0A7ED42,A11
290 DATA 22D8A018C523DD7E01BC2003DD66001B,633
300 DATA FD7E00BAC0FD56011EFFC90608CB06EB,7F9
310 DATA 10FB1A060817CB1E10FB1712C97E47CD,5C2
320 DATA BDA0F51A47F11278CDBDA077C9E655CB,99E
330 DATA 274F78E6AACB3FB1C9C0C8D0D8E0F7FF,B08
340 DATA EFF7E7EFDFE70000000000000000000,582
```

Lilavati (6)

„PIĘTNASTKA"

Danuta KWASIŻUR,
Mieczysław SKONIECZNY

W szóstym odcinku cyklu poświęconego mikrokomputerowej realizacji rozrywek matematycznych, zgromadzonych przez Szczepana Jeleńskiego w „Lilavati", chcemy zaprezentować znaną grę o nazwie „piętnastka". Gra znana również pod angielską nazwą „Fifteen puzzle" pochodzi ze Stanów Zjednoczonych, gdzie wynalazł ją pewnien głuchoniemy w roku 1878. Szybko zyskała popularność — również w Europie. Jako ciekawostkę można podać, że swego czasu władze rządowe USA musiały wydać zakaz posiadania tej gry przy sobie przez urzędników w godzinach pracy.

Zasady gry są bardzo proste. Są one przedstawione na ekranie mikrokomputera po zainicjowaniu programu w następującej postaci:

```
Gra sklada sie z ramki, w ktorej miesci
sie pietnascie ruchomych kwadracikow -
po 4 w kazdej kolumnie z wyjatkiem osta-
tniego szeregu, w ktorym miejsce odpo-
wiadajace szesnastemu kwadracikowi jest
puste. Na kwadracikach widnieja numery
od 1 do 15. Korzystajac z owego jednego
wolnego miejsca nalezy kwadraciki prze-
suwac dopoty, dopoki nie zostana upo-
rzadkowane w kolejnosci od 1 do 15,
a ostatni kwadracik bedzie pusty.

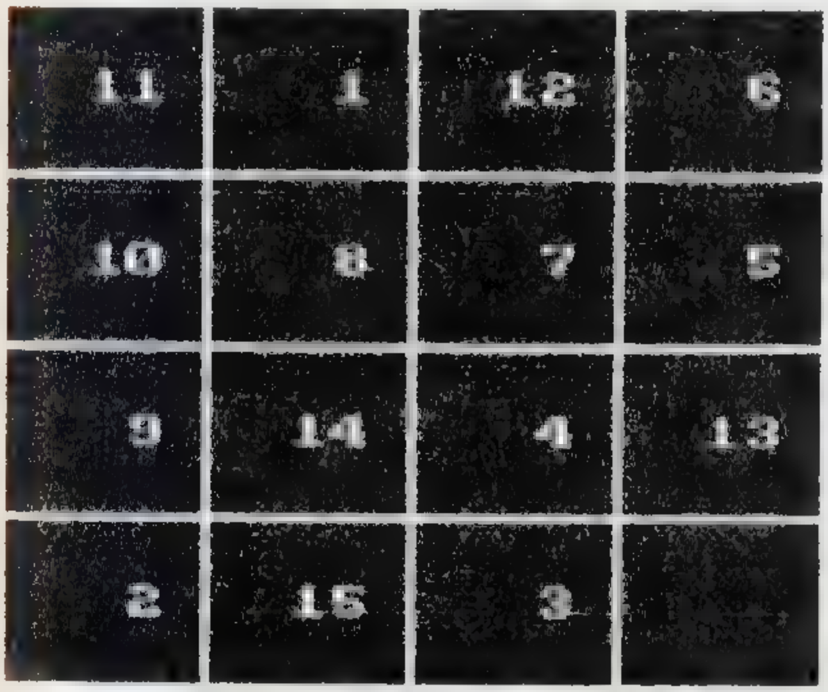
STEROWANIE KLAWISZAMI:

↑,↓,←,→

NACISNIJ DOWOLNY KLAWISZ
```

Następnie pojawi się plansza gry z losowym, generowanym przez program, rozstawieniem numerów.

Przykładowe rozstawienie prezentuje poniższy rysunek:



Przesuwanie kwadracików odbywa się za pośrednictwem klawiatury funkcyjnej (4 klawisze ze strzałkami). W sytuacji, gdy przesunięcie jest niemożliwe, program informuje o tym gracza komunikatem:

PRZESUNIĘCIE JEST NIEMOŻLIWE

Po każdym przesunięciu program bada, czy gra zakończyła się sukcesem. Jeżeli ustawienie ponumerowanych kwadracików jest prawidłowe, na ekranie pojawia się komunikat:

BRAWO! UDAŁO SIĘ

i program kończy działanie.

Wkrótce po rozpowszechnieniu się „Piętnastki" pojawiły się liczne jej teorie. Zwłaszcza dwaj znani matematycy, Johnson i Story podali sporo ciekawego materiału naukowego dotyczącego tej gry. Udowodnili oni, że w zależności od ustawienia początkowego można przewidzieć rozwiązalność lub nierozwiązalność zadanego układu.

Dodamy tylko, że nie zawsze możliwe jest osiągnięcie układu kwadracików gwarantującego sukces w grze. Niejednokrotnie w szeregu najniższym zamiast kolejności 13, 14, 15 otrzymuje się ustawienie 13, 15, 14. Nie radzimy wtedy mozolić się dalej nad przestawieniem kwadracików — najlepiej nacisnąć klawisz BREAK i rozpocząć grę od początku.

```
10 MODE 1:INK 0,26:INK 1,0:INK 2,24:INK 3,14:BORDER 0:PAPER 1:CLS
20 PEN 2:PRINT"Gra sklada sie z ramki, w ktorej miesci sie pietnascie
ruchomych kwadracikow-   po 4 w kazdej kolumnie z wyjatkiem os
ta-tniego szeregu, w ktorym miejsce odpo-  wiadajace szesnastemu kwadr
acikowi jest"
30 PRINT "puste. Na kwadracikach widnieja numery  od 1 do 15. Korzysta
jac z owego jednego wolnego miejsca nalezy kwadraciki prze- su
wac dopoty, dopoki nie zostana upo-   rzadkowane w kolejnosci od 1 do
15,    a ostatni kwadracik bedzie pusty."
40 PRINT:PRINT "STEROWANIE KLAWISZAMI:":PRINT:PRINT CHR$(240);",";CHR$
(241);",";CHR$(242);",";CHR$(243)
50 LOCATE 1,20:PRINT "NACISNIJ DOWOLNY KLAWISZ"
60 IF INKEY$="" THEN GOTO 60
70 CLS
80 WINDOW £1,11,30,6,25:PAPER £1,2:CLS £1
90 WINDOW £2,1,40,1,1:PAPER £2,1:CLS £2:PEN £2,0
95 REM  'rysowanie planszy gry'
100 GRAPHICS PEN 0:FOR k=0 TO 3
110 PLOT 160+k*80,0:DRAWR 0,318
120 PLOT 238+k*80,0:DRAWR 0,318
130 NEXT k
140 FOR k=0 TO 3
150 PLOT 160,K*80:DRAWR 318,0
160 PLOT 160,78+K*80:DRAWR 318,0
170 NEXT K
180 DIM TABL(16)
190 GOSUB 420
200 TABL(16)=0
210 PAPER 2:PEN 1
220 FOR K=1 TO 15
230 X=13-INT((K-1)/4)*20+(K-1)*5:Y=8+INT((K-1)/4)*5
240 LOCATE X,Y:PRINT USING "££";TABL(K)
250 NEXT K
255 REM  'sterowanie przesuwaniem kwadracikow'
260 I=16
270 A$=INKEY$
280 IF A$="" THEN GOTO 270
290 IF ASC(A$)=240 THEN GOTO 340
300 IF ASC(A$)=241 THEN GOTO 360
310 IF ASC(A$)=242 THEN GOTO 380
320 IF ASC(A$)=243 THEN GOTO 400
330 GOTO 270
335 REM  'kontrola poprawnosci przesuniecia'
340 IF I=13 OR I=14 OR I=15 OR I=16 THEN GOTO 500
350 TABL(I)=TABL(I+4):TABL(I+4)=0:GOSUB 510:LOCATE X,Y:PRINT USING "££
";TABL(I):I=I+4:GOSUB 510:LOCATE X,Y:PRINT "  ":GOSUB 520:GOTO
 270
360 IF I=1 OR I=2 OR I=3 OR I=4 THEN GOTO 500
370 TABL(I)=TABL(I-4):TABL(I-4)=0:GOSUB 510:LOCATE X,Y:PRINT USING "££
";TABL(I):I=I-4:GOSUB 510:LOCATE X,Y:PRINT "  ":GOSUB 520:GOTO
 270
380 IF I=4 OR I=8 OR I=12 OR I=16 THEN GOTO 500
390 TABL(I)=TABL(I+1):TABL(I+1)=0:GOSUB 510:LOCATE X,Y:PRINT USING "££
";TABL(I):I=I+1:GOSUB 510:LOCATE X,Y:PRINT "  ":GOSUB 520:GOTO
 270
400 IF I=1 OR I=5 OR I=9 OR I=13 THEN GOTO 500
410 TABL(I)=TABL(I-1):TABL(I-1)=0:GOSUB 510:LOCATE X,Y:PRINT USING "££
";TABL(I):I=I-1:GOSUB 510:LOCATE X,Y:PRINT "  ":GOSUB 520:GOTO
 270
415 REM  'generowanie kolejnosci kwadracikow na planszy'
420 FOR I=1 TO 15
430 LOS=1+INT(RND*15)
440 FOR J=1 TO I-1
450 IF TABL(J)=LOS THEN GOTO 430
460 NEXT J
470 TABL(I)=LOS
480 NEXT I
490 RETURN
500 CLS £2:PRINT £2,"PRZESUNIECIE NIEMOZLIWE":FOR T=1 TO 1000:NEXT T:C
LS £2:GOTO 270
510 X=13-INT((I-1)/4)*20+(I-1)*5:Y=8+INT((I-1)/4)*5:RETURN
520 IF TABL(16)>0 THEN RETURN
530 FOR J=1 TO 15
540 IF NOT TABL(J)=J THEN RETURN
550 NEXT J
560 PRINT £2,"BRAWO! UDALO SIE!"
570 FOR a=0 TO 26:BORDER a:FOR t=1 TO 100:NEXT t:NEXT a:END
```

UWAGI PROGRAMOWE

1. Program opracowano w języku BASIC mikrokomputera AMSTRAD CPC 6128.

2. Tablica TABL służy do przechowywania aktualnego układu kwadratów.

3. Instrukcje 100—170 realizują rysowanie planszy gry.

4. Instrukcje 260—410 sterują przesuwaniem kwadracików.

5. Generowanie początkowego ustawienia ponumerowanych kwadratów realizuje podprogram 420—490.

6. Instrukcje 520—570 badają czy gra zakończyła się sukcesem. Jeśli tak, to generują stosowny komunikat i kończą pracę programu.

Oprogramowanie układów transputerowych

Początkowo programowanie komputerów polegało na zapisaniu sekwencji rozkazów opisującej algorytm rozwiązania danego problemu w postaci numerycznych kodów maszynowych, zrozumiałych bezpośrednio przez procesor. Ta metoda programowania, pracochłonna i nieefektywna była obarczona szczególną podatnością na błędy. Mała czytelność stosowanego zapisu programu powodowała znaczne trudności w czasie procesów uruchomieniowych. Wkrótce udoskonalono metodologię programowania przez wprowadzenie języków niskiego poziomu (asemblery) oraz problemowo (języki wysokiego poziomu) — FORTRAN, ALGOL, COBOL.

Języki wysokiego poziomu stały się bardzo wygodnym narzędziem dla programisty, gdyż nie wymagały znajomości organizacji używanego sprzętu. Jednakże w programach, w których czas wykonywania był elementem krytycznym, w szczególności dotyczy to oprogramowania systemowego, języki wysokiego poziomu nie były przydatne, ze względu na swą nadmiarowość. Do najbardziej rozpowszechnionych obecnie języków zaliczyć należy BASIC, PASCAL, ADA, LISP, PROLOG i LOGO, ale wskutek zmiany charakteru programowania, zbliżania go do modelowania systemowego istnieje dziś duże zainteresowanie językami typu FORTH, PL/M, C i MODULA. Języki te mając konstrukcję języka wysokiego poziomu, wzbogacone są o zestaw instrumentów, które nadają im pewne własności „asemblerowe".

Transputer a OCCAM

W pierwszych zastosowaniach, język maszynowy transputera wykorzystywał nieznaczną ilość instrukcji, których funkcyjna natura określała działanie. Instrukcje te były jednobajtowe, podzielone na cztery „bity funkcji" oraz cztery „bity danych". W ten sposób podstawowy zbiór instrukcji liczył 13 funkcji (*Load Constant, *Add Constant, *Load Local, *Store Local, *Load Local Pointer, *Jump, *Store Non Local, *Conditional Jump, *Call, *Load Non Local, *Prefix, *Negative Prefix i *Operate). Dziesięć pierwszych funkcji, to operacje obliczeniowe. Pozostałe trzy są szczególnie ciekawe, gdyż zezwalają na budowę słów 32-bitowych złożonych z 4-bitowych danych (*Prefix, *Negative Prefix) oraz powodują interpretowanie słowa 8-bitowego jako kodu funkcji (*Operate). Tym prostym sposobem INMOS formalnie zapewniła sobie znaczne możliwości rozbudowywania listy rozkazów. Konwencjonalne języki programowania nie są zbyt przydatne do konstruowania programów przeznaczonych dla współbieżnej pracy układów wieloprocesorowych.

Równolegle z pracami badawczo-projektowymi nad transputerami rozpoczęto tworzenie odpowiednich języków programowania. OCCAM — specjalizowany język wysokiego poziomu powstawał pod kątem przyszłych zastosowań w środowisku transputerowym, stąd też jest on wewnętrznie dostosowany do architektury i możliwości tych układów. OCCAM tworzy szkielet koncepcyjno-konstrukcyjny projektowanego współbieżnego systemu w analogiczny sposób, jak algebra Bool'a czyni to w stosunku do systemów elektronicznych złożonych z bramek logicznych. OCCAM wyróżnia głęboka spójność z modelem współbieżnie pracujących wielu procesów, komunikujących się za pomocą dwuprzewodowych łączy lub ich odpowiedników programowych — kanałów. Dwojakie rozumienie kanału przez język sprowadza się w efekcie do nierozróżniania ich fizycznej implementacji. Z punktu widzenia programisty nie ma znaczenia lokalizacja pamięci i miejsce wykonywania poszczególnych procesów. Wzajemna komunikacja między procesami odbywa się automatycznie i nie ma potrzeby stosowania specjalnych procedur zapewniających kontrolę jej poprawności oraz właściwą synchronizację wewnętrzną. Dla transputera T-800 powstała nowa, znacznie wzbogacona wersja OCCAM-2, oparta na 16-, 32- i 64-bitowych danych. Zastosowane w niej mechanizmy sprzętowe zapewniają wykonywanie operacji zmiennoprzecinkowych z podobną szybkością, jak analogicznych operacji na liczbach całkowitych. Zasadniczą zmianę wprowadza to, że kanały mogą przesyłać dane zdefiniowane strukturalnie zamiast 16-bitowych wartości. Oznacza to, że łańcuchy, macierze oraz skomplikowane rekordy Pascala i struktury języka C mogą tworzyć wyjście pojedynczych operacji. Opracowano specjalne wersje kompilatorów: Pascala, C, Fortranu, Prologu i Macro Assemblera przeznaczone dla układów transputerowych. Moduły pisane w tych językach po rekompilacji mogą być powiązane programem napisanym w OCCAM, który wywołuje te moduły jako procesy. Dotychczas nie ma systemu operacyjnego dla urządzeń bazujących na układach transputerowych. Niektóre z firm proponują stosowanie techniki nakładek systemowych, a także adaptację systemu UNIX.

OCCAM - 2

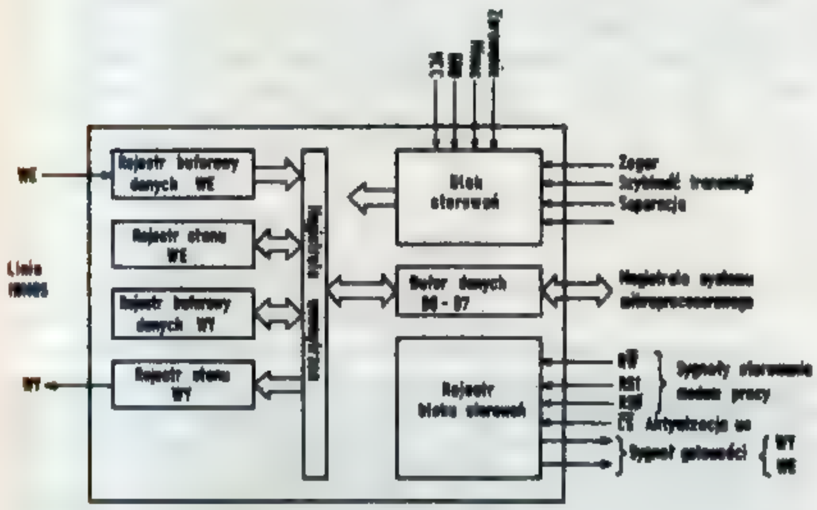
Jądrem koncepcji języka OCCAM są pojęcia procesu i kanału. W OCCAM występują trzy podstawowe operacje zwane też procesami prymitywnymi (INPUT, OUTPUT, ASSIGN — w symbolicznej notacji odpowiednio: ?, !, :=). Operacje te wraz ze słowami kluczowymi oraz innymi, typowymi dla każdego języka składnikami (wyrażenia, zmienne, wartości, literały, operatory itp.), stanowią materię poszczególnych bloków programowych zwanych w OCCAM konstrukcjami. Każdy blok poprzedzony jest specjalnym „konstruktorem", który informuje procesor o sposobie realizacji programu i stosowanej w nim komunikacji. Konstrukcje same w sobie są procesami i mogą być zanurzane jedna w drugiej. Pojęcie procesu jest bardzo szerokie i należy je rozumieć jako formę niezależną, tzn. wykonywanych w tym procesie operacji od lokalizacji fizycznej programu i danych (tj. pamięci, jak również transputera). Każdy proces po inicjacji wykonuje określone instrukcje, a następnie zostaje zatrzymany lub się kończy. W OCCAM-2 wyróżnia się 4 podstawowe konstruktory. Konstruktor SEQ określa blok, w którym instrukcje są realizowane jedna po drugiej. PAR umożliwia realizację współbieżności, tzn. poszczególne komponenty konstrukcji wykonywane są w tym samym czasie, a prawidłowe zakończenie procesu może nastąpić po uprzednim zakończeniu wszystkich jego elementów. Konstrukcję warunkową rozpoczyna IF. W zależności od wartości logicznej warunków wykonywana jest sekwencja instrukcji umieszczona bezpośrednio po warunku mającym wartość TRUE. Ostatni konstruktor ALT jest związany bezpośrednio z komunikacją wyrażając alternatywę wykonania odpowiednich części bloku od gotowości kanału komunikacyjnego. Słowo kluczowe WHILE z występującym łącznie warunkiem umożliwia powtarzanie sekwencji instrukcji do czasu, gdy warunek będzie miał wartość logiczną FALSE. Tworzenie pętli umożliwi też tzw. replikator konstrukcji FOR. Pętla może być w poszczególnych przebiegach różna, gdyż FOR odnosi się do występującej z nim konstrukcji. Selektor CASE zestawia w programie rozgałęzienia (analogicznie jak np. w BASIC-u instrukcja IF... WHEN...ELSE). OCCAM-2 umożliwia tworzenie procedur, definiowanie funkcji, deklarowanie wartości zmiennych, przypisanie typu kanałom, zmiennym, tablicom i wartościom. Posługuje się znakami ASCII wyróżniając pewną grupę znaków specjalnych. Łańcuchy mogą być tworzone ze wszystkich znaków z wyjątkiem: *,'," . W wyrażeniach stosuje się operatory opisane w tabeli. Niektóre z nich mają swoje odpowiedniki w postaci znaków (pary znaków). Gdy łańcuch zawiera parę znaków, z których pierwszy jest * , to jego wartość jest równa 0, a znaczenie opisuje drugi znak

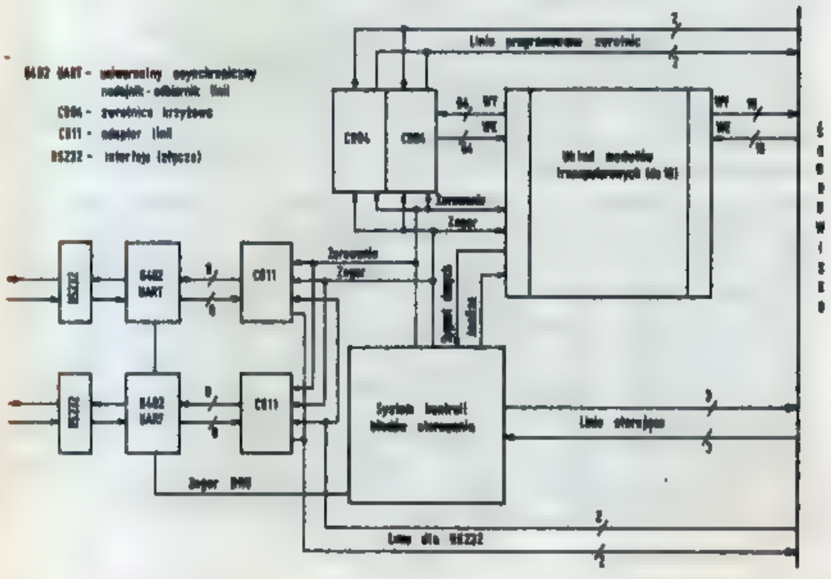
Do prymitywnych typów w OCCAM zalicza się: INT, INT32, INT 64, REAL32, REAL64, BYTE, TIMER oraz CHAN OF type. Wszystkim zmiennym wyrażeniom i wartościom przypisywany jest typ, który może należeć do prymitywów albo być typem tablicy, rekordu lub struktury. CHAN OF protocol (type określa kanał i specyfikuje protokół komunikacji). TIMER jest typem określanym dla zmiennych, które w programie (procesie) odmierzają czas. Forma [*n*] BYTE przedstawia łańcuch o n znakach, a [*n*]INT opisuje

Operator	Typ operandu	Opis
+, -, *, /, \ (REM)	INTEGER, REAL	op. arytmetyki
PLUS, MINUS, TIMER, AFTER	INTEGER	op. arytmetyki modu
=, <>	wszystkie typy	op. relacji
>, <, <=, >=	INTEGER, REAL	op. relacji
AND, OR, NOT	BOOLEAN	op. logiczne
><, /\ (BITAND), \/ (BITOR, ~ (BITNOT)	INTEGER	op. logiczne bitów
>>, <<	INTEGER	op. przesunięcia

Reprezentacja łańcuchowa znaków specjalnych				
*c – CR	*n – LF	*t – TAB	*s – Spacja	*' – '
*" – "	** – *	*# – screen HEX		

tablicę o *n* elementach typu INT. Przy definiowaniu procedur i funkcji przypisuje się im nazwy oraz parametry formalne ujęte w nawiasach. Wywoływanie tych struktur dokonywane jest przez nazwę z listą parametrów. W OCCAM-2 można uwzględnić konfigurację systemu, tym samym zapewniając prawidłowy rozdział programu na poszczególne transputery. Konfiguracja nie wpływa na samą logikę zachowań programowych. Przez użycie formuły PLACED PAR określamy lokalizację procesu w pamięci lub przypisujemy konkretny procesor jako wykonawcę procesu. Używane zmienne i zegar muszą być deklarowane oddzielnie dla każdej lokalizacji procesu. Formuły PRI PAR i PRI ALT nadają priorytety procesom wykonywanym przez ten sam transputer. Formuła PLACE.......AT... określa adres fizyczny zmiennej, kanału, tablicy lub portu I/O w pamięci. Oto ilustracja programowa.





```
Val ekran IS #B800: —— nadanie wartości zmiennej ekranu
[32000] BYTE ekran_bufor: —— deklaracja tablicy bajtowej
PLACE ekran_bufor AT ekran: —— lokalizacja bufora w pamięci
. . . . . .
ekran_bufor 57: = FF —— 57 element tablicy przyjmuje wartość FF
. . . . . . .
```

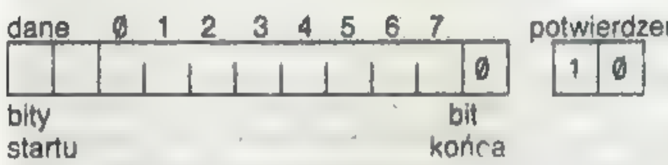
Poniższy fragment programu ilustruje przydział procesów w procedurze sortującej elementy.

```
PLACED PAR i = 0 FOR numer
  PROCESOR i —— Lokalizacja
    input    IS p[i]: —— przypisanie
    output   IS p[i + 1]:
    INT max: —— deklaracja typu zmiennej max
      SEQ
        input ? max
        SEQ j = 0 FOR numer -1 —— przemieszczenie danych
          INT następny : —— deklaracja typu zmiennej następny
          SEQ
            input ? następny
            IF
              następny <= max
                output ! następny
              następny > max
                SEQ
                  output ! max
                  max := następny
        output ! max
.................. —— procedury kopiowania posortowanych elementów
```

Transputery zawierają w swojej architekturze zegar. Umożliwia to bezpośrednie i niezależne modelowanie czasu dla danego procesu, tzn. pracę w czasie rzeczywistym. Zmienne typu TIMER stosowane są do wewnętrznych pomiarów czasu lub ustalania harmonogramu czynności. Poniższy przykład ilustruje realizację opóźnienia PROC czekaj (VAL INT cykl_op) —— procedura z parametrem typu INT

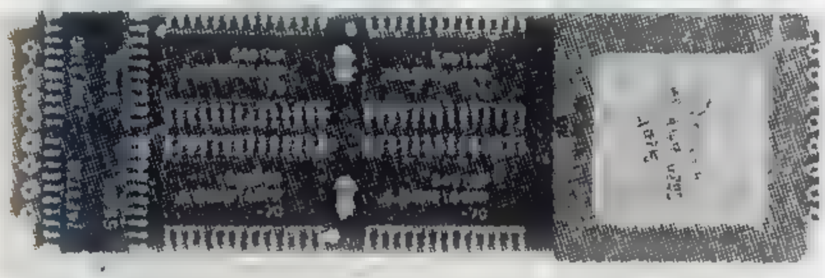
```
TIMER zegar : —— deklaracja typu zmiennej zegar
INT aktualny_czas : —— deklaracja typu zmiennej aktualny_czas
SEQ
  zegar ? aktualny_czas
  zegar ? AFTER aktualny_czas PLUS cykl_op
.
INT y : —— własny proces
SEQ
  input1 ? y —— y przyjmuje wartość z kanału input1
  czekaj (1000)
  output1 ! y —— wyślij wartość zmiennej y do kanału output1
...
```

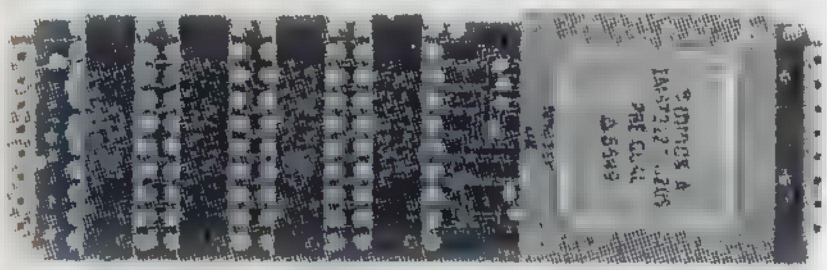
Niezmiernie trudno jest w zwięzły sposób przedstawić język, jego strukturę, formuły zapisu i cechy charakterystyczne, dlatego ograniczyłem się do omówienia podstawowych zasad i wiadomości o OCCAM — 2. Zainteresowanych niestety muszę odesłać do opracowań aplikacyjnych firmy INMOS lub wydawnictw zagranicznych. Na zakończenie chciałbym przedstawić kilka uwag o komunikacji. Będzie to krótki wstęp do następnego odcinka poświęconego zastosowaniom transputerów. Komunikacja między procesami odbywa się podobnie do komunikacji sprzętowej. Jeśli pewien kanał jest używany jako wejście jednego procesu oraz jako wyjście w drugim procesie, to transmisja może się odbywać tylko wtedy, gdy oba procesy są gotowe. Wartość, która będzie przesyłana, jest kopiowana z procesu wysyłającego do przyjmującego. W ten sposób interpretuje się współbieżną pracę procesów w jednym transputerze. Kanałem w tym wypadku jest pamięć. W wypadku sieci transputerów, każdy z nich wykonuje proces w nim ulokowany. Transmisja dokonuje się bezpośrednio przez parę jednokierunkowych przewodów. Dane są przesyłane szeregowo wg poniższego protokołu.

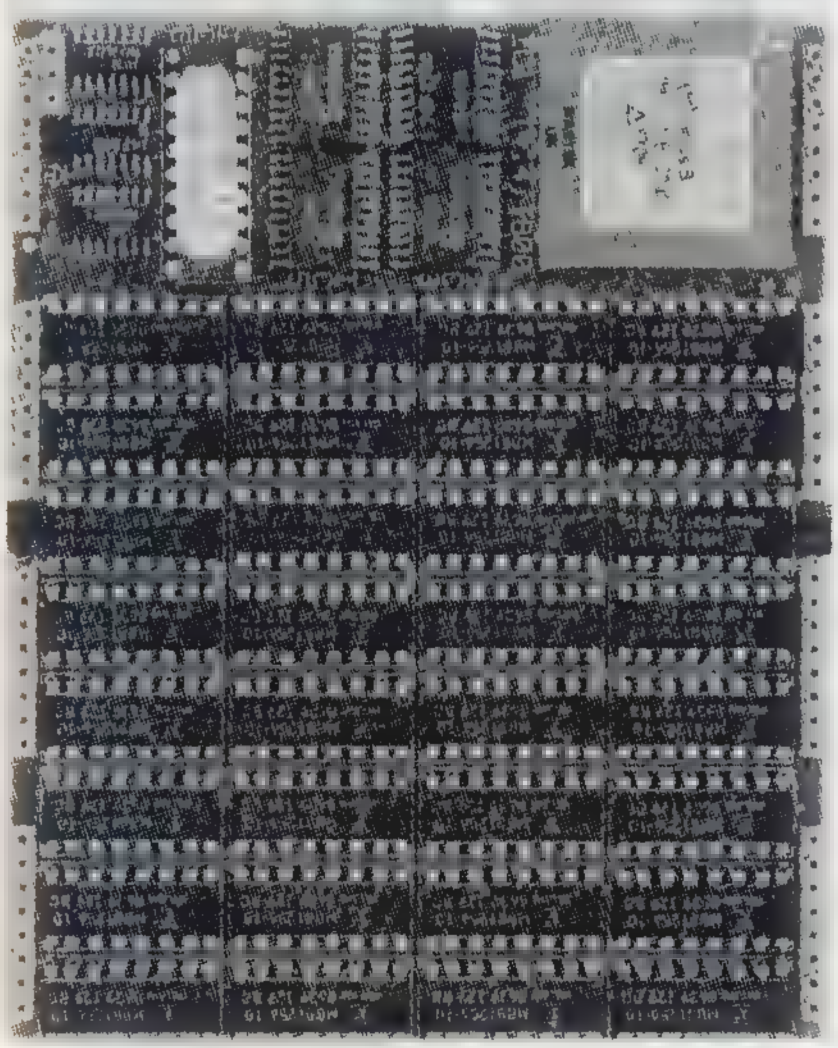


Protokół zapewnia prawidłową komunikację między transputerami z różną długością słowa. Linie komunikacyjne nie są czułe na fazę zegara. Transputerowa rodzina zawiera adaptery zapewniające sprzężenia transputerowych linii komunikacyjnych z urządzeniami nietransputerowymi. Adapter IMS C012 przedstawiony na rysunku jest uniwersalnym interfejsem sprzężenia między interfejsami współpracujących urządzeń, zapewniającym pełny duplex i dużą szybkość transmisji (10 lub 20 Mbit/s). Może być używany w połączeniach między systemami transputerowymi, do połączeń transputera z kontrolerami peryferiali f-my INMOS, systemem I/O lub układami mikroprocesorowymi INTEL, MOTOROLA i innych firm. Adapter ten przyjmuje z linii komunikacyjnej INMOS szeregowo i przekształca je w postać 8-bitowego słowa lub odwrotnie. Służą w tym celu rejestry we/wy danych. Kontrolę zapewnia system sterujący, który bada słowo stanu rejestru sterującego oraz rejestry statusu stowarzyszone z rejestrami we/wy danych. 8-bitowe dane są wysyłane, bądź pobierane z bufora danych D0—D7

od strony magistrali systemowej. Do bloku sterującego doprowadzone są sygnały zasilania, zegar oraz sygnał separacji i sygnał wyboru szybkości transmisji. Rejestr stanu bloku sterującego zatrzaskuje informacje o trybie pracy (bity RW, RS1, RSØ) oraz o gotowości linii komunikacyjnej INMOS. Na kolejnym rysunku przedstawiona została karta firmy Transtech TSMB-16. Jest ona kartą umożliwiającą budowanie sieci złożonej z modułów transputerowych (do 16 modułów z transputerem T-414 lub T-800 z 64 MB







DRAM). W skład jej wchodzą adaptery linii z 2 interfejsami RS232 oraz 2 moduły INMOS IMS CØØ4 — które stanowią programowalną łącznicę z zespołem 32 linii we i 32 linii wy. Programowanie przełącznicy odbywa się poprzez 2 linie zwane liniami konfiguracji, ponadto może ono się odbywać w sposób dynamiczny. Możliwe jest łączenie kaskadowe modułów CØØ4. Szybkość transmisji wynosi 10 lub 20 Mb/s zapewniając maksymalne opóźnienie do 2 bitów. Karta TSMB ma także 2 linie (z 8 możliwych, wybieranych programowo), które są wyprowadzone na zewnątrz (backplane) W ten sposób można do niej dołączać inne karty modułów transputerowych firmy Transtech lub INMOS i kompatybilne z nimi TSMB-16 znacznie zwiększa możliwości budowania na jej podstawie wielkich systemów transputerowych i mikroprocesorowych. Jest ona wyposażona w niezbędne sygnały sterujące i testujące, zapewniając właściwy poziom współpracy wszystkich elementów sieci.

W.M.S

Przedstawiamy zaktualizowaną listę (por. „IKS" nr 1/87 oraz nr 12/87) skrótów, które związane są z mikrokomputerem, łącznie z ich oryginalną interpretacją. Czekamy na inne, Waszym zdaniem, ważne skróty

W jednym z najbliższych numerów „IKS-a" postaramy się przedstawić uaktualnioną listę skrótów wraz z ich tłumaczeniem (interpretacją) w języku polskim

SLAM – SCAN-LINE ACCESS MEMORY,
SLIC – SELECTIVE LISTING COMBINATION,
SLT – SOLID LOGIC TECHNOLOGY,
SNA – SYSTEM NETWORK ARCHITECTURE,
SNI – SERIAL NETWORK INTERFACE
SNR – SIGNAL-TO-NOISE RATIO,
SOH – START OF HEADER,
SOL – SIMULATION ORIENTED LANGUAGE,
SP – SPACE CHARACTER,
SP – STACK POINTER,
SPOOL – SIMULTANEOUS PERIPHERAL OPERATIONS OFF-LINE,
SPP – SPECIAL-PURPOSE PROCESSOR,
SPRS – SIGNAL PROCESSING ROUTER/SCHEDULER,
SPT – SHORTEST PROCESSING TIME,
SQL – STANDARD QUERY LANGUAGE,
SR – SEQUENCE REGISTER,
SRAM – STATIC RAM,
SRQ – SERVICE REQUEST,
SSI – SMALL SCALE INTEGRATION,
SST – SYNCHRONOUS SYSTEM TRAP,
STX – START OF TEXT,
SYN – SYNCHRONOUS IDLE,
SYSIN – SYSTEM INPUT,
TBF – TIME BETWEEN FAILURES,
TCC – TRANSMISSION CONTROL CHARACTER,
TCP – TRANSMISSION CONTROL PROTOCOL,
TDM – TIME DIVISION MULTIPLEXING,
TDMA – TIME DIVISION MULTIPLE ACCESS,
TELEX – TELEPRINTER-EXCHANGE,
TPI – TRACKS PER INCH,
TRL – TRANSISTOR-RESISTOR LOGIC,
TS – TRANSPORT STATION,
TSL – THREE-STATE-LOGIC,
TSS – TIME-SHARING SYSTEM,
TSS – TASK STATE SEGMENT,
TTD – TEMPORARY TEXT DELAY,
TTL – TRANSISTOR-TRANSISTOR LOGIC,
UART – UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER,
UC – UPPER CASE,
UDC – UNIVERSAL DECIMAL CODE,
UPC – UNIVERSAL PRODUCT CODE,
US – UNIT SEPARATOR,
USART – UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER-TRANSMITTER.
VC – VIRTUAL CALL,
VDU – VIDEO DISPLAY UNIT,
VDU – VISUAL DISPLAY UNIT,
VGA – VIDEO GATE ARRAY,
VIC – VIDEO INTERFACE CHIP,
VLSI – VERY LARGE SCALE INTEGRATION,
VOLSER – VOLUME SERIAL NUMBER,
VSAM – VIRTUAL STORAGE ACCESS METHOD,
VRC – VERTICAL REDUNDANCY CHECK,
VT – VERTICAL TAB,
VTAM – VIRTUAL TELECOMMUNICATION ACCESS METHOD,
VTOC – VOLUME TABLE OF CONTENTS,
VW-GRAMMAR – VAN WIJNGAARDEN GRAMMAR,
WACK – WAIT BEFORE TRASMITTING POSITIVE ACKNOWLEDGEMENT,
WADEX – WORD AND AUTHOR INDEX,
WE – WRITE ENABLE,
WPR – WRITE PERMISSION RING,



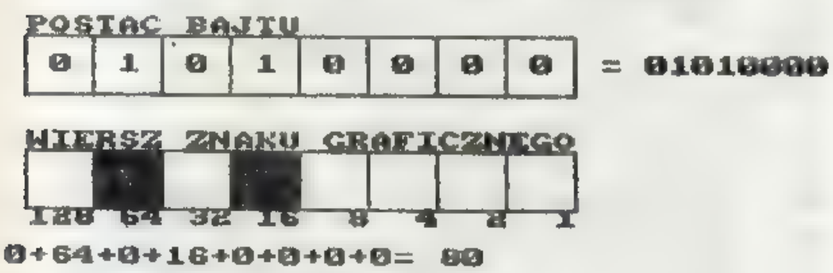
— Komputer prosi Kowalskiego.

SAMI TWORZYMY SYMBOLE GRAFICZNE

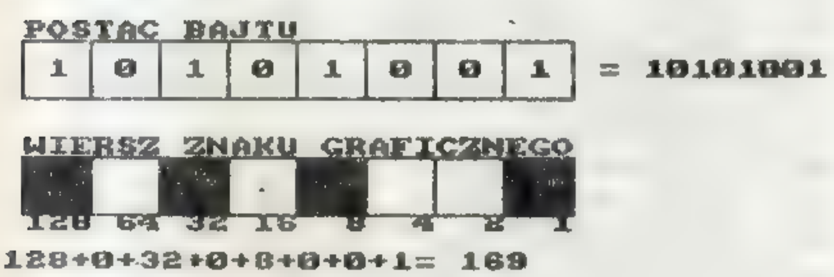
Mieczysław SKONIECZNY , Robert ZUGEHOR

Początkującemu użytkownikowi mikrokomputera wiele trudności nastręcza definiowanie własnych symboli graficznych

Prezentowany program ma na celu wyjaśnienie tego problemu na przykładzie mikrokomputera AMSTRAD CPC 6128. Program kreśli na ekranie powiększoną siatkę znaku, złożoną z 64 punktów. Następnie prezentowana jest: postać jednego wiersza znaku graficznego oraz odpowiadająca mu binarna wartość bajtu. Ma to na celu wyjaśnienie zależności pomiędzy wypełnieniem odpowiedniego pola wiersza znaku graficznego a jego interpretacją liczbową i binarną. Wypełniając pixelami kolejne pola, obserwujemy zmianę obliczanych wartości bajtu. Tę sytuację prezentują poniższe rysunki:



rys 1

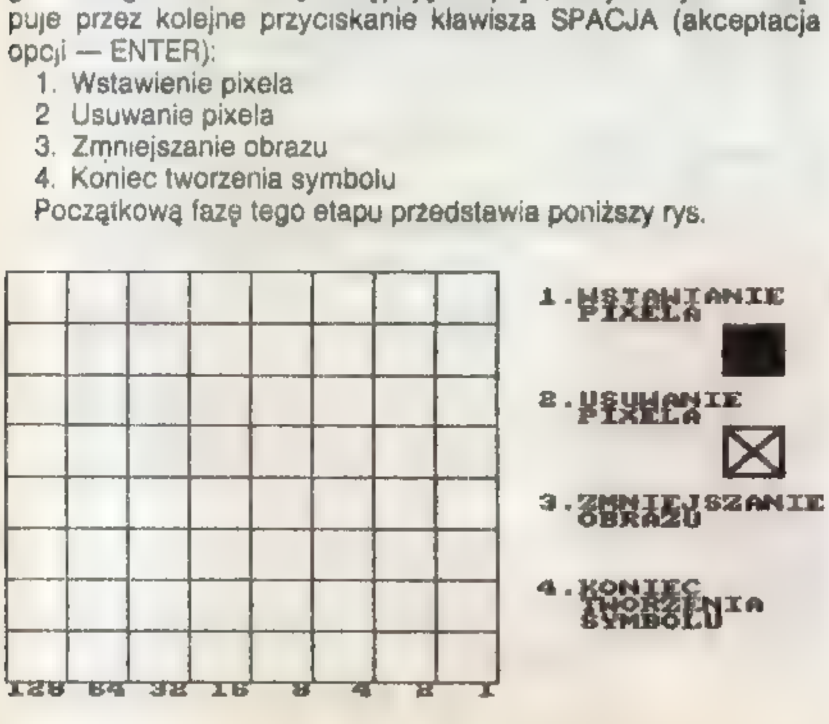


rys. 2

Dalsza część programu pozwala na tworzenie całego znaku graficznego. Możliwe są następujące opcje, których wybór następuje przez kolejne przyciskanie klawisza SPACJA (akceptacja opcji — ENTER):

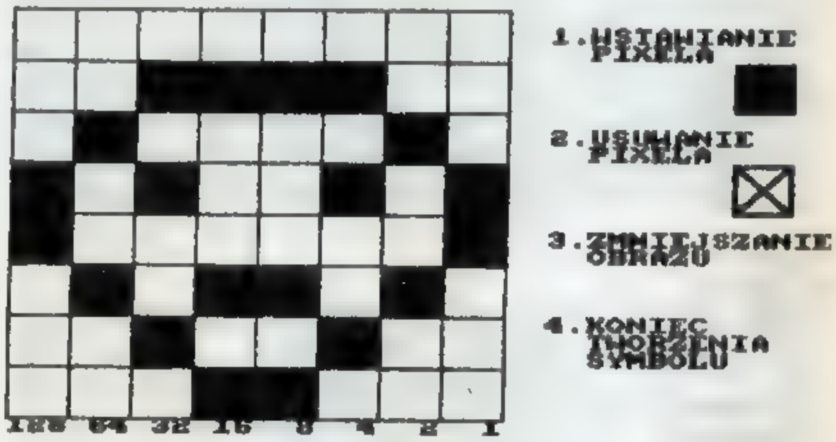
1. Wstawienie pixela
2. Usuwanie pixela
3. Zmniejszanie obrazu
4. Koniec tworzenia symbolu

Początkową fazę tego etapu przedstawia poniższy rys.



rys 3

Przykładowo utworzony znak pokazany jest na poniższym rysunku.



rys. 4

Wersja źródłowa tego programu ma następującą postać:

```
10 INK 0,24:INK 1,6:INK 2,0:BORDER 24:PAPER 0:MODE 0
20 t2$="        GRAFIKA        ":t3$="  MIKROKOMPUTEROWA  "
30 PEN 1
40 FOR x=1 TO 20
50 LOCATE 1,13:PRINT MID$(t2$,21-x,x);:LOCATE 21-x,14:PRINT MID$(t3$,1,x);
:NEXT
60 GOSUB 2700


70 GOSUB 2550
80 MODE 1:INK 1,6:INK 0,0:INK 2,9:INK 3,26:PEN 0:PAPER 3:BORDER 26:CLS
90 GRAPHICS PEN 0:PEN 1
100 WINDOW £1,12,29,11,14:PAPER £1,0
110 DIM c(8)
120 FOR x=127 TO 511 STEP 48
130 PLOT x,400:DRAWR 0,-384
140 NEXT
150 FOR y=399 TO 0 STEP -48
160 PLOT 127,y:DRAWR 384,0
170 NEXT
180 |SCREENCOPY,2,1
190 a$=INKEY$
200 CLS £1:PEN £1,1:LOCATE £1,6,2:PRINT £1,"POSTAC":LOCATE £1,7,3:PRINT £1,
"ZNAKU"
210 FOR t=1 TO 300:NEXT
220 |SCREENCOPY,1,2
230 FOR t=1 TO 300:NEXT
240 IF a$="" GOTO 190
250 CLS
260 LOCATE 3,5:PRINT"0  0  0  0  0  0  0  0  = 00000000":GOSUB 2730
270 LOCATE 2,3:PRINT"POSTAC BAJTU"
280 LOCATE 2,9:PRINT"WIERSZ ZNAKU GRAFICZNEGO":LOCATE 2,13:PRINT
"128 64 32 16  8  4  2  1"
290 GOSUB 2760
300 |SCREENCOPY,2,1
310 PEN 0:CLEAR INPUT
320 |SCREENCOPY,1,2
330 a$=INKEY$
340 FOR j=1 TO 3:LOCATE 26,9+j:PRINT CHR$(201);CHR$(201);CHR$(201);:NEXT
350 FOR t=1 TO 200:NEXT
360 FOR j=1 TO 3:LOCATE 26,9+j:PRINT"   ";:NEXT
370 IF a$="" GOTO 330
380 IF a$=" " GOTO 660
390 px=2:kx=23:x=26:y=10
400 a$=INKEY$
410 IF a$="" THEN |SCREENCOPY,1,2:GOSUB 2470:GOSUB 2510:GOTO 400
420 IF ASC(a$)=13 GOTO 520
430 IF ASC(a$)=32 GOTO 660
440 IF ASC(a$)<242 OR ASC(a$)>243 GOTO 400
450 IF ASC(a$)=242 THEN x=x-3
460 IF ASC(a$)=243 THEN x=x+3
470 IF x<px THEN x=px
480 IF x>kx THEN x=kx
490 |SCREENCOPY,1,2
500 GOSUB 2510
```

```
510 GOTO 400
520 IF x>kx GOTO 330
530 c(INT(x/3)+1)=1:PEN 1
540 FOR i=1 TO 8
550 LOCATE 3*(i-1)+2,5:PRINT c(i);:NEXT:GOSUB 2730
560 LOCATE 29,5
570 FOR i=1 TO 8
580 PRINT MID$(STR$(c(i)),2,1);:NEXT
590 |SCREENCOPY,2,1:LOCATE 1,15:s=0:PEN 0
600 FOR i=1 TO 8
610 m=c(i)*2^(8-i):s=s+m:PRINT MID$(STR$(m),2,3);
620 IF i=8 THEN PRINT"="; ELSE PRINT"+";
630 NEXT
640 PRINT s
650 PEN 0:GOTO 330
660 |SCREENCOPY,1,2:ERASE c
670 GOSUB 2700:CLS
680 LOCATE 1,25:PRINT"128 64 32 16  8  4  2  1";
690 GRAPHICS PEN 0
700 FOR x=0 TO 384 STEP 48
710 PLOT x,400:DRAWR 0,-384
720 NEXT x
730 FOR y=399 TO 0 STEP -48
740 PLOT 0,y:DRAWR 384,0
750 NEXT y
760 |SCREENCOPY,3,1
770 GOSUB 2320:GOSUB 2370:GOSUB 2420:GOSUB 2440
780 |SCREENCOPY,2,1
790 bord=0:GOSUB 2320:GOSUB 2370:GOSUB 2420:GOSUB 2440
900 GOSUB 2700
810 IF ASC(a$)<>32 AND ASC(a$)<>13 GOTO 800
920 IF ASC(a$)=13 AND bord<>0 GOTO 890
830 IF ASC(a$)=13 AND bord=0 THEN GOTO 800
940 bord=bord+1
850 IF bord>4 THEN bord=1
860 ON bord GOSUB 2800,2820,2840,2860

870 LOCATE 38,24:PRINT bord:PAPER 3
880 GOTO 800
890 BORDER 26:ON bord GOSUB 1000,1310,1620,2020
900 GOTO 790
910 END
```

Poszczególne funkcje realizują następujące podprogramy:
1 Stawianie punktu w dowolnym miejscu siatki znaku.

```
1000 PEN 0:CLEAR INPUT
1010 |SCREENCOPY,1,3
1020 a$=INKEY$
1030 FOR j=1 TO 3:LOCATE 25,21+j:PRINT CHR$(201);CHR$(201);CHR$(201);:NEXT
1040 FOR t=1 TO 200:NEXT
1050 FOR j=1 TO 3:LOCATE 25,21+j:PRINT"   ";:NEXT
1060 IF a$="" GOTO 1020
1070 IF a$=" " GOTO 1290
1080 px=1:kx=22:py=1:ky=22:x=25:y=22
1090 a$=INKEY$
1100 IF a$="" THEN |SCREENCOPY,1,3:GOSUB 2470:GOSUB 2510:GOTO 1090
1110 IF ASC(a$)=13 GOTO 1250
1120 IF ASC(a$)=32 GOTO 1290
1130 IF ASC(a$)<240 OR ASC(a$)>243 GOTO 1090
1140 IF ASC(a$)=240 THEN y=y-3
1150 IF ASC(a$)=241 THEN y=y+3
1160 IF ASC(a$)=242 THEN x=x-3
1170 IF ASC(a$)=243 THEN x=x+3
1180 IF x<px THEN x=px
1190 IF x>kx THEN x=kx
1200 IF y<py THEN y=py
1210 IF y>ky THEN y=ky
1220 |SCREENCOPY,1,3
1230 GOSJB 2510
1240 GOTO 1090
1250 IF x>kx GOTO 1020
1260 |SCREENCOPY,3,1
1270 a(INT(x/3)+1,INT(y/3)+1)=1
1280 GOTO 1020
1290 |SCREENCOPY,1,3
1300 RETURN
```

Sterowanie ruchem „kostki" odbywa się za pomocą klawiszy funkcyjnych (↑,↓,→,←). Umieszczenie „kostki" w wybranym miejscu — ENTER, przejście do innej funkcji programu — SPACJA.

2. Usuwanie pixela z dowolnego pola.

```
1310 PEN 1:CLEAR INPUT
1320 |SCREENCOPY,1,3
1330 a$=INKEY$
1340 FOR j=1 TO 3:FOR i=1 TO 3:LOCATE 24+j,21+i:PRINT CHR$(201+3*(i-1)+j);
:NEXT i:NEXT j
1350 FOR t=1 TO 200:NEXT
1360 FOR j=1 TO 3:LOCATE 25,21+j:PRINT"   ";:NEXT
1370 IF a$="" GOTO 1330
1380 IF a$=" " GOTO 1600
1390 px=1:kx=22:py=1:ky=22:x=25:y=22
1400 a$=INKEY$
1410 IF a$="" GOTO 1400
1420 IF ASC(a$)=13 GOTO 1560
1430 IF ASC(a$)=32 GOTO 1600
1440 IF ASC(a$)<240 OR ASC(a$)>243 GOTO 1400
1450 IF ASC(a$)=240 THEN y=y-3
1460 IF ASC(a$)=241 THEN y=y+3
1470 IF ASC(a$)=242 THEN x=x-3
1480 IF ASC(a$)=243 THEN x=x+3
1490 IF x<px THEN x=px
1500 IF x>kx THEN x=kx
1510 IF y<py THEN y=py
1520 IF y>ky THEN y=ky
1530 |SCREENCOPY,1,3
1540 FOR j=1 TO 3:FOR i=1 TO 3:LOCATE x-1+j,y-1+i:PRINT CHR$(201+3*(i-1)+j)
;:NEXT i:NEXT j
1550 GOTO 1400
1560 IF x>kx GOTO 1330
1570 PEN 0:LOCATE x,y:PRINT CHR$(211);CHR$(212);CHR$(212);:LOCATE x,y+1:
PRINT CHR$(213);"  ";:LOCATE x,y+2:PRINT CHR$(213);"  ";:PEN
 1:|SCREENCOPY,3,1
1580 a(INT(x/3)+1,INT(y/3)+1)=0
1590 GOTO 1330
1600 |SCREENCOPY,1,3
1610 RETURN
```

Sterowanie symbolem „usuwania" odbywa się jak wyżej.
3. Pomniejszanie obrazu.

```
1620 |SCREENCOPY,1,3
1630 skala=1
1640 a$=INKEY$
1650 IF a$="" GOTO 1640
1660 IF a$=" " GOTO 2000
1670 IF ASC(a$)<240 OR ASC(a$)>241 GOTO 1640
1680 IF ASC(a$)=241 THEN skala=skala*2
1690 IF ASC(a$)=240 THEN skala=skala/2
1700 IF skala<1 THEN skala=1
1710 IF skala>16 THEN skala=32
1720 CLS
1730 IF skala =1 THEN |SCREENCOPY,1,3:GOTO 1640
1740 GRAPHICS PEN 0
1750 IF skala=16 GOTO 1830
1760 IF skala=32 GOTO 1940
1770 FOR x=0 TO 384/skala STEP 48/skala
1780 PLOT x,400:DRAWR 0,-384/skala
1790 NEXT
1800 FOR y=399 TO 399-(8*48/skala) STEP -48/skala
1810 PLOT 0,y:DRAWR 384/skala,0
1820 NEXT y
1830 FOR i=1 TO 8
1840 FOR j=1 TO 8
1850 IF a(i,j)=0 GOTO 1910
1860 poczk=(i-1)*48/skala:konk=poczk+48/skala
1870 poczl=399-(j-1)*48/skala:konl=poczl-48/skala
1880 FOR k=poczk TO konk
1890 PLOT k,poczl:DRAWR 0,-48/skala
1900 NEXT
1910 NEXT j
1920 NEXT i
1930 GOTO 1640
1940 FOR i=0 TO 7
1950 FOR j=0 TO 7
1960 IF a(i+1,j+1)=0 THEN GRAPHICS PEN 3 ELSE GRAPHICS PEN 0
1970 PLOT 2*i,399-2*j
1980 NEXT j:NEXT i
1990 GOTO 1640
2000 |SCREENCOPY,1,3
2010 RETURN
```

Znak jest przedstawiany w 6 kolejnych, pomniejszanych skalach aż do naturalnej wielkości.
Pomniejszanie — klawisz ↓
Powiększanie — klawisz ↑
4. Koniec tworzenia symbolu.

```
2020 |SCREENCOPY,1,3
2030 PEN 0:DIM b(8)
2040 FOR i=1 TO 8
2050 licz=0
2060 FOR j=1 TO 8
2070 licz=licz+a(j,i)*2^(8-j)
2080 NEXT j
2090 LOCATE 26,3*(i-1)+2:PRINT licz;:b(i)=licz
2100 NEXT i
2110 GOSUB 2700
2120 CLS
2130 FOR i=0 TO 7
2140 FOR j=0 TO 7
2150 IF a(i+1,j+1)=0 THEN GRAPHICS PEN 3 ELSE GRAPHICS PEN 0
2160 PLOT 2*i,399-2*j
2170 NEXT j:NEXT i
2180 LOCATE 3,1:PEN 1:PRINT"SYMBOL GRAFICZNY":LOCATE 3,11:PRINT"INSTRUKCJE
PROGRAMU":PEN 0
2190 LOCATE 1,12:PRINT"...":LOCATE 1,13:PRINT"...":LOCATE 1,14:PRINT"1000
SYMBOL AFTER n"
2200 LOCATE 1,15:PRINT"1010 SYMBOL n";
2210 FOR i=1 TO 8
2220 PRINT",";MID$(STR$(b(i)),2,3);
2230 NEXT i
2240 GOSUB 2700
2250 CLS:LOCATE 1,12:PRINT"            NOWY SYMBOL (T/N)"
2260 a$=INKEY$
2270 IF a$=""GOTO 2260
2280 IF UPPER$(a$)="T" THEN ERASE a:ERASE b:CLS:GOTO 680
2290 IF UPPER$(a$)="N" THEN |SCREENCOPY,1,3:ERASE b:GOTO 2310
2300 GOTO 2260
2310 RETURN
```

Z prawej strony siatki znaku wyświetlane są na wysokości odpowiednich wierszy ich wartości liczbowe. Następnie jest pokazany fragment programu definiujący dany symbol graficzny.

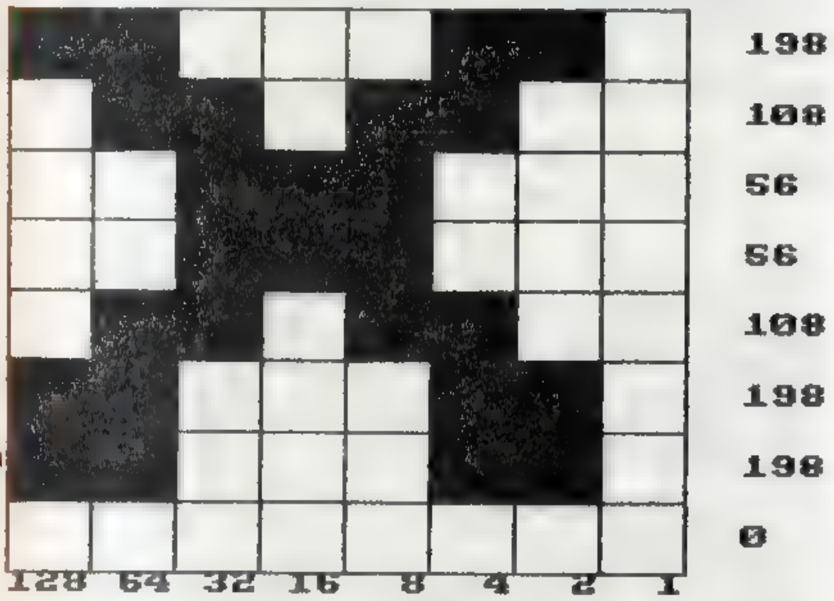
Tę sytuację prezentują poniższe rysunki.

```
X SYMBOL GRAFICZNY

 INSTRUKCJE PROGRAMU

1000 SYMBOL AFTER n
1010 SYMBOL n,198,108,56,56,108,198,198,
0
```

rys. 5



rys. 6

Dodatkowo są wykorzystywane następujące podprogramy pomocnicze

```
2320 FOR i=1 TO 3:FOR j=1 TO 3
2330 LOCATE 35+j,3+i:PEN 0:PRINT CHR$(201);
2340 NEXT j:NEXT i
2350 LOCATE 27,2:PRINT"1.WSTAWIANIE":LOCATE 27,3:PRINT"  PIXELA";
2360 RETURN
2370 FOR i=1 TO 3:FOR j=1 TO 3
2380 LOCATE 35+j,9+i:PEN 1:PRINT CHR$(201+3*(i-1)+j);
2390 NEXT j:NEXT i
2400 LOCATE 27,8:PRINT"2.USUWANIE ":LOCATE 27,9:PRINT"  PIXELA";
2410 RETURN
2420 LOCATE 27.14:PEN 2:PRINT"3.ZMNIEJSZANIE";:LOCATE 27,15:PRINT"  OBRAZU";
2430 RETURN
2440  WINDOW £1,26,40,18,22:PAPER £1,0:CLS £1
2450 LOCATE £1,2,2:PEN £1,3:PRINT £1,"4.KONIEC":LOCATE £1,2,3:PRINT£1,
"  TWORZENIA";:LOCATE £1,2,4:P
RINT£1,"  SYMBOLU"
2460 RETURN
2470 FOR i=0 TO 2
2480 LOCATE x,y+i:PRINT "   ";
2490 NEXT i
2500 RETURN
2510 FOR i=0 TO 2
2520 LOCATE x,y+i:PRINT CHR$(201);CHR$(201);CHR$(201);
2530 NEXT i
2540 RETURN
2550 SYMBOL AFTER 201
2560 SYMBOL 201,255,255,255,255,255,255,255,255
2570 SYMBOL 202,255,255,240,248,220,206,199,195
2580 SYMBOL 203,255,255,0,0,0,0,0,129
2590 SYMBOL 204,255,255,15,31,59,115,227,195
2600 SYMBOL 205,193,192,192,192,192,192,192,193
2610 SYMBOL 206,195,231,126,60,60,126,231,195
2620 SYMBOL 207,131,3,3,3,3,3,3,131
2630 SYMBOL 208,195,199,206,220,248,240,255,255
2640 SYMBOL 209,129,0,0,0,0,0,255,255
2650 SYMBOL 210,195,227,115,59,31,15,255,255
2660 SYMBOL 211,255,128,128,128,128,128,128,128
2670 SYMBOL 212,255
2680 SYMBOL 213,128,128,128,128,128,128,128,128
2690 RETURN
2700 a$=INKEY$
2710 IF a$="" GOTO 2700
2720 RETURN
2730 PLOT 15,351:DRAWR 384,0:PLOT 15,303:DRAWR 384,0
2740 FOR x=15 TO 399 STEP 48:PLOT x,351:DRAWR 0,-48:NEXT
2750 RETURN
2760 PLOT 15,255:DRAWR 384,0:PLOT 15,207:DRAWR 384,0
2770 FOR x=15 TO 399 STEP 48:PLOT x,255:DRAWR 0,-48:NEXT
2780 RETURN
2800 BORDER 0:PEN 0
2810 RETURN
2820 BORDER 6:PEN 1
2830 RETURN
2840 BORDER 9:PEN 2
2850 RETURN
2860 BORDER 26:PAPER 0:PEN 3
2870 RETURN
```

Realizacja programu odbywa się z wykorzystaniem dodatkowych 64 KB. Niezbędne jest więc uprzednie załadowanie programu BANKMANAGER

Robert ZUGEHOR

Mieczysław SKONIECZNY

Spearmen i Pearson powiedzą Ci prawdę

Mariusz KUBIAK

W czasie jednych z ostatnich ćwiczeń z podstaw informatyki w Wojskowej Akademii Politycznej byłem uczestnikiem atrakcyjnego, a przede wszystkim dającego wiele do myślenia eksperymentu (testu) z wykorzystaniem komputera.

Otóż grupa słuchaczy drugiego roku Wydziału Nauk Pedagogicznych w trakcie zajęć z wykorzystaniem sieci mikrokomputerów BBC miała m.in. za zadanie sprawdzenie możliwości zastosowania mikrokomputerów do obliczeń statystycznych. Wykorzystywane były do tego celu programy na „rachowanie" współczynników Spearmana i Pearsona. Jako przykładowy cel badań (by nie były zbyt odległe od rzeczywistości) przyjęliśmy szukanie zależności między naszymi wynikami nauki uzyskanymi w trzeciej sesji egzaminacyjnej, a następującymi zmiennymi: wiekiem, odległością od miejsca stałego zamieszkania, ilością wyjazdów do domu w ciągu miesiąca, ilością posiadanych dzieci, dochodami na osobę w rodzinie, ilością wypalanych papierosów w czasie dnia.

Oczywiście, aby wyniki testu w sposób maksymalnie obiektywny obrazowały wielkość wpływu poszczególnych zmiennych na ową średnią egzaminacyjną, każdy z nas musiał zdobyć się nie tylko na szczere przeprowadzenie „rachunku sumienia", lecz także ujawnienie swoich słabości do palenia papierosów.

Po wprowadzeniu danych do komputera, a następnie wypisaniu na tablicy wyników, najbardziej zdziwiony i zaskoczony był prowadzący ćwiczenie. Co się bowiem okazało? Ano to, że im słuchacze częściej wyjeżdżają do rodziny i im dalej mieszkają, a także... im więcej palą papierosów, tym lepsze osiągają wyniki w nauce. Natomiast wpływ pozostałych zmiennych okazał się nieco mniejszy, ale także bardzo istotny. Wobec tak nieoczekiwanych wyników sprawdziliśmy dodatkowo na mikrokomputerze raz jeszcze rezultaty testu, licząc — przynajmniej niektórzy z nas — na błąd maszyny. Niestety, mikrokomputer okazał się nieprzekupnym, perfekcyjnym „mózgiem".

Kończąc to krótkie sprawozdanie z ćwiczeń komputerowych, chciałbym zaznaczyć, że moim zdaniem przydatność komputera do pracy dydaktyczno-wychowawczej czy naukowej niewspółmiernie wzrasta, jeśli jest on wykorzystywany do rozwiązywania problemów bezpośrednio dotyczących człowieka, który go obsługuje, ewentualnie zaspokaja jego ciekawość poznawczo-naukową. W każdym razie jeszcze długo po ćwiczeniu komentowaliśmy w swoim gronie jego wyniki, wielu z nas chętniej sięgnęło po czasopisma i specjalistyczną literaturę komputerową, a niektórzy poważnie myślą o zakupie „Spectrum" czy „Atari".

Po raz kolejny przekonaliśmy się, że komputer to nie tylko gry i zabawy wymagające pewnej sprawności manualnej, lecz również cenna pomoc w zdobywaniu wiedzy nie tylko o otaczającej nas rzeczywistości, ale także o nas samych.

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

END OF MEDIUM CHARACTER – znak fizycznego zakończenia nośnika,
END OF MESSAGE – koniec komunikatu,
END OF RECORD – koniec zapisu (rekordu),
END-OF-RECORD FLAG – identyfikator końca zapisu,
END OF REEL – koniec szpuli,
END OF TEXT CHARACTER – znak zakończenia tekstu (treści),
END OF THE TAPE – znacznik końca taśmy,
END OF TRANSMISSION – koniec transmisji,
END OF TRANSMISSION CHARACTER – znak zakończenia transmisji,
END OF TRANSMISSION BLOCK CHARACTER – znak zakończenia transmisji bloku,
END OFFICE – lokalna centrala telefoniczna,
ENDPOINT – punkt końcowy,
END-TO-END PROTOCOL – protokół końcowy,
ENERGIZE – wzbudzać, zasilać, także: zapalać,

ESCAPE SEQUENCE – sekwencja zarządzająca,
ESCAPING KEY – klawisz powodujący posuw papieru,
ESD – patrz: EXTERNAL SYMBOL DICTIONARY,
ESS – patrz: ELECTRIC SWITCHING SYSTEM,
ESSENCE – istota (rzeczy),
ESSENTIAL – istotny, zasadniczy,
ESTABLISH – ustanowić, założyć,
ESTABLISH A RELATIONSHIP – ustalić zależność,
ESTABLISHMENT – firma, przedsiębiorstwo,
ESTIMATE – oszacowanie, szacunek, ocena, także: szacować, oznaczać,
ESTIMATION – ocena, oszacowanie, także: estymacja,
ETB – patrz: END OF TRANSMISSION BLOCK CHARACTER,
ETCHING – trawienie, wytrawianie,
ETX – patrz: END OF TEXT CHARACTER,
EU – patrz: EXECUTION UNIT,
EURONET – sieć komputerowa EURONET,
EUROPEAN ACADEMIC RESEARCH NETWORK – Międzynarodowy Skomputeryzowany System Wymiany Informacji Naukowej,
EUROPEAN COMPUTER MANUFACTURES ASSOCIATION – Europejskie Stowarzyszenie Producentów Sprzętu Komputerowego,
EUROPEAN INFORMATION NETWORK – sieć komputerowa EIN,
EVADE – unikać, obchodzić,
EVALUATE – obliczać (np. wartość wyrażenia arytmetycznego), oceniać,
EVALUATION – wyznaczanie wartości, ocena, obliczanie,
EVALUATION SYSTEM – system oceny,
EVEN – parzystość, także: równać, wyrównywać,
EVEN NUMBER – liczba parzysta,
EVEN-ODD – parzysty-nieparzysty,
EVEN PARITY BIT – bit parzystości,
EVEN PARITY CHECK – kontrola parzystości,
EVENT – zaszłość, także: zdarzenie, wydarzenie,
EVENT-DRIVEN – zarządzający przerwaniami w trybie przerwania,
EVENT FLAG – znak wystąpienia zdarzenia,
EVENT MODE INPUT – wprowadzenie według kolejki, wprowadzenie z buforowaniem,
EVENT TRAPPING – obsługa przerwania,
EVIDENCE – dowód, także: świadczyć, udowadniać,
EVOLUTION – rozwijanie, rozwój, ewolucja, także: pierwiastkowanie,
EVOLUTIONARY SYSTEM – system rozwijający się,

Słownik na dysku

Od kilku miesięcy zawsze na tych samych kolumnach „IKSa" publikujemy kolejne odcinki informatycznego słownika angielsko-polskiego. Cieszy nas, że ma on liczną grupę odbiorców.

Tym przyjemniej jest nam poinformować, o udanym oprogramowaniu niemal 16-tysięcznego zbioru terminów angielskich i ich polskich odpowiedników. Oprogramowania słownika dokonała firma DORAN, instalując je jako integralną część edytora tekstów, rozbudowującą jego możliwości o funkcję „tłumacza". Mieliśmy okazję korzystać ze słownika w nowej postaci, jest prosty w obsłudze, a szybkość dostępu do każdego z haseł jest mniejsza niż 1/2 sekundy.

Tyle wiemy. O bliższe informacje radzimy się zwracać listownie do firmy DORAN, której specjalnością jest oprogramowanie. 03-756 W-wa ul. Siedlecka 1/15 m. 74.

EVOLVE – wymyśleć, wykombinować,
EXACT – dokładny, ścisły,
EXACT DIVISION – dzielenie bez reszty,
EXACTITUDE – dokładność, ścisłość,
EXAMINE – badać, rozpatrywać,
EXAMPLE – przykład,
EXCEED – przekraczać (np. pojemność pamięci),
EXCEED CAPACITY CHECK – kontrola przepełnienia,
EXCEEDING – niezmierzony, nadzwyczajny,
EXCELLENT – doskonały, znakomity,
EXCEPT – wykluczać, wyłączać, także: z wyjątkiem,
EXCEPTION – wyjątek,
EXCEPTION CONDITION – wydarzenie wyjątkowe,

EXCEPTION HANDLER – program obsługi zdarzenia wyjątkowego,
EXCEPTION HANDLING – obsługiwanie zdarzenia wyjątkowego, (przerwanie wewnętrzne),
EXCEPTIONAL – wyjątkowy,
EXCESS – nadmiar, nadwyżka, przekroczenie,
EXCESS-THREE CODE – kod z nadmiarem trzy, kod Stibitza,
EXCHANGE – wymiana, zamiana, także: wymieniać, zamieniać,
EXCHANGEABLE – wymienny,
EXCHANGEABLE DISK – dysk wymienny,
EXCHANGEABLE DISK STORE – pamięć na dysku wymiennym,
EXCITATION – pobudzenie, wzbudzenie,
EXCITE – wzbudzać, pobudzać,
EXCLAMATION MARK – znak wykrzyknika, "!",
EXCLUDE – wykluczać, wyłączać,
EXCLUSIVE LOCKING – blokada wybiórcza, także: blokowanie dla zapisu,
EXCLUSIVE OR – "LUB" wykluczające,
EXCLUSIVE SEGMENTS – wzajemnie wykluczające się moduły,
EXCLUSIVE USAGE MODE – tryb wyłączności,
EXE – patrz: EXECUTE,
EXE FILE – moduł ładowania, moduł wprowadzania,
EXECUTE – wykonywać,
EXECUTE ONLY PROGRAM – program bez opisów, program bez komentarzy,
EXECUTE PROGRAM – program wykonawczy,
EXECUTING – wykonanie, także: uruchomienie,
EXECUTION – wykonanie,
EXECUTION CYCLE – cykl wykonania,
EXECUTION ENVIRONMENT – warunki (środki) wykonania programu,
EXECUTION TIME – czas wykonania, czas obliczeń,
EXECUTION UNIT – układ wykonawczy,
EXECUTIVE – program zarządzający, egzekutor, także: system operacyjny,
EXECUTIVE DEVICE – człon operacyjny (komputera analogowego),
EXECUTIVE DIRECTIVE – dyrektywa (operacja) systemu operacyjnego,
EXECUTIVE INSTRUCTION – instrukcja (rozkaz) systemu operacyjnego,
EXECUTIVE MODE – priorytetowy tryb pracy,
EXECUTIVE PROGRAM – program organizacyjny, program sterujący, program wykonawczy, program nadzorczy, program zarządzający,
EXECUTIVE REPLIES – informacje EXECUTIVE'a umieszczane w słowach odpowiedzi,
EXECUTIVE RESIDENT – część systemu operacyjnego, rezydująca na stałe w pamięci operacyjnej,
EXECUTIVE SUPERVISOR – program zarządzający systemu operacyjnego,
EXECUTIVE SYSTEM – system operacyjny,
EXECUTIVE SYSTEM UTILITY – systemowy program serwisowy,
EXEMPLIFY – ilustrować (na przykładzie),
EXERCISE CONTROL OVER ... – sprawować kontrolę nad ..., nadzorować,
EXERCISER – program testujący, program kontrolny,
EXHAUSTED ARGUMENT – parametr wyczerpany (oznacza przeanalizowanie wszystkich wariantów problemu),
EXHAUSTIVE SEARCH – pełne przeszukanie,
EXHIBIT – wykazywać (cechę), wskazywać,
EXIST – istnieć,
EXISTENCE – istnienie, egzystencja,
EXIT – wyjście,
EXIT CONDITIONS – warunki wyjścia, także: realizacja warunków po wyjściu z podprogramu,
EXJUNCTION – "LUB" wykluczające,
EXPAND – rozszerzać, rozwijać (np. wzór matematyczny),
EXPANDABILITY – możliwość rozszerzania (np. pojemności pamięci),
EXPANDABLE SYSTEM – system rozszerzalny, system dostosowany do rozbudowy,
EXPANDER – ekspander (sygnału),
EXPANDING – rozszerzanie (się), rozprężanie (się), także: rozwijanie (funkcji),
EXPANSION – rozwinięcie, rozszerzenie, rozbudowa, uaktywnienie,
EXPANSION BOX – blok rozbudowy,
EXPANSION BUS – szyna rozszerzenia,

EXPANSION SLOT - złącze do rozbudowy funkcjonalnej,
EXPECTATION - wartość oczekiwana (zmiennej losowej), wartość przeciętna, także: oczekiwanie,
EXPECTED VALUE - wartość oczekiwana (zmiennej losowej), nadzieja matematyczna, wartość przeciętna,
EXPEL - wydalać, usuwać,
EXPENDABLE - jednorazowego użytku,
EXPENDITURE - wydatki, rozchód,
EXPENDITURE QUOTA - normy zużycia,
EXPENSES - wydatki,
EXPERIENCE - doświadczenie (nabyte),
EXPERIMENT - doświadczenie, eksperyment, także: przeprowadzać doświadczenie, przeprowadzać eksperyment,
EXPERIMENTAL - doświadczalny,
EXPERIMENTAL CENTRE - ośrodek doświadczalny,
EXPERIMENTAL CONDITIONS - warunki doświadczalne,
EXPERIMENTAL ERROR - błąd doświadczenia,
EXPERIMENTATION - wykonywanie doświadczeń, przeprowadzanie doświadczeń,
EXPERT OPINION - ekspertyza,
EXPERT SYSTEM - system ekspertowy, system ekspercki,
EXPIRATION DATE - termin upływu (np. ważności), data ważności,
EXPLANATION - objaśnienie, wyjaśnienie,
EXPLANATION FACILITIES - środki umotywowania, środki uzasadniania (dot. systemów ekspertowych),
EXPLANATORY - wyjaśniający, tłumaczący,
EXPLICIT - jawny, jawnie, także: jasny, sprecyzowany, wyraźny, czytelny,
EXPLODED VIEW - przedstawienie częściowe, także: obraz częściowy,
EXPLORATION - poszukiwanie, badanie,
EXPONENT - cecha, wykładnik potęgi,
EXPONENTIAL - wykładniczy,
EXPONENTIALLY - wykładniczo,
EXPONENTIAL NOTATION - zapis liczb w postaci wykładniczej,
EXPONENTIATE - podnosić do potęgi,
EXPONENTIATION - potęgowanie,
EXPORTED - przesyłany (eksportowany),
EXPRESS - wyrażać się,
EXPRESSION - wyrażenie,
EXTEND - rozciągać, przedłużać, rozszerzać,

EXTEND ACTIVITY - rozszerzać działalność,
EXTENDED ASCII - rozszerzony kod ASCII,
EXTENDED BACKUS-NAUR FORM - rozszerzona notacja Backusa-Naura,
EXTENDED BCD-CODE - patrz: EXTENDED BINARY-CODED DECIMAL CODE,
EXTENDED BINARY-CODED DECIMAL INTERCHANGE CODE - rozszerzony kod binarny do kodowania liczb dziesiętnych,
EXTENDED BINARY-CODED DECIMAL CODE - patrz: EXTENDED BINARY-CODED DECIMAL INTERCHANGE CODE,
EXTENDED BINARY INTERCHANGE CODE - kod 8-bitowy bez bitu kontrolnego (dający 256 możliwości),
EXTENDED CODE - patrz: EXTENDED BINARY INTERCHANGE CODE,
EXTENDED INSTRUCTION SET - repertuar rozkazów rozszerzony,
EXTENDED-PRECISION - podwyższona precyzja,
EXTENDER CARD - karta ekspandera,
EXTENDER CIRCUIT - ekspander,
EXTENDER INPUT - wejście ekspandera,
EXTENDING A FILE - rozszerzanie pliku (zbioru), powiększanie pliku,
EXTEND THE RANGE - zwiększyć zasięg,
EXTENSIBLE - rozszerzony, także: odkryty,
EXTENSIBLE ADDRESSING - adresacja w trybie rozszerzonym, adresacja rozszerzona,
EXTENSIBLE LANGUAGE - język rozszerzony,
EXTENSIBLE REGISTER - rejestr rozszerzony,
EXTENSION - rozwinięcie, rozszerzenie, rozbudowanie, także: zakres (pojęcia),
EXTENSION BUCKET - porcja (blok programowy) w obszarze przepełnień drugiego stopnia,
EXTENSION BUCKET HEADER - etykieta porcji dodatkowej (bloku dodatkowego),
EXTENSION REGISTER - rejestr rozszerzenia,
EXTENSIVE - obszerny, rozległy,
EXTENSIVE DAMAGE - poważne uszkodzenie,
EXTENT - podobszar, zasięg, także: stopień,
EXTERNAL ARITHMETIC - procesor arytmetyczny (dodatkowy),
EXTERNAL DATA REPRESENTATION - zewnętrzna reprezentacja danych,
EXTERNAL DELAY - strata czasu (np. komputera) spowodowana przyczynami zewnętrznymi,
EXTERNAL DEVICE - urządzenie zewnętrzne, urządzenie peryferyjne,
EXTERNAL ENVIRONMENT - warunki eksploatacji,
EXTERNAL INTERRUPT - przerwanie zewnętrzne,
EXTERNAL KEY - klucz zewnętrzny (w relacyjnych bazach danych oznacza dostęp do danych),
EXTERNAL MEMORY - pamięć zewnętrzna,
EXTERNAL PERFORMANCE - rzeczywista szybkość działania,
EXTERNAL REFERENCE - odsyłacz zewnętrzny,
EXTERNAL STORAGE - pamięć zewnętrzna,
EXTERNAL STORE - patrz: EXTERNAL STORAGE,
EXTERNAL SYMBOL DICTIONARY - słownik symboli zewnętrznych, także: lista symboli do tłumaczenia programów,
EXTRA-LARGE-SCALE INTEGRATION - bardzo duża skala integracji,
EXTRA SEGMENT - segment pomocniczy, segment dodatkowy,

EXTRACT - wybór, wyciąg, wypis, także: wydobywać, wybierać, wydostawać,
EXTRACT INSTRUCTION - rozkaz maskowania,
EXTRACTION - wydzielanie, maskowanie,

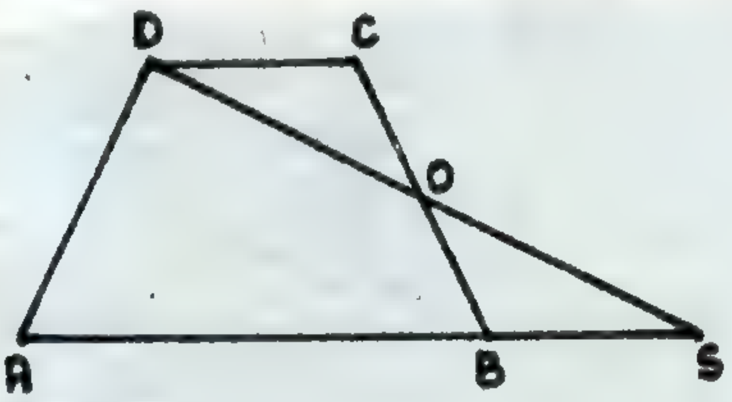
F

FALLBACK - rezerwa awaryjna, przejście na awaryjny tryb pracy, zmiana trybu pracy lub konfiguracji sprzętu w celu uniknięcia niesprawności,
FALLOUT - znikanie, zanikanie, przepadanie,
FALSE - fałszywy, także: fałsz (pojęcie używane w logice),
FALSE ACCEPTANCE - przyjęcie fałszywej hipotezy,
FALSEHOOD - przekłamanie, fałszywy wniosek, fałszywa decyzja,
FALSE REJECTION - odrzucenie hipotezy fałszywej,
FALSE RETRIEVAL - uzyskanie niepotrzebnej informacji w automatycznym systemie wyszukiwania,
FALSE STATEMENT - twierdzenie fałszywe,
FAMILY - rodzina,
FAMILY OF CHARACTERISTICS - rodzina charakterystyk,
FAMILY OF CURVES - rodzina krzywych,
FANFOLD - druk ciągły składany w "harmonię",
FANFOLD FORM - formularze ciągłe złożone w postaci harmonii,
FAN-IN - obciążalność wejściowa (bramki logicznej),
FAN-OUT - obciążalność wyjściowa (bramki logicznej),
FAR CALL - wywołanie długie, wywołanie odległe,
FAR END CROSS TALK - przesłuch skrośny,
FAR PLANE - daleki plan (w grafice komputerowej określenie płaszczyzny ograniczającej przestrzeń zobrazowania na ekranie),
FAR-REACHING - dalekosiężny,
FAST - szybki, szybko,
FAST-ACCESS STORAGE - pamięć szybka, pamięć o dostępie szybkim,
FAST-ACTING - szybki, szybko działający,
FAST LINE PRINTER - szybka drukarka wierszowa,
FAST MEMORY - szybka pamięć (operacyjna),
FAT - patrz: FILE ALLOCATION TABLE,
FATAL ERROR - błąd uniemożliwiający kontynuację programu,
FATHER FILE - początkowa (wyjściowa) wersja pliku (zbioru),
FAULT - 1. błąd, 2. uszkodzenie, zakłócenie, niesprawność, 3. wada (fizyczna wada sprzętu),
FAULT CLEARING - likwidacja zakłóceń, usuwanie uszkodzeń,
FAULT DETECTOR - wskaźnik uszkodzeń,
FAULT FINDING - wykrywanie uszkodzeń, także: usuwanie zakłóceń,
FAULTLESS - bezusterkowy, bezbłędny,
FAULTLESS TAPE - taśma sprawdzona, taśma bezbłędna,
FAULT LOCATION - lokalizacja uszkodzeń,
FAULT SIMULATOR - symulator uszkodzeń, symulator zakłóceń,
FAULT TIME - czas przestoju z powodu uszkodzenia,
FAULT-TOLERANT SYSTEM - system odporny na uszkodzenia,
FAULT TRACING - wyszukiwanie uszkodzeń,
FAULT TREE GATE - element drzewa niesprawności (usterek),
FAULTY - 1. błędny, 2. uszkodzony, zakłócony, niesprawny, 3. wadliwy,
FAULTY OPERATION - działanie wadliwe, działanie błędne,
FAX - patrz: FACSIMILE,
FAY - przylegać, dokładnie pasować,
FC - patrz: FONT-CHANGE CHARACTER,
FCB - patrz: FILE CONTROL BLOCK,
FCFS - patrz: FIRST-COME FIRST-SERVED,
FCP - patrz: FILE CONTROL PROCESSOR,

LIGA MYŚLĄCYCH

ZADANIE 1

Trapez ABCD na poniższym rysunku jest równoramienny. Punkt O jest środkiem boku BC, a punkt S punktem wspólnym prostej AB i prostej OD. Które z tych figur mają równe pola?



ZADANIE 2

Znaleźć najmniejszą liczbę naturalną n>1, dla której suma kwadratu kolejnych liczb naturalnych od 1 do n byłaby kwadratem liczby naturalnej.

ZADANIE 3

Siostra jest 3 lata młodsza od brata. Brat ma obecnie 2 razy tyle lat, ile miała siostra wtedy, kiedy brat miał tyle, ile siostra ma teraz. Ile lat ma siostra, a ile brat?

ZADANIE 4

Jeżeli cyfrę dziesiątek pewnej liczby dwucyfrowej zwiększymy o 4, a jej cyfrę jedności zmniejszymy o 2, to otrzymamy liczbę mniejszą od 86. Jeżeli zaś cyfrę dziesiątek tej liczby zmniejszymy o 2, a cyfrę jedności powiększymy o 1, to otrzymamy liczbę większą od 27. Jaka to liczba?

ZADANIE 5

Pociąg wyjechał ze stacji A do stacji C mijając po drodze stację B. Odległość pomiędzy stacjami A i B pociąg przejechał z pewną stałą prędkością, a odległość pomiędzy stacjami B i C z prędkością mniejszą od niej o 25 proc. W drodze powrotnej od stacji C do B pociąg jechał z prędkością taką samą, z jaką w pierwszą stronę jechał od stacji A do B, a ze stacji B do A z prędkością o 25 proc. mniejszą. Ile godzin jechał pociąg ze stacji A do C, jeżeli wiadomo, że odległość od stacji A do B przejechał w tym samym czasie co odległość od stacji B do C oraz że w kierunku od stacji A do C pociąg jechał o 5/12 godziny krócej niż z powrotem (tj. od C do A)?

Rozwiązanie zadań prosimy przesyłać do redakcji do końca września br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

TEŻ KOMPUTERY...

DOKOŃCZENIE ZE STR. 32

komputer od razu przełoży to na język włoski, francuski, niemiecki lub hiszpański i poda syntetycznym komputerowym głosem. Niezbyt miłym, ale zrozumiałym dla kogoś, kto zna któryś z tych języków.

„Voice" tłumaczy zdania w wybranym z tych pięciu języków. Jego pamięć zawiera ponad 2 tys. prostych zdań stanowiących zazwyczaj podstawowy zasób tzw. rozmówek dla turystów. Niedostatkiem tego zmyślnego urządzenia jest to, iż robi to tylko w jedną stronę. Nie można go natychmiast wykorzystać do przekładania odpowiedzi na język zrozumiały dla jego posiadacza. Za to „Voice" ma płaski ekran, na którym można wyświetlać 16 wierszy tekstu w wybranym języku.

Zdaniem prezesa firmy APT, Steve'a Rondela, natychmiastowe tłumaczenie w obie strony technologicznie jest całkiem realne. Lecz wówczas urządzenie byłoby zbyt masywne, by mogło służyć w wersji przenośnej, a poza tym kosztowałoby zbyt dużo. Obecnie ustalona cena — 1500 dolarów w Stanach Zjednoczonych — nie wydaje się zbyt wygórowana, a czas powinien sprzyjać rozwiązaniu problemów gabarytowych.

Firma Sharp dla nieco innych celów takie problemy już rozwiązała. Zbudowała mikrokomputer PC-1100, przy czym skrót PC nie oznacza Personal Computer, lecz Pocket Computer, czyli komputer kieszonkowy. Łączy on w sobie funkcje minikalkulatora i podręcznej bazy danych. Mimo cech podręcznego notatnika i terminarza PC-1100 jest mniejszy od typowego kalendarzyka, jego wymiary bowiem wynoszą 14x8x1 cm.

Wymienne karty pamięci RAM oraz możliwość programowania w uproszczonym języku Basic otwierają przed tym zminiaturyzowanym urządzeniem duże możliwości zastosowania. Miniekranik zbudowany z wyświetlaczy ciekłokrystalicznych umożliwia wyświetlanie dwóch linii po 16 znaków w każdej. Specjalna funkcja przeszukująca dane umożliwia dostęp do ponad setki zapamiętanych nazwisk i numerów telefonów. Wyszukane nazwisko wyświetlane jest w górnej części ekranu, inne dane w dolnej. PC-1100 umożliwia również zapamiętanie innych informacji takich, jak czas odjazdów pociągów czy terminy spotkań. Do pamięci urządzenia można wprowadzić także wszystkie dane personalne właściciela. Wielostronność zastosowań ułatwiają wymienne karty pamięci RAM w pakietach 2, 4 lub 8 KB.

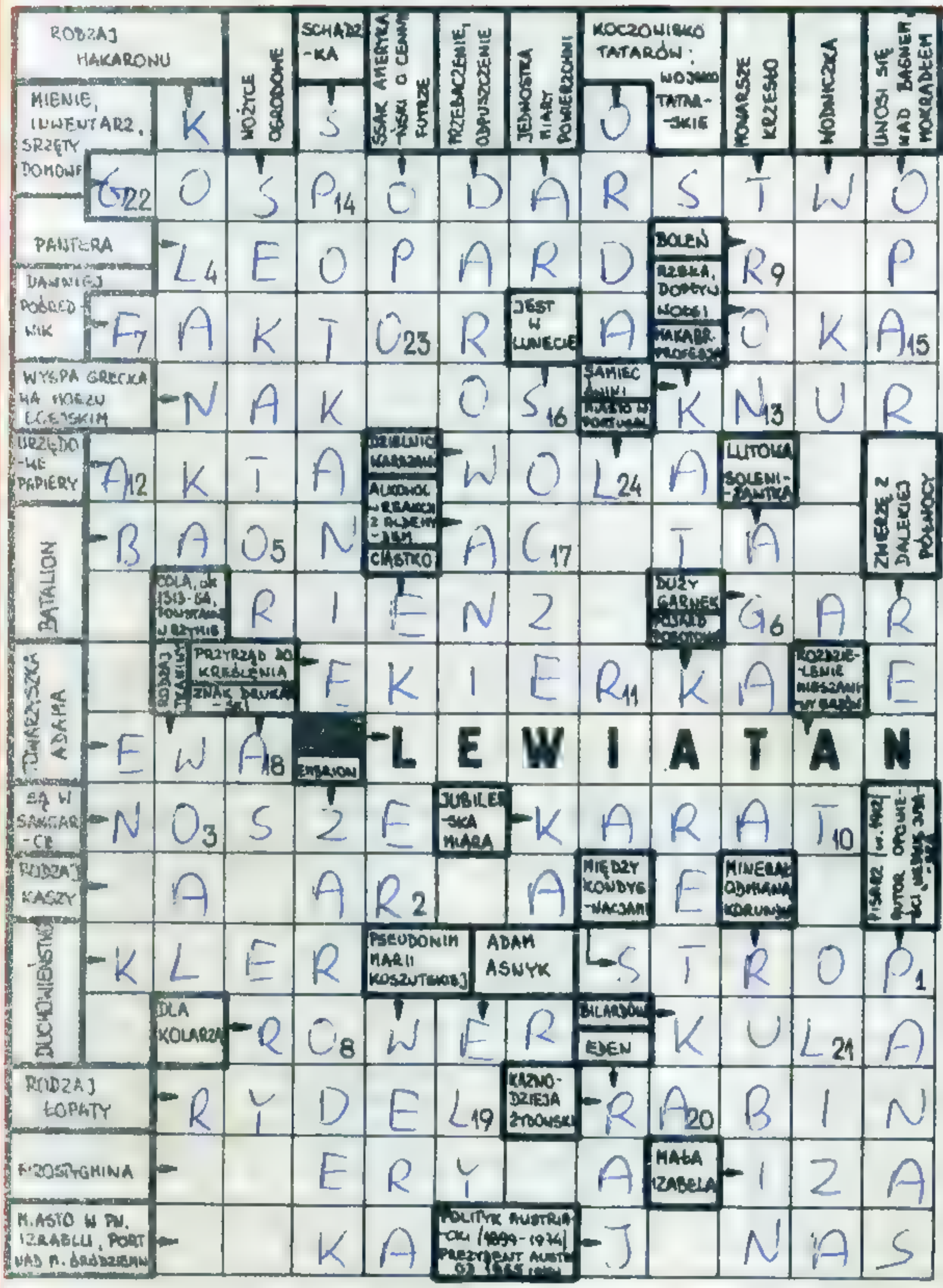
Oprogramowane kalkulatory są partnerem dla dużych komputerów, na przykład Atari lub IBM. W umysłach konstruktorów zrodził się pomysł, by je udoskonalić, wykorzystując do zapisu i edycji programów, dostępnych na kalkulatorze, obszaru pamięci komputerów personalnych. Jeden z nich, Atari ST ma zamkniętą stację dyskietek i ekran monitora mieszczący kilkadziesiąt liczb naraz. Ta zaleta skłoniła firmę Yellow Computing do wyprodukowania przyrządu elektronicznego, który za pomocą programu „Transfile ST" umożliwia połączenie kalkulatora kieszonkowego Sharp z wymienionym już komputerem.

Ten interfejs przenosi dane i programy z kalkulatora do pamięci komputera. Przez to użytkownik może opracować niezbędną informację i zapisać ją na dyskietce. Jednocześnie korzystać można z programów pomocniczych dla Atari, niedostępnych na zwykłym kalkulatorze. Po opracowaniu w ten sposób danych uzyskane informacje można z powrotem przesłać do kalkulatora. Odbywa się to przy użyciu rozkazów Cload, Csave i Print przez wejście Centronics, do którego podłączony jest interfejs. Zaletą tego urządzenia jest ominięcie zawodnej pamięci taśmowej i możliwość przeglądu zawartości wnętrza Sharpa.

Czego to ludzie nie wymyślą. Entuzjasta komputerów z Sydney, Barry Tucker, wpadł na pomysł wydawania gazety bez... jej drukowania. Takie pismo, dotyczące gier liczbowych, sprzedawane jest na dyskietkach. Nosi tytuł „Hazardzista" i przeznaczone jest — na razie — na komputery Amstrad serii PCW. Jego cena wynosi 6,5 dolara australijskiego, czyli prawie tyle samo, co normalne wydawnictwo tego typu. Cena obejmuje również koszt zwrotnej przesyłki. Klienci przysyłają swoje dyskietki, a Tucker kopiuje na nich swój magazyn i odsyła czytelnikom. Liczba „czytelników" rośnie, kolejne „nakłady" mają coraz więcej odbiorców.

J. RAJCH

KRZYŻÓWKA NR 6



Litery z kratek ponumerowanych od 1 do 24 utworzą 4 terminy związane z komputerami, które wystarczy podać jako rozwiązanie zadania przesyłając pod adresem redakcji na kartkach pocztowych w terminie do końca września (naklejając kupon „IKS-a".) Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.



TEŻ KOMPUTERY

Jerzy RAJCH

Wydaje się, że nie ma końca zastosowaniom komputera. Przed kilkoma laty amerykańska firma Intel wypuściła na rynek pierwszą bitową kostkę pamięci. Dzięki temu skonstruowana została karta zastępująca dowód osobisty, prawo jazdy lub każdy dokument tożsamości. Rzecz polegała na tym, że w obwód scalony sztywnej karty, czyli kostki pamięci, wpisywano wszelkie dane personalne. Plus takiej karty polega na tym, że dane nie dają się zniekształcić ani sfałszować.

Zwiększenie pojemności pamięciowej kostki do 32 bitów — w perspektywie do 64 bitów — doprowadziło do rozszerzenia zastosowań karty. Można już w niej zapisać nie tylko nazwisko, wiek i stan konta bankowego, ale także dane samochodu czy sposób otwarcia szyfrowanego sejfu. Taka utwardzona plastykiem wizytówka — znana w świecie jako „chipcard" — może być prawem do posiadania broni, książeczką zdrowia, legitymacją związkową i klubową, a nawet nie podrabianym kluczem do domu czy samochodu.

Aby wyposażyć obywatela w pamięciowy plastykowy dowód tożsamości, nie wystarczy wyprodukować dla każdego kartę i wprowadzić do niej niezbędne dane. Trzeba jeszcze stworzyć do tego niezbędny system, który zastąpiłby dotychczasowe metody zarządzania i rozliczeń. Robią to już niektóre zachodnie banki, gdzie wszelkie operacje księgowo-rozliczeniowe dokonywane są tylko w pamięci komputerów, a użytkownicy korzystają z nich za pomocą swych kart. Zdaniem futurologów taka karta-wizytówka będzie niebawem do życia niezbędna i posiadać ją będzie dosłownie każdy.

Inne zastosowanie komputera proponuje amerykańska firma Advanced Products and Technologies. Już w tym roku zamierza sprzedawać przenośne urządzenia elektroniczne tłumaczące natychmiast mowę z jednego języka na drugi. „Voice" — bo tak nazywa się ów komputerowy tłumacz osobisty, waży niespełna 2 kg i zasilany jest z baterii. Gdy jego użytkownik powie coś, przykładowo po angielsku,

DOKOŃCZENIE NA STR. 31

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Celera (kierownik zespołu); Rada programowa: Krzysztof Chmerra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1444 Nr ind. 361682. U-23

INFORMATYKA KOMPUTERY SYSTEMY



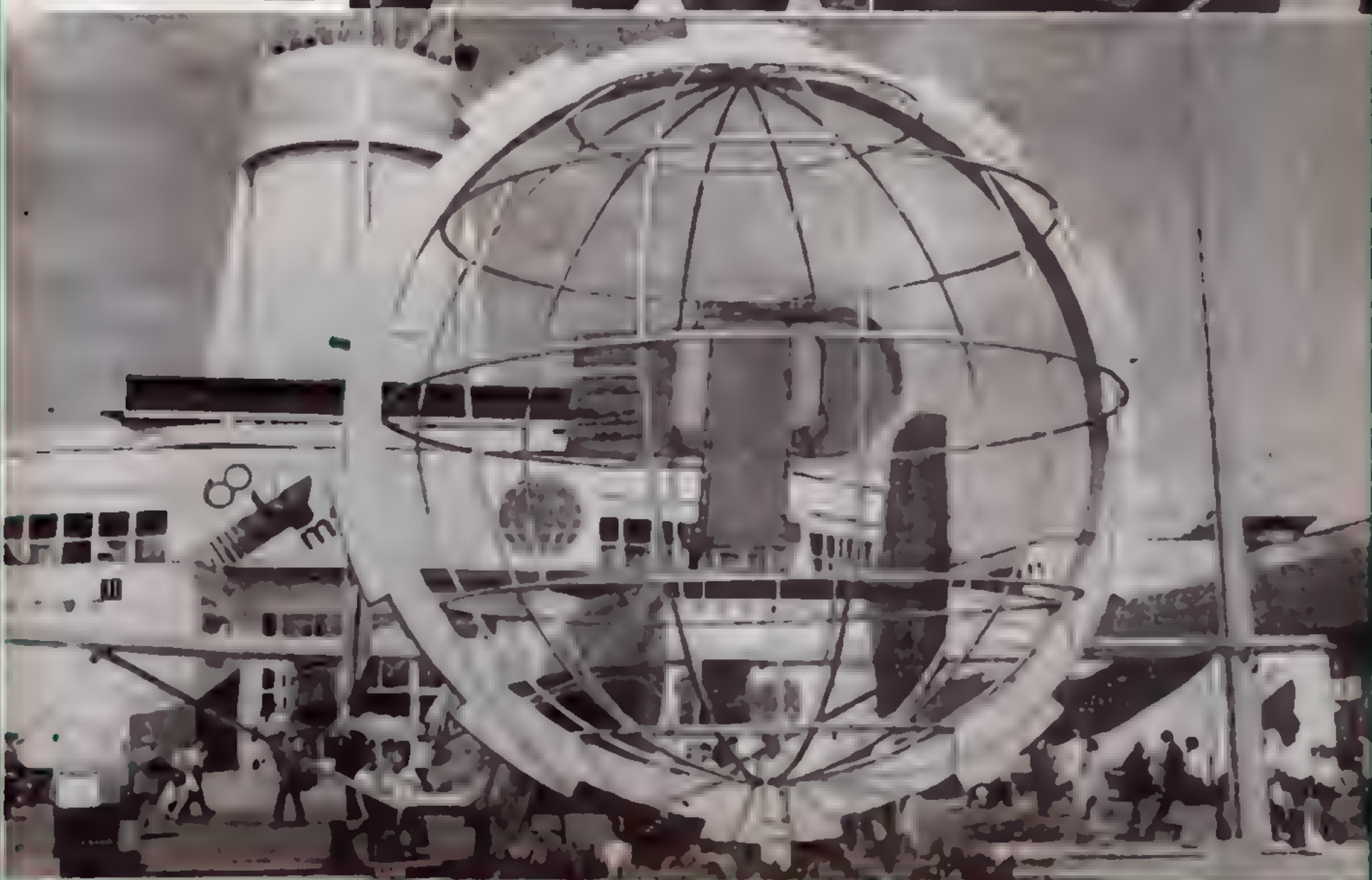
CENA 240 ZŁ

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" [illegible]

Czy komputer zastąpi nauczyciela czyli „Informatyka w szkole"







Jak jubilat z jubilatem

Rozmowa z dyrektorem Centralnej Składnicy Harcerskiej mgr. Wojciechem Szanterem.

● *Czy uważa Pan, że trzydzieści lat istnienia jest dla przedsiębiorstwa wiekiem dojrzałym?*

— Trzydzieści lat, to z pewnością już daleko posunięta dojrzałość. Dla przedsiębiorstwa handlowego każdy rok istnienia na rynku ma olbrzymie znaczenie. Umacniamy przecież swoje miejsce i mamy nadzieję — zdobywamy nowych klientów...

● *Jeśli jest czym...*

— ...i gdzie.

● *To problem każdej firmy*

— Dla nas los był łaskawy Kiedy w 1958 roku Centralna Składnica Harcerska została wydzielona ze struktury spółdzielni „Społem" i rozpoczęła działalność jako samodzielna jednostka gospodarki uspołecznionej, faktycznie — nie mieliśmy nic. Wkrótce jednak ówczesny przewodniczący Rady Przyjaciół Harcerstwa, marszałek Marian Spychalski, przyczynił się do przeznaczenia na cele handlowe CSH znanego dziś budynku przy ul. Marszałkowskiej. I tak rozwinęliśmy działalność handlową

● *Stając się z czasem monopolistą w sprzęcie i umundurowaniu harcerskim.*

— Nie tylko. Obok handlu prowadziliśmy od początku działalność produkcyjną Nasze zakłady wytwarzają artykuły politechniczne. Oczywiście początkowo zajmowały się także przeróbkami sprzętu technicznego, pochodzącego z tak zwanego demobilu i przeznaczonego dla harcerzy

● *Podobno zatrudnialiście wtedy jedynie „złote rączki" do wszystkiego.*

— Rzeczywiście podzespoły techniczne obejmowały bardzo szeroki wachlarz wyrobów, a ich „produkcja" była rzemieślnicza Robiliśmy między innymi zabawki mechaniczne sterowane kablowo, mikrosilniki, sprzęt łączności...

● *I tak staliście się potentatem...*

— Nie. Tak się po prostu rozwijaliśmy Rozszerzaliśmy działalność produkcyjną i rozbudowywaliśmy bazę handlową, dysponując wkrótce trzydziestoma sklepami na terenie całego kraju. Nie mamy też szans być monopolistą. Tylko w zaopatrzeniu ZHP w umundurowanie jesteśmy jedyni.

● *I dlatego jest ono tak drogie?*

— Nic podobnego Cena umundurowania nie zależy od nas. Sprzedajemy uwzględniając jedynie koszt własny i marżę handlową, a ta nie zależy od nas.

● *To na czym zarabiacie?*

— Na sprzęcie

● *Ale teraz jakby go więcej było w sklepach CSH.*

— To efekt importu, głównie z NRD Tę współpracę rozpoczęliśmy w latach sześćdziesiątych. Sprowadzaliśmy artykuły techniczne, a przede wszystkim sprzęt modelarski, narzędzia, mikronapędy. Mniejsze szanse mamy na rozwinięcie eksportu, a przyczyna jest prozaiczna — brak producentów

● *Importowany sprzęt jest bez wątpienia atrakcyjny, podobnie jak i jego cena...*

— Jakość kosztuje. Ale mamy i tańszy, na przykład obok narzędzi do obróbki drewna firmy Blacken-Decker wyroby z CELMY.

● *Pojawiające się niczym meteoryty...*

— Bo są niezłe i tanie Zresztą nie jest ich tak mało — sprzedajemy przecież 25—30 proc. produkcji tej firmy. Prowadzimy również w dziewięciu punktach w kraju serwis narzędzi przez nas sprzedawanych.

● *A jeśli towar nie idzie...*

— Tak bywa. Wtedy obniżamy ceny i to znacznie, jak na przykład w wypadku nart. A jest to przecież sprzęt dobry. Na przecenie nie tracimy. Przeciwnie, uwalniamy zamrożony kapitał. Przecież handel polega na obrocie kapitałem

● *Chyba nie zawsze jest to łatwe?*

— Nie przeżywamy kryzysu

● *A na początku lat osiemdziesiątych?*

— W 1982 roku mieliśmy 2,8 mld obrotu, w 1988 realne jest osiągnięcie 22 mld

● *Uwzględniając dewaluację..*

— Każdy rachunek daje pokaźny plus po naszej stronie.

● *To wynik ekonomiczny, ale również rozszerzacie swoją ofertę o sprzęt komputerowy.*

— Od 1985 roku najpierw w Warszawie, a później w innych miastach (w sumie w 8 sklepach) sprzedajemy komputery i osprzęt. Naszym celem było dostarczenie na polski rynek taniego, a tym samym dostępnego mikrokomputera

● *Timexa, bardzo krytykowanego za jego stosunkowo małe możliwości..*

— Ale to nie przeszkodziło w sprzedaniu 18 000 egzemplarzy tego modelu. 90 procent Timexa kupiło ówczesne Ministerstwo Oświaty i Wychowania i kluby Młodych Mistrzów Techniki. A poza tym Timex miał zasadnicze dwie zalety: po pierwsze był zgodny z bardzo popularnym Spectrum, a po drugie i najważniejsze — po prostu był

● *Do dzisiaj nie doczekaliśmy się natomiast mikrokomputera z prawdziwego zdarzenia, przeznaczonego dla szkół..*

— ...a bardzo chętnie byśmy nim handlowali

● *Tymczasem, zamiast wyrobów z ELWRO na półkach w CSH gości firma PROSYSTEM*

— I jak dotąd ten sprzęt sprzedaje się dobrze, choć komputery do najtańszego towaru przecież nie należą. W 1987 roku sprzęt komputerowy dał nam obrót 8 mld.

● *Przy takim obrocie pracownikom powodzi się chyba dobrze..*

— Średnie zarobki wynoszą 36 tysięcy, ale są oczywiście zróżnicowane Spośród 1800 pracowników 350 pracuje w produkcji. Poza tym naszym celem nie jest maksymalizacja zysku — chcemy być dla młodzieży atrakcyjni, dlatego szkolimy załogę, inwestujemy, zabiegamy o towary innowacyjne.

● *I tylko takich życzę w całym następnym trzydziestoleciu istnienia.*

Dziękuję za rozmowę.

Wiesław CETERA

W NUMERZE:

W szponach Atari
— Okna — str. 4
Komputer i wideo — str. 8
Transputery — str. 9
Nuty z komputera — str. 10
Zadanie konikowe — str. 19
Masterword — str. 20
Kompilatory C — str. 22
Jak oszukać edytor ZX — str. 23
Tajemnica żółtego światła — str. 26
Gra — str. 27
Informatyczny słownik angielsko-polski — str. 29
Liga Myślących — str. 31



Dyrektor Wojciech Szanter przekazał redakcji komputer SVI-738 — sprowadzany do kraju przez firmę Prosystem i sprzedawany w sklepach Centralnej Składnicy Harcerskiej.

Informatyka w medycynie

Czy ryzyko zapadnięcia na chorobę serca jest większe u osoby prowadzącej higieniczny tryb życia, ale obciążonej genetycznie skłonnością do takich właśnie dolegliwości, czy też w wypadku osoby nie przestrzegającej wprawdzie zasad racjonalnego odżywiania i aktywnego wypoczynku, wolnej jednak od genetycznych obciążeń? Próby oceny rzeczywistej skali ryzyka z uwzględnieniem wszystkich realnych przyczyn zagrożenia wydają się tak trudne, że z pomocą może przyjść tylko... komputer!

Nikogo już dziś nie zaskakuje wykorzystanie komputerów w medycynie. Powszechnie stosuje się je przy przeprowadzaniu i analizie różnego rodzaju badań i testów. Okazuje się jednak, że na tym nie kończą się ich możliwości. Ostatnio pojawiły się próby spożytkowania techniki komputerowej do opracowywania indywidualnych ocen statystycznego stanu zagrożenia zdrowia.

Najpierw trzeba odpowiedzieć na sto pytań dotyczących trybu i warunków życia, chorób występujących w rodzinie, wyników badań medycznych. Odpowiedzi wprowadzone są do pamięci komputera, który dysponuje także danymi na temat stopnia umieralności i stanu epidemiologicznego. Korzystając z tych wiadomości, maszyna stawia diagnozę w postaci wydruku komputerowego. Zawiera on ogólną ocenę stanu zdrowia — bardzo dobry, dobry, słaby — oraz określenie „wieku zdrowotnego" w porównaniu z wiekiem rzeczywistym. Wydruk przedstawia także indywidualne ryzyko zachorowania w porównaniu z przeciętnym zagrożeniem daną chorobą wśród osób tej samej płci i tego samego wieku. Może też udzielić wskazówek, co robić, aby zdrowiej i dłużej żyć.

Wiele amerykańskich przedsiębiorstw — wśród nich również te największe, jak Ford czy IBM — wykorzystuje komputerowe diagnozy do określenia stanu zdrowotności załogi, co z kolei służy jako podstawa do opracowywania planów ochrony zdrowia zatrudnionych.

Środowisko lekarskie niechętnie jednak korzysta z usług elektronicznych „kolegów". Składa się na to kilka przyczyn. Nie chodzi tu bynajmniej tylko o pracochłonność wszystkich czynności związanych z przeprowadzeniem bardzo dokładnego i szczegółowego wywiadu oraz wprowadzeniem go do pamięci komputera. Istnieją także inne, o wiele chyba istotniejsze powody. Naukowe podstawy takich ocen nie wydają się zupełnie sprawdzone. W przypadku niektórych chorób z całą pewnością potwierdzono negatywny wpływ czynników mogących je wywoływać. Jest tak chociażby z rakiem płuc czy chorobami serca, chorobami związanymi z paleniem papierosów. Jednakże informacje dotyczące ewentualnych powiązań potencjalnych czynników chorobotwórczych, ze wskaźnikiem zapadania na inne dolegliwości, są już daleko mniej precyzyjne.

Komputerowe oceny opracowywane są przy wykorzystaniu danych statystycznych, a statystyka — jak powszechnie wiadomo — lubi kłamać. Podobnie jest i w tym wypadku. Obraz statystyczny uzyskany w wyniku badania dużej grupy w niewielkim tylko



Foto: „U.S. News and World Report"

stopniu może być wykorzystany do nakreślenia wizerunku indywidualnego pacjenta, odznaczającego się jedynym, jemu tylko właściwym kodem genetycznie uwarunkowanych mocniejszych i słabszych cech. Toteż komputerowa diagnoza jest raczej tylko ogólną, przybliżoną informacją na temat stanu zdrowia i możliwych rokowań na przyszłość. Zdaniem Edwina Hutchinsa, dyrektora uniwersyteckiego centrum w Atlancie prowadzącego badania nad programem komputerowych diagnoz, ocena taka powinna być traktowana jako zbiór wskazówek bardziej co prawda wiarygodnych niż wróżby i horoskopy, ale nie pretendujących w żadnym wypadku do dokładnego i ścisłego przepowiadania przyszłości.

Częste są przecież przypadki, że ludziom nie dbającym zupełnie o swoje zdrowie, bez umiaru korzystającym z uciech życia, dopisuje zdrowie i dożywają długich lat. Zdarza się także, i to wcale nierzadko, że stosunkowo młodo umierają osoby, które — jeśli by sądzić z ich trybu życia — powinny żyć o wiele dłużej. Poza tym informacje komputera mogą być mylnie interpretowane. Pięćdziesięciolatek palący dwie paczki papierosów dziennie może odczytać ze swojego wydruku, że w jego wypadku ryzyko zachorowania na raka płuc jest dziesięć razy większe w porównaniu z przeciętnym zagrożeniem tą chorobą; elektroniczny „diagnostyk" nie powie już jednak, iż nie każdy palacz musi na pewno zachorować i że rak płuc rozwija się nie częściej niż u jednego z dziesięciu palaczy.

Należy jeszcze zwrócić uwagę na jeden niebezpieczny aspekt całego tego przedsięwzięcia. Otóż może się okazać, że człowiek zupełnie zdrowy, nie zagrożony żadną z ciężkich chorób, otrzyma diagnozę niezbyt dla niego pomyślną. Może to zostać spowodowane na przykład niedokładnym, czy niepełnym wywiadem, nieznajomością rodzinnych obciążeń genetycznych, czy wreszcie fałszywą interpretacją danych statystycznych, o co przecież nie tak trudno. Szkody powstałe w takim wypadku są trudne do przewidzenia. Pacjent nie tylko traci dobre samopoczucie, ale lęk spowodowany świadomością bliskiej choroby w znacznym stopniu wpływa na obniżen[illegible]ego kondycji zarówno psychicznej, jak [illegible]cznej, co może doprowadzić do znacz[illegible] nawet utraty odporności organizmu i [illegible] konsekwencji — choroby, a nawet śm[illegible] Prawdopodobieństwo tak skrajnego przypadku nie jest wprawdzie duże, ale zawsze należy się liczyć z takim niebezpieczeństwem. Podstawową zasadą medycyny jest przecież „po pierwsze — nie szkodzić", toteż wszystkie czynniki mogące w najmniejszym nawet stopniu przyczynić się do osłabienia kondycji pacjenta powinny zostać wyeliminowane.

Oprócz wyżej wymienionych wątpliwości, nasuwa się jeszcze jedna. Nauka rozwija się w niezwykle szybkim tempie. Ośrodki prowadzące badania nad przyczynami różnych chorób opracowują wciąż nowe programy, służące do wykorzystania przy komputerowych diagnozach. Przyszłe osiągnięcia w dziedzinie genetyki i biochemii w połączeniu z dokładnym określeniem wpływu warunków życiowych, zaowocują z pewnością dużymi sukcesami na tym polu. Jednakże w miarę wzrostu dokładności i trafności ocen, trzeba sobie zadać pytania: jak dużo tak naprawdę chcemy wiedzieć? Jaki margines zostawić dla nadziei?

Małgorzata NOWOTNY

Nauka • Technika • Nauka • Technika

Poznawaniem tajników gaszenia ognia postanowili zająć się uczeni z Uniwersvtetu w Tibilisi. W tym celu nie rozniecano jednak próbnych pożarów. Radzieccv uczeni do odpowiedzi na nurtujące ich pytanie wykorzvstali komputer. Wpisali do zaprogramowanego "pożaru" umowne cząsteczki związku o średnicach rzędu kilkudziesięciu mikronów, będące odpowiednikiem proszków gaśniczych. W wyniku operacji symulacvjnych otrzvmano odpowiedź, według której płonący w powietrzu metan gaszonv jest wskutek zachodzących reakcji chemicznych węglowodorów z cząsteczkami soli. Natomiast czvstv wodór okazał się nadzwvczaj podatnv na termicznv kontakt z substancją gaśniczą. Tak więc to w jaki sposób gaszonv jest pożar zależy przede wszystkim od rodzaju płomienia.

Amstrad

Kulki

```
10 GOTO 430
20 n=2
30 DIM x(n):DIM y(n):DIM a(n):DIM b(n)
40 GOSUB 370:s=INT(TIME/300):p=0:z=20:v=
12
50 MOVE 10,8: DRAWR 0,378:DRAWR 618,0:DR
AWR 0,-378:DRAWR -618,0
60 GOSUB 310
70 FOR i=1 TO n
80 LOCATE x(i),y(i):PRINT " ";
90 x(i)=x(i)+a(i):y(i)=y(i)+b(i)
100 LOCATE x(i),y(i):PRINT CHR$(231);
110 IF x(i)<=2 OR x(i)>=39 THEN a(i)=-a(
i)
120 IF y(i)<=2 OR y(i)>=24 THEN b(i)=-b(
i)
130 IF x(i)=z AND y(i)=v THEN 240
140 IF x(i)=18 AND y(i)<20 AND y(i)>6 TH
EN a(i)=-a(i)
150 IF x(i)=24 AND y(i)<16 AND y(i)>5 TH
EN a(i)=-a(i)
160 SOUND 1,x(i)*y(i),5,5
170 IF INKEY(71)=0 AND z>=3 THEN GOSUB 2
90:z=z-1
180 IF INKEY(63)=0 AND z<=38 THEN GOSUB
290:z=z+1
190 IF INKEY(29)=0 AND v>=3 THEN GOSUB 2
90:v=v-1
200 IF INKEY(31)=0 AND v<=23 THEN GOSUB
290:v=v+1
210 LOCATE z,v:PRINT CHR$(249);
220 NEXT
230 GOTO 60
240 FOR k=239 TO 60 STEP -50
250 SOUND 3,k,10:NEXT
260 FOR i=1 TO 100
270 INK 0,9,24:NEXT:INK 0,1
280 GOSUB 370:p=p+1:CLS:GOTO 50
290 LOCATE z,v:PRINT " ";:SOUND 3,800,1,
,,13
300 RETURN
310 t=INT(TIME/300)-s:LOCATE 6,1
320 IF t>=100 THEN 350
330 PRINT t:LOCATE 35,1:PRINT p
340 RETURN
350 LOCATE 15,1:PRINT "K O N I E C"
360 FOR i=1 TO 3000:NEXT:RUN
370 FOR i=1 TO n
380 x(i)=INT(RND*100):IF x(i)<5 OR x(i)>
35 THEN 380
390 y(i)=INT(RND*100):IF y(i)<5 OR y(i)>
20 THEN 390
400 IF x(i)=20 AND y(i)=12 THEN 380
410 a(i)=1:b(i)=-1
420 NEXT:RETURN
430 CLS:LOCATE 14,1:PRINT "K  U  L  K  I
 ";:LOCATE 20,12
440 PRINT CHR$(249)
450 LOCATE 20,5:PRINT "*"
460 LOCATE 20,19:PRINT ">"
470 LOCATE 13,12:PRINT "x"
480 LOCATE 27,12:PRINT "x"
490 MOVE 310,228:DRAWR 0,92
500 MOVE 310,204:DRAWR 0,-87
510 MOVE 297,213:DRAWR -80,0
520 MOVE 320,212:DRAWR 89,0
530 TAG:FOR x=620 TO 50 STEP -20
540 MOVE x,20:PRINT "   S T A R T
 S P A C J A      ";
550 SOUND 1,x,10
560 NEXT
570 SOUND 1,50,100
580 IF INKEY$ <> CHR$(32) THEN 580
590 MODE 1: RUN 20
```

C.S

W szponach Atari (17)

Okna

Tomasz MROWIEC,
Ludwik PIELA

Podczas tworzenia programów musimy pamiętać, że korzystać z nich będą zwykle ludzie mający niewielkie doświadczenie informatyczne. Wymaga to zastosowania prostych i przystępnych metod komunikowania się z programem. Obsługa programu powinna być tak prosta i wygodna, aby każdy mógł się nim posługiwać po krótkim treningu. W tym celu powinien on spełniać następujące warunki:

— wyświetlać na ekranie wykaz funkcji, które może zrealizować, w postaci menu,

— komunikować się z użytkownikiem na zasadzie dialogu, sprawdzać poprawność wprowadzanych danych i poleceń,

— wyświetlać na ekranie wszystkie informacje potrzebne użytkownikom,

— podpowiadać, na życzenie, jakie czynności należy wykonać, aby zrealizować określone zadanie.

Interesującym i efektownym środkiem prezentowania na ekranie dodatkowych informacji, różnorodnych objaśnień i podpowiedzi jest technika okien. Przez pojęcie okna rozumiemy wyróżniony fragment ekranu przeznaczony do wyświetlania określonych informacji. Okna mogą być tworzone w dowolnych miejscach ekranu, mieć różne rozmiary, być umieszczone obok siebie, albo zasłaniać się częściowo lub całkowicie. W każdej chwili mogą być usunięte z ekranu, na którym pojawia się poprzednio jego zawartość.

Jeśli ktoś z czytelników nie widział jeszcze używania okien, stosowanych powszechnie w komputerach profesjonalnych, to nic straconego. Prezentowane programy demonstracyjne nie tylko pokażą, jak to wygląda, lecz także umożliwią tworzenie okien we własnych programach.

Zdajemy sobie sprawę, że najlepszy opis nie zastąpi najmarniejszego pokazu. Musimy jednak krótko opisać, jak wygląda używanie okien, zanim omówimy, na czym polega ich realizacja.

Z punktu widzenia użytkownika wygląda to następująco. W wybranym miejscu ekranu wyświetlany jest wyróżniony komunikat, niekiedy obramowany stosowną obwódką. Zasłania on informacje tekstowe lub graficzne, istniejące w tym obszarze ekranu. Na razie nie ma w tym nic interesującego. Podobny efekt można uzyskać środkami dostępnymi w języku BASIC. Najciekawsze pojawia się dopiero w chwili usuwania okna. W jego miejsce pojawia się poprzednia zawartość ekranu.

W celu uzyskania takiego efektu musimy działać bezpośrednio na obszarze pamięci ekranu. Dlatego dokładnie musimy określić położenie okna nie we współrzędnych ekranu, lecz w terminach przesunięcia (liczby bajtów) początku okna (lewego górnego rogu) względem początku obszaru pamięci ekranu. Musimy również podać liczbę kolumn i wierszy zajmowanych przez okno.

Procedura tworzenia (otwierania) okien musi realizować następujące czynności:

— przechować, w zmiennej tekstowej, zawartość komórek pamięci obszaru ekranu, w którym będzie umieszczone okno,

— przepisać zawartość innej zmiennej tekstowej, przechowującej treść komunikatu umieszczonego w oknie, do wskazanego obszaru pamięci ekranu.

Dodatkowo musi ona wykonać jeszcze jedną, niewidoczną funkcję. Należy przekodować znaki, przechowywane w zmiennej tekstowej, z kodu ATASCII na kod wewnętrzny. Tylko w takiej postaci dadzą one czytelny tekst na ekranie.

W celu utworzenia okna musimy określić następujące parametry:

— położenie okna (lewy górny róg),

— wymiary (poziomy i pionowy),

— zawartość.

Następnie należy przechować zawartość obszaru pamięci ekranu, która będzie zakrywana przez okno. Kolejnym krokiem jest umieszczenie w tym obszarze pamięci, po uprzednim przekodowaniu, zawartości okna, którą przygotowaliśmy w zmiennej tekstowej.

Otwieranie kolejnego okna odbywa się w identyczny sposób. Usuwanie okna z ekranu (zamykanie) powoduje tylko odtworzenie poprzedniej zawartości używanego obszaru ekranu. Może to być początkowa zawartość ekranu, zawartość wcześniej utworzonego okna lub obie.

W przypadku tworzenia na ekranie wielu wzajemnie zasłaniających się okien należy pamiętać o właściwej kolejności ich zamykania, odwrotnej do otwierania.

Opisane czynności mogą być wykonane za pomocą procedury napisanej w języku BASIC, lecz czas jej realizacji może zniechęcić nas do używania okien. Dlatego najlepiej napisać ją w języku wewnętrznym komputera.

Program nr 1 zawiera dwie procedury w języku maszynowym. Pierwsza z nich, umieszczona w zmiennej tekstowej OTW$, służy do otwierania okna na ekranie. Realizuje ona następujące funkcje:

— wyznacza obszar ekranu przeznaczony na okno,

— przechowuje jego zawartość w wyznaczonym buforze,

— przekodowuje symbole umieszczone w oknie z kodu ATASCII na kod ekranu i umieszcza je w obszarze pamięci ekranu przydzielonym dla okna.

Sposób wywołania procedury tworzenia okna jest następujący:

**X = USR(ADR(OTW$),A,S,W,
ADR(BUF$),ADR(O$))**

gdzie: **OTW$** — zmienna tekstowa zawierająca procedurę w języku maszynowym,

A — adres lewego górnego rogu okna; adres bezwzględny komórki pamięci określającej ten róg,

S — szerokość okna w bajtach,

W — wysokość okna w liniach,

BUF$ — zmienna tekstowa służąca do przechowania zawartości obszaru ekranu zakrywanego oknem,

O$ — zmienna tekstowa zawierająca informacje wyświetlane w oknie.

Wielkości zmiennych tekstowych BUF$ i O$ muszą być jednakowe i nie mniejsze od liczby bajtów umieszczanych w obszarze okna. Nie ma żadnych ograniczeń na wielkość okna. Może ono zajmować nawet cały ekran

Druga procedura w języku maszynowym, umieszczana w zmiennej tekstowej ZAM$, służy do zamykania okna. Realizuje ona następujące funkcje:

— wyznacza obszar ekranu przeznaczony na okno,

— odtwarza poprzednią zawartość tego obszaru na podstawie danych przechowywanych w zmiennej tekstowej.

Sposób wywołania tej procedury jest następujący:

X=USR(ADR(ZAM$),A,S,W,ADR(BUF$))

gdzie: **ZAM$** — zmienna tekstowa zawierająca procedurę w języku maszynowym,

A — adres bezwzględny lewego górnego rogu okna,

S — szerokość okna w bajtach,

W — wysokość okna w liniach,

BUF$ — zmienna tekstowa, w której przechowywana jest zawartość obszaru ekranu zakrywanego oknem.

W omawianym programie przedstawiony jest sposób przygotowania zawartości okna oraz dwa sposoby wywołania go na ekran. Za pierwszym razem za pomocą klawisza (HELP), za drugim poprzez wybranie odpowiedniej litery. Zamykanie obu okien następuje po naciśnięciu (HELP). Proszę zauważyć, że w celu zachowania pierwotnej zawartości ekranu w postaci niezmienionej, należy przestrzegać właściwej kolejności zamykania okien oraz używania tych samych zmiennych przechowujących zawartość obszaru ekranu zakrywanego oknem.

Charakterystyczną cechą prezentowanych procedur jest używanie dwóch odrębnych obszarów. Jednego do przechowywania zawartości wyświetlanego okna i drugiego do przechowywania zawartości obszaru ekranu zakrywanego oknem

Proszę pamiętać, że obie procedury umożliwiają otwieranie i zamykanie okien tylko w trybie tekstowym (najlepiej GRAPHICS 0).

Przedstawione powyżej rozwiązanie nie jest jedynym sposobem tworzenia okien na ekranie. Poniżej prezentujemy inne. Umożliwia ono, podobnie jak poprzednie, zdefiniowanie okna o zadanej wielkości i umieszczenie go w dowolnym miejscu ekranu. Po przydzieleniu tekstu do zmiennej, to znaczy określeniu zawartości okna, następuje wymiana zawartości ciągu z wybranym obszarem pamięci ekranu. W tym celu należy wywołać dwa podprogramy w języku maszynowym. Pierwszy z nich dokonuje konwersji znaków z kodu ATASCII na kod wewnętrzny, a drugi dokonuje wymiany

Powyższy sposób wykorzystany został w **programie nr 2.** W pierwszej kolejności umieszcza on procedurę konwersji w zmiennej tekstowej C$. Kody znaków przed i po konwersji przechowywane są w tym samym ciągu. Jeśli będzie to zmienna o nazwie D$, to wywołanie procedury kontrowersji ma następującą postać:

X=USR=(ADR(C$),ADR(D$))

Druga procedura, która dokonuje wymiany zawartości wybranego obszaru ekranu.

Uwaga: Litery z lewej strony numerów linii nie są częścią programu, lecz sumą kontrolną ułatwiającą wprowadzenie programu za pomocą Edytora BASIC'u (IKS nr 4 z 1987 roku). Znaki, które są podkreślone, należy wprowadzić w negatywie. Znaki wypisane drukiem rozszerzonym należy wprowadzić z wciśniętym klawiszem CONTROL.

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Program nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
XN 10 DATA 104,104,133,204,104,13
   3,203,104,104,133,205,104,104,
   170,104,133,207,104,133,206,10
   4,133,209,104,133,208
FX 20 DATA 160,0,177,203,145,206,
   177,208,72,41,128,133,210,104,
   41,127,201,32,176,4,105,64,144
   ,7,201,96,176,3,56,233
RO 30 DATA 32,5,210,145,203,200,1
   96,205,144,218,24,165,203,105,
   40,133,203,165,204,105,0,133,2
   04,24,165,206
AA 40 DATA 101,205,133,206,165,20
   7,105,0,133,207,24,165,208,101
   ,205,133,208,165,209,105,0,133
   ,209,202,16,174,96,-1
NB 50 DATA 104,104,133,204,104,13
   3,203,104,104,133,205,104,104,
   170,104,133,207,104,133,206,16
   0,0
AG 60 DATA 177,206,145,203,200,19
   6,205,144,247,24,165,203,105,4
   0,133,203,165,204,105,0,133,20
   4,24,165,206,101
FN 70 DATA 205,133,206,165,207,10
   5,0,133,207,202,16,216,96,-1
NA 100 DIM OTW$(109),ZAM$(61),E$(
   1)
AS 110 FOR I=1 TO 200:READ A:ON (
   A=-1) GOTO 120:OTW$(I,I)=CHR$(
   A):NEXT I
VC 120 FOR I=1 TO 200:READ A:ON (
   A=-1) GOTO 130:ZAM$(I,I)=CHR$(
   A):NEXT I
OC 130 ADRE=PEEK(88)+256*PEEK(89)
SS 140 O1=ADRE+5+5*40
IK 150 O2=ADRE+10+10*40
BU 160 DIM O1$(200),O2$(200),BUF1
   $(200),BUF2$(200)
PF 200 O1$(1,7)="QRRRRRR"
HZ 204 O1$(8,14)="RRRRRRR"
MV 208 O1$(15,20)="RRRRRE"
JI 210 O1$(21,40)="|   WYKAZ KOME
   ND   |"
QI 220 O1$(41,47)="ARRRRRR"
DO 224 O1$(48,54)="RRRRRRR"
PJ 228 O1$(55,60)="RRRRRD"
MI 230 O1$(61,80)="| A=Dodaj  C=Z
   mien |"
ZG 240 O1$(81,100)="| D=Kasuj  E=
   Koniec|"
JA 250 O1$(101,120)="| P=Drukuj H
   =Pomoc |"

XF 255 O1$(121,127)="ZRRRRRR"
TJ 257 O1$(128,134)="RRRRRRR"
RH 258 O1$(135,140)="RRRRRC"
PX 260 O2$(1,7)="QRRRRRR"
IR 264 O2$(8,14)="RRRRRRR"
NN 268 O2$(15,20)="RRRRRE"
EG 270 O2$(21,40)="|      DODAJ
        |"
RA 280 O2$(41,47)="ARRRRRR"
EG 284 O2$(48,54)="RRRRRRR"
QB 288 O2$(55,60)="RRRRRD"
LX 290 O2$(61,80)="|Tekst wyjasni
   ajacy|"
RK 300 O2$(81,100)="| dzialanie k
   omendy|"
GY 310 O2$(101,120)="|      DODAJ
        |"
EK 320 O2$(121,127)="ARRRRRR"
TB 324 O2$(128,134)="RRRRRRR"
SG 328 O2$(135,140)="RRRRRD"
YP 330 O2$(141,147)="ZRRRRRR"
UZ 334 O2$(148,154)="RRRRRRR"
TG 338 O2$(155,160)="RRRRRC"
SK 1000 POSITION 0,0:IF PEEK(732)
   <>17 THEN 1000
NQ 1005 POKE 732,0
PD 1010 X=USR(ADR(OTW$),O1,20,6,A
   DR(BUF1$),ADR(O1$))
TY 1020 INPUT E$:IF E$<>"A" THEN
   2000
UN 1030 X=USR(ADR(OTW$),O2,20,7,A
   DR(BUF2$),ADR(O2$))
XG 1040 IF PEEK(732)<>17 THEN 104
   0
HD 1050 POKE 732,0:X=USR(ADR(ZAM$
   ),O2,20,7,ADR(BUF2$))
YT 2000 X=USR(ADR(ZAM$),O1,20,6,A
   DR(BUF1$))
MS 2010 GOTO 1000
```

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Program nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
MX 100 GRAPHICS 0:POKE 752,1:JA=1
   20
CK 110 DIM A$(961),D$(113),B$(241
   ),W$(116),C$(61)
FT 120 ? :? "WCZYTANIE W$":FOR I=
   1 TO 116:READ IT:W$(I,I)=CHR$(
   IT):NEXT I
LU 130 ? :? "WCZYTANIE C$":FOR I=
   1 TO 61:READ IT:C$(I,I)=CHR$(I
   T):NEXT I
GZ 140 A$=".":A$(960)=A$:A$(2)=A$
   :A$(961)=CHR$(0)
PE 150 A$(131,154)=" Program   de
   monstruje "
HC 160 A$(171,194)="sposob tworze
   nia okien "
CS 170 D$="****************    OKN
   A    **moga pokazac** co  chce
   sz **     "
UE 180 D$(63,113)="i        **   G
   DZIE    ** chcesz    *********
   ******"
IA 190 B$="_":B$(240)=B$:B$(2)=B$
   :B$(241)=CHR$(0)
XV 200 B$(35,55)=" Podprogramy te
    mozna  "
JD 210 B$(92,119)="uzywac w tryba
   ch tekstowych"
UU 220 B$(155,176)="Okna moga byc
    dowolne "
HK 230 B$(184,208)="w kazdym miej
   scu ekranu"
AO 240 GRAPHICS 0:POKE 752,1:?
WF 250 X=USR(ADR(C$),ADR(A$))
YY 260 X=USR(ADR(C$),ADR(D$))
XG 270 X=USR(ADR(C$),ADR(B$))
DC 280 SOUND 0,0,2,2:GOSUB 370:GO
   SUB 340
FI 290 SOUND 0,2,2,2:GOSUB 340:GO
   SUB 370
FW 300 IF LG=0 THEN LIST 470,670:
   LG=1
PE 310 LICZ=LICZ+1:IF LICZ>=4 THE
   N GOSUB 380:GOSUB 390:GOSUB 38
   0:JA=10
DL 320 IF LICZ>=7 THEN GRAPHICS 0
   :? :? "KONIEC PRACY":END
PI 330 GOTO 280
ZE 340 IF JA=10 THEN L=41
BR 350 IF JA=120 THEN L=39
XE 360 X=USR(ADR(W$),ADR(D$),412,
   8,14,L):GOSUB 390:RETURN
CC 370 X=USR(ADR(W$),ADR(A$),0,24
   ,40,40):GOSUB 390:RETURN

FG 380 X=USR(ADR(W$),ADR(B$),285,
   8,30,40):GOSUB 390:RETURN
DP 390 POKE 540,JA:SOUND 0,0,0,0
WY 400 IF PEEK(540) THEN 400
```

```
ZC 410 RETURN
QG 420 DATA 104,104,133,206,104,1
   33,205
SG 430 DATA 104,141,1,6,104,141,0
SO 440 DATA 6,104,133,208,104,133
   ,207
LH 450 DATA 104,104,133,209,104,1
   41,3
NG 460 DATA 6,104,141,2,6,165,88
AO 470 DATA 133,203,165,89,133,20
   4,24
RF 480 DATA 165,203,109,0,6,133,2
   03
TZ 490 DATA 165,204,109,1,6,133,2
   04
AI 500 DATA 160,0,177,203,170,177
   ,205
GB 510 DATA 145,203,138,145,205,2
   00,196
LI 520 DATA 209,208,241,169,0,198
   ,207
KF 530 DATA 197,207,208,6,197,208
   ,240
SB 540 DATA 30,198,208,24,152,101
   ,205
CL 550 DATA 133,205,144,3,230,206
   ,24
SU 560 DATA 165,203,109,2,6,133,2
   03
VO 570 DATA 165,204,109,3,6,133,2
   04
MZ 580 DATA 24,144,197,96
MF 590 DATA 104,104,133,204,104,1
   33,203
KA 600 DATA 160,0,177,203,170,201
   ,0
MD 610 DATA 240,44,41,96,201,0,20
   8
ID 620 DATA 6,138,9,64,24,144,22
AQ 630 DATA 201,96,240,20,138,41,
   64
KJ 640 DATA 201,0,208,6,138,73,32
MA 650 DATA 24,144,5,138,9,32,73
WX 660 DATA 64,145,203,200,208,21
   0,230
HN 670 DATA 204,24,144,203,96
```

```
FK 1 REM ***************
FJ 2 REM *             *
YO 3 REM * Program nr 3 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
SH 100 GRAPHICS 0:POKE 752,1:POKE
    82,4
AD 110 ? :? "        OKNA TEKSTOW
   E "
RL 120 ? :? "     W TRYBACH GRAFI
   CZNYCH"
LJ 130 DIM W$(116),KOM$(11),ROB$(
   11*16),E$(263)
BN 140 ? :? "WCZYTANIE W$"
ME 150 FOR I=1 TO 116:READ A:W$(I
   ,I)=CHR$(A):NEXT I
ND 160 ? :? "WCZYTANIE E$"
VL 170 FOR I=1 TO 263:READ A:E$(I
   ,I)=CHR$(A):NEXT I
WR 180 G=8:W=40:GRAPHICS G+16:GOS
   UB 260
WM 190 G=7:W=40:GRAPHICS G+16:GOS
   UB 260
VO 200 G=6:W=40:GRAPHICS G+16:GOS
   UB 260
UN 210 G=5:W=20:GRAPHICS G+16:GOS
   UB 260
TX 220 G=4:W=10:GRAPHICS G+16:GOS
   UB 260
TS 230 G=3:W=10:GRAPHICS G+16:GOS
   UB 260
ZW 240 GRAPHICS 0:POKE 82,2:? :?
   "KONIEC PROGRAMU":END
NL 250 REM PODPROGRAM WYSWIETLANI
   A KOMUNIKATOW
JA 260 SETCOLOR 0,9,10:SETCOLOR 1
   ,3,8:SETCOLOR 2,9,0:SETCOLOR 4
   ,1,2
II 270 REM ZEROWANIE CIAGU
UB 280 ROB$=CHR$(0):ROB$(11*16)=R
   OB$:ROB$(2)=ROB$:KOM$=ROB$
OB 290 FOR L=1 TO 3
```

```
BE 310 KOM$="GR.? Typ  ":KOM$(4,4
   )=CHR$(G+48):KOM$(10,10)=CHR$(
   L+48):KOM$(11,11)=CHR$(0)
SS 320 REM KONWERSJA NA FORMAT GR
   AFICZNY
VH 330 X=USR(ADR(E$),ADR(ROB$),AD
   R(KOM$),LEN(KOM$)-1,L)
SX 340 REM UMIESZCZENIE NA EKRANI
   E
FT 350 X=USR(ADR(W$),ADR(ROB$),W*
   8*(L-1),8,20,W)
RA 360 REM
NT 370 SOUND 0,L*75,14,5:X=25*L:P
   OKE 20,0
DA 380 A=PEEK(20):POKE 708,45:IF
   A<X THEN POKE 708,40:GOTO 380
AA 390 POKE 708,40:SOUND 0,0,0,0:
   NEXT L:RETURN

TT 400 REM DANE DLA W$
OG 410 DATA 104,104,133,206,104,1
   33,205,104,141,1,6,104,141,0,6
NZ 430 DATA 104,133,208,104,133,2
   07,104,104,133,209,104,141,3,6
   ,104,141,2,6
AT 450 DATA 165,88,133,203,165,89
   ,133,204,24,165,203,109,0,6,13
   3
JA 470 DATA 203,165,204,109,1,6,1
   33,204,160,0,177,203,170,177,2
   05,145
OC 490 DATA 203,138,145,205,200,1
   96,209,208,241,169,0,198,207,1
   97,207,208
MU 510 DATA 6,197,208,240,30,198,
   208,24,152,101,205,133,205,144
   ,3,230
XF 530 DATA 206,24,165,203,109,2,
   6,133,203,165,204,109,3,6,133,
   204,24,144,197,96
HU 560 REM DANE DLA E$
BK 570 DATA 104,104,133,204,104,1
   33,203,104,133,212,104,133,211
   ,104,104,133
MK 590 DATA 216,10,133,217,104,10
   4,41,3,24,106,106,106,41,192,1
   33,207
XX 610 DATA 133,208,169,0,133,219
   ,169,4,133,220,169,0,160,145,2
   19,200
GL 630 DATA 192,16,208,249,169,25
   5,133,215,160,0,177,211,201,12
   8,144,4
TV 650 DATA 132,215,73,128,201,96
   ,144,4,160,3,176,19,201,64,144
   ,4
RS 670 DATA 160,1,176,11,201,32,1
   44,5,176,5,24,144,205,160,2,41
JA 690 DATA 31,10,10,10,133,213,2
   4,152,109,244,2,133,214,160,0,
   132
RU 710 DATA 209,132,210,162,0,134
   ,205,164,210,177,213,133,206,1
   64,209,165
BJ 730 DATA 206,48,6,177,219,5,20
   7,145,219,24,102,207,24,102,20
   7,6
IZ 750 DATA 206,232,224,4,208,231
   ,165,215,240,5,56,241,219,145,
   219,165
HE 770 DATA 208,133,207,230,209,2
   30,205,165,205,201,2,208,208,2
   30,210,164
UG 790 DATA 210,192,8,208,190,240
   ,3,24,144,160,165,203,72,165,2
   04,72
EN 810 DATA 160,0,177,219,145,203
   ,200,192,2,208,247,177,219,170
   ,200,177

XA 830 DATA 219,72,200,132,218,16
   4,217,138,145,203,104,200,145,
   203,24,165
LA 850 DATA 203,101,217,133,203,1
   44,2,230,204,164,218,192,16,20
   8,220,104
JB 870 DATA 133,204,24,104,105,2,
   133,203,144,2,230,204,230,211,
   208,2
KP 890 DATA 230,212,198,216,208,1
   77,96
```

```
FK 1 REM ***************
FJ 2 REM *             *
ZI 3 REM * Program nr 4 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
SH 100 GRAPHICS 0:POKE 752,1:POKE
    82,4
AD 110 ? :? "        OKNA TEKSTOW
   E "
RL 120 ? :? "     W TRYBACH GRAFI
   CZNYCH"
WB 130 DIM W$(116),KOM$(11),ROB$(
   22*16),E$(263),A$(11*16)
BN 140 ? :? "WCZYTANIE W$"
ME 150 FOR I=1 TO 116:READ A:W$(I
   ,I)=CHR$(A):NEXT I
ND 160 ? :? "WCZYTANIE E$"
VL 170 FOR I=1 TO 263:READ A:E$(I
   ,I)=CHR$(A):NEXT I
EB 175 G=15:W=40:GRAPHICS G+16:GO
   SUB 260
CY 176 G=14:W=20:GRAPHICS G+16:GO
   SUB 260
WR 180 G=8:W=40:GRAPHICS G+16:GOS
   UB 260
WM 190 G=7:W=40:GRAPHICS G+16:GOS
   UB 260
US 200 G=6:W=20:GRAPHICS G+16:GOS
   UB 260
UN 210 G=5:W=20:GRAPHICS G+16:GOS
   UB 260
TX 220 G=4:W=10:GRAPHICS G+16:GOS
   UB 260
TS 230 G=3:W=10:GRAPHICS G+16:GOS
   UB 260
TW 240 GRAPHICS 0:POKE 82,2:? :?
   "KONIEC PROGRAMU":END
NL 250 REM PODPROGRAM WYSWIETLANI
   A KOMUNIKATOW
GK 260 SETCOLOR 0,9,10:SETCOLOR 1
   ,3,6:SETCOLOR 2,9,0:SETCOLOR 4
   ,1,2
II 270 REM ZEROWANIE CIAGU
UZ 275 COLOR 1:PLOT W*3,0:FOR I=1
    TO 40:X=RND(0)*W*4:Y=RND(0)*W
   *2:DRAWTO INT(X),INT(Y):NEXT I
VW 280 ROB$=CHR$(0):ROB$(22*16)=R
   OB$:ROB$(2)=ROB$:KOM$=ROB$
OB 290 FOR L=1 TO 3
IZ 300 REM TWORZENIE OKNA
LX 305 A$=CHR$(0):A$(11*16)=A$:A$
   (2)=A$
MP 310 KOM$="Gr. ":KOM$(6-LEN(STR
   $(G)))=STR$(G)
ME 315 X=USR(ADR(E$),ADR(A$),ADR(
   KOM$),LEN(KOM$),L)
KM 320 ROB$=A$
ZG 325 WSK=(LEN(KOM$)-0)*16+1

MN 330 KOM$="Typ ":KOM$(5)=STR$(L
   )
MD 335 A$=CHR$(0):A$(11*16)=A$:A$
   (2)=A$
LV 340 X=USR(ADR(E$),ADR(A$),ADR(
   KOM$),LEN(KOM$),L)
KX 345 ROB$(WSK)=A$
XC 360 REM UMIESZCZANIE OKNA
OD 370 FOR I=1 TO 5
UC 380 X=USR(ADR(W$),ADR(ROB$),W*
   8+W/2-5,16,LEN(KOM$)*2,W)
XM 385 FOR J=15 TO 0 STEP -15/200
AG 386 SOUND 0,L*75,14,J
HO 387 NEXT J
CE 390 NEXT I:NEXT L:RETURN
BV 400 REM PODPROGRAM UMIESZCZANI
   A OKNA
OG 410 DATA 104,104,133,206,104,1
   33,205,104,141,1,6,104,141,0,6
NZ 430 DATA 104,133,208,104,133,2
   07,104,104,133,209,104,141,3,6
   ,104,141,2,6
AT 450 DATA 165,88,133,203,165,89
   ,133,204,24,165,203,109,0,6,13
   3
JA 470 DATA 203,165,204,109,1,6,1
   33,204,160,0,177,203,170,177,2
   05,145
OC 490 DATA 203,138,145,205,200,1
   96,209,208,241,169,0,198,207,1
   97,207,208
MU 510 DATA 6,197,208,240,30,198,
```

```
    208,24,152,101,205,133,205,144
    ,3,230
XF 530 DATA 206,24,165,203,109,2,
    6,133,203,165,204,109,3,6,133,
    204,24,144,197,96
CR 560 REM PODPROGRAM KONWERSJI Z
    NAKOW NA FORMAT CZTEROKOLOROWY
BK 570 DATA 104,104,133,204,104,1
    33,203,104,133,212,104,133,211
    ,104,104,133
MK 590 DATA 216,10,133,217,104,10
    4,41,3,24,106,106,106,41,192,1
    33,207
XX 610 DATA 133,208,169,0,133,219
    ,169,4,133,220,169,0,168,145,2
    19,200
GL 630 DATA 192,16,208,249,169,25
    5,133,215,160,0,177,211,201,12
    8,144,4
TV 650 DATA 132,215,73,128,201,96
    ,144,4,160,3,176,19,201,64,144
    ,4
RS 670 DATA 160,1,176,11,201,32,1
    44,5,176,5,24,144,205,160,2,41
JA 690 DATA 31,10,10,10,133,213,2
    4,152,109,244,2,133,214,160,0,
    132
RU 710 DATA 209,132,210,162,0,134
    ,205,164,210,177,213,133,206,1
    64,209,165
BJ 730 DATA 206,48,6,177,219,5,20
    7,145,219,24,102,207,24,102,20
    7,6
IZ 750 DATA 206,232,224,4,208,231
    ,165,215,240,5,56,241,219,145,
    219,165
HE 770 DATA 208,133,207,230,209,2
    30,205,165,205,201,2,208,208,2
    30,210,164
UG 790 DATA 210,192,8,208,190,240
    ,3,24,144,160,165,203,72,165,2
    04,72
EN 810 DATA 160,0,177,219,145,203
    ,200,192,2,208,247,177,219,170
    ,200,177
XA 830 DATA 219,72,200,132,218,16
    4,217,138,145,203,104,200,145,
    203,24,165
LA 850 DATA 203,101,217,133,203,1
    44,2,230,204,164,218,192,16,20
    8,220,104
JB 870 DATA 133,204,24,104,105,2,
    133,203,144,2,230,204,230,211,
    208,2
KP 890 DATA 230,212,198,216,208,1
    77,96
```

przechowywana jest w zmiennej W$. Wywołujemy ją następująco:

X=USR(ADR(W$), ADR(D$), O,W,S,L)

gdzie: **O** — wielkość przesunięcia lewego górnego rogu okna względem lewego górnego rogu ekranu (w bajtach),

W — liczba linii wyświetlanego tekstu (wysokość okna),

S — szerokość okna w znakach,

L — przesunięcie między początkiem kazdej kolejnej linii, dla okna prostokątnego, w trybie GR.0,L powinno być równe 40, używając L=39 lub L=41 otrzymamy okno pochylone w jedną lub drugą stronę

Podczas używania tych procedur należy pamiętać, aby w instrukcji USR dostarczyć odpowiednią liczbę parametrów. Nieprzestrzeganie tego może zablokować komputer.

Program nr 2 demonstruje kilka możliwych sposobów używania okien. Proszę zwrócić uwagę na wykorzystanie tej techniki do szybkiej wymiany zawartości całego ekranu W tym celu zmienna tekstowa musi mieć wielkość 960 bajtów

Powyższe procedury w prezentowanej postaci mogą być używane tylko w trybie tekstowym. W związku z tym interesujące może być szybkie wyświetlenie tekstu, najlepiej kolorowego, w dowolnym z trybów graficznych. Wtedy uwolnimy się od konieczności używania okna tekstowego, któremu na stałe przydzielone są cztery dolne linie ekranu

Idea tworzenia okien w trybach graficznych jest identyczna jak w tekstowych Różnica w realizacji polega na konwersji znaków. W trybie tekstowym należało zamienić kody ATASCII na kody ekranu. W trybie graficznym należy zamienić kody znaków na ich reprezentacje bitowe. Wykonuje to procedura przechowywana w zmiennej E$ (program nr 3).

Wywołujemy ją następująco

X = USR(ADR(E$), ADR(ROB$),ADR (KOM$),LEN(KOM$)—1,K)

gdzie **E$** — zmienna tekstowa zawierająca procedurę konwersji,

ROB$ — pomocnicza zmienna tekstowa służąca do przechowywania reprezentacji bitowej znaków tekstu; musi być 16 razy dłuższa od tekstu,

KOM$ — zmienna tekstowa przechowująca wyświetlany tekst,

K — kod koloru wybrany dla okna.

Wyświetlenie okna na ekranie wykonujemy tą samą procedurą co w trybie tekstowym. Występują jednak pewne różnice w znaczeniu parametrów. Wynikają one z faktu, że ciąg do wyświetlania zawiera reprezentacje bitowe znaków. A każdy punkt elementarny definiowany jest przez dwa bity Na jedną linię wyświetlanych symboli przypada 8 linii obrazu, których grubość zależy od używanego trybu graficznego.

Sposób wywołania tej procedury jest następujący

X=USR(ADR(W$),ADR(ROB$),O,W,S,L)

gdzie: **O** — wielkość przesunięcia lewego górnego rogu okna względem lewego górnego rogu ekranu (w bajtach),

W — wysokość okna (w liniach obrazu),

S — szerokość linii w bajtach (ilość znaków w linii razy dwa),

L — przesunięcie między początkiem każdej kolejnej linii, jego wielkość zależy od używanego trybu graficznego.

Tworzenie okien w trybach graficznych nie jest, niestety, takie proste jak w tekstowych. Pewne komplikacje spowodowane są koniecznością konwersji ciągu znaków na jego odwzorowanie bitowe. Z tego powodu każdą linię tekstu umieszczonego w oknie musimy poddawać konwersji oddzielnie Następnie ciąg wynikowy umieszczamy w zmiennej tekstowej, której zawartość zostanie wyświetlona w oknie. Wysokość okna musi być podana w liniach obrazu (8 linii na jedną linię tekstu). Szerokość linii musi być dwa razy większa od liczby wyświetlanych znaków, ponieważ kazdy punkt elementarny opisywany jest za pomocą dwóch bitów.

W celu zapoznania się ze sposobami tworzenia okien należy dokładnie przeanalizować program nr 3, który wyświetla na ekranie okna zawierające jedną linię tekstu w rozmaitych trybach graficznych. Z kolei program nr 4 demonstruje sposób tworzenia okien zawierających dwie linie tekstu (także w trybach 14 i 15). W podobny sposób tworzy się okna zawierające trzy lub więcej linii tekstu

Musimy przyznać, że nie udało się nam uzyskać spodziewanych efektów tworzenia okien w trybach graficznych. Jednak ze względu na nieprzekraczalny termin złożenia w redakcji kolejnego odcinka cyklu, wyniki naszych dalszych prac zamieścimy w następnym numerze „IKS-a".

Podczas przygotowywania tego odcinka korzystaliśmy z artykułów zamieszczonych w czasopiśmie ANTIC.

Tomasz MROWIEC
Ludwik PIELA

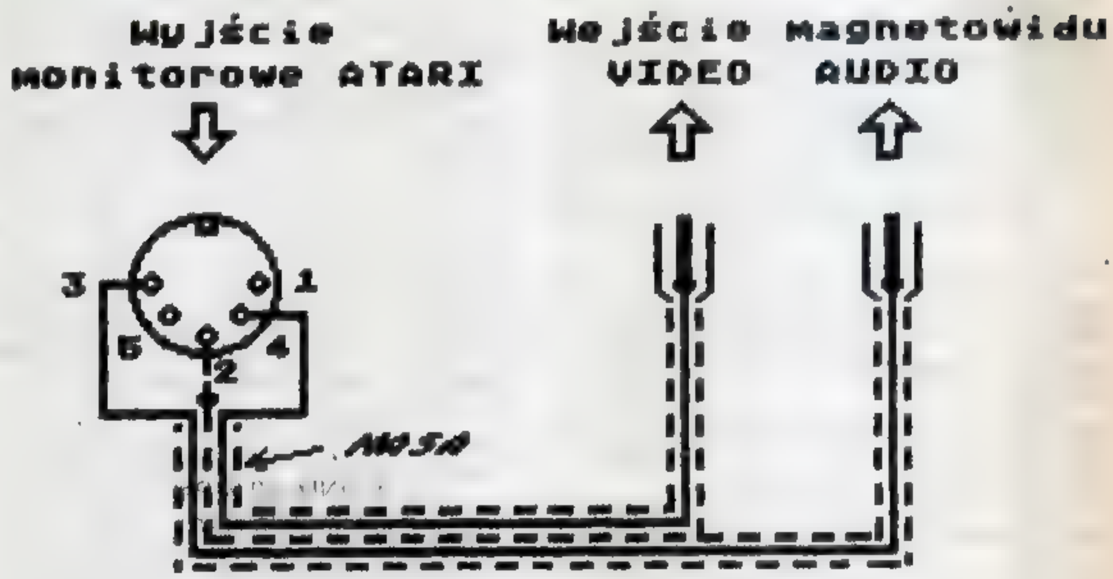
Atari

J. Janiec

```
PG 500 REM ***********************
YA 502 REM *                     *
HT 504 REM *       RACHUNKI      *
DI 505 REM *     * Atari *       *
YP 507 REM *                     *
QH 509 REM ***********************
BU 510 DIM B$(11),D$(3),U$(13)
QB 515 PRINT CHR$(125):P=0
WR 520 ? :? "R A C H U N K I"
FF 522 ? "==============="
YB 525 B$="B R A W O !":U$="P O P R
    A W !"
SM 530 FOR K=1 TO 10:D=INT(RND(0)*3
    )+1:A=INT(RND(0)*7)+2:B=INT(RND(
    0)*8)+1
DF 540 IF B>A THEN A=B
UE 545 IF D=1 THEN X=A+B:D$=" + "
XT 550 IF D=2 THEN X=A-B:D$=" - "
TE 555 IF D=3 THEN X=A*B:D$=" * "
VF 560 ? :? "POLICZ"
SB 565 ? :? A;D$;B;" = ";:INPUT Y:G
    OSUB 630
WW 570 IF X=Y THEN ? :? B$:GOTO 590
DQ 580 P=P+1:? :? U$:GOTO 560
HH 590 NEXT K
NT 600 ? :? P;" POPRAWEK"
RY 610 D$="":? :? "Jeszcze raz ?"
DW 615 ? :? "            ";CHR$(212);"/
    ";CHR$(206);:INPUT D$
TX 620 IF D$="T" THEN 515
ER 625 END :REM ******************
HD 630 FOR D=10 TO 1 STEP -1:SOUND
    0,100,10,D:NEXT D:SOUND 0,0,0,0
ZX 635 RETURN
```

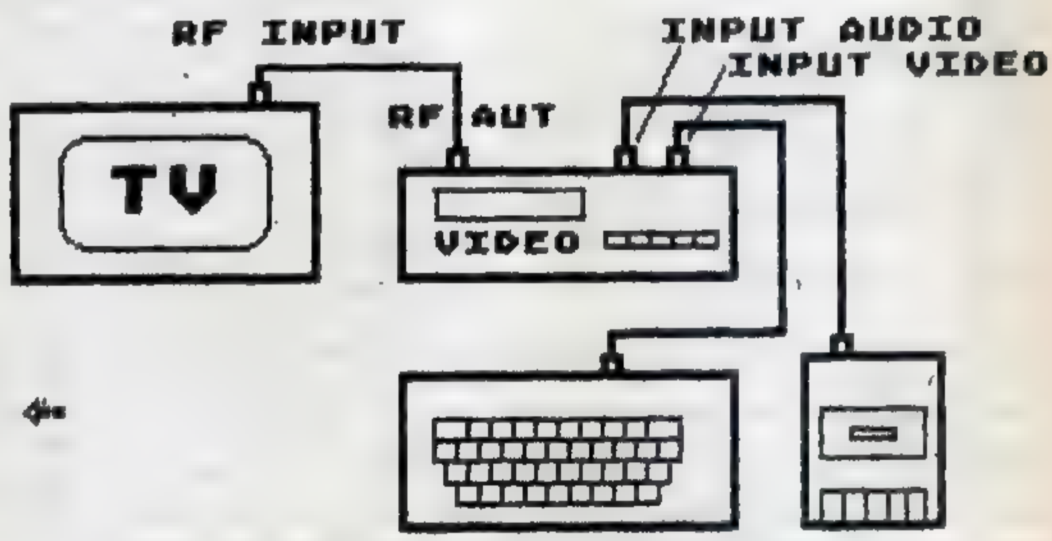
KOMPUTER I VIDEO

W wielu domach, obok komputera drugim ważnym przedmiotem użytkowym jest magnetowid. Oba te sprzęty egzystują równolegle nie „wchodząc" sobie w drogę. Tylko czy tak musi być? Jako pierwsze wspólne ich przedsięwzięcie proponuję wykonać czołówkę do swoich kaset, coś w rodzaju exlibrisu. Zrobić to można za pomocą dowolnego komputera, a skopiować na każdy magnetowid. Nim do tego przystąpimy, powinniśmy wykonać kabel połączeniowy między komputerem a magnetowidem. Typy wtyków będą różne i zależne od stosowanego sprzętu. W wypadku stosowania do tworzenia grafiki komputera Atari, na jednym końcu kabla powinniśmy przylutować wtyk stereofoniczny GM-5, a w wypadku Spectrum typu mini „Jack". Drugi koniec kabla zaopatrujemy w dwa wtyki WW-1 (SANYO, FISHER) lub po jednym WW-1 i BNC (NEC). Długość kabla powinna być oczywiście tak dobrana by pozwoliła w sposób wygodny korzystać z zestawu, tzn 1,2÷1,5 m. Schemat kabla do połączenia komputera Atari i dowolnego magnetowidu przedstawiono na rysunku 1.



To byłoby wszystko, jeśli chodzi o podłączenie komputera do magnetowidu. Drugi problem, to stworzenie czołówki „exlibrisu" swoich taśm. Metod jest wiele. Jedną z nich może być rozwiązanie programowe, to znaczy stworzenie programu z częścią graficzną, tekstową i ewentualnie dźwiękową. Nie jest to czynność łatwa, ale z pewnością dająca dużo satysfakcji. Inny sposób stworzenia czołówki, to wykorzystanie jednego z wielu programów graficznych Tu chciałbym zaproponować program pod tytułem „MAGIC-PAITER". W przeciwieństwie do innych programów najczęściej o statycznej grafice, można stworzyć za jego pomocą efekt „płynącej tęczy". Może on zostać zwielokrotniony, gdy do kreślenia wykorzystamy kilka kolorów. Każdy z kolorów będzie tworzył odrębną „tęczę". Aby uzyskać w tym programie efekt „płynącej tęczy", należy po otrzymaniu na ekranie monitora zestawu możliwości, zainicjować Multicolor-effect, a z niego np. Set Color 0. Następnie z opcji szerokość linii (Select breadth of Stripes) wybrać dla przykładu najszerszą linię (Very wide Stripes) i dalej z wyboru rodzaju linii (Select kind of Stripes), przewijanie w górę (Stripes up). Teraz pozostaje nam tylko powrót do opcji Multicolor-effect i uruchomienie swojego dzieła za pomocą Start Effect. Jedynym mankamentem tego i podobnych programów jest brak dźwięku. Upartym, udźwiękowienie czołówki też nie powinno sprawić większego kłopotu. Potrzebne są w tym celu trzy rzeczy: magnetofon, program muzyczny i przewód koncentryczny do połączenia magnetofonu z magnetowidem i komputerem. Do utrwalenia stworzonej w komputerze melodii można użyć każdego magnetofonu kasetowego lub szpulowego z wejściem na mikrofon i wyjściem do przegrywania. Melodię można stworzyć dowolnym programem muzycznym lub wykorzystać gotowe melodie. Z gotowych programów polecamy np: Music, Organy, Kompozytor

Kiedy mamy przygotowany obraz i melodię, możemy przystąpić do przekopiowywania ich na taśmę magnetowidową. Przedtem jednak musimy stworzyć układ do kopiowania obrazu i dźwięku, którego schemat przedstawiono poniżej.



Życzymy powodzenia.

Janusz JANIEC

Chochlik dał znać o sobie. Na ostatniej stronie 6 nr „IKS-a" popełniliśmy błąd. Otóż w tekście „Też komputery..." pojemność informacyjna „kostki" wynosi oczywiście nie 32 bity a 32 Kb i analogicznie nie 64 bity a 64 Kb.

Przepraszamy!

Transputery — rodzina INMOS

Układy rodziny transputerowej zostały opracowane w pracowniach brytyjskiej firmy INMOS, znanej dotychczas z produkcji pamięci statycznych i dynamicznych RAM, wykonanych przy użyciu technologii 1,6 μ lub 1,2 μ CMOS. Dzisiaj najważniejszą ofertą INMOS są układy rodziny transputerowej oraz szeroki asortyment kart, modułów, pakietów i systemów rozszerzania układów transputerowych i mikrokomputerów PC. Przy ich produkcji INMOS stosuje 1,5 μ technologii CMOS. Oferta firmy obejmuje także sterowane mikroprocesorowo układy specjalistyczne. Są one często mylnie zaliczane do rodziny transputerów. Należą do nich kaskadowe, 32-stopniowe filtry cyfrowe (IMS A100) stosowane m.in w analizatorach stanów oraz specjalne przetworniki analogowo-cyfrowe (IMS G17x) dla potrzeb kolorowej grafiki komputerowej, systemów wideo i TV. Produkty INMOS bazujące na transputerach zestawiono w tabeli nr 1.

W poprzednim odcinku przedstawiłem jeden z nich, adapter linii komunikacyjnych IMS C012, dziś kilka słów o innych produktach. **IMS M212** jest specjalizowanym 16-bitowym transputerem o mocy obliczeniowej równej 10 MIPS[1], przeznaczonym do obsługi dysków typu WINCHESTER lub stacji dysków elastycznych łącznie do 4 napędów. IMS M212 zawiera w jednym układzie jednostkę procesora, 4 KB ROM, 2 KB RAM, kontroler dysku, układy interfejsów oraz system sterujący. Dodatkowo poprzez interfejs pamięci zewnętrznej może obsługiwać 64 KB DRAM. W pamięci ROM rezyduje system wspomagania obsługi napędu dyskowego, natomiast RAM wykorzystuje się jako bufor sektora lub dla zapamiętania parametrów, formatu zapisu i odczytu danych. Ponadto w RAM zapisuje się program przetwarzający dane z linii komunikacyjnych łączących M212 bezpośrednio z innymi transputerami lub poprzez adapter z konwencjonalnymi systemami mikroprocesorowymi. Trzeci interfejs jest sprzężony poprzez 2 jednokierunkowe, 8-bitowe porty z kontrolerem dysku. Kontroler ten wyposażony jest w dwa dwukierunkowe porty, używane też do kontroli takich funkcji dyskowych, jak ustawienie początkowe, selekcja, sprawdzanie słowa statusu itp. Możliwości transputera M212 wykorzystano w karcie rozszerzenia IMS B005, zawierającej WINCHESTER i dysk elastyczny oraz opcjonalnie dołączane duże stacje dysków poprzez interfejsy dodatkowo buforowane według standardu SA400/ST506

IMSB007 jest podwójną Eurokartą przeznaczoną do realizacji funkcji graficznych. Zawiera ona 32-bitowy transputer IMS T414, wspomniany przetwornik IMS G170, 512 KB DRAM i podwójny port (512 KB) wideo RAM oraz buforowanie linii komunikacyjnych i wyjście RGB. Dodatkowe wspomaganie karty poprzez software w OCCAM zapewnia realizację podstawowych funkcji graficznych (przy kreśleniu linii i wielokątów) oraz generowanie alfanumerycznych tekstów ze skalowaniem. Obraz jest mapowany w pamięci z wykorzystaniem podwójnego ekranu dla szybkiego odświeżania zawartości wyświetlanego obrazu. Karta zapewnia wybór 262 lub 144 kodów umożliwiając grafikę 512x512 przy 8-bitowym kodzie opisującym dowolny punkt i jest przeznaczona dla monitorów o częstotliwości wybierania linii w zakresie od 20 do 50 kHz.

Moduły podstawowe, karty rozszerzania, pakiety specjalistyczne, a także urządzenia i systemy bazujące na technice transputerów wyposażone w odpowiednie oprogramowanie można znaleźć w ofertach wielu firm m.in. Transtech, Meiko, Sension, MikroWay, Nichie.

Kompletne oprogramowanie INMOS może być używane łącznie z systemem operacyjnym UNIX v.2.2. Pakiet stanowi integralną część systemu i zawiera transputer T1414 z 32-bitową magistralą lokalną, do której podłączono 2 MB dodatkowej pamięci DRAM i 64-256 KB EPROM oraz podwójny układ uniwersalnych ansychronicznych nadajników-odbiorników linii (DUART), dwa szeregowe porty RS232 oraz zwrotnicę linii komunikacyjnych. Karta posiada dwa złącza umożliwiające opcjonalne dołączanie modułów kompatybilnych z IMS B0xx. Ponadto na karcie zainstalowano niezbędne elementy i bloki dla systemu VME oraz adapter linii, który sprzęga (łącznie z DRAM) środowisko transputerowe z magistralą systemową.

Sension jest producentem transputerowych urządzeń TES, znacznie zwiększających parametry użytkowe i moc obliczeniową mikrokomputerów. TES są przeznaczone do bezpośredniego połączenia z mikrokomputerami IBM PC XT/AT i kompatybilnymi, a także stosując odpowiednie adaptery z innymi komputerami PC. Można w nich zainstalować maksymalnie 4 transputery T414-20 (opcjonalnie T800-20 i 4 MB DRAM. Ponadto firma oferuje system PARALLAX umożliwiając współbieżną wieloprocesorową pracę wykorzystując magistralę VME. PARALLAX jest uniwersalną otwartą konstrukcją pozwalającą w znacznym zakresie na konfigurowanie systemu oraz dalsze rozwijanie przy jednoczesnym bieżącym testowaniu poprawności współdziałania poszczególnych jego elementów. Cały system składa się z pewnej ilości nie powiązanych ze sobą subsystemów budowanych w oparciu o podwójne Eurokarty. Każdy subsystem może zawierać do 5 kart bazujących na 32-bitowych transputerach, PARALLAX jest wyposażony w moduły przetwarzające, pamięć masową, karty buforowanych interfejsów RS422 oraz optyczne linie komunikacyjne (dla transmisji danych), szybkie przetworniki analogowo-cyfrowe, pakiety zasilania, monitorowanie i analizy. Komunikacja jest trzypoziomowa. Dla realizacji sieci transputerowych dynamicznie rekonfigurowanych system posiada kartę zespołów programowanych 32-liniowych łącznie elektronicznych (IMS C004), adaptery i inne niezbędne elementy sieciowe.

W październiku 1987 roku powstało amerykańsko-brytyjskie konsorcjum NICHE. Produkty NICHE mają stanowić szereg integralnych systemów o budowie modularnej. Pierwszym pakietem, który opracowała Nichie Technology Limited jest rozwinięty system obliczeniowy NT-1000 ACP (rys. 2) Jest on przeznaczony do bezpośredniego włączenia w hardware amiteksurę mikrokomputerowych stacji roboczych Sun-3 i Sun-4 produkowanych przez Sun Microsystem. NT-1000 zawiera szeregowe i równoległe porty I/O, integralny szybki kontroler portów, złącza dla 32 modułów, pakiet konfiguracji sieci (na bazie układów C004) oraz interfejsy linii komunikacyjnych INMOS. Moduły posiadają 32-bitowy transputer i do 16 MB DRAM. Moc obliczeniowa stacji roboczych dla modelu Sun-3/160 wynosi 2 MIPS, Sun-3/260 — 4 MIPS, Sun-4/10 MIPS, zaś maksymalna moc obliczeniowa bloku NT/1000 jest równa 320 MIPS (48 MFLOPS). Zadanie wykonywane przez stację Sun-3/10 w ciągu 8 godzin przy zastosowaniu bloku NT-1000 może być zrealizowane w czasie 3 minut. Do stacji Sun można dołączyć 8 bloków NT-1000. Stację wyposażono w pełne oprogramowanie systemów transputerowych (TDS) oraz kompilatory języków FORTRAN, PASCAL, C, PROLOG oraz system operacyjny Sun-UNIX.

Stosowanie transputerów lub transputerowych kart rozszerzenia w konwencjonalnych mikrokomputerach klasy IBM PC XT/AT. DEC MicroVax czy komputerowych stacjach roboczych SUN może mieć tylko uzasadnienie w zwiększaniu możliwości użytkowych posiadanego sprzętu. Nie jest to jednak właściwe i pełne wykorzystanie potencjału i własności tych procesorów. Jest wiele przykładów, gdzie moc obliczeniowa decyduje o jakościowych walorach i zdolnościach techniczno-użytkowych urządzeń — na przykład skomplikowane obliczenia dla precyzyjnego kreślenia rysunków, krojów pisma, i innych znaków graficznych przez graficzne drukarki laserowe i plotery, grafika i animacja o bardzo dużej rozdzielczości, a także problemy trójwymiarowości obrazów.

Stosunek środowisk naukowych i inżynierskich do transputerów, ich znaczenia w rozwoju technik obliczeniowych i technologii układów mikro wyraża się w powołaniu przy SERC/DTI [2]/ towarzystwa fundacyjnego na rzecz popierania technik transputerowych oraz licznych regionalnych grup inicjatywnych. SERC/DTI Transputer Inititive skupia w swoich szeregach członkowskich ponad 1000 osób, przede wszystkim ze środowisk naukowych (z 40 placówek naukowo-uniwersyteckich i politechnicznych), działających w ramach 10 sekcji. Ich wkład w rozwój zagadnień technik komputerowych, to organizacja sesji i konferencji naukowych, popularyzacja wiedzy poprzez wydawnictwo materiałów, publikacji oraz podręczników programowania, organizacja kursów i, szkoleń a także prowadzenie prac naukowo-badawczych i wdrożeniowych. Wydaje się, że stosunek środowisk naukowych najlepiej wyraża wartość produktów firmy INMOS —

Tabela 1. Transputerowe produkty firmy INMOS

	Ozn. kat.	Opis
	1	2
Podsta[illegible] układy rodziny INMOS	IMS T800	32-bit. transputer, 64-bit. FPU, 4 KB on-RAM[a]
	IMS T414	32-bit transputer, 2 KB on-RAM[b]
	IMS T212	16-bit transputer, 2 KB on-RAM[c]
	IMS M212	16-bit. disc procesor, 2 KB on-RAM, 2 KB on-ROM[b]
	IMS C004	zespół programowanej łącznicy elektronicznej[d]
	IMS C011	adapter linii z dwoma modami pracy[d]
	IMS C012	adapter linii[d]
Karty	IMS [illegible]	T414 (T800 + 64 KB SRAM + 128 KB ROM, 2 porty RS232
	IMS [illegible]	T414 + 2 MB DRAM + 128 KB ROM, 2 porty RS232
	IMS [illegible]	cztery.T414 (T800 + 1 MB DRAM, konfiguracja kwadratu
	IMS [illegible]	karta do IBM PC XT (AT z T414 + 2 MB DRAM, IBM PC interfejs I/O
	IMS [illegible]	M212 + 20 MB 3,5' Winchester + 1 MB 3,5' floppy disc
	IMS [illegible]	M212 + 56 KB SRAM + 2 porty RS232 z możliwością zainstalowania do 9 T212 i 128 KB SRAM łącznie sterowaniem UART i EPROM
Pakiety	[illegible]	karta grafiki: T414 + G170 + 512 KB DRAM + KB VRAM
	IMS B009	cztery A100 + T212 + moduł dekodera adresu i generatora, interfejs IBM PC — 1280 MOPS[e] — opcjonalnie T414/T800, 1MB RAM
	IMS B010	T414 + 1 MB DRAM + C012 — karta dla komputera NEC PC-9801
	IMS B011	T414/T800 + 2 MB DRAM + 2 porty RS232, 2 miejsca dla modułów rozszerzenia INMOS, system magistrali VME
Moduły	IMS B203	10 B003 + 10 MB DRAM + D600 lub D701, możliwość rekonfiguracji poprzez wymianę karty na inną typu B0xx
	IMS B401	T414/T800 + 32 KB SRAM
	IMS B402	T212 + 8 KB SRAM
	IMS B403	T414/T800 + 1 MB DRAM
	IMS B404	T800 + 2 MB DRAM
	IMS B405	T800 + 8 MB DRAM
Transputerowe systemy/opro[illegible]ne	IMS D600	OCCAM 2 z pełnym edytorem, kompilatorem i konfiguratorem systemu dla komputera VAX/VAM 11/7xx firmy DEC mikroVAX
	IMS D700	OCCAM 2 z pełnym edytorem, kompilatorem i konfiguratorem systemu dla mikrokomputera IBM PC XT AT
	IMS D 701	D700 + B004
	[illegible]	symulator A100 w OCCAM
	IMS D704A	D700 + D703A + B009
	IMS D711	D700 + kompilator C
	IMS D712	D700 + kompilator PASCAL
	IMS D713	D700 + kompilator FORTRAN-u

[a] — Zegar procesora: 17, 20, 30 MHz
[b] — Zegar procesora: 15, 20 MHz
[c] — Zegar procesora: 17, 20 MHz
[d] — Szybkość transmisji: 10/20 Mbitów/s.
[e] — operacje na sekundę

1/ MIPS — jednostka mocy obliczeniowej wyrażona w milionach instrukcji wykonywanych w ciągu sekundy

2/ SERC/DTI — Science and Engineering Research Council (Departament of Trade and Industry — Rada Badań Naukowo-Technicznych) Dep. Handlu i Przemysłu. Fundacja na rzecz popierania technik transputerowych powstała przy SERC/DTI w lutym 1987 r.

W. M. S.

Nuty z komputera — cz. 3

W trzeciej części artykułu zajmiemy się współpracą z całym zbiorem taktów. Jest to zagadnienie niezwykle istotne i (niestety) dosyć skomplikowane, zarówno od strony programowej, jak i użytkowej.

Na wstępie chcemy przekazać Czytelnikom kilka ważnych uwag.

UWAGA 1: **brzmienie, klawiatura dynamiczna** — oprócz wysokości dźwięku istnieje olbrzymia grupa parametrów opisująca przebieg dźwięku w czasie. Niestety, ich ilość i format zależą od typu syntezatora. Nowoczesne „klawisze" mają klawiaturę dynamiczną, stąd przesyłanie tylko wysokości dźwięku jest niewystarczające. Wynika z tego istotny wniosek: najlepszym wyjściem jest możliwość przesyłania w każdym kroku nie tylko wysokości dźwięku, ale i dowolnej ilości parametrów brzmieniowych. W przedstawianej wersji sekwencera nie będziemy tych parametrów programować, ale możliwość ich zapisu przewidzieliśmy (szczegóły podczas opisu „kompresji").

UWAGA 2: **listing** — łączymy z pierwszą częścią programu. Jeżeli linia o danym numerze już występuje, to należy ją wymienić na aktualnie zamieszczoną.

UWAGA 3: **zmiana skali dźwięków** — ze względu na przeprowadzaną kompresję zapisu taktu dźwięk **g7** (g siedmiokreślne) został przyporządkowany **pauzie**. Aktualna skala rozciąga się od **C2** do **fis7**. Konsekwencją tej zmiany jest przyporządkowanie w zapisie cyfrowym pauzy wartości 127.

UWAGA 4: **wprowadzanie nazw taktów** — procedura umożliwia wprowadzenie słowa dziesięcioznakowego
1) możemy wprowadzić małe litery, cyfry, spację oraz znaki ASCII o kodach od 33 do 47
2) pierwszym znakiem musi być litera
3) funkcje dużych liter:
<**W**> — wprowadza nazwę: ukazuje się tekst „<t/n>" — wciśnięcie <t> wprowadza nazwę
<**D**> — kasuje ostatnio wprowadzony znak
<**P**> — przerwanie procedury

EDYTOR TAKTU (nowe funkcje)

klawisz <**m**> — umożliwia wprowadzenie podzielnika tempa (tzn. ile razy wolniej dany takt ma być wykonywany w stosunku do tempa podstawowego) parametry: 1) podzielnik

uwaga: opcjonalnie podzielnik jest ustawiany na wartość maksymalną: 255

klawisz <**u**> — pozwala na zapis taktu do pamięci (zmiennej tekstowej m$) i powrót do edytora głównego

klawisz <**U**> — powrót do edytora głównego bez zapisu taktu

klawisz <**n**> — pozwala podzielić edytowany takt na dwa niezależne takty
parametry: 1) nr kroku, od którego rozpoczynać się będzie drugi takt
2) nazwa tworzonego taktu

uwaga: jeżeli drugi takt będzie rozpoczynał się od kroku nr 1, to procedura podziału jedynie wprowadzi podaną nazwę taktu na miejsce starej

uwaga: po wykonaniu podziału następuje powrót do edytora głównego

EDYTOR GŁÓWNY

klawisz <**w**> — wywołanie taktu z pamięci (tzn. odczyt ze zmiennej tekstowej m$), wejście do edytora taktu
parametry: 1) nr taktu

klawisz <**u**> — utworzenie taktu
parametry: 1) nazwa taktu
2) ilość kroków

warunki: a) jest to pierwszy wprowadzany takt
3) następuje zgłoszenie funkcji <d> edytora taktu: należy podać parametry pierwszej ścieżki — wtedy nastąpi wejście do edytora taktu z pierwszą zadeklarowaną ścieżką

b) istnieje co najmniej jeden takt w pamięci
3) należy wybrać, czy takt jest tworzony od początku, czy wg ustawienia ście-

żek określonego taktu
wciśnięcie <n> — wykonanie procedury z punktu a) wciśnięcie <t> — spowoduje skorzystanie z wzorcowego taktu·
4) numer wzorcowego taktu

klawisz <t> — grupowa transpozycja w górę (przeprowadzana wg zawartości pola aktywności)
parametry: ilość półtonów transpozycji
klawisz <T> — jak wyżej — transpozycja w dół
klawisz <k> — kasowanie grupowe przeprowadzane na podstawie zawartości pola aktywności
uwaga: jeżeli nastąpi skasowanie wszystkich ścieżek, wtedy takt jest usuwany z pamięci
klawisz <d> — dane taktu — funkcja wypisuje podstawowe informacje o takcie i o ścieżkach (przynależność do instrumentu, źródła i typ zapisu)
parametry: 1) numer taktu
klawisz <l> — lista taktów — drukuje na ekranie ponumerowane takty, ich nazwy, ilość ścieżek, kroków i podzielnik tempa
uwaga: kolejne numery wypisywane przy taktach są obowiązujące podczas żądania: *podaj numer taktu*

```
   1 REM
********************************
     EDYTOR UTWORU I TAKTU
********************************
   7 DEF FN z$(y)=CHR$ (y-256*IN
T (y/256))+CHR$ INT (y/256)
   8 DEF FN z(y)=256*CODE m$(y+1
)+CODE m$(y)
  30 LET w$=CHR$ 0+CHR$ 0+CHR$ 0
+CHR$ 0: LET w$=w$+w$+w$+w$+CHR$
 0
  31 LET cike=10: LET cise=8
  40 LET t$="c0c1d0d1e0f0f1g0g1a
0a1h021001234567"
  50 LET m$=CHR$ 0
 103 LET pnk=(str-1)*mik+1: PRIN
T "  ";: LET knk=pnk+eik-1: FOR
t=pnk TO knk: PRINT " " AND t<10
0; INVERSE 1;"0" AND (t<10);t; I
NVERSE 0;: NEXT t
 172 LET str=FN m(str,INT ((ik-1
)/mik)+1): LET pnk=mik*(str-1)+1
1000 REM EDYTOR GLOWNY
1001 LET it=CODE m$(1)
1005 GO SUB 1910
1010 GO SUB 9198
1011 LET x$=INKEY$: IF x$="" THE
N GO TO 1011
1012 IF x$="w" AND it>0 THEN GO
SUB 1115: GO TO 1005
1014 IF x$="u" AND it<16 THEN GO
 SUB 1100: GO TO 1005
1020 IF x$="a" AND it>0 THEN GO
SUB 8900: GO TO 1010
1024 IF x$="k" AND it>0 THEN GO
SUB 1130: GO TO 1010
1026 IF x$="t" AND it>0 THEN GO
SUB 1140: GO TO 1010
1027 IF x$="T" AND it>0 THEN GO
SUB 1160: GO TO 1010
1029 IF x$="d" AND it>0 THEN GO
SUB 1170: GO TO 1010
1031 IF x$="l" AND it>0 THEN GO
SUB 1180: GO TO 1010
1090 GO TO 1005
1100 GO SUB 8975: IF q$="" THEN
RETURN
1101 IF k<=it THEN RETURN
1104 LET q=0: LET n$=q$: LET md=
256: LET z$="ilosc krokow:": GO
SUB 9190: IF q=0 THEN RETURN
1105 LET pt=255: LET e$="": LET
ik=q: IF it=0 THEN GO TO 1108
1106 LET z$="instr. w/q wzorca 1
-tak": LET q=1: GO SUB 9190
1107 IF q=1 THEN GO TO 1110
1108 LET is=0: LET q=1: LET eis=
1: GO SUB 9151: IF is=1 THEN LET
 nt=it+1: GO SUB 9000
1109 RETURN
1110 LET nt=q: GO SUB 1120: IF q
>0 THEN LET nt=it+1: FOR k=1 TO
ik: LET e$=e$+CHR$ 128: NEXT k:
LET n=FN z(2*q)+2: LET is=CODE m
$(n): LET n$=q$+m$(n+11+2*is TO
n+10+5*is): FOR k=1 TO is-1: LET
 e$=e$+e$: NEXT k: GO SUB 9000
1112 RETURN
1115 GO SUB 1120: IF q>0 THEN LE
T nt=q: GO SUB 9255: GO SUB 9000
1117 RETURN
1120 LET z$="podaj nr taktu:": L
ET md=it: LET q=0: GO TO 9190
1121 LET q=0: LET md=2: LET z$="
transp. 1-sc. 2-takt.": GO SUB 9
190: RETURN
1130 LET q=0: LET md=2: LET z$="
kasow. 1-sc. 2-takt.": GO SUB 91
90: IF q=0 THEN RETURN
1131 IF q=1 THEN GO SUB 8840: RE
TURN
1132 LET t=1
1133 FOR t=t TO it: IF w$(t)=CHR
$ 1 THEN GO SUB 8825: GO TO 1133
1134 NEXT t: RETURN
1140 LET q=0: LET md=126: LET z$
="transp w gore o:": GO SUB 9190
: IF q=0 THEN RETURN
1141 LET r=q: GO SUB 1121: IF q=
0 THEN RETURN
1142 IF q=2 THEN GO TO 1147
1143 LET t=1: GO SUB 8800
1144 IF t=0 THEN RETURN
1145 IF CODE m$(j+2)<>1 THEN GO
SUB 1150
1146 GO SUB 8810: GO TO 1144
1147 FOR t=1 TO it: IF w$(t)=CHR
$ 0 THEN GO TO 1149
1148 LET n=FN z(2*t): LET k=CODE
 m$(n+2): LET j=n+13+2*k: FOR g=
1 TO k: GO SUB 1150: LET j=j+3:
NEXT g
1149 NEXT t: RETURN
1150 LET q=n+FN z(n+11+2*g): GO
SUB 8830: LET b=w+q-1: IF m$(j+2
)=CHR$ 1 THEN RETURN
1151 FOR q=q TO b
1152 LET f=CODE m$(q): IF f>127
THEN LET f=f-128
1153 IF f=127 THEN GO TO 1156
1154 LET f=f+r: IF f<1 OR f>126
THEN LET f=127
1156 IF CODE m$(q)<128 THEN LET
m$(q)=CHR$ f: GO TO 1159
1157 LET m$(q)=CHR$ (f+128): LET
 q=q+1: IF CODE m$(q)<128 THEN G
O TO 1159
1158 LET q=q+1
1159 NEXT q: RETURN
1160 LET q=0: LET md=126: LET z$
="transp w dol o:": GO SUB 9190
 IF q=0 THEN RETURN
1161 LET r=-q: GO SUB 1121: IF q
=0 THEN RETURN
1162 IF q=2 THEN GO TO 1147
1163 GO TO 1143
1170 GO SUB 1120: IF q=0 THEN RE
TURN
1171 CLS : LET n=FN z(2*q): PRIN
T "nazwa taktu: "; INVERSE 1;m$(
n+3 TO n+12)

1172 LET s=CODE m$(n+2): PRINT "
ilosc sciezek: ";TAB 19;s: PRINT
"ilosc krokow: ";TAB 19;CODE m$(
n+1)+1: PRINT "podzielnik metrum
```

```
: ";CODE m$(n)
1173 LET n=n+13+2*s: PRINT : PRI
NT "  instrument    zrodlo    ty
p": PRINT : FOR q=1 TO s: LET i=
12*CODE m$(n)-9: LET z=6*CODE m$
(n+1)-5: LET t=CODE m$(n+2): PRI
NT q;TAB 4;i$(i TO i+9);TAB 16;s
$(z TO z+5);TAB 23;("literowy" A
ND (t=0));("symbol." AND (t=1));
("cyfrowy" AND (t=2)): LET n=n+3
: NEXT q
1174 RETURN
1180 CLS : PRINT "nr    nazwa
 is   ik   pt": PRINT
1181 FOR t=1 TO it: LET n=FN z(2
*t): PRINT t;TAB 3; INVERSE 1;m$
(n+3 TO n+12)
1182 LET s=CODE m$(n+2): PRINT T
AB 15,s;TAB 20;CODE m$(n+1)+1;TA
B 25;CODE m$(n)
1183 NEXT t: RETURN
1185 CLS : LET n=FN z(2*q): PRIN
T "nazwa taktu: "; INVERSE 1;m$(
n+3 TO n+12)
1186 LET s=CODE m$(n+2): PRINT "
ilosc sciezek:";TAB 23;s: PRINT
"ilosc krokow: ";TAB 23;CODE m$(
n+1)+1: PRINT "podzielnik taktow
ania: ";CODE m$(n)
1200 LET q=0: LET it=CODE m$(1):
 LET md=it: LET z$="wybierz takt
:": GO SUB 9190: IF q=0 THEN GO
TO 9198
1910 CLS : PRINT TAB 7; INVERSE
1;"SEKWENCER v.1.0": PRINT : PRI
NT TAB 8; INVERSE 1;"EDYTOR GLOW
NY"
1912 PRINT TAB 5;"w - wywolanie
taktu"
1914 PRINT TAB 5;"u - utworzenie
 taktu"
1916 PRINT TAB 5;"t - transpozyc
ja w gore"
1917 PRINT TAB 5;"T - transpozyc
ja w dol"
1918 PRINT TAB 5;"k - kasowanie"
1930 PRINT TAB 5;"d - dane taktu
"
1932 PRINT TAB 5;"l - lista takt
ow"
1934 PRINT TAB 5;"a - aktywnosc"
1950 RETURN
8800 REM wyszukiwanie sciezki
8801 FOR t=t TO it
8802 IF w$(t)=CHR$ 0 THEN GO TO
8813
8804 FOR i=1 TO ii
8805 IF w$(it+i)=CHR$ 0 THEN GO
TO 8812
8806 LET s=CODE i$((i-1)*12+1):
LET d=CODE i$((i-1)*12+2): FOR z
=1 TO s
8807 IF w$(it+ii+d)=CHR$ 0 THEN
GO TO 8811
8808 LET n=FN z(2*t): LET k=CODE
 m$(n+2): LET g=1
8809 FOR g=g TO k: LET j=n+10+2*
k+3*g: IF m$(j)=CHR$ i AND m$(j+
1)=CHR$ d THEN RETURN
8810 NEXT g
8811 LET d=d+1: NEXT z
8812 NEXT i
8813 NEXT t: LET t=0: RETURN
8820 REM skasow. sciezki w pam.
8822 GO SUB 8830: LET b=w+5: LET
 q=2*t+2: FOR l=t+1 TO it: LET m
$(q TO q+1)=FN z$(FN z(q)-b): LE
T q=q+2: NEXT l
8823 LET q=n+13: FOR l=1 TO k: L
ET m$(q TO q+1)=FN z$(FN z(q)-5-
w*(l>g)): LET q=q+2: NEXT l: LET
 m$(n+2)=CHR$ (k-1): LET q=n+11:
 LET e=q+2*k+3*g-2: LET b=n+FN z
(q+2*g)+5: LET k=k-1: LET m$=m$(
 TO q+2*g-1)+m$(q+2*g+2 TO e)+m$
(e+4 TO b-1)+m$(b+w TO ): RETURN
8825 REM skasowanie taktu w pam.
8826 LET u=FN z(2*t): LET r=FN z
(2+2*t)-FN z(2*t): FOR l=2 TO 2*
it STEP 2: LET m$(l TO l+1)=FN z
$(FN z(l)-r*(l>=(2+2*t))-2): NEX
T l
8827 LET m$=m$( TO 2*t-1)+m$(2+2
*t TO u-1)+m$(u+r TO )
8828 LET it=it-1: LET m$(1)=CHR$
 it: LET w$=w$( TO t-1)+w$(it+1 T
O ): RETURN
8830 REM oblicz. dlug. sciezki
8831 IF g<k THEN LET w=FN z(n+13
+2*g)-FN z(n+11+2*g): RETURN
8832 IF t<it THEN LET w=FN z(2*t
+2)-FN z(n+11+2*g)-n: RETURN
8833 LET w=LEN m$-FN z(n+11+2*g)
-n+1: RETURN
8840 REM proc. kasowania sciezek
8842 LET t=1: GO SUB 8800
8844 IF t=0 THEN RETURN
8846 IF k=1 THEN GO SUB 8825: GO
 SUB 8800: GO TO 8844
8848 GO SUB 8820: GO SUB 8809: G
O TO 8844
8849 RETURN
8900 REM procedura wyboru
8901 CLS : LET zu=1: LET pt=1: L
ET iwt=FN m(16,it): LET pi=1: LE
T iwi=FN m(16,ii): LET n=1: LET
iwz=FN m(16,CODE i$(1)): LET at=
0: LET ai=it: LET az=it+ii: LET
x=1: LET y=1: LET knc=iwt
8903 GO SUB 8940: GO SUB 8945: G
O SUB 8950: GO SUB 8960
8905 GO SUB 9198
8906 LET q$=INKEY$: IF q$="" THE
N GO TO 8906
8907 BEEP 0.01,20
8910 IF q$="6" AND y<knc THEN GO
 SUB 8965: LET y=y+1: GO SUB 896
0: GO TO 8906
8911 IF q$="7" AND y>1 THEN GO S
UB 8965: LET y=y-1: GO SUB 8960:
 GO TO 8906
8912 IF q$="8" AND x<3 THEN LET
knc=iwi*(x=1)+iwz*(x=2): GO SUB
8965: LET x=x+1: LET y=FN m(knc,
y): GO SUB 8960: GO TO 8906
8913 IF q$="5" AND x>1 THEN LET
knc=iwt*(x=2)+iwi*(x=3): GO SUB
8965: LET x=x-1: LET y=FN m(knc,
y): GO SUB 8960: GO TO 8906
8914 IF q$=" " AND x=1 THEN LET
w$(y+pt-1)=CHR$ NOT CODE w$(y+pt
-1): GO SUB 8940: GO TO 8906
8915 IF q$=" " AND x=2 THEN LET
w$(y+pt+it-1)=CHR$ NOT CODE w$(y
+pt+it-1): GO SUB 8945: GO TO 89
06
8916 IF q$=" " AND x=3 THEN LET
w$(y+az+pz-1)=CHR$ NOT CODE w$(y
+pz+az-1): GO SUB 8950: GO TO 89
06
8917 IF q$<>"k" AND q$<>"K" AND
q$<>"u" AND q$<>"U" THEN GO TO 8
921
8918 LET m=(q$>"a"): IF x=1 THEN
 GO SUB 8955: IF q=1 THEN FOR k=
1 TO iwt*m+it*NOT m: LET w$(k+(p
t-1)*m)=CHR$ (q$="u" OR q$="U"):
 NEXT k: GO SUB 8940: GO TO 8905
8919 IF x=2 THEN GO SUB 8955: IF
 q=1 THEN FOR k=1 TO iwi*m+ii*NO
T m: LET w$(k+it+(pi-1)*m)=CHR$
(q$="u" OR q$="U"): NEXT k: GO S
UB 8945: GO TO 8905
```

```
8920 IF x=3 THEN GO SUB 8955: IF
 q=1 THEN FOR k=1 TO iwz*m+(LEN
 w$-it-ii)*NOT m: LET w$(k+(az+pz
-1)*m+((it+ii)*NOT m))=CHR$ (q$=
"u" OR q$="U"): NEXT k: GO SUB 8
950: GO TO 8905
8921 IF q$="c" THEN LET q=0: LET
 z$="1-kas. 2-ust. calosci": LET
 md=2: GO SUB 9190: IF q>0 THEN
FOR k=1 TO LEN w$: LET w$(k)=CHR
$ (q=2): NEXT k: GO TO 8903
8929 IF q$="p" THEN RETURN
8930 GO TO 8905
8940 PRINT AT 0,3;"TAKTY": LET a
x=1: FOR k=1 TO iwt: LET l=FN z(
(k+pt-1)*2)+3: PRINT AT k+1,1; I
NVERSE CODE w$(k+at);m$(l TO l+9
): NEXT k: GO TO 8970
8945 PRINT AT 0,13;"INSTRUMENTY"
: LET ax=13: FOR k=1 TO iwi: LET
 l=3+(k+pi-2)*12: PRINT AT k+1,1
3; INVERSE CODE w$(ai+k+pt-1);i$
(l TO l+9); INVERSE 0;("*" AND z
u=k);(" " AND zu<>k): NEXT k: GO
 TO 8970
8950 PRINT AT 0,26;"ZRODLA": LET
 ax=26: FOR k=1 TO iwz: LET l=n+
(k-1)*6: PRINT AT k+1,26; INVERS
E CODE w$(az+k);s$(l TO l+5): NE
XT k: GO TO 8970

8955 LET q=1: LET z$=("kasowanie
" AND (q$="k" OR q$="K"))+("usta
wianie" AND (q$="u" OR q$="U"))+
(" kolumny" AND NOT m)+" 1-tak":
 GO TO 9190
8960 PRINT AT y+1,(x=2)*12+(x=3)
*25; FLASH 1;">": IF x=2 THEN LE
T zu=y: GO SUB 8945: LET uz=y: L
ET nri=y-1+pi: LET aii=1+(nri-1)
*12: LET n=6*(CODE i$(aii+1)-1)+
1: LET az=it+ii+CODE i$(aii+1)-1
: LET pz=1: LET iwz=FN m(16,CODE
 i$(aii)): GO TO 8950
8962 RETURN
8965 PRINT AT y+1,(x=2)*12+(x=3)
*25;" "
8966 RETURN
8970 FOR k=k TO 16: PRINT AT k+1
,ax;"          "+("      " AND ax<>26)
: NEXT k: RETURN
8975 REM wprow. nazwy taktu
8976 LET md=10: LET z$="nazwa ta
ktu: ": LET q$="": GO SUB 8980:
IF q$="" THEN RETURN
8977 FOR k=1 TO it: LET n=FN z(2
*k)+3: IF m$(n TO n+9)=q$ THEN G
O SUB 8979: RETURN
8978 NEXT k: RETURN
8979 PRINT AT 19,0; FLASH 1;" N
AZWA JUZ ISTNIEJE  -  Enter  ":
PAUSE 1: PAUSE 0: RETURN
8980 GO SUB 9198: PRINT AT 19,0;
 INVERSE 1;z$;: PRINT ;" ";: IF
q$<>"" THEN PRINT INVERSE 1;q$;
8982 LET x$=INKEY$: IF x$="" THE
N GO TO 8982
8983 IF x$="P" THEN LET q$="": R
ETURN
8984 IF NOT (x$>="a" AND x$<="z"
 OR (x$="W" OR x$="D" OR x$>=" "
 AND x$<="9") AND q$<>"") THEN G
O SUB 9326: GO TO 8982
8985 IF x$="W" THEN PRINT ;" ";
FLASH 1;"(t/n)": PAUSE 1: PAUSE
0:: LET x$=INKEY$: LET q$=q$ AND
 (x$="t"): FOR k=LEN q$+1 TO 10*
(q$<>""): LET q$=q$+" ": NEXT k:
 BEEP .1,30: RETURN
8986 IF x$="D" THEN LET q$=q$( T
O LEN q$-1): GO TO 8980
8987 LET q$=q$+x$: IF LEN q$>md
THEN LET q$="": GO SUB 9326: GO
TO 8980
8988 PRINT INVERSE 1;x$;: GO TO
8982
9045 IF x$="d" THEN LET q=0: GO
SUB 9150: GO SUB 100: GO TO 9025
9046 IF x$="D" THEN LET q=by: GO
 SUB 9150: GO SUB 100: GO TO 902
5
9059 IF x$="u" THEN GO TO 9225
9060 IF x$="U" THEN GO TO 9228
9061 IF x$="m" THEN LET q=0: GO
SUB 9260: GO TO 9025
9062 IF x$="n" THEN LET q=0: GO
TO 9265

9090 GO TO 9025
9150 LET md=is+1: LET z$="dolacz
. sciez. nr:": GO SUB 9190: IF q
=0 THEN RETURN
9151 LET ans=q: GO SUB 9180: LET
 md=ii: LET q=0: LET z$="podaj n
r instumentu:": GO SUB 9190: IF
q=0 THEN RETURN
9152 LET ani=q: PRINT AT 3+q,2;
INVERSE 1;q;"-";i$(12*(q-1)+3 TO
 12*q): LET aai=1+12*(q-1): LET
ais=CODE i$(aai TO aai): LET asp
=CODE i$(aai+1 TO aai+1): GO SUB
 9185: LET md=ais: LET q=0: LET
z$="podaj nr zrodla:": GO SUB 91
90: IF q=0 THEN RETURN
9153 PRINT AT 3+q,20; INVERSE 1;
";q;"-";s$(6*(asp+q-2)+1 TO 6*(as
p+q-1)): LET ansi=q: LET md=3: L
ET q=0: LET z$="1-liter 2-graf 3
-cyfr": GO SUB 9190: IF q=0 THEN
 RETURN
9154 LET k=FN o(ans): LET z$="":
 FOR n=1 TO ik: LET z$=z$+CHR$ 1
28: NEXT n: LET e$=e$( TO k-1)+z
$+e$(k TO ): LET k=FN t(ans): LE
T n$=n$( TO k-1)+CHR$ ani+CHR$ (
asp+ansi-1)+CHR$ (q-1)+n$(k TO )
: LET eis=eis+(is<cise): LET is=
is+1: RETURN
9202 LET k=q: LET md=FN m(ik-q+1
,ik-1): LET q=0: LET z$="ile kro
kow kasowac:": GO SUB 9190: IF q
=0 THEN GO SUB 9198: RETURN
9210 LET md=is: LET z$="nr sciez
ki transpon.:": GO SUB 9190: IF
q=0 OR CODE n$(10+3*q)=1 THEN GO
 TO 9198
9211 LET k=q: LET q=0: LET md=12
6: LET z$="transpozycja w gore o
:": GO SUB 9190: IF q=0 THEN GO
TO 9198
9213 LET n=n+q: IF n>=127 OR n<0
 THEN LET n=128
9215 LET md=is: LET z$="nr sciez
ki transpon.:": GO SUB 9190: IF
q=0 OR CODE n$(10+3*q)=1 THEN GO
 TO 9198
9216 LET k=q: LET q=0: LET md=12
6: LET z$="transpozycja w dol o:
": GO SUB 9190: IF q=0 THEN GO T
O 9198
9225 LET q=1: LET z$="powrot z z
apisem? t=1:": GO SUB 9190: IF q
=0 THEN GO SUB 9198: GO TO 9025
9226 LET q=nt: IF q<=it THEN LET
 t=q: GO SUB 8825
9227 GO TO 9230
9228 LET q=1: LET z$="powrot bez
 zapisu? t=1:": GO SUB 9190: IF
q=0 THEN GO SUB 9198: GO TO 9025
9229 RETURN : REM
       zapis i kompres. bufora
9230 LET n=1: LET l=1: FOR m=1 T
O is: LET n$=n$( TO 8+2*m)+FN z$
(n+5*is+12)+n$(9+2*m TO ): LET k
```

```
=0
9231 LET h=CODE e$(l): LET h=h+(
h<=127)+127*(h>127): LET e$(n)=C
HR$ h: LET l=l+1: FOR i=2 TO ik:
 LET o=CODE e$(l): IF o>126 AND
o<>129 THEN LET o=127
9232 IF h=127 AND o=127 OR o=129
 THEN LET k=k+1: GO TO 9240
9233 IF k=0 THEN LET n=n+1: LET
e$(n)=CHR$ o: LET h=o: GO TO 924
0
9234 LET e$(n)=CHR$ (128+h): LET
 h=o: LET n=n+2: IF k<128 THEN L
ET e$(n-1)=CHR$ k: GO TO 9239
9235 LET e$(n-1)=CHR$ l: LET e$(
n)=CHR$ 128: LET n=n+1
9239 LET e$(n)=CHR$ o: LET k=0
9240 LET l=l+1: NEXT i: IF k>0 T
HEN LET e$(n)=CHR$ (128+h): LET
n=n+1: LET e$(n)=CHR$ k: IF k>=1
28 THEN LET n=n+1: LET e$(n)=CHR
$ 128
9241 LET n=n+1: NEXT m: LET e$=e
$(1 TO n-1): LET n$=CHR$ pt+CHR$
 (ik-1)+CHR$ is+n$
9242 LET it=CODE m$(1)
9243 IF q>it THEN FOR l=2 TO 2*i
t STEP 2: LET m$(l TO l+1)=FN z$
(FN z(l)+2): NEXT l: LET m$=m$(
TO 2*it+1)+FN z$(LEN m$+3)+m$(2*
it+2 TO )+n$+e$: GO TO 9246
9244 LET n=FN z(2*q): LET k=5*is
+13+LEN e$: FOR l=2 TO 2*it STEP
 2: LET m$(l TO l+1)=FN z$(FN z(
l)+k*(l>=2*q)+2): NEXT l
9245 LET m$=m$( TO 2*q-1)+FN z$(
n+2)+m$(2*q TO n-1)+n$+e$+m$(n T
O )
9246 LET it=it+1: LET m$(1)=CHR$
 it: LET w$=w$( TO q-1)+CHR$ 0+w
$(q TO ): RETURN
9255 REM utworzenie bufora taktu
9256 LET k=FN z(2*q): LET pt=COD
E m$(k): LET ik=CODE m$(k+1)+1:
LET is=CODE m$(k+2): LET n$=m$(k
+3 TO k+12)+m$(13+2*is+k TO 12+5
*is+k): LET n=k+13: LET n=k+FN z
(n): LET e$="": FOR l=1 TO is*ik
: LET i=CODE m$(n): LET e$=e$+CH
R$ (i-128*(i>127)+(i=127 OR i=25
5)): IF i<128 THEN : GO TO 9259
9257 LET n=n+1: LET m=CODE m$(n)
: LET l=l+m: FOR j=1 TO m: LET e
$=e$+(CHR$ 129 AND i<255)+(CHR$
128 AND i=255): NEXT j: IF m>127
 THEN LET n=n+1
9259 LET n=n+1: NEXT l: RETURN
9260 LET md=255: LET z$="podziel
. metrum d.="+STR$ pt+":": GO SU
B 9190: IF q=0 THEN RETURN
9261 LET pt=q: RETURN
9265 LET z$="nr pierwsz. kroku":
 LET md=ik: GO SUB 9190: IF q=0
THEN GO TO 9025
9266 GO SUB 9198: GO SUB 8975: I
F q$="" THEN GO TO 9025
9267 IF k<=it THEN GO TO 9266
9268 IF n$(1 TO 10)=q$ THEN GO S
UB 8979: GO TO 9266
9269 IF q=1 THEN LET n$(1 TO 9)=
q$: GO TO 9226
9270 LET k=q: LET q=ik-k+1: LET
z$="": LET l=k-1: LET n=(ik-q)*i
s: FOR m=l TO n STEP ik-q: LET z
$=z$+e$(m+1 TO q+m): LET e$=e$(
TO m)+e$(q+m+1 TO ): NEXT m: LET
 ik=ik-q: LET zik=q
9271 LET x$=n$(11 TO ): GO SUB 9
226: LET e$=z$: LET n$=q$+x$: LE
T ik=zik: LET nt=it+1: GO TO 922
6
9337 LET m=m+cis+12*((2-VAL x$)*
n+(3+VAL x$)*NOT n): IF m>=127 T
HEN GO SUB 9326: LET m=128
9342 LET m=127*(x$="0")+1: RETUR
N
```

klawisz <a> — wywołanie pola aktywności

pole składa się z trzech kolumn: taktów, instrumentów i źródła

b) uaktywniony element jest wypisywany na ekranie w trybie inwersyjnym

c) uaktywniać możemy niezależnie takty, instrumenty i ich źródła

d) po polu poruszamy się kursorem

funkcje pola aktywności: klawisz <6> — kursor w dół

klawisz <7> — kursor w górę

klawisz <5> — kursor w lewo

klawisz <8> — kursor w prawo

<SPACE> — włączanie/wyłączanie elementu, przy którym znajduje się kursor

klawisz <k> — kasowanie aktywności widocznionego na ekranie fragmentu kolumny, przy której znajduje się kursor

klawisz <u> — jak <k>, ale ustawienie aktywności

klawisz <K> — kasowanie aktywności całej kolumny, przy której ustawiony jest kursor

klawisz <U> — jak <K>, ale ustawienie aktywności

klawisz <c> — dotyczy całego zbioru. parametr: 1 — kasowanie

2 — ustawianie

uwaga: kolumna „źródła" wyświetla źródła instrumentu, po którego nazwie jest wyświetlona gwiazdka

uwaga: funkcje <w>, <t>, <T>, <k>, <d>, <l>, <a> są wykonywane pod warunkiem istnienia przynajmniej jednego taktu

uwaga: funkcje <t>, <T>, <k> jako drugi parametr mają wybór trybu funkcji: 1 — funkcja wykonywana jest na poszczególnych ścieżkach pod warunkiem zgodności taktu, instrumentu i źródła

2 — funkcja dotyczy całego taktu bez względu na zadeklarowane instrumenty i ich ścieżki

INFORMACJE DLA CIEKAWSKICH

zapis taktu w pamięci, procedura kompresji:

Treść taktu aktualnie edytowanego (podczas pracy edytora taktu) znajduje się w zmiennej tekstowej **e$**. Na podstawie opisu z poprzedniej części wynika, że jest to zapis pochłaniający dużą ilość pamięci, chociaż wygodny od strony edycyjnej.

Dlatego w momencie zapisu taktu do pamięci (czyli wprowadzenia go do zmiennej tekstowej **m$**) ulega on kompresji.

Na takie postępowanie zdecydowaliśmy się, aby zaoszczędzić jak najwięcej pamięci i w ten sposób umożliwić wpisanie dużej ilości taktów. Duże marnotrawstwo powodują dźwięki przedłużone, ciągi pauz, czy całe puste ścieżki.

Mechanizm jest prosty: jeżeli dźwięk trwa dłużej niż jeden krok, to zostaje zapisana wysokość dźwięku i informacja, na ile kroków jest on przedłużony. Jako informację o przedłużeniu dźwięku wykorzystujemy 8 (najstarszy) bit słowa wysokości dźwięku (nie jest on używany przy zapisie wysokości dźwięku).

Format zapisu przedstawia rysunek 1:

Przykładowe zapisy przedstawiają rysunki 2 i 3

uwaga: jak wspomnieliśmy we wstępie, w prezentowanej wersji nie przewidujemy stosowania parametrów brzmieniowych i dlatego ich ilość stale wynosi 0

pamięć taktów:

Zbiór taktów jest trzymany w zmiennej tekstowej **m$**

it — ilość taktów

pt — podzielnik tempa

ik — 1 — zmniejszona o jeden ilość kroków

is — ilość ścieżek

Krzysztof PÓŹNIAK
Michał ROLLINGER
Jarosław ZIEMBICKI

INFORMATYKA W SZKOLE

Jak uczyć informatyki w szkole? Ten problem budzi gorące namiętności informatyków, nauczycieli i uczniów. I nie jest to spór teoretyczny, coraz częściej bowiem w szkolnych pracowniach pojawiają się komputery. Od dwóch lat mamy nawet program nauczania tego uzupełniającego przedmiotu, jakim jest informatyka. Ten program budzi jednak kontrowersje, na przykład, że jego nierozsądna realizacja doprowadzić może do uczynienia z informatyki „sztuki dla sztuki" pozbawiając w ten sposób szans na wpisanie tej dyscypliny w codzienność.

Jaki powinien być zatem program nauczania tego przedmiotu? A może nie należy tej dyscypliny traktować zbyt samodzielnie i stworzyć raczej program jej zastosowania w nauczaniu fizyki, matematyki, chemii...?

Poszukajmy odpowiedzi na te pytania, a od początku roku szkolnego towarzyszyć będziemy wszystkim uczniom w poznawaniu właśnie informatyki.

Redakcja

Rola komputera w edukacji

Uchwała Prezydium Rządu z 5 listopada 1984 r. zakłada wprowadzenie do wszystkich szkół średnich do roku 1990 mikrokomputerów wraz z oprzyrządowaniem. Za pomocą tych urządzeń możliwe jest kształcenie informatyczne naszej młodzieży oraz wspomaganie nauczania innych przedmiotów. Komputery zastosowane właściwie są w szkole cennymi środkami dydaktycznymi, rozwijającymi aktywność umysłową uczniów, wspomagają one i uatrakcyjniają proces dydaktyczny. By komputer mógł dobrze spełniać swoją funkcję w szkole, nauczyciel musi być zaopatrzony w dobre, sprawdzone programy edukacyjne. Obserwujemy obecnie olbrzymie, jak zawsze, zaangażowanie polskich nauczycieli w rozwój oświaty przez tworzenie świetnych programów edukacyjnych. Bardzo istotną sprawą jest ich upowszechnienie, by zainteresowani nauczyciele i uczniowie mogli wspólnie z nich korzystać. W tej dziedzinie przychodzi nam z dużą pomocą czasopismo wojskowe „Żołnierz Wolności", wydając specjalny dodatek komputerowy „IKS", który na swoich gościnnych łamach umieszcza również programy edukacyjne. Nie czynią tego niestety inne czasopisma komputerowe. Najliczniejszą grupę czytelników tych czasopism stanowi młodzież, a młodzież uczęszcza do szkół i poszukuje tych programów. Nieocenione usługi w upowszechnianiu programów edukacyjnych wyświadcza nam Polskie Radio, zwłaszcza audycje Rozgłośni Harcerskiej emitowane w 4 programie. Wielu uczniów ma swoje własne komputery i oni przynoszą do szkoły, za moją namową, programy edukacyjne zarejestrowane na kasetach magnetofonowych, które ukazały się w prasie (niestety, ciągle jeszcze za mało) lub nagrane bezpośrednio za pomocą magnetofonu z radia. Jeśli program mi się podoba, przegrywam go na swoją kasetę lub dyskietkę; w ten sposób je kolekcjonuję. Nauczyciele, wbrew mylnie panującym opiniom, nie mają dużo wolnego czasu, dlatego uważam, że w zdobywaniu tak bardzo potrzebnych programów edukacyjnych powinni nam pomagać nasi uczniowie, dla których będzie to połączenie przyjemnego z pożytecznym. Również wykorzystany jest do tego celu szkolny komputer. Jeśli wszyscy nauczyciele, zajmujący się wprowadzeniem komputerów do dydaktyki, będą zachęcać dostępnymi środkami swoich uczniów do kolekcjonowania tak potrzebnych programów, to żaden egzemplarz „IKS-a" nie będzie leżał w kiosku, w oczekiwaniu na swego czytelnika. Istnieją szkoły, które mają komputery, ale z nich nie korzystają z braku programów edukacyjnych.

Kształcenie informatyczne zależy od postaw i przygotowania nauczycieli. Dla wielu z nich, szczególnie starszych stażem, komputer na lekcji, to tajemniczy przyrząd, pochłaniający nauczycielowi wiele czasu. Każda nowość ma swoich zwolenników i przeciwników. Komputer wymaga od nauczyciela zapoznania się z nową dziedziną wiedzy, co pochłania wiele czasu. Sam również pokonywałem barierę psychologiczną i lęk przed nieznanym. Przecież 25 lat uczyłem bez komputera i wcale nie odczuwałem jego braku. Mamy w szkołach wielu zapaleńców, którzy wciągają do stosowania komputerów innych nauczycieli. Próby pisania własnych programów przez nauczycieli, to zjawisko naturalne. Z własnego doświadczenia wiem, i to musimy sobie uświadomić, że wiedzę informatyczną o wiele łatwiej jest opanować ludziom młodym. Dlatego nasi uczniowie powinni szybko prześcignąć swoich mistrzów w umiejętności posługiwania się komputerem i pisaniu programów. Dla nas nauczycieli, stanowić to będzie źródło satysfakcji z dobrze wykonanej pracy. Pisanie programu jest bardzo czasochłonne, dlatego nauczyciel może być autorem scenariusza, polecając pisanie samego programu uczniowi. W technikach napisanie programu edukacyjnego może być tematem pracy końcowej z przygotowania zawodowego. Uważam, że programy edukacyjne nie powinny wychodzić poza już i tak przeciążone programy nauczania poszczególnych przedmiotów. W czasopiśmie „Mikroclan", wydawanym na pięknym papierze, znalazłem program na Spectrum „W pracowni fizycznej", traktujący o drganiach, które nie występują w żadnym programie szkolnym, ani wyższej uczelni i nie występują... w przyrodzie, jedynie w wyobraźni programisty. Komputer rysuje wszystko, nawet rzeczy, które nie istnieją, jeśli jest źle zaprogramowany. Jest on szybki i dokładny, ale nie potrafi myśleć. Myślenie jest domeną nauczyciela-programisty. Komputer może być niewłaściwie wykorzystany w dydaktyce. Sprowadza on wtedy ucznia do roli niemyślącego wykonawcy, odbierając mu inicjatywę i pomysłowość.

W pierwszym szeregu entuzjastów wprowadzenia komputerów do szkół są nauczyciele fizyki. Są oni zainteresowani każdą możliwością wprowadzenia nowych technik do dydaktyki. Komputer na lekcjach fizyki ma szerokie pole działania. Może być zastosowany do różnych celów. Służy do modelowania zjawisk fizycznych, analizując je numerycznie, do szybkich obliczeń i przeprowadzania analizy wyników doświadczeń, dając dużą oszczędność czasu na lekcji. Wyniki pomiarów przedstawia łatwo na wykresie o zmiennej skali w celu przejrzystszego odwzorowania badanej zależności. Za pomocą komputera można symulować na ekranie różne doświadczenia lub działanie urządzeń niedostępnych w szkole, np. akceleratory, spektograf masowy, reaktor jądrowy. Nie wolno jednak zastąpić symulacją komputerową rzeczywistych eksperymentów, przeprowadzanych przyrządami fizycznymi na lekcji. Komputer może pomóc w uzupełnieniu i utrwaleniu przeprowadzonych wcześniej doświadczeń. Symuluje on doświadczenia w warunkach idealnych (bez tarcia) lub ekstremalnych. Za pomocą komputera, dzięki jego możliwościom graficznym, szybkości i niezawodności, można uzyskać duży postęp w nauczaniu fizyki. Najefektywniejszym wykorzystaniem komputera do badania zjawisk fizycznych jest sprzężenie go z obserwowanym obiektem. Komputer spełnia wtedy dwie funkcje: jest przyrządem pomiarowym i analizatorem otrzymanych wyników. Za pomocą komputera nauczyciel może sprawdzić wiadomości uczniów, używając do tego testu z omawianego działu. Wystawianie uczniom ocen przez komputer ożywia lekcje powtórzeniowe, najmniej lubiane przez uczniów i nauczycieli. Przy użyciu programu z instrukcją DATA sprawdzam wyniki obliczeń, dokonanych przez uczniów, jeśli każdy z nich otrzymał inne dane, co uniemożliwia odpisywanie od sąsiada.

Program edukacyjny powinien, moim zdaniem, być tak napisany, by zawierał w odpowiednim miejscu wszystkie wskazówki i polecenia

dla obsługującego go, by używać go mógł każdy nauczyciel lub uczeń, nawet nie znający zasad programowania. Dodatkowy tekst w publikacji prasowej jest wtedy zbędny, nie zostaje on przecież umieszczony na taśmie magnetofonowej lub dyskietce. Jeśli program przewiduje wprowadzenie do komputera pewnych danych za pomocą instrukcji INPUT, należy wtedy koniecznie w programie umieścić tekst objaśniający, w jakich granicach winny być te dane, by symulacja komputerowa zmieściła się na ekranie. Sposób przekazywania wiedzy za pomocą komputera jest efektowny, bardziej koncentruje uwagę ucznia. Po kilkuletnim stosowaniu na lekcjach komputerów MERITUM (szkolny) oraz SPECTRUM i ATARI (własne), zauważyłem, że uzyskuję zdecydowanie lepsze wyniki nauczania. Przeprowadziłem kilkakrotnie następujący eksperyment pedagogiczny: ucząc w 4 równoległych klasach, w 2 z nich utrwaliłem tematy z optyki geometrycznej — obrazy w zwierciadłach i soczewkach, po narysowaniu ich na tablicy i pokazaniu na ławie optycznej, za pomocą symulacji komputerowej. Wyniki przeprowadzanych sprawdzianów w tych klasach były dużo lepsze. Gdy na lekcji używany jest komputer, uczniowie nie słyszą często dzwonka, ogłaszającego oczekiwaną zwykle przerwę, a do opuszczania pracowni muszę ich zmuszać.

Prawdziwą przyjemność w stosowaniu komputera na lekcji odczułem jednak dopiero po zastosowaniu stacji dysków do ATARI (własna). Rodzice powinni zdać sobie sprawę z tego, że komputer w domu może w edukacji dziecka lub młodzieńca wyrządzić wiele szkody, jeśli używany będzie jedynie do zabawy. Kiedy uczeń odrobi lekcje, gdy całymi godzinami będzie ślęczał przed ekranem telewizora, pochłonięty grami komputerowymi.

Głównym celem obecności komputera w szkole jest przygotowanie do życia i działania w nowoczesnym społeczeństwie. Pamiętajmy o tym, że ci uczniowie, którzy dziś zasiadają w ławkach naszych szkół, będą pracować w XXI wieku, a ich narzędziem pracy z pewnością często będzie komputer. Brak umiejętności w posługiwaniu się komputerem w niedalekiej przyszłości będzie tak dotkliwy, jak obecnie brak umiejętności pisania i czytania.

mgr Jan GOLLA
nauczyciel fizyki
w Zespole Szkół Chemiczno-Mechanicznych
w Gliwicach

Jak uczyć informatyki

Zważywszy na duże zainteresowanie informatyką w szkole z jednej strony, a niewielkie doświadczenie w stosowaniu komputerów (jako przedmiotu i narzędzia) w klasach z drugiej, proponujemy wzajemną pomoc w tym zakresie. Zapraszamy przede wszystkim nauczycieli, którzy prowadzą podstawy informatyki w szkole średniej lub uczelni humanistycznej. Interesują nas zarówno rzeczywiste programy zajęć, jak i konkretne rozwiązania metodyczne nauczania informatyki — szczególnie programy wykorzystujące komputer, jako narzędzie i jako przedmiot nauczania.

Najlepsze propozycje zostaną opublikowane i godziwie nagrodzone:

I nagroda 50 tys. zł
II nagroda 30 tys. zł
III nagroda 20 tys. zł

PROGRAM LICEUM OGÓLN
(profil podstawowy i matem
ELEMENTY INFO
(uzupełniający przedm

CELE KSZTAŁCENIA I WYCHOWANIA

Zasadniczym celem zajęć z elementów informatyki jest nauczanie metod rozwiązywania za pomocą komputera prostych problemów na poziomie programu liceum ogólnokształcącego, dostosowanych do wiedzy i umiejętności uczniów. W trakcie tych zajęć uczniowie powinni poznać podstawy programowania oraz zdobyć praktyczne umiejętności posługiwania się szkolnym sprzętem informatycznym i jego oprogramowaniem. Dodatkowym celem nauczania elementów informatyki jest stworzenie warunków sprzyjających korzystaniu z komputera podczas uczenia się matematyki, fizyki, chemii, pracy-techniki i innych przedmiotów.

TREŚCI KSZTAŁCENIA I WYCHOWANIA

WIADOMOŚCI

1. Obsługa mikrokomputera

Zapoznanie się ze szkolnym mikrokomputerem. Konfiguracja mikrokomputera. Prawidłowe połączenie części składowych. Podstawowe zasady obsługi mikrokomputera. Włączenie. Układ klawiatury. Monitor ekranowy. Wskaźnik na ekranie. Wprowadzenie komend i tekstów. Kasowanie pomyłek. Rozumienie typowych odpowiedzi i reakcji mikrokomputera. Wprowadzanie i uruchamianie programu z pamięci zewnętrznej. Przerywanie wykonywania programu. Możliwe nieprawidłowości pracy mikrokomputera i sposoby postępowania w razie ich wystąpienia.

2. Praktyczne zastosowanie mikrokomputera

Poznanie działania wybranych programów, pokazujących typowe zastosowanie mikrokomputera; symulowanie procesów lub działania urządzeń, układanie i przekształcanie obrazów na ekranie, zbieranie i przechowywanie informacji, wykonywanie prostych obliczeń i zestawień rachunko-wych, zapisywanie i redagowan itp. Pokazanie uczniom sposobów wania się takimi programami; wpływania na przebieg symulowa cesów, doboru parametrów symu urządzeń, komponowanie obrazów kształtu formularza w zestawien chunkowych, dobieranie formy nego tekstu.

Analiza działania wybranych pro Określanie, jakie problemy moż zywać za pomocą danego progra jak wygląda proces rozwiązywania mu. Zbadanie, czy da się wyodręb problemy, które są rozwiązywane osobno oraz czy da się ustalić, jak ków cząstkowych jest składany wynik rozwiązania całego problemu.

3. Tworzenie rysunków na

Podstawowe komendy graficzne giwanie się wskaźnikiem (znaczn ekranie. Układ współrzędnych ek sowanie odcinków, punktów i łuk tworzenie z nich prostych rysun ekranie. Operowanie kolorami rysu

4. Procedury

Programowanie czynności, któ być wielokrotnie wykonywane. T procedur. Parametry procedur i po nie się nimi. Procedury z obliczan cią (funkcje, operacje). Wyrażenia dy posługiwania się funkcjami i wy przy wykonywaniu obliczeń. Przeka wartości między procedurami; zmie balne i lokalne. Współpraca proce cedury rekurencyjne. Procedura re ca (edytor); posługiwanie się nią prawianiu procedur. Zapisywanie na stałe w pamięci zewnętrznej. Wc nie procedur z pamięci zewnętrzne krokomputera w celu ich wykonan prawienia.

5. Styl programowania

Metoda zstępująca tworzenia procedur: od ogólnych założeń i pr zarządzającej wykonaniem całości cedur zapewniających wykonan ności szczegółowych.

Opanowywanie i utrwalanie p wych nawyków przy pisaniu proced nowywanie i utrwalanie podsta umiejętności programowania stru go, w szczególności: czytelnego za

A W SZKOLE

...OKSZTAŁCĄCEGO ...atyczno-fizyczny)

...RMATYKI ...ot nauczania)

...e tekstów ...w posługi- ...sposobów ...nych pro- ...lowanych ...w, doboru ...niach ra- ...edagowa-

...ogramów. ...a rozwią- ...amu oraz ...a proble- ...bnić pod- ...kolejno i ...ak z wyni- ...stateczny ...

...ekranie

...ne Postu- ...ikiem) na ...ranu. Ry- ...ków oraz ...nków na ...nku i tła.

...óre mają ...Tworzenie ...osługiwa- ...ą wartoś- ... Przykła- ...rażeniami ...azywanie ...enne glo- ...edur Pro- ...redagują- ...przy pd- ...procedur ...czytywa- ...ej do mi- ...a lub po-

... zestawu ...rocedury ...i do pro- ...e czyn-

...prawidło- ...dur. Opa- ...awowych ...kturalne- ...apisywa-

nia algorytmów, stosowania czytelnych nazw procedur i zmiennych oraz dobierania przejrzystych schematów współdziałania procedur. Używania w procedurach konstrukcji strukturalnych: powtarzania, instrukcji warunkowych i rekurencji.

Niezmienniki procedur: wartości, które są takie same przed wykonaniem procedury i po jej wykonaniu. Zależność rezultatów wykonania procedury od wartości początkowych zmiennych, mających wpływ na jej wykonanie. Przykłady niezmienników: współrzędne wskaźnika na ekranie, kierunek wskaźnika, kolor kreski. Tworzenie procedur tak, by pewne wielkości były z zasady niezmiennikami; korzyści wynikające z takiego podejścia przy planowaniu współpracy procedur. Pisanie przez uczniów własnych zestawów procedur do tworzenia złożonych rysunków

6. Nieelementarne metody grafiki

Tworzenie procedur graficznych tak, aby ułatwić przekształcenia i modyfikacje rysunków, na przykład zmianę skali lub położenia elementów rysunku, rysowanie scen trójwymiarowych z różnych punktów widzenia. Zastosowanie list do opisywania elementów rysunku i operacji na listach umożliwiających programowanie przesunięć, obrotów i zmian skali.

7. Działania na tekstach

Działania na zdaniach, przedstawianych jako ciąg wyrazów, zapisany w formie listy. Operacje składania wyrazów w zdanie (listę). Wydobywanie ze zdania wyrazów: pierwszego, ostatniego, dowolnie wybranego. Przykłady procedur działających na listach i zasady tworzenia takich procedur. Użycie rekurencji przy przeglądaniu lub modyfikowaniu list. Tworzenie losowo generowanych zdań i danego zasobu wyrazów. Programowanie prostych, sensownych odpowiedzi na zdania wprowadzane z klawiatury. Projektowanie i programowanie prostych dialogów z komputerem, opartych na zasadzie rozpoznawania tekstów. Działania na słowach, rozumianych jako ciągi znaków.

8. Ćwiczenia samodzielne

Programowanie rozwiązań wybranych przez uczniów problemów. Utrwalanie i pogłębianie przyswojonych wiadomości. Analiza i ocena uzyskanych rozwiązań i stosowanych metod

ORIENTACYJNY PRZYDZIAŁ GODZIN

1. Obsługa mikrokomputera	**2/ 3%**
2. Praktyczne zastosowania mikrokomputera	**6/ 8%**
3. Tworzenie rysunków na ekranie	**4/ 5%**
4. Procedury	**12/ 16%**
5. Styl programowania	**12/ 16%**
6. Nieelementarne metody grafiki	**12/ 16%**
7. Działania na tekstach	**12/ 16%**
8. Ćwiczenia samodzielne	**15/ 20%**
Razem godzin	**75/100%**

PROGRAM LICEUM OGÓLNOKSZTAŁCĄCEGO — profil podstawowy i matematyczno-fizyczny (ELEMENT INFORMATYKI — uzupełniający przedmiot nauczania), to tytuł owocu niewątpliwie ogromnej pracy dziewięcioosobowego zespołu autorów. Czytanie tego opracowania potwierdza ogromną wiedzę jego twórców. Wydaje się jednak, że przeważyły akademickie (politechniczne?) doświadczenia autorów. Proponowany materiał bowiem, przy znacznym wysiłku nauczycieli i uczących się (najlepiej studentów), można opanować zapewne nie szybciej niż w 200—250 godzin, a nie w zakładanym czasie 75 godzin

Skąd ta opinia? Czy aby rzeczywiście „zasadniczym celem zajęć z elementów informatyki jest nauczanie metod rozwiązywania za pomocą komputera prostych problemów na poziomie programu liceum ogólnokształcącego, dostosowanych do wiedzy i umiejętności uczniów"?[1] Wydaje się, że stwierdzenie to stwarza niebezpieczeństwo uznania, iż celem lekcji informatyki jest nauczanie ograniczonych możliwości komputerów. W każdym razie uczniowie mogą odnieść wrażenie, że komputery służą do rozwiązywania prostych zadań. PROGRAM nie przewiduje prezentacji rzeczywistych, możliwych i przyszłych zastosowań informatyki Szkoda także, że nie zostało zdefiniowane znaczenie dostosowania „do wiedzy i umiejętności uczniów". Czym mierzyć owo „dostosowanie" — klasą? Wielkością miasta? Czy kwalifikacjami nauczyciela WF?

Jako inny cel kształcenia i **wychowania** (podkr. wł.) proponowana jest nauka programowania (o języku nie wspomniano, na wszelki wypadek, ani słowem). Rodzi to kolejną wątpliwość: „nauka programowania"? Kto potrafi nauczyć programowania (bo po co nauka, skoro niemożliwe jest nauczenie?) w szkole średniej w 75 godzin?

Podczas definiowania celów kształcenia i wychowania — wychowanie pozostało w tytule. Zapomniano o emocjach, „sympatiach", granicach, o których należy pamiętać poznając komputery Podobnie jeśli idzie o treści kształcenia i wychowania (?) — wychowanie pozostało w tytule

Praktyczne zastosowanie komputerów, które (wg PROGRAMU) powinny być zrozumiałe (z zachowaniem wymagań dydaktyki) przedstawione uczniom, to „tylko"
— symulowanie procesów działań urządzeń,
— układanie i przekształcanie obrazów na ekranie,
— zbieranie i przechowywanie informacji,
— wykonywanie prostych obliczeń i zestawień rachunkowych, m.in. dobór kształtu formularza w zestawieniach rachunkowych,
— zapisywanie i redagowanie tekstów,
— analiza działania wybranych programów,
— itp.!

Jest to zaledwie jeden z 8 (ośmiu) punktów treści kształcenia i wychowania przewidzianych na 75 godzin. Darując sobie analizę pozostałych punktów, warto zwrócić uwagę na „Zagadnienia związane z realizacją programu". Podkreślono tu bardzo ważny problem edukacji komputerowej w ogóle — przygotowanie nauczycieli. Ale już za chwilę napisano, że: „komputer, używany w szkole, powinien mieć... pamięć zewnętrzną na nośnikach magnetycznych lub możliwość przyłączenia magnetofonu kasetowego" — czyżby magnetofon kasetowy był pamięcią z optycznym nośnikiem informacji? Na marginesie tej uwagi, ciekawy jestem, czym kierowali się autorzy, pisząc o magnetofonie kasetowym, a nie o magnetofonie w ogóle — przecież wszędzie tam, gdzie pasuje magnetofon kasetowy, można wstawić szpulowy — prawda?

Trochę ironicznie brzmi niepokój o efekt — „zniechęcenie uczniów i nauczycieli do dalszych przedsięwzięć tego typu"

Autorzy PROGRAMU twierdzą, że „rozsądek nakazuje", by jednak program ten zweryfikować (a jednak!)... w odpowiednich szkołach wyposażonych między innymi w specjalistyczne laboratoria informatyczne Cóż z takiej weryfikacji, skoro jej wyniki mają być reprezentatywne dla wszystkich szkół (także tych bez laboratoriów)

Całkowicie się nie zgadzam z zaleceniem: „Należy unikać podawania wiadomości o charakterze teoretycznym, jak szczegóły bu-

dowy komputerów. Jak zatem wytłumaczyć uczniom problem miniaturyzacji, szybkości, niezawodności, źródła gubienia informacji (RAM) i jej pamiętania (ROM), ekonomii eksploatacji sprzętu itp.?

Zalecana jest natomiast analiza przykładowych programów. Autorzy nie zaryzykowali jednak choćby jakiejkolwiek egzemplifikacji tej propozycji, np.: najlepiej, według naszych doświadczeń, nadaje się do tego celu analiza programu sortowania. (Wiele przykładów zawiera prasa informatyczna).

Niestety, lekturę PROGRAMU utrudnia także „hermetyczny język", np. „Uzyskane wiadomości powinny być niezwłocznie wykorzystane i utrwalone w pracy na komputerze" — to jest PROGRAM, nie brudnopis.

Natomiast przykłady zawarte w PROGRAMIE wyraźnie nobilizują poziom licealistów — „przykład algorytmu jedzenia kaszki:

jedz kaszkę (czynność do wykonania)
jeśli talerz jest pusty, to koniec (warunek zakończenia)
zjedz łyżkę kaszki (czynność podstawowa każdego kroku)
jedz kaszkę (rekurencja)" (s. 12)

Doświadczenie uczy, że każdy początek jest trudny — nowe rodzi się w bólach (bolesne było czytanie PROGRAMU) — niemniej stanowi podstawę do tworzenia rzeczy innych, czasami lepszych. Stąd też w najbliższym numerze przedstawimy naszą propozycję PROGRAMU. Sprawdzoną w praktyce. Dlatego PROGRAM uzupełni „konspekt" na wszystkie zajęcia, które przewidziane są na około 60 godzin. Planujemy, że konspekt ten będzie przedstawiony w „odcinkach" co miesiąc — każda część będzie obejmowała sześciogodzinne zajęcia — wykłady lub ćwiczenia. Uzupełnieniem będą uwagi państwa oraz zespołu redakcji.

Włodzimierz GOGOŁEK

Co nas interesuje

(zajęcia z informatyki)

Sądzę, że potwierdzeniem naszego zainteresowania informatyką (jestem słuchaczem II roku Wojskowej Akademii Politycznej) były jedne z ostatnich zajęć z tego przedmiotu.

Celowym wydaje się podkreślenie oryginalnej ich formy. Słuchacze bowiem stając w roli wykładowcy starali się przez 5 minut zainteresować odbiorców dowolnie podjętą tematyką z dziedziny informatyki. Ze względu na ograniczenia czasowe głos zabrało 9 słuchaczy. Jako pierwszy „przejął pałeczkę" od wykładowcy kol. Kos. Tematem jego wystąpienia była prezentacja możliwości dysku twardego typu „Winchester". Referujący przytoczył parametry jednego z typów dysku, podkreślając jego dużą pojemność (360 MB) przy jednocześnie małych rozmiarach. Wspomniał o zastosowaniu nowych cienkich głowic manganowych, a także o cieńszej i wydajniejszej powłoce magnetycznej dysku. Te i inne zalety dysku plus niska cena jednostkowa w stosunku do jego pojemności stawiają firmę IBM w roli potentata w tej dziedzinie.

Z kolei głos zabrał kol. Dela prezentując sposoby wykorzystywania komputera w sferze teleinformacji. Mówił on między innymi o zastosowaniu mikrokomputera domowego, a także o możliwościach dialogu za pośrednictwem ekranów telewizyjnych. Ponadto przedstawił nowości brytyjskiej firmy PRESTER z dziedziny systemu teleinformatycznego.

Natomiast kol. Kosiński „przykuł" uwagę pozostałych słuchaczy problemem płytek krzemowych. Obecnie możemy mówić o toczącej się rywalizacji w dziedzinie konstruowania najbardziej doskonałych płytek krzemowych między Japonią (TOSHIBA) i Europą Zachodnią (SIEMENS-PHILIPS). Kol. Kosiński mówił także o problemach w zdobyciu surowca tj. krzemu, co powoduje nagły wzrost cen na pamięci komputera. Ponadto bardzo ciekawie wypowiedział się na temat promieniowania synktronowego tj. tzw. promieniowania X, które daje możliwość budowania pamięci RAM do 64 megabitów.

Następne 5 minut przeznaczone zostało dla kol. Górskiego. Podjął on temat dotyczący zastosowania komputera w rolnictwie. Współczesne rolnictwo, to także odpowiednio „uzbrojone" komputery, które informują człowieka, np. jakie składniki posiada próbka danej gleby i co należy na niej zasiać lub też jakie jest zapotrzebowanie na nawozy sztuczne itd. W sumie komputer staje się dziś wręcz nieodzownym doradcą dla rolnika.

Kolejnym referującym był kol. Rzeczycki. Swoje wystąpienie oparł o artykuł z tygodnika „NEWSWEEK" dotyczący przyszłościowego zastosowania komputera. Mówił on o współczesnym trendzie doskonalenia komputera na wzór mózgu człowieka, a także o jego zastosowaniu na rzecz przewidywania ludzkich działań. Warto jednak pamiętać, że mózg człowieka posiada ok. 100 mln połączeń między neuronami, natomiast współczesny komputer ma tylko 100 połączeń, a więc tyle co mózg mrówki. Jednakże generalnym celem jest zminimalizowanie roli programisty na rzecz samodzielności maszyny. Negatywnym natomiast jest fakt, iż najnowsze zdobycze w tej dziedzinie znajdują swoje zastosowanie przede wszystkim w przemyśle zbrojeniowym.

Kol. Kasprzak natomiast zabrał głos w kwestii zastosowania procesora i pamięci w komputerze. Wystąpienie to było oparte o najnowsze dane w tej dziedzinie. Wspomniano między innymi o Conection Machine, a więc o komputerze wyposażonym w wiele procesorów. Możliwość zastosowania jak największej liczby procesorów w komputerze ma na celu jak najszybsze i dokładne dokonanie określonej operacji (np. w dziedzinie grafiki). Programista komunikuje się z takim komputerem za pośrednictwem tradycyjnego komputera, korzystając ze standardowego języka programowania.

Możliwościami zastosowania komputera w dziedzinie marynistyki zajął się kol. Sykulak — oficer marynarki wojennej. Przedstawiona przy okazji plansza zobrazowała, iż każdy współczesny statek (okręt), to potencjalne „siedlisko" komputerów. Zastosowanie ich zaś dotyczy: dowodzenia statkiem czy okrętem, jego bezpieczeństwa, nawigacji i eksploatacji, w tym optymalizacji zużycia paliwa. W tym wypadku komputer okazuje się wielkim sprzymierzeńcem człowieka.

W dalszej części zajęć ciekawe wystąpienie zaprezentował kol. Tessikowski. Poruszył on kwestię możliwości wykorzystania komputera do gry w szachy. Już w końcu lat pięćdziesiątych pojawiły się pierwsze programy tego typu dla maszyn cyfrowych. Faktem jest, iż właśnie „królewska gra" okazała się doskonałym polem do badań nad sztuczną inteligencją. „Elektroniczny szachista" jednak nie jest doskonały, nie potrafi wygrać z arcymistrzami tej gry. Nie oznacza to jednak, że tak musi być, gdyż wraz z postępem w dziedzinie elektroniki komputer staje się coraz bardziej znaczący, a jego możliwości nie są dzisiaj jeszcze w pełni znane.

I wreszcie jako ostatni zabrał głos kol. Minkina. Skoro zajęcia odbywały się w Wojskowej Akademii Politycznej, nie można było pominąć tematyki militarnej, a ściślej wykorzystania komputera w siłach zbrojnych. Kol. Minkina zajął się kwestią komputeryzacji w armii Stanów Zjednoczonych. Przedstawił on słuchaczom między innymi możliwości systemu taktyczno-operacyjnego (Tactical Operation System) przeznaczonego do uzyskiwania, diagnozowania i przekazywania informacji. System ten może być wykorzystywany np. w kierowaniu ogniem artylerii, obronie powietrznej, kierowaniu tyłami czy też w dowodzeniu taktycznym. Znamienny jest fakt, iż już dzisiaj mówi się w USA o tzw. elektronicznym polu walki. Zaś dowództwo SZ USA ma opracowanych 8 planów wojny światowej i 60 planów działań w sytuacjach krytycznych, gdzie dane mogą być przekazywane w ciągu kilku minut. I wszystko za sprawą komputera. Jest to jednak w tym wypadku przerażające.

Niewątpliwie, omawiane zajęcia osiągnęły swój zasadniczy cel, tzn. poszerzyły wiedzę, a także zainteresowania problematyką dotyczącą roli komputera w życiu człowieka. Wypowiedzi słuchaczy nie były, co prawda stricte fachowe, ale takie nie miały być. Były natomiast odpowiednio dostosowane do odbiorcy, dzięki czemu w swej wymowie, w tym wypadku, były bardziej wyraziste i zrozumiałe.

Sądzę, iż taka forma zajęć powinna być częściej stosowana, bowiem prowadzących i słuchających mobilizuje oraz bardziej przybliża tego typu wiedzę i w takim zakresie, niż wykład tradycyjny.

Zbigniew CYWIŃSKI

Zadanie konikowe

Danuta KWASIŻUR,
Mieczysław SKONIECZNY

Dzisiejszy odcinek cyklu rozwiązywania anegdot matematycznych chcemy poświęcić zadaniu konikowemu na szachownicy. Podanie mówi, że uczony indyjski SISSA-NASIR, twórca szachów, zażądał od władcy, oczarowanego nową grą, pozornie skromnej zapłaty. Chciał dostać tyle zboża, ile otrzyma, gdy poprzez 64 pola szachownicy podwajane będzie 1 ziarno położone na pierwszym polu. Jak łatwo odgadnąć, jest to suma pierwszych 64 wyrazów postępu geometrycznego, którego wyrazem początkowym jest 1, a ilorazem 2. Władca Indii, człowiek bardzo bogaty, nie był niestety w stanie uhonorować autora gry. Dlaczego — cierpliwi Czytelnicy mogą sami policzyć, ile powinno wynosić honorarium wynalazcy.

Zadanie konikowe jest sformułowane następująco:
jak można skokami konika szachowego obiec wszystkie 64 pola szachownicy, korzystając z każdego pola tylko 1 raz.

Zagadnieniami skoków konika po szachownicy zajmowało się wielu znanych matematyków. Stworzyli oni szereg metod rozwiązania przedstawionego wyżej zadania, z których najbardziej znane to.

1. Metoda Eulera (koniec wieku XVII).
2. Metoda ramowa Moora (1843 rok).
3. Metoda ramowa Moivre'a (początek XVIII wieku).
4. Metoda podziału na ćwiartki Rogeta (połowa XIX wieku).

Zainteresowanych szczegółami Czytelników odsyłamy do rozdziału 6 „Lilavati".

Liczba różnych rozwiązań tego zadania, co wydaje się wręcz niemożliwe, jest ogromna — przekracza 31 milionów. Będzie to chyba zachętą dla Czytelników, którzy zechcą wprowadzić program do pamięci mikrokomputera i spróbować swoich sił.

CHARAKTERYSTYKA PROGRAMU

Po zainicjowaniu pracy, program przedstawia na ekranie zasady gry w następującej postaci:

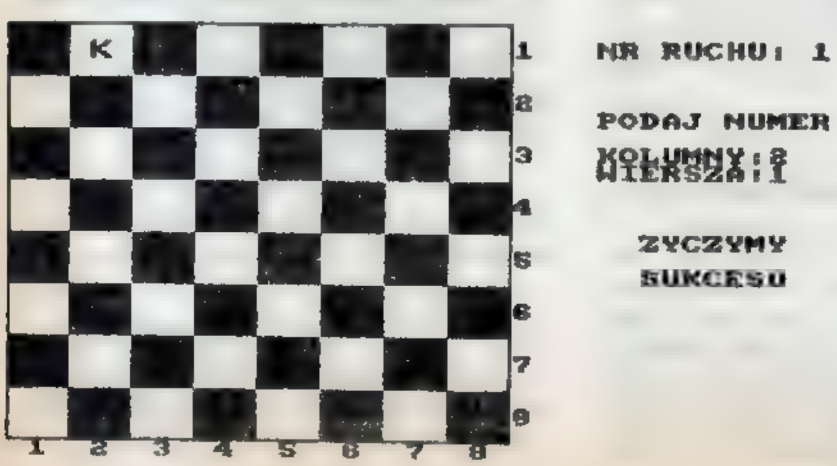
```
                         PROGRAM - ZADANIE KONIKOWE

Twoje zadanie polega na obejsciu wszystkich 64 pol szachownicy skokami konika
szachowego. Obowiazuje zasada, ze kazde pole moze byc wykorzystane tylko jeden
raz. Proba ponownego wykorzystania pola jest sygnalizowana komunikatem:
                        TO POLE JUZ WYKORZYSTANO"
Jezeli bedziesz probowal wykonac ruch, jakim nie porusza sie konik, to program
zareaguje komunikatem:
                        "TAK NIE PORUSZA SIE KONIK"
Za chwile ukaze sie szachownica. W oknie gornym jest podawany nr ruchu. Aby
przesunac konika musisz podac nr kolumny i wiersza kazdorazowo wciskajac po
liczbie klawisz ENTER ( okienko srodkowe ). Numeracja jest widoczna obok szacho-
wnicy ( na dole nr kolumny, a z prawej strony nr wiersza ).

Przed podaniem wspolrzednych pierwszego ruchu na szachownicy nie widac konika.
W nastepnych ruchach aktualna pozycja konika jest sygnalizowana litera K.

Pola juz raz wykorzystane sa oznaczane kwadratem w srodku pola
```

Następnie po naciśnięciu dowolnego klawisza pojawia się plansza szachownicy i użytkownik może rozpocząć grę. Sytuację po 1 ruchu przedstawia poniższy rysunek



Program opracowany w języku BASIC mikrokomputera AMSTRAD CPC 6128 ma następującą wersję źródłową:

CHARAKTERYSTYKA PODPROGRAMÓW

Do programu głównego należy dołączyć 3 podprogramy spełniające następujące funkcje

1. Czyszczenie 2 okienka i sprawdzanie poprawności wprowadzonych danych (nr wiersza i kolumny).

```
10 DIM s(8,8)
20 'Ekran informacyjny
30 MODE 2:INK 0,26:INK 1,0:PAPER 1:PEN 0:CLS:BORDER 0
40 WINDOW £1,1,80,1,3:PAPER £1,0:PEN £1,1:CLS£1
50 WINDOW £2.1,80,23,25:PAPER £2,0:PEN £2,1:CLS£2
60 LOCATE £1,26,2:PRINT£1."PROGRAM - ZADANIE KONIKOWE"
70 LOCATE 1,5:PRINT"Twoje zadanie polega na obejsciu wszystkich 64 pol
 szachownicy skokami konika   szachowego. Obowiazuje zasada, z
e kazde pole moze byc wykorzystane tylko jeden  raz. Proba ponownego w
ykorzystania pola jest sygnalizowana komunikatem:"
80 PRINT"                              "TO POLE JUZ WYKORZYSTANO"
                   Jezeli bedziesz probowal wykonac ruch, jaki
m nie porusza sie konik, to program  zareaguje komunikatem:"
90 PRINT"                              "TAK NIE PORUSZA SIE KONIK"
                   Za chwile ukaze sie szachownica. W oknie go
rnym jest podawany nr ruchu. Aby    przesunac konika musisz podac nr
kolumny i wiersza kazdorazowo wciskajac po"
100 PRINT"liczbie klawisz ENTER ( okienko srodkowe ). Numeracja jest w
idoczna obok szacho-wnicy ( na dole nr kolumny, a z prawej str
ony nr wiersza )."":PRINT
110 PRINT"Przed podaniem wspolrzednych pierwszego ruchu na szachownicy
 nie widac konika.  W nastepnych ruchach aktualna pozycja koni
ka jest sygnalizowana litera K."
120 PRINT:PRINT"Pola juz raz wykorzystane sa oznaczane kwadratem w sro
dku pola"
130 LOCATE £2,26,2:PRINT£2,"NACISNIJ DOWOLNY KLAWISZ"
140 CLEAR INPUT:A$=INKEY$
150 IF A$="" GOTO 140
160 'Deklaracja kolorow i okien w trybie 1
170 MODE 1:INK 0,26:INK 1.0:INK 2,6:INK 3,19
180 PAPER 2:BORDER 6:CLS
190 WINDOW £1.28,40,1,3:PEN £1,1:PAPER £1,0:CLS£1
200 WINDOW £2,28,40,5,10:PEN £2,1:PAPER £2,0:CLS£2
210 WINDOW £3,28,40,12,16:PEN £3,1:PAPER £3,0:CLS£3
220 'Zerowanie tablicy s
230 FOR i=1 TO 8
240 FOR j=1 TO 8:s(i,j)=0:NEXT:NEXT
250 ruch=1
260 'Rysowanie szachownicy
270 PAPER 0:PEN 1:FOR i=0 TO 6 STEP 2
280 FOR i1=6*i+1 TO 6*i+3
290 FOR i2=1 TO 4
300 PRINT CHR$(143);CHR$(143);CHR$(143);"   ";
310 NEXT i2:PRINT: NEXT i1
320 FOR i1=6*i+4 TO 6*i+6
330 FOR i2=1 TO 4
340 PRINT "   ";CHR$(143);CHR$(143);CHR$(143);
350 NEXT i2:PRINT: NEXT i1:NEXT i
360  PAPER 2:PRINT" 1  2  3  4  5  6  7  8"
370 FOR i=1 TO 8 :LOCATE 25,i*3-1:PRINT USING"£";i:NEXT
380 'Czynnosci wstepne do pierwszego ruchu
390  PEN 3:PEN £1,2:PEN £2,2:PEN £3,2
400 ob=0:GOSUB 720:suk=0
410 SYMBOL AFTER 255
420 SYMBOL 255,199,206,220,240,240,220,206,199
430 'Kolejny ruch
440 CLS £1:LOCATE £1,2,2
450 PRINT £1,"NR RUCHU:";USING"££";ruch
460 GOSUB 600
470 ob=2:GOSUB 720:IF blad=2 GOTO 440
480 ob=3:GOSUB 720
490 IF blad=3 GOTO 440
500 'Sprawdzenie koloru pola, na ktore ma skoczyc konik
510 IF (in=1 OR in=3 OR in=5 OR in=7) AND (jn=1 OR jn=3 OR jn=5 OR jn=
7) THEN PAPER ):GOTO 530
520 IF (in=2 OR in=4 OR in=6 OR in=8) AND (jn=2 OR jn=4 OR jn=6 OR jn=
8) THEN PAPER 1 ELSE PAPER 0
530 IF ruch<>1 THEN LOCATE (i-1)*3+2,(j-1)*3+2:PRINT CHR$(143)
540 LOCATE (in-1)*3+2,(jn-1)*3+2:PRINT CHR$(255)
550 ruch=ruch+1
560 i=in:j=jn:s(i,j)=1
```

```
570 GOSUB 1080
580 IF suk=1 GOTO 20
590.GOTO 440
```

2. Obsługa błędów

Umownie możliwym błędom nadano kody 0 — 3.
Wystąpienie każdego z tych kodów powoduje zmianę zawartości tekstu wyświetlanego w oknie nr 3. Interpretacja kodów jest następująca

0 — brak błędów (w oknie wyświetla się komunikat „Życzymy sukcesów",

1 — wprowadzono niedopuszczalny nr kolumny lub nr wiersza (komunikat: „Ten numer musi być od 1 — 8");

2 — próba wykonania niedozwolonego ruchu (komunikat: „Tak nie porusza się konik"),

3 — próba powtórnego wykorzystania pola (komunikat: „To pole już wykorzystano").

```
600 'Podprogram czyszczenia 2 okna i sprawdzania poprawnosci wprowadzanych liczb
610 CLS £2:LOCATE £2,2,2
620 PRINT £2,"PODAJ NUMER"
630 LOCATE £2,2,4
640 INPUT £2,"KOLUMNY:",in
650 ob=1:nr=in:GOSUB 720
660 IF blad=1 GOTO 610
670 LOCATE £2,2,5
680 INPUT £2,"WIERSZA:",jn
690 nr=jn:GOSUB 720
700 IF blad=1 GOTO 610
710 RETURN
```

3. Zakończenie gry w sytuacji poprawnego rozwiązania zadania.

```
720 'Podprogram obslugi bledow
730 blad=0
740 IF ob=0 GOTO 1040
750 'Obsluga bledu 1 rodzaju
760 IF ob<>1 GOTO 820
770 IF (nr>0 AND nr<9) GOTO 1040
780 CLS £3:blad=1
790 LOCATE £3,3,2:PRINT £3,"TEN NUMER"
800 LOCATE £3,3,3:PRINT £3,"MUSI BYC"
810 LOCATE £3,3,4:PRINT £3,"OD 1 DO 8":GOTO 1030
820 'Obsluga bledu 2 rodzaju
830 IF ob<>2 GOTO 970
840 IF ruch=1 GOTO 1040
850 IF in=i+1 AND jn=j-2 GOTO 1040
860 IF in=i+2 AND jn=j-1 GOTO 1040
870 IF in=i+2 AND jn=j+1 GOTO 1040
880 IF in=i+1 AND jn=j+2 GOTO 1040
890 IF in=i-1 AND jn=j+2 GOTO 1040
900 IF in=i-2 AND jn=j+1 GOTO 1040
910 IF in=i-2 AND jn=j-1 GOTO 1040
920 IF in=i-1 AND jn=j-2 GOTO 1040
930 CLS £3:blad=2
940 LOCATE £3,4,2:PRINT £3,"TAK NIE"
950 LOCATE £3,2,3:PRINT £3,"PORUSZA SIE"
960 LOCATE £3,5,4:PRINT £3,"KONIK":GOTO 1030
970 'Obsluga bledu 3 rodzaju
980 IF s(in,jn)=0 GOTO 1040
990 CLS £3:blad=3
1000 LOCATE £3,4,2:PRINT £3,"TO POLE"
1010 LOCATE £3,4,3:PRINT £3,"JUZ WY-"
1020 LOCATE £3,3,4:PRINT £3,"KORZYSTANO"
1030 FOR T=1 TO 2000:NEXT
1040 CLS £3
1050 LOCATE £3,4,2:PRINT £3,"ZYCZYMY"
1060 LOCATE £3,4,4:PRINT £3,"SUKCESU"
1070 RETURN
1080 'Podprogram badania sukcesu
1090 FOR a=1 TO 8
1100 FOR b=1 TO 8
1110 IF s(a,b)=0 GOTO 1230
1120 NEXT b:NEXT a
1130 CLS £2:LOCATE £2,4,3:PRINT£2,"BRAWO!"
1140 LOCATE £2,2,4:PRINT£2,"ZWYCIEZYLES"
1150 CLS £3:LOCATE £3,2,2:PRINT £3,"CZY CHCESZ"
1160 LOCATE £3,3,3:PRINT £3,"POWTORZYC"
1170 LOCATE £3,2,4:INPUT £3,"GRE (T/N):",a$
1180 suk=1
1190 IF a$="T" OR a$="t" GOTO 1230
1200 PAPER 2:CLS:PEN 0
1210 LOCATE 12,12:PRINT"KONIEC ZADANIA KONIKOWEGO"
1220 GOTO 1220
1230 RETURN
```

Życzymy wytrwałym Czytelnikom sukcesu w tej nieskomplikowanej grze konwersacyjnej

Danuta KWASIŻUR
Mieczysław SKONIECZNY

„MASTERWORD"

Masterword jest kolejnym edytorem tekstu przeznaczonym dla komputera Commodore — 64. W wersji opisanej poniżej (wersja 2.8J) napisany został przez Green Valey Publishing należący do Share Data Inc. w roku 1986.

Program ten zasadniczo różni się od opisywanego wcześniej edytora Geowrite. Może być wykorzystywany tylko przez osoby dysponujące stacją dysków.

Po uruchomieniu programu ukazuje się pierwsze menu o następującej zawartości:

1. MASTER WORD
2. BRIEF INSTRUCTIONS (krótka instrukcja obsługi)
3. QUIT — zakończenie pracy z edytorem

Po wybraniu „1" następuje załadowanie edytora do pamięci komputera. Program na samym początku przypomina nam o konieczności sformatowania dyskietki, na której zamierzamy przechowywać tworzone dokumenty. Do formatowania należy użyć opcji PREPARE DOCUMENT DISK zawartej w menu DISK UTILITIES o czym będzie mowa w dalszej części artykułu.

I. Menu główne edytora Masterword

Zarówno menu główne edytora, jak i poszczególne funkcje przedstawione są w Masterword za pomocą ikon. Główne menu składa się z następujących funkcji: PRINT, EDIT, FILE, LAYOUT, CUSTOMIZE, DISK UTILITIES. Odpowiednią funkcję wybieramy za pomocą klawisza kursora, oznaczając za pomocą przesuwającej się ramki tą, która nas aktualnie interesuje, co potwierdzamy naciskając RETURN. Powrót do stanu poprzedniego menu następuje zawsze po naciśnięciu klawisza funkcyjnego F1.

W pamięci komputera stale rezydują tylko funkcje EDIT i LAYOUT, stąd po wybraniu którejkolwiek z pozostałych należy chwilę odczekać na jej załadowanie.

II. Opis poszczególnych funkcji zawartych w menu głównym.

1. PRINT

Po wybraniu tej funkcji na ekranie po pewnej chwili ukazuje się menu w postaci ikon, zawierające:

TYPE OF PAPER — rodzaj papieru (pojedyncze kartki lub papier podawany w sposób ciągły).
TYPEWRITER — przejście do trybu pracy maszyny do pisania — dana linia jest drukowana po wciśnięciu RETURN.
PRINT DOCUMENT — drukowanie dokumentu.
SEE FINAL DOCUMENT — obejrzenie na ekranie gotowego dokumentu.
STARTING PAGE NUMBER — numer pierwszej strony. Numer ten drukowany jest na dole z napisem PAGE (strona).

2. EDIT

Funkcja ta posiada następujące menu:
ERASE TEXT — usunięcie tekstu.
INSERT EREASED TEXT — umieszczenie tekstu usuniętego.
MOVE TEKST — przesunięcie (przeniesienie) tekstu w inne miejsce dokumentu.
COPY TEXT — skopiowanie tekstu w inne miejsce dokumentu, bez usuwania go z miejsca poprzedniego.
FIND — znalezienie danego tekstu
FIND AND REPLACE — znalezienie tekstu i zastąpienie go innym. Tekst przeznaczony do usunięcia, przeniesienia lub skopiowania zaznaczamy przez „podświetlenie" kursorem.

3. FILE

Opcja FILE składa się z następujących funkcji:
INSERT DOCUMENT — umieszczenie dokumentu w pisanym tekście, w miejscu gdzie znajduje się aktualnie kursor.
GET DOCUMENT — wywołanie na ekran napisanego wcześniej dokumentu. Jeżeli na ekranie znajduje się tekst, to ulega on kasowaniu przez tekst wywoływany.
SAVE DOCUMENT — zapisanie dokumentu na dyskietkę.
BEGIN NEW DOCUMENT — rozpoczęcie pisania nowego dokumentu.

4. LAYOUT

Funkcja ta pozwala na nadanie pisanym dokumentom odpowiedniej formy. Jej menu składa się z:
SET SPACING — (SET TOP/BOTTOM MARGINS, SET LEFT/ RIGHT MARGINS, SET LINE SPACING) — ustalenie marginesów: dolnego, górnego, lewego i prawego oraz odstępu między wierszami.
START NEW PAGE — rozpoczęcie nowej strony.
ALIGNMENT — (ALIGN RIGHT, ALIGN LEFT, CENTER NEXT LINE) — wyrównanie tekstu do prawej lub lewej strony, ustawienie centralne następnego wiersza.
PRINT STYLE — (BOLDFACE TEXT, NORMAL TEXT, UNDERLINE TEXT) — rodzaj druku — tekst pogrubiony, normalny i podkreślony.
HEADINGS/FOOTINGS — DRUKOWANIE TEKSTU JAKO NAGŁÓWKA LUB STOPKI po lewej/prawej stronie lub centralnie.

5. CUSTOMIZE

Funkcja ta posiada następujące możliwości:
MAKE BACKUP DOCUMENTS — wykonywanie kopii dokumentu podczas zapisywania go na dyskietce.
CHANGE PRESET MARGINS — (SET TOP/BOTTOM MARGINS, SET LEFT/ /RIGHT MARGINS, SET LINE SPACING, SET TAB STOPS, SET PHYSICAL PAGE LENGTH, SET PHYSICAL PAGE WIDTH) — zmiana poprzednio ustalonych parametrów takich jak: marginesy górny, dolny, lewy, prawy, odstęp między wierszami, tabulacji, długości i szerokości strony (w calach).
SAVE CUSTOMIZE CHOICES — zapisanie na dysku ustalonych wyżej parametrów.
TYPE OF PRINTER — wybór typu drukarki.
ACCESSORIES — określenie, czy praca z jedną, czy z dwiema stacjami dysków.

6. DISK UTILITIES

Funkcja ta zawiera następujące możliwości:
PREPARE DOCUMENT DISK — sformatowanie dyskietki w celu zapisywania na niej dokumentów pisanych za pomocą programu Masterword.
ERASE DOCUMENT — wymazanie dokumentu z dyskietki.
CATALOG DISK — wyświetlenie katalogu dyskietki (katalogu dokumentów na dyskietce)
EXIT — wyjście z programu (uruchomienie innego programu).

III. Klawisze funkcyjne programu

W programie Masterword znajduje się plik o nazwie HELP.DOC, w którym opisano funkcje przypisane pewnym klawiszom, a użycie których jest bardzo przydatne podczas tworzenia dokumentów.
Do klawiszy tych należą:
D i INST DEL — usunięcie znaku na lewo od kursora.
CRSR prawy — przesunięcie kursora o 1 spację w prawo.
SHIFT CRSR prawy lub F7 — przesunięcie kursora jedno miejsce w lewo.
CLR HOME — przejście do początku dokumentu.
F3 — klawisz tabulacji.
Naciśnięcie klawisza CTRL i jednego z niżej wymienionych daje następujące możliwości:
W — przejście kursora do poprzedniego wyrazu
E — przejście kursora do następnego wyrazu
N — wymazanie następnego słowa
O — wyświetlenie zawartości poprzedniego ekranu
P — wyświetlenie zawartości następnego ekranu
T — przejście do początku dokumentu
B — przejście do końca dokumentu
F — przejście do następnej strony (strony wydruku)
G — przejście do poprzedniej strony (strony wydruku)
C — zmiana trybu pisania (pisanie z rozsuwaniem istniejącego tekstu lub pisanie z zamazywaniem tekstu istniejącego).
V — wyświetlenie wartości ASCII danego znaku.
X — umieszczenie wartości znaku ASCII.

Możliwe jest także bezpośrednie wybieranie funkcji określonych przez ikony za pomocą równoczesnego naciśnięcia klawisza Commodore i następujących klawiszy:
A — umieszczenie dokumentu w pisanym tekście,
G — załadowanie dokumentu,
H — pokazanie na ekranie pliku HELP,
S — zapis dokumentu na dyskietce,
C — kopiowanie tekstu,
E — usunięcie tekstu,
M — umieszczenie usuniętego tekstu,
F — znalezienie tekstu,
R — znalezienie tekstu i zastąpienie go innym,
Y — przesunięcie tekstu,
B — drukowanie (litery) pogrubione,
D — początek nowej strony,
K — ustalenie prawego/lewego marginesu,
L — wyrównanie tekstu do lewej strony,
N — tekst normalny,
O — napisanie następnego wiersza centralnie,
Q — ustalenie górnego i dolnego marginesu,
T — nagłówek/stopka,
U — podkreślenie tekstu,
W — ustalenie odstępu między wierszami,
Z — wyrównanie tekstu do prawej strony.

Stosowanie wyżej wymienionych klawiszy przyspiesza znacznie pracę podczas pisania dokumentów, ponieważ nie wymaga poruszania się, przeglądania i wybierania poszczególnych ikon.

IV. Pisanie dokumentów za pomocą edytora Masterword.

Pracę z edytorem, jak już wcześniej wspomniano należy zacząć od przygotowania dyskietki na dokumenty. Należy w tym miejscu zaznaczyć, że dokumenty (pliki) umieszczone na takiej dyskietce, utworzone podczas pracy z Masterword, nie są widoczne przy próbie ich wylistowania za pomocą instrukcji LOAD „$", 8 i LIST. Chcąc utworzyć nowy dokument wybieramy z menu FILE opcję BEGIN NEW DOCUMENT.

Podczas pisania ekran podzielony jest na trzy okna. Okno główne przeznaczone jest do wpisywania tekstu. W oknie po lewej stronie na dole znajduje się informacja o numerze aktualnej strony, ilość wolnej pamięci komputera oraz ilość wolnego miejsca na dyskietce, na którą wpisywane są tworzone dokumenty (dwie ostatnie informacje w postaci jasnych pasków, zmniejszających swoją długość).
W oknie na dole po prawej stronie ekranu widoczny jest rzeczywisty układ tekstu na pisanej stronie (układ taki jaki zostanie wydrukowany).
Podczas pisania dłuższych tekstów należy co jakiś czas wpisać na dyskietkę tekst już napisany, aby w sytuacji awaryjnej (zanik napięcia) nie było konieczne zaczynanie całej pracy od nowa.

V. Ocena edytora

Masterword jest programem bardzo przyjaznym i łatwym w obsłudze. Zastosowane ikony ułatwiają bardzo pracę, szczególnie początkującym użytkownikom. Masterword w zasadzie spełnia wszystkie wymagania stawiane „poważnym" edytorom tekstu przeznaczonym na komputery osobiste.
Do jego wad należy zaliczyć jedynie dosyć ubogi wybór rodzajów pisma (zastosowany krój czcionki może się jednak podobać) oraz oczywiście brak polskiego alfabetu.

Tadeusz CISEK

Kompilator języka C firmy HISOFT

(Amstrad CPC-6128)

Język C z racji nieco skomplikowanej syntaktyki jest jeszcze mniej popularny od BASIC'a, czy PASCAL'a, lecz pomimo to istnieje wiele jego wersji mikrokomputerowych. Do opanowania podstawowych konstrukcji świetnie nadaje się kompilator wyprodukowany przez firmę HISOFT, która z konsekwencją lansuje własny styl. Dzięki temu, ten kto opanował obsługę kompilatora PASCAL na Spectrum, czy Amstradzie, nie powinien mieć w zasadzie kłopotów z obsługą C. Niniejszy opis dotyczy wersji Amstrada.

1. URUCHOMIENIE KOMPILATORA

Kompilator składa się z jednego zbioru (HISOFTC. BIN). Wystarczy wpisać

RUN"HISOFT

i włożyć dyskietkę lub uruchomić magnetofon z kasetą. Po prawidłowej operacji ładowania powinien ukazać się napis:

HISOFT — C Compiler v1.2
Copyright C 1984 HISOFT

oraz znak zachęty > będący sygnałem zgłoszenia się edytora. Obok pojawi się kursor. Ekran będzie zorganizowany w trybie 2.

2. EDYTOR

Komendy edytora są jednoliterowe i nie mogą mieć więcej niż 4 parametry. Pierwsze dwa są numeryczne (początkowo równe 10), a kolejne dwa tekstowe (początkowo puste). Nie wszystkie są zapamiętywane, co stanowi pewne odstępstwo od standardu HISOFT. Oto skrócony opis opcji edytora:

I m, n	—	wpisywanie linii programu od m co n;
L m, n	—	listowanie programu od linii m do n;
D m, n	—	kasowanie bloku linii od m do n; (kasowanie jednej linii D m).
R m, n	—	przenumerowanie linii od m co n.
W m, n	—	analogicznie jak L, lecz na drukarkę;
S, znak	—	zmiana separatora (początkowo przecinek);
V	—	opcja informacyjna podająca bieżące wartości separatora oraz 4 parametrów komend, poza tym wyświetlenie adresu początku i końca pamięci zawierającej program źródłowy;
F m, n, f, s	—	szukanie ciągu znakowego f w tekście między liniami m, n; po edycji linii można wybrać dwie kombinacje: CTRL f — dalsze szukanie; CTRL s — zamiana łańcucha f na s.
P m, n, s	—	zapis programu źródłowego od linii m do n pod nazwą s na dyskietkę lub kasetę.
G,, s	—	odczyt programu źródłowego o nazwie s.
E n	—	redagowanie linii n; opcje dostępne dla tej czynności są następujące:
RETURN	—	zakończenie edycji linii z uwzględnieniem poprawek;
ESC	—	zakończenie edycji linii bez uwzględniania poprawek;
prawo	—	kursor w prawo;
lewo	—	kursor w lewo
CTRL prawo	—	kursor na koniec linii;
CTRL lewo	—	kursor na początek linii;
COPY	—	kopiowanie drugim kursorem;
CTRL TAB	—	przełącznik trybu wkładania znaków w tekst na tryb nakładania znaków na tekst i vice versa;
CLR	—	kasowanie znaku pod kursorem;
DEL	—	kasowanie znaku na lewo od kursora;

Jak można zauważyć, opcje trybu redagowania linii są bardziej zbliżone do edytora BASIC'a w Amstradzie, niż do poprzednich rozwiązań firmy.

Kolejne linie programu wprowadza się przy wykorzystaniu opcji I lub bezpośrednio wczytując numer linii i instrukcje. Jeśli w pewnym momencie edytor zareaguje raportem:

ERROR 60
LIMIT: no more memory

co oznacza przepełnienie pamięci, to należy wpisać dyrektywę kompilatora #error, która spowoduje odzyskanie pamięci zajętej przez komunikaty o błędach (ok. 2KB).

Powrót do BASIC'a następuje po naciśnięciu CTRL—SHIFT—ESC, przy czym niszczy to kompilator i „ciepły" restart nie jest możliwy.

3. KOMPILACJA PROGRAMU UTWORZONEGO W EDYTORZE

Kompilacja utworzonego pod edytorem programu zachodzi po wprowadzeniu opcji c (po znaku zachęty). Kompilator przedstawi się wówczas znaną już formułą. Teraz wystarczy podać dyrektywę #include, po której zachodzi kompilacja z jednoczesnym listowaniem programu źródłowego na ekranie. W przypadku używania magnetofonu proces może być przerywany napisem Press PLAY then any key, szczególnie gdy dołączamy zbiory biblioteczne.

Przejście do fazy uruchomienia programu nastąpi po naciśnięciu kombinacji CTRL-z. Wygodnie jest zasygnalizować ten moment w programie odpowiednim komentarzem. Oto typowy schemat przebiegu obsługi kompilacji w skrócie:

```
>c
        (formuła)
#include
        (kompilacja)
CTRL-z
        (type y to run)
y
```

4. KOMPILACJA „NA GORĄCO"

Jak wspomniałem wyżej, opcja c pozwala wejść do trybu kompilacji. Możliwe jest wtedy podawanie dyrektyw kompilatora oraz redagowanie programu (bez numerów linii), który zostanie natychmiast przetłumaczony po CTRL-z. Wygodne do testowania, gdyż kompilacja taka nie niszczy programu pod edytorem.

5. PRACA INTERPRETACYJNA

Kompilator C ma dość ciekawą własność. Jeśli znajdujemy się na poziomie kompilatora, to dyrektywa #direct+ spowoduje przejście do trybu pracy interpretacyjnej. Możliwe jest wpisywanie pojedynczych instrukcji, które zostaną natychmiast wykonane, podobnie jak w trybie bezpośrednim w BASIC'u. Wyłączenie tego trybu zajdzie po #direct—.

6. DYREKTYWA #INCLUDE

Poznana już dyrektywa #include jest podstawą dla procesu kompilacji programu znajdującego się w pamięci operacyjnej, a napisanego pod edytorem. Używa się jej również do innych celów:

a) kompilacji zbioru o podanej nazwie

#include nazwa. typ

b) włączeniu innego zbioru zawierającego tekst źródłowy do kompilacji tekstu głównego przez użycie dyrektywy #include nazwa. typ jako linii w programie głównym;

c) kompilacji warunkowej; jeśli użyjemy w programie linii:

#include? nazwa. typ?

to do kodu wynikowego zostaną włączone tylko te funkcje, które są używane w programie głównym;

Inne postacie tej dyrektywy w standardzie języka, a mianowicie:

#include <nazwa. typ>
#include „nazwa. typ"

są równoważne zapisowi #include nazwa. typ.

7. TWORZENIE SAMODZIELNEGO ZBIORU WYNIKOWEGO

Dyrektywa #translate nazwa. typ pozwala na uzyskanie niezależnego zbioru wynikowego, który nie będzie wymagał otoczenia kompilatora c w celu uruchomienia. Zbiór taki zostanie odłożony na dyskietce lub kasecie, a z poziomu AMSDOS'u może być uruchamiany przez:

RUN" nazwa. typ

8. INNE DYREKTYWY KOMPILATORA

#list+	włączenie wydruku podczas kompilacji,
#list-	wyłączenie wydruku podczas kompilacji;
#define	w kompilatorze HISOFT jest to opcja ograniczona w zasadzie do przedefiniowywania nazw stałych.

W porównaniu do standardu języka brakuje implementacji dyrektyw #undef, #line oraz #if i pochodnych.

Krzysztof MAMCARZ

Jak oszukać, czyli jak rozszerzyć edytor ZX...

Niestety, edytor ZX Spectrum nie jest zbyt udaną konstrukcją. Poprawianie linii programu za jego pomocą, to żmudna praca, zwłaszcza gdy są one długie i zawierają błędy składniowe. Aby dostać się do ostatniego znaku, kursor musi przejść przez całą linię i bardzo często trwa to tyle czasu, że można sobie bez pośpiechu zrobić nawet herbatę.

Proponuję zwiększyć możliwości edytora, trochę go oszukując. Edytor wykonuje polecenia użytkownika przez kontrolę klawiatury, a ściślej: sprawdza wartości zmiennych systemowych LAST-K, FLAGX, FLAGS, FLAS2, MODE. Te zmienne systemowe informują m.in., czy i jaki klawisz został wciśnięty, w jakim trybie znajduje się kursor, itp. Są one jakby „oknem na świat" edytora. O początku i końcu edytowanej linii informują zmienne E-LINE i WORKSP, a o położeniu kursora zmienna K-KUR. O wszystkich tych zmiennych systemowych była już mowa w poprzednich numerach „IKS-a" i dlatego nie będę ich omawiał.

Władzę nad edytorem uzyskamy, kontrolując i odpowiednio zmieniając wartości tych zmiennych. W tym celu skorzystamy z drugiego typu przerwań (IM 2), co oznacza, że co 1/50 s. oprócz kontroli klawiatury zostanie uruchomiony nasz program. Opiszę sposób oszukania edytora na przykładzie: tryb specjalny oznaczony kursorem X uzyskujemy naciskając jednocześnie Symbol Shift i Spację (SS+Sp). W rzeczywistości jest inaczej: w czasie przerwania nasz program sprawdza czy jednocześnie wciśnięto SS+Sp, jeśli tak, to ustawia zmienną LAST.K, tak jakby wciśnięto CAPS LOOK (do LAST-K wpisuje wartość 6). Dodatkowo do zmiennej MODE wpisuje 138. Tymczasem edytor na podstawie LAST-K ustawia tryb C kursora, ale ponieważ zamiast MODE=0 jest MODE=138, zostaje wyświetlony kursor X. Dodatkowo program zapamiętuje aktualny tryb pracy, aby po ponownym wciśnięciu SS+Sp przywrócić dawny stan kursora.

W trybie X uzyskujemy następujące możliwości:

kursory — pozwalają poruszać się w odpowiednich kierunkach po edytowanej linii

P — ustawia kursor na początku edytowanej linii

```
 100 CLEAR 65014
1000 RESTORE 9000: FOR k=0 TO 25
1010 LET s1=0: LET s2=0: LET z=1
1030 FOR l=0 TO 19: READ a: LET
s1=s1+a: LET s2=s2+z*a: LET z=-z
: POKE 65015+20*k+l,a: NEXT l: R
EAD a1: READ a2: IF s1<>a1 OR a2
<>s2 THEN PRINT "blad w linii :"
;9000+10*k: STOP
1040 NEXT k
1050 READ a: POKE 65535,a: IF a<
>134 THEN PRINT "blad w linii 92
60": STOP
2000>RANDOMIZE USR 65015: REM wl
aczenie rozszerzenia edytora

9000 DATA 62,254,237,71,237,94,1
75,50,208,255,201,205,56,0,245,5
8,113,92,203,111,2927,547
9010 DATA 194,211,254,197,213,22
9,58,8,92,254,32,194,80,254,205,
142,2,62,24,186,2691,-563
9020 DATA 194,80,254,58,208,255,
203,127,202,56,254,203,191,50,20
8,255,58,209,255,50,3370,684
9030 DATA 65,92,195,72,254,203,2
55,50,208,255,58,65,92,50,209,25
5,62,138,50,65,2693,203
9040 DATA 92,62,6,50,8,92,195,20
8,254,58,208,255,203,127,202,208
,254,237,86,58,2863,153
9050 DATA 8,92,254,8,202,206,254
,254,10,202,213,254,254,11,202,3
3,255,254,9,202,3177,145
9060 DATA 206,254,254,245,202,99
,255,254,112,202,99,255,254,80,2
02,99,255,254,240,202,4023,135
9070 DATA 113,255,254,107,202,11
3,255,254,75,202,113,255,254,233
,202,135,255,254,100,202,3833,-1
67
9080 DATA 135,255,254,68,202,135
,255,254,255,202,120,255,254,122
,202,120,255,254,90,202,3889,155
9090 DATA 120,255,254,238,202,16
8,255,254,105,202,168,255,254,73
,202,168,255,254,7,202,3891,-247
9100 DATA 148,255,254,13,202,166
,255,58,59,92,203,175,50,59,92,2
37,94,225,209,193,3039,93
9110 DATA 241,201,6,33,42,91,92,
126,254,13,202,113,255,254,165,2
10,244,254,254,32,3082,428
9120 DATA 210,235,254,4,35,34,91
,92,16,230,195,106,255,205,7,255
,218,105,255,195,2996,76
9130 DATA 1,255,255,195,2,254,42
,91,92,195,234,254,33,128,255,22
,0,203,63,245,2819,-865
9140 DATA 95,25,241,126,218,26,2
55,15,15,15,15,230,15,79,120,145
,71,201,6,32,1945,157
9150 DATA 237,91,91,92,26,254,16
5,210,86,255,254,32,210,56,255,2
54,13,202,56,255,3094,-308
9160 DATA 4,27,42,89,92,175,237,
82,210,99,255,237,83,91,92,16,21
9,26,254,32,2362,614
9170 DATA 218,55,255,19,237,83,9
1,92,195,105,255,205,7,255,218,1
05,255,237,91,91,3069,575
9180 DATA 92,195,55,255,42,89,92
,34,91,92,62,8,50,8,92,195,206,2
54,42,97,2051,-403
9190 DATA 92,43,195,102,255,237,
91,91,92,42,97,92,43,43,205,229,
25,195,113,255,2537,-121
9200 DATA 237,91,89,92,42,91,92,
205,229,25,195,99,255,58,110,92,
50,73,92,58,2275,507
9210 DATA 111,92,50,74,92,205,17
9,255,195,206,254,237,94,205,179
,255,62,13,50,8,2816,-284
9220 DATA 92,195,208,254,175,50,
208,255,58,209,255,50,65,92,58,1
06,92,203,95,202,2922,-310
9230 DATA 202,255,203,159,195,20
4,255,203,223,50,106,92,201,0,0,
3,98,54,133,52,2688,544
9240 DATA 85,66,68,68,68,52,68,6
8,83,69,84,68,82,34,102,70,133,1
34,119,135,1656,126
9250 DATA 134,133,119,137,101,13
5,102,105,88,165,85,120,118,101,
116,103,101,107,69,103,2244,-174
9260 DATA 134
```

```
00012  ; program wykorzystuje przerwa-
00013  ; nia maskowalne, ustawia tryb 2
00014  ; przerwan.
00015  ; program nie jest relokowalny.
00016  ;
00017  rej.i   equ  0feh;wartosc rej. i
00018  ; kody rozkazow
00019  EDIT    equ  7
00020  DIM     equ  233
00021  COPY    equ  255
00022  LIST    equ  240
00023  INPUT   equ  238
00024  ENTER   equ  13
00025  PRINT   equ  245
00026  kur.X   equ  138 ,kursor X
```

```
00027  kur.C  equ  6   ;kursor C
00028  kur.lf equ  8   ;kursor w lewo
00029  kur.rt equ  9   ;kursor w prawo
00030  kur.dn equ  10  ;kursor w gore
00031  kur.up equ  11  ;kursor w dol
00032  ; adresy procedur ROM-u
00033  recmin equ  19e5h
00034  keyb   equ  38h
00035  keybsc equ  28eh
00036  ; adresy zmie nnych systemowych
00037  flags  equ  5c3bh
00038  flags2 equ  5c6ah
00039  flagx  equ  5c71h
00040  e_ppc  equ  5c49h
00041  oldppc equ  5c6eh
00042  e_line equ  5c59h
00043  mode   equ  5c41h
00044  k_kur  equ  5c5bh
00045  worksp equ  5c61h
00046  last_k equ  5c08h
00047  ;
00048         org  65015
00049  ; inicjalizacja trybu 2
00050  ustim2 ld   a,rej.i
00051         ld   i,a
00052         im   2
00053         xor  a
00054         ld   (flag.p),a
00055         ret
00056  ;
00057  ; rozszerzenie przerwania
00058  ; program wykonywany w czasie
00059  ; przerwania
00060  przerw call keyb
00061         push af
00062         ld   a,(flagx)
00063         bit  5,a
00064         jp   nz,tryb.i ;tryb inp.
00065         push bc
00066         push de
00067         push hl
00068         ld   a,(last_k)
00069         cp   32
00070         jp   nz,etyk2
00071         call keybsc
00072         ld   a,18h
00073         cp   d
00074         jp   nz,etyk2
00075  ;zmiana trybu
00076         ld   a,(flag.p)
00077         bit  7,a
00078         jp   z,ustaw7
00079         res  7,a ;wyl. trybu X
00080         ld   (flag.p),a
00081         ld   a,(modez)
00082         ld   (mode),a
00083         jp   etyk1
00084  ustaw7 set  7,a ;wlacz. trybu X
00085         ld   (flag.p),a
00086         ld   a,(mode)
00087         ld   (modez),a
00088         ld   a,kur.X
00089         ld   (mode),a
00090  etyk1  ld   a,kur.C
00091         ld   (last_k),a
00092         jp   konprw
00093  etyk2  ld   a,(flag.p)
00094         bit  7,a
00095         jp   z,konprw
00096         im   1
00097  ; program obslugujacy rozkazy
00098  ; trybu X
00099         ld   a,(last_k)
00100         cp   kur.lf
00101         jp   z,koniec
00102         cp   kur.dn
00103         jp   z,kurDN
00104         cp   kur.up
00105         jp   z,kurUP
00106         cp   kur.rt
00107         jp   z,koniec
00108         cp   PRINT
00109         jp   z,obsl.p
00110         cp   'p'
00111         jp   z,obsl.p
00112         cp   'P'
00113         jp   z,obsl.p
00114         cp   LIST
00115         jp   z,obsl.k
00116         cp   'k'
00117         jp   z,obsl.k
00118         cp   'K'
00119         jp   z,obsl.k
00120         cp   DIM
00121         jp   z,obsl.d
00122         cp   'd'
00123         jp   z,obsl.d
00124         cp   'D'
00125         jp   z,obsl.d
00126         cp   COPY
00127         jp   z,obsl.z
00128         cp   'z'
00129         jp   z,obsl.z
00130         cp   'Z'
00131         jp   z,obsl.z
00132         cp   INPUT
00133         jp   z,obsl.i
00134         cp   'i'
00135         jp   z,obsl.i
00136         cp   'I'
00137         jp   z,obsl.i
00138         cp   EDIT
00139         jp   z,obslED
00140         cp   ENTER
00141         jp   z,obslER
00142  ; zablokowanie pozostalych
00143  ; klawiszy
00144         ld   a,(flags)
00145         res  5,a
00146         ld   (flags),a
00147  koniec im   2
00148  konprw pop  hl
00149         pop  de
00150         pop  bc
00151  tryb.i pop  af
00152         ret
00153  ;
00154  ; procedury obslugujace
00155  ;
00156  ; procedura: kursor w dol
00157  kurDN  ld   b,33
00158  naprz  ld   hl,(k_kur)
00159         ld   a,(hl)
00160         cp   ENTER
00161         jp   z,obsl.k
00162         cp   165
00163         jp   nc,rozk
00164         cp   32
00165         jp   nc,znak
00166  znaknd inc  b
00167  znak   inc  hl
00168         ld   (k_kur),hl
00169         djnz naprz
00170         jp   lastLF
00171  rozk   call rozkwk
00172         jp   c,lastLF
00173         jp   most
00174  ;
00175         org  256*rej.i+255
00176  ; adres obslugi przerwania trybu
00177  ; im 2
00178         dw   przerw
```

```
00179  ;
00180  most    ld    hl,(k_kur)
00181          jp    znaknd
00182  ;wewn. proc. przeskakiw. rozk.
00183  rozkwk  ld    hl,tabl-164/2
00184          ld    d,0
00185          srl   a
00186          push  af
00187          ld    e,a
00188          add   hl,de
00189          pop   af
00190          ld    a,(hl)
00191          jp    c,nieprs
00192          rrca
00193          rrca
00194          rrca
00195          rrca
00196  nieprs  and   00001111b
00197          ld    c,a
00198          ld    a,b
00199          sub   c
00200          ld    b,a
00201          ret
00202  ;
00203  ; procedura: kursor w gore
00204  kurUP   ld    b,32
00205  cofaj   ld    de,(k_kur)
00206          ld    a,(de)
00207          cp    165
00208          jp    nc,rozk1
00209          cp    32
00210          jp    nc,znak1
00211          cp    ENTER
00212          jp    z,znak1
00213  znakn1  inc   b
00214  znak1   dec   de
00215          ld    hl,(e_line)
00216          xor   a
00217          sbc   hl,de
00218          jp    nc,obsl.p
00219          ld    (k_kur),de
00220          djnz  cofaj
00221          ld    a,(de)
00222          cp    32
00223          jp    c,znakn1
00224          inc   de
00225          ld    (k_kur),de
00226          jp    lastLF
00227  rozk1   call  rozkwk
00228          jp    c,lastLF
00229          ld    de,(k_kur)
00230          jp    znakn1
00231  ;
00232  ; procedura: kursor na poczatek
00233  obsl.p  ld    hl,(e_line)
00234  krs.lf  ld    (k_kur),hl
00235  lastLF  ld    a,kur.lf
00236          ld    (last_k),a
00237          jp    koniec
00238  ;
00239  ; procedura: kursor na koniec
00240  obsl.k  ld    hl,(worksp)
00241          dec   hl
00242          jp    krs.lf
00243  ;
00244  ; procedura: kasuj za kursorem
00245  obsl.z  ld    de,(k_kur)
00246          ld    hl,(worksp)
00247          dec   hl
00248          dec   hl
00249          call  recmin
00250          jp    obsl.k
00251  ;
00252  ; procedura: kasuj do kursora
00253  obsl.d  ld    de,(e_line)
00254          ld    hl,(k_kur)
00255          call  recmin
00256          jp    obsl.p
00257  ;
00258  ; procedura: EDIT
00259  obslED  ld    a,(oldppc)
00260          ld    (e_ppc),a
00261          ld    a,(oldppc+1)
00262          ld    (e_ppc+1),a
00263          call  norm.k
00264          jp    koniec
00265  ;
00266  ; procedura: ENTER
00267  ; procedura: ENTER+tryb 1
00268  obslER  im    2
00269  obsl.i  call  norm.k
00270          ld    a,ENTER
00271          ld    (last_k),a
00272          jp    konprw
00273  norm.k  xor   a
00274          ld    (flag.p),a
00275          ld    a,(modez)
00276          ld    (mode),a
00277          ld    a,(flags2)
00278          bit   3,a
00279          jp    z,ustaw3
00280          res   3,a
00281          jp    etyk4
00282  ustaw3  set   3,a
00283  etyk4   ld    (flags2),a
00284          ret
00285  ; zmienne systemowe procedury
00286  flag.p  ds    1
00287  modez   ds    1
00288  ;
00289  ; tablica ilosci liter rozkazow
00290  ; od RND (kod 165) do COPY (kod
00291  ; 255). Kazdy bajt zawiera dlu-
00292  ; gosci dwoch kolejnych rozkazow
00293  ; zapisanych jako liczby BCD
00000  ;
00294  tabl    ;     45  67  89  01  23
00295  t16...  db    03h,62h,36h,85h,34h
00296  t17...  db    55h,44h,44h,44h,44h
00297  t18...  db    34h,44h,44h,53h,45h
00298  t19...  db    54h,44h,52h,22h,66h
00299  t20...  db    46h,85h,86h,77h,87h
00300  t21...  db    86h,85h,77h,89h,65h
00301  t22...  db    87h,66h,69h,58h,0a5h
00302  t23...  db    55h,78h,76h,65h,78h
00303  t24...  db    67h,65h,6bh,45h,67h
00304  t25...  db    86h
00305          end
```

K — ustawia kursor na końcu edytowanej linii
D — kasuje znaki edytowanej linii od początku linii do kursorą
Z — kasuje znaki edytowanej linii za kursorem
ENTER — działa, jak klasyczne ENTER, ale wyłącza tryb X
I — tak samo jak ENTER, ale zostaje ustawiony tryb pierwszy przerwań
EDIT — edytuje linię wskazywaną przez CONTINUE
Uwaga: pozostałe klawisze w trybie X są ignorowane
SS+Sp — włącza i wyłącza tryb X

Ponieważ program jest przedstawiony w zapisie mnemonicznym, to ze względu na czytelność nie mógł być zbyt obszerny (stąd niewiele dodatkowych funkcji). Należy go więc traktować jako przykład, a nie jak ostateczne rozwiązanie.

Proponuję natomiast czytelnikom wtajemniczonym w programowanie w assemblerze uzupełnienie programu o nowe funkcje. Myślę, że wspólnymi siłami ulepszymy edytor ZX Spectrum. Wraz z redakcją „IKS-a" czekam na propozycje.

Dla tych, którzy chcą skorzystać z prezentowanego programu, zamieszczam program ładujący kod maszynowy

Włączył magnetofon. Muzyka wypełniła pokój. Erta Kitt śpiewała C'est si bon. Ulubioną piosenkę jego młodości. Słuchał jej zawsze, zanim wyruszył w trasę, zajadając sandwicza i popijając sok grejpfrutowy. Chwycił przewieszoną przez poręcz krzesła cienką kurtkę z impregnowanego papieru, z której uśmiechało się słońce i przyciągał wzrok tęczowy napis Miami Beatch. Zbiegł schodami do holu, podszedł do wiszącego na ścianie „wiecznego" kalendarza i ustawił pokrętłami datę. Był 28 sierpnia 1979 roku.

Sprawnie wspiął się do umieszczonej wysoko nad ziemią szoferki swej potężnej ciężarówki, włączył radio i w dostojnym rytmie jakby najważniejszej w życiu pracy uruchomił silnik. Skontrolował dokładnie wskaźniki rozbłysłe kolorowymi światełkami na pulpicie. Niedbale wrzucił do skrytki obok kierownicy potężnego colta kalibru 32.

Uśmiechnął się do swych myśli: — Z bronią nie rozstawaj się nigdy, chłopcze — przestrzegał szef. Tak jakby na swojej nowoczesnej, niezwykłej, trzydziestometrowej ciężarówce z naczepą przewoził złoto, a nie artykuły żywnościowe na lokalnej linii transportowej pomiędzy Winnchester a Friedrickburgiem. — Z bronią bezpieczniej i pewniej — tłumaczył szef. Tak jakby on, 28-letni kierowca zawodowy, Hary Joe Turner, musiał się czegoś lub kogoś bać. On, barczyście zbudowany i wysportowany, młody przecież jeszcze człowiek, były żołnierz amerykańskiej piechoty morskiej, były zawodowy bokser — miałby mieć stracha? Niedorzeczność...

Nastawił głośniej radio. Powietrze, czyste i świeże, jakby naraz drgnęło. Pierwsze promienie ukośnie padły na ziemię i zajrzały do wnętrza szoferki. Na niebie, aż po horyzont, rozsypały się białe, drobniutkie obłoczki. Słońce pomarańczową kulą wygramoliło się zza widnokręgu. Koła nawijały kolejne mile asfaltowej szosy. Silnik pracował z nużącą monotonią. I nagle... Co za licho? Zamilkło radio. Cisza i zupełna pustka. Stało się to tak niespodziewanie, że niemal poczuł się nieswojo. Spojrzał w boczne lusterko. Wypełniało je żółte światło. Zbliżało się z dużą prędkością do samochodu. Jakiś „pirat" drogowy pędzi na złamanie karku — pomyślał Turner i zjechał na prawą stronę drogi. Po chwili jednak żółty świetlisty snop objął ze wszystkich stron ciężarówkę.

Joemu nieznośnie załopotało serce. Mieszanina stężonego niepokoju i podniecenia wypełniła mózg i ciało. Żółtawa plama światła wpełzła do środka szoferki, poruszała się po ścianach, wspięła aż po dach, potem opadła i rozpełzła się po wszystkich zakamarkach. Lepki żar wypełnił wnętrze wozu. Poczuł, że zabrakło mu powietrza. Kleiły się oczy, rwała głowa. Coraz trudniej było zwalczać obezwładniające zmęczenie. Do tego jeszcze ten dziwny powiew ciepła, przygniatający ciężar w piersiach i dźwięk — wysoki, nieznośny. Uległ sennemu bezwładowi.

Dopiero teraz uświadomił sobie, że czuje lęk. Strach przed nieznanym niebezpieczeństwem. Niczego nie mógł dojrzeć przed samochodem. Nie widział nawet swoich rąk, zaciśniętych kurczowo na kierownicy. Tylko to mocne, jaskrawe światło z fioletowym odcieniem, sprawiające wrażenie płynu. Tak mu się przynajmniej wydawało — to światło było lepkie.

Samochód przestał go słuchać. Joe popadł w stan ni to snu, ni to odrętwienia. Nie widział już niczego, o niczym nie potrafił myśleć, wszystko zatraciło się jakby w gęstej, przytłaczającej żółtej mgle.

Nie pamięta, jak długo to trwało. Wie tylko, że w jakiejś chwili wróciła świadomość otaczającej rzeczywistości. Znowu wszystko widział i słyszał, czuł, że trawi go gorączka, męczą dreszcze. Z przerażeniem stwierdził, że znajduje się we wnętrzu obszernego pomieszczenia, jakby kopulastej hali o przymglonych i słabo widocznych ścianach. W przypływie rozpaczy próbował uruchomić silnik — bezskutecznie. I wtedy usłyszał rozedrgany szept, jakby ktoś przewijał od tyłu, w przyspieszonym tempie, taśmę magnetofonową, a potem, w refleksie dziwnej poświaty, pojawiły się zwaliste, jakby z grubej bryły wyciosane, istoty. Podobne były do ludzi wzrostem i wyglądem. Także ich twarze — nieruchome, zamknięte i mroczne — przypominały ludzkie.

Turner wyszarpnął ze skrytki pistolet, odbezpieczył i prawie nie celując, wypalił osiem razy — przez szybę — w kierunku dziwnych postaci, ubranych w obcisłe kombinezony. I wtedy poczuł, że jakaś tajemnicza siła nakazuje mu opuścić broń, wyjść z kabiny. Nie mógł się oprzeć tym telepatycznym rozkazom. Posłusznie je wykonał.

Czuł się, jakby śnił jakiś makabryczny, nierealny sen. Czyż to mogła być jawa? A przecież... Czemu oni tak dokładnie oglądali samochód. Brrr... Jakaż ich skóra była zimna w dotyku. I te dziwne błony między palcami, jakby minipłetwy. I ta szybka, śpiewna, niezrozumiała mowa... Czyż to mógł być sen?

Kto to, gdzie ja jestem? — zadawał sobie pytania, umierając z przerażenia. Czegoż oni mogą ode mnie chcieć? Po co ta istota o kamiennej twarzy wchodzi pod samochód... Ależ to są chyba... Tak, to... roboty. Przynajmniej tych trzech. Na pewno. Temu pod ciężarówką, kiedy się pochylił, spadł z głowy kaptur. Na skórze, nad oczami miał wytatuowany ciąg cyfr albo liter, jakiegoś dziwnego, hieroglificznego pisma.

Ten czwarty — to szef. Daje jakieś znaki rękami i tamci trzej skrupulatnie wykonują polecenia. Może to tylko gestykulacja? Może porozumiewają się także telepatycznie? Tak jak z nim. Tak, z pewnością. Nie można przecież kilkoma gestami przekazać tak szczegółowo złożonych poleceń...

Czuł straszny ból głowy. Jakby ktoś napompował czaszkę do granic wytrzymałości powietrzem. Humanoidalne istoty nakazały mu podejść do mglistej ściany kopuły. Przedziwne to wszystko. Jakby wyglądał zza najczystszej, niewidzialnej niemal szyby. Ależ tam rozciąga się... kosmos. Ujrzał, że ich olbrzymi pojazd znajduje się nad jakąś dziwną planetą. Z góry widać było bardzo dokładnie „nibymiasta", nakryte czymś w rodzaju gigantycznych, przezroczystych kopuł.

Chwycił się oburącz za głowę. Trudno powiedzieć, czy w natłoku wrażeń, czy też dlatego, że ten sam spokojny, przenikliwy głos wtłaczał mu do świadomości wiadomość, której tak naprawdę, nie rozumiał: — Znajdujemy się o dwa i pół roku świetlnego od gwiazdy Alfa Centaurii... Co to jest ta Alfa Centaurii? Lata świetlne? Co to wszystko ma znaczyć? Toż to jakieś majaki w malignie...

Tajemnica żółtego światła

Ależ nie... Za przezroczystą kopułą wehikułu przesuwała się monotonna czerń, rozjaśniona od czasu do czasu skąpym światłem różnej wielkości punktów — jakby patrzył na gwieździste niebo w bezchmurną noc. Przez głowę przelatywały mu strzępy myśli... Gorączkowo próbował obmyśleć jakiś plan działania, ale nic mu nie przychodziło do głowy. Na plecach czuł wzrok tych przedziwnych istot. Śledzili czujnie każdy jego ruch. Tego był pewny i to w jakiś dziwny sposób nie pozwalało mu na wykonanie żadnego ruchu.

I znowu nachodziły go uparcie monotonne fale snu. Starał się walczyć z tym koszmarnym obezwładnieniem. Na próżno. Musiał zrezygnować z tej męczącej walki. Zdawało mu się, że czas mijał niemiłosiernie wolno. Wszystko dokoła zaczęło wirować w coraz szybszym tempie. Stracił świadomość.

Ocknął się na parkingu w Friedricksburgu. Jak się tutaj znalazł? Przecież... Rozejrzał się po szoferce. Rzeczy osobiste oraz pieniądze, które wiózł ze sobą, narzędzia, leżały bezładnie porozrzucane na podłodze. Wzrok ślizgał się przez chwilę po tym bałaganie... Nie, niczego nie brakowało. Zegar na ścianie domu dworcowego, przed którym zawsze parkował, przywożąc towar, wskazywał godzinę trzecią po południu. Miał mętlik w głowie, czuł się źle.

Samochód, co z samochodem? — pomyślał po chwili, kiedy nareszcie doszedł do siebie. Niezdarnie wygramolił się z kabiny. Na plac wjeżdżała jakaś ciężarówka. Może ktoś ze znajomych? Musi mu opowiedzieć o swojej historii, potrzebuje przecież pomocy.

Obcy kierowca jechał wprost na niego, nie zmniejszając prędkości. Był już tylko 20 metrów, 10, 5... Turner osłupiał z wrażenia. Tamten zatrzymał się zaledwie metr od niego. Przecież jeszcze sekunda i... Podniósł z wysiłkiem rękę, gestem człowieka, który pilnie potrzebuje pomocy. Obcy kierowca nie reagował. Wyszedł niespiesznie z wozu i skierował się w stronę marketu. Roztrzęsiony powlókł się do wozu. Przybity przeżyciami ostatnich godzin oparł głowę na kierownicy i zapadł w głęboki sen.

Poczuł silne szarpnięcie. Przez chwilę wydawało mu się, że walczy z niewidzialnym wrogiem. Kiedy nareszcie otworzył oczy, nie bardzo wiedział, gdzie jest. Trwał przez chwilę w nieruchomym odrętwieniu. Powieki miał jak z kamienia. Wreszcie doszło do niego: pochyla się nad nim, mówiąc coś bez przerwy kierowca, który przed chwilą o mało go nie rozjechał. Przed chwilą..? Spojrzał na zegar... To już siódma rano. Spał więc prawie dobę.

Czego on chce? Acha, żebym mu pozwolić obejrzeć ciężarówkę. Nie widział jeszcze czegoś takiego. Ktoś niedawno też ją oglądał... A ten tu, w ostatniej chwili przed nim zatrzymał wóz.

— Słuchaj przyjacielu, dlaczego chciałeś mnie wczoraj przejechać — wypowiedziane słowa wydały mu się jakieś dziwne, jakby nie swoje. — Przecież wczoraj, o trzeciej, mało mnie nie rozwaliłeś. Tak, tak, tutaj stałem, w tym właśnie miejscu.

Spoglądał w wytrzeszczone ze zdziwienia oczy kolegi po fachu. Tłumaczył mu przez chwilę raz jeszcze, jak to było, ale cierpliwość tamtego, z każdym jego słowem zmieniała się w złość. Popukał się wymownie palcem w czoło i mamrocząc coś o pomyleńcach — odszedł pospiesznie.

Dlatego uznał za stosowne zgłosić się do szpitala? To przez te paskudne dolegliwości: mdłości, potworny ból głowy, ogólne rozbicie. Opowiedział dokładnie to, co pamiętał. Nic więcej naprawdę nie wie. Została, gdzieś w pamięci, jedynie ta garść faktów, które z niemałym trudem starał się teraz wyłuskać i ułożyć w jakąś logiczną całość. O niczym więcej nie wie... Niczego już więcej nie potrafi sobie przypomnieć.

— I co pan o tym myśli, dyrektorze? — w głosie ordynatora, który asystował przy kolejnej próbie wydobywania z niezwykłego pacjenta, jakim był Hary Joe Turner, informacji na temat zadziwiających historii, zabrzmiała jakby nutka niepokoju. Co dalej robić z tym przypadkiem?

— Myślę, że już nic więcej nam nie powie. Może pan doktorze obudzić Joego z hipnozy. Wygląda to tak, jakby ktoś zablokował mu świadomość... Ale dlaczego do licha, wypuściliście go tak wcześnie ze szpitala — znany futurolog, dyrektor stanowy Północno-Zachodniej Wirginii, Fred Whiting, który prowadził dochodzenie w tej sprawie, nie ukrywał irytacji.

— Żądał tego — tłumaczył ordynator. — Był zdenerwowany, opowiadał, że oni jeszcze czegoś od niego chcą. Że nie znajdzie spokoju, póki nie spełni ich żądania. Nie muszę chyba panu tłumaczyć, iż niektórzy nasi lekarze patrzyli na niego z niechęcią, uważali to wszystko jedynie za pozę, szpanowanie.

— Tak, rozumiem. — Dyrektor wziął do ręki arkusz z ostatnimi „zeznaniami" Turnera. — Wszystko się powtórzyło — mówił nie odrywając wzroku od gęsto zapisanych kartek. Znów to żółte światło dopadło go w tym samym miejscu — 18 mil do Winnchesteru. Znalazł się ponownie w kopulastej hali. Teraz jednak mniej już się bojąc, sam wysiadł z samochodu i zapytał ze zdziwieniem o co chodzi: Czy niemało tych kłopotów, których narobili.

Twierdzi, że otrzymał telepatyczną odpowiedź, że istoty, które go porwały, bardzo przepraszają, ale zwracając samochód zapomniano, go wysterylizować. Na jego oczach jakaś niewidzialna siła odwróciła wóz do góry kołami. Słoje z ketchupem i musztardą, które przewoził, wymieszały się dokładnie. Samochód został zlany jakąś dziwną cieczą... Po pewnym czasie znalazł się znów na szosie, dokładnie w tym samym miejscu, skąd został porwany po raz pierwszy. Na siedemnastej mili...

— I co pan na to, dyrektorze? — powtórzył pytanie ordynator. Ale teraz w jego głosie było zaciekawienie.

— Można by — dyrektor mówił wolno, cedząc każde słowo — uznać to za wierutne brednie, wymysł chorej wyobraźni, gdyby nie pewne fakty, niepodważalne, choć trudne do wytłumaczenia. Przeprowadzono bardzo dokładną ekspertyzę samochodu, zaraz po tym, jak powiadomiliście nas o swym dziwnym pacjencie. Już podczas wstępnych badań odkryto niezwykłości, dla których trudno znaleźć jakieś logiczne wytłumaczenie. Wnętrze kabiny, wszystkie metalowe i szklane powierzchnie pokryte zostały warstewką spalenizny. Jakby w szoferce eksplodował ładunek elektryczny, np. piorun kulisty. Ale wtedy Turner by już nie żył.

Taki sam nalot znaleziono na skórze kierowcy. Antena radiowa na zewnątrz samochodu została częściowo stopiona. Licznik wskazywał, że samochód przejechał jedynie 17 mil z 80 dzielących go od miasta. W jaki sposób więc się tam znalazł? Mam też meldunek od patrolu policyjnego, który nadjechał akurat wtedy, kiedy Turner wysiadał z samochodu, aby obejrzeć go po drugiej przygodzie. Policjanci piszą o swym zdziwieniu. Zarówno olbrzymia ciężarówka, trzydziestotonowy ładunek, wnętrze szoferki oraz sam Turner ociekali jakąś szybko parującą, eterycznie pachnącą cieczą.

Wie pan dobrze, że teren, przez który jechał samochód, jest pustynny, nigdzie w pobliżu nie znaleziono żadnego zbiornika wodnego. Żeby tak zmoczyć wóz, trzeba by tej cieczy ogromną ilość. W ciężarówce nie znaleziono także żadnego pojemnika.

— Cóż to więc było, dyrektorze? — nie dawał za wygraną ordynator.

— Nie wyjaśnione zjawisko. UFO. Tak to trzeba zakwalifikować. W holu czekają dziennikarze „Evening Star" z Winnchester i „Times Dispatch" z Richmond. Będzie pan z nimi rozmawiał?

— Oczywiście. Chodźcie i wy, panowie.

Anna KALETA

Oto gra dla dwóch osób. Każdy z graczy prowadzi swoją linię czterema klawiszami (w prawo, w lewo, w dół i w górę). Przegrywa ten gracz, który pierwszy przetnie swoją linią, linię przeciwnika lub linię pola, po którym poruszają się obaj gracze. W wypadku gdy gracze spotkają się — obaj przegrywają. Gra ta posiada 9 stopni trudności.

K. POŹNIAK

```
  10 REM GRA W DWIE LINIE
  15 INK 7: PAPER 0: BORDER 0: C
LS
  20 PRINT AT 3,6,"'GRA W DWIE L
INIE'"
  21 PRINT AT 7,4;"W GRE GRA DWO
CH GRACZY"
  22 PRINT TAB 0;"  TEN GRACZ PR
ZEGRYWA, KTORY         PIERWSZY PR
ZETNIE LINIE"
  23 PRINT AT 13,4;"GRACZ 1
   GRACZ 2"
  24 PRINT TAB 3;"lewo - 'w'
 lewo - 'o'"
  25 PRINT TAB 2;"prawo - 'e'
 prawo - 'p'"
  26 PRINT TAB 3;"gora - 'q'
 gora - 'i'"
  27 PRINT TAB 4;"dol - 'a'
 dol - 'k'"
  30 INPUT "  STOPIEN TRUDNOSCI
(1,9) ? ";t: IF t<0 OR t>9 OR t<
>INT t THEN GO TO 30
  35 CLS
  36 LET stop1=0: LET stop2=0
  40 LET gx=INT (255/t)*t
  41 LET gy=INT (175/t)*t
  50 PLOT 0,0: DRAW gx,0: DRAW 0
,gy: DRAW -gx,0: DRAW 0,-gy
  60 LET x1=t: LET y1=t
  61 LET x2=gx-t: LET y2=gy-t
  65 LET kx1=t: LET ky1=0
  66 LET kx2=-t: LET ky2=0
 100 IF IN 64510=190 THEN LET kx
1=0: LET ky1=t: GO TO 150
 110 IF IN 64510=189 THEN LET kx
1=-t: LET ky1=0: GO TO 150
 120 IF IN 64510=187 THEN LET kx
1=t: LET ky1=0: GO TO 150
 130 IF IN 65022=190 THEN LET kx
1=0: LET ky1=-t: GO TO 150
 150 IF IN 57342=187 THEN LET kx
2=0: LET ky2=t: GO TO 200
 160 IF IN 57342=189 THEN LET kx
2=-t: LET ky2=0: GO TO 200
 170 IF IN 57342=190 THEN LET kx
2=t: LET ky2=0: GO TO 200
 180 IF IN 49150=187 THEN LET kx
2=0: LET ky2=-t: GO TO 200
 200 LET nx1=x1+kx1: LET ny1=y1+
ky1
 201 LET nx2=x2+kx2: LET ny2=y2+
ky2
 210 IF POINT (nx1,ny1) THEN LET
 stop1=1
 211 IF POINT (nx2,ny2) THEN LET
 stop2=1
 212 IF nx1=nx2 AND ny1=ny2 THEN
 LET stop1=1: LET stop2=1
 220 PLOT x1,y1: DRAW kx1,ky1: L
ET x1=nx1: LET y1=ny1
 221 PLOT x2,y2: DRAW kx2,ky2: L
ET x2=nx2: LET y2=ny2
 230 IF NOT stop1 AND NOT stop2
THEN GO TO 100
 240 FLASH 1: PRINT AT 8,6;
 250 IF stop1 AND stop2 THEN PRI
NT "OBAJ GRACZE PRZEGRALI"
 260 IF stop1 AND NOT stop2 THEN
 PRINT "WYGRAL GRACZ DRUGI"
 270 IF stop2 AND NOT stop1 THEN
 PRINT "WYGRAL GRACZ PIERWSZY"
 280 FLASH 0: BRIGHT 1: INVERSE
1: PRINT AT 12,9;"WCISNIJ 'ENTER
'": BRIGHT 0: INVERSE 0
 290 IF IN 49150=190 THEN RUN
 300 GO TO 290
```

Systemy na sprzedaż

W świecie informatyki, komputerów i oprogramowania obowiązują takie same prawa i zasady, jak w każdej innej, mogącej przynieść zyski, dziedzinie. Po nasyceniu rynków wewnętrznych najnowszymi osiągnięciami myśli naukowo-technicznej, firmy — zarówno najpotężniejsze, jak i te całkiem małe — zaczynają szukać możliwości rozszerzenia pola działania, co w tym wypadku oznacza zdobycie nowych rynków zbytu. Coraz częściej są to przedsięwzięcia na dużą skalę, przynoszące realne korzyści obu uczestnikom transakcji.

Do takich właśnie udanych przedsięwzięć można z pewnością zaliczyć zawartą właśnie w Moskwie po czterech miesiącach negocjacji umowę, o zastosowaniu brytyjskiego systemu expert jako standardu przy opracowywaniu tego typu programów w Związku Radzieckim. Opiewający na ponad 3 miliony funtów kontrakt przewiduje zacieśnienie współpracy między brytyjskimi i radzieckimi programistami w ciągu najbliższych trzech lat. Zgodnie z osiągniętym porozumieniem, ZSRR będzie stosował program Xi Plus firmy Expertech z Berkshire, jako podstawę przy wszelkiego typu pracach związanych z tym systemem. Umowa ma także dalsze konsekwencje — wszystkie firmy zagraniczne, zainteresowane współpracą ze Związkiem Radzieckim, będą w pewnym sensie zmuszone do przechowywania danych i przetwarzania informacji właśnie w systemie Xi Plus.

Jeszcze w tym roku, radzieccy specjaliści do oprogramowania przybędą do Wielkiej Brytanii w celu doskonalenia umiejętności odpowiedniego wykorzystania systemu Xi. Do jego zapisu używa się jeszcze obecnie dwóch języków — C i Prolog, ale eksperci Expertechu opracowują nową wersję, w całości zakodowaną przy użyciu C.

Podpisane w Moskwie porozumienie jest jednym z największych transferów brytyjskiego software do Związku Radzieckiego i największym radzieckim zakupem importowym w dziedzinie expert system. Firma Expertech zajmuje się także opracowywaniem programów dla joint-ventures, w którego działalność zaangażowany jest Algorytm — przedsiębiorstwo zajmujące się doskonaleniem radzieckiej techniki software, działające pod auspicjami Państwowego Komitetu ds. Informatyki i Komputeryzacji, utworzonego w 1986 roku.

Brytyjczycy nie ukrywają zadowolenia z zawartej umowy. Wiążą z jej podpisaniem olbrzymie nadzieje. Chodzi przecież o naprawdę dużą rzecz — rozpowszechnienie w całym, tak dużym, kraju zakupionego u nich systemu. Korzyści można upatrywać także z innych powodów. Kontrakt otwiera szeroko drzwi do radzieckiego know-how w takich dziedzinach, jak medycyna, rolnictwo czy oświata, których zasady mogą docierać do Wielkiej Brytanii i innych państw europejskich, właśnie w postaci zapisu w systemie Xi Plus. W firmie Expertech trwają właśnie badania nad zastosowaniem zautomatyzowanego tłumacza, który w znacznym stopniu uprości przechodzenie z angielskiej na rosyjską wersję systemu, z możliwością ponownego przetłumaczenia na wersję angielską.

Radziecko-brytyjskie przedsięwzięcie, oprócz wymienionych już korzyści dla obu stron, jest także potwierdzeniem faktu, że osiągnięcia najnowszej techniki mogą służyć nawiązywaniu bliskich kontaktów w dziedzinie ekonomiczno-gospodarczej.

Małgorzata NOWOTNY

ZX Spectrum

Warto przypomnieć Czytelnikom ten mało znany program z roku 1983 pokazujący w górnym prawym rogu ekranu numer aktualnie wykonywanego wiersza

Program uruchamia się przez RANDOMIZE USR 65250, a zatrzymuje przez RANDOMIZE USR 65257

Jan RUBINOWICZ

```
   1 DATA 1841,62,254,237,71,237
,94,201,237,86,62,63,237
   2 DATA 1546,71,201,255,245,22
9,42,69,92,36,40,69,197
   3 DATA 1047,213,37,175,24,2,2
40,254,71,79,17,232,3
   4 DATA 1743,237,62,60,48,251,
25,61,229,205,65,255,225
   5 DATA 1070,17,100,0,175,237,
62,60,48,251,25,61,14
   6 DATA 1299,1,229,205,65,255,
225,17,10,0,125,147,20
   7 DATA 1447,48,252,21,131,103
,229,14,2,122,205,65,255
   8 DATA 1885,241,14,3,205,65,2
55,209,193,225,241,201,33
   9 DATA 1051,28,64,9,135,135,1
35,235,79,33,128,61,9
  10 DATA 786,6,8,126,47,18,35,2
0,16,249,201,0,60
  15 RESTORE : CLEAR 65249: LET
adr=65250
  16 FOR n=1 TO 10: READ suma: L
ET s=0: FOR m=1 TO 12:: READ bt:
 POKE adr,bt: LET s=s+bt: LET ad
r=adr+1: NEXT m: IF s<>suma THEN
 PRINT "Blad w linii ";n: LIST n
: STOP
  17 NEXT n: RESTORE : RANDOMIZE
 USR 65250
  30 REM  Program demonstracyjny
  40 GO SUB 9800: PAUSE 50: GO T
O 300
  50 GO SUB 9800: PAUSE 50: GO T
O 5550
  70 GO SUB 9800: PAUSE 50: GO T
O 40
 300 GO SUB 9800: PAUSE 50: GO T
O 9000
5550 GO SUB 9800: PAUSE 50: GO T
O 70
9000 GO SUB 9800: PAUSE 50: GO T
O 50
9800 PAUSE 20: IF INKEY$<>"" THE
N RANDOMIZE USR 65257: PRINT "cz
y wlaczyc program ? (T/N)": PAUS
E 0: IF INKEY$="t" THEN RANDOMIZ
E USR 65250
9801 LET b=INT (RND*7): BORDER b
: BEEP .05,b: CLS : RETURN
9997 REM  Zapis na tasme
9998 SAVE "TRASAcode"CODE 65250,
118: STOP
9999 SAVE "TRASA" LINE 1:
```

W poprzednim numerze rozpoczęliśmy drukować hasła zaczynające się na literę „F", niestety i tu namieszał złośliwy Chochlik — przez nieuwagę opuściliśmy pierwsze hasła tej litery. Dziś powtarzamy.

F

FABRIC - budowa, struktura,
FABRICATION - wyrób, fabrykacja,
FACE - 1. druk, nadruk, 2. powierzchnia czołowa, czoło, lico, 3. licować, okładać,
FACE BONDING - montaż czołowy,
FACE-CHANGE CHARACTER - znak zmiany rodzaju druku,
FACE DOWN - nadrukiem do dołu,
FACEDOWN FEED - podawanie (kart) prawą stroną do dołu,
FACE OF A CARD - prawa (w znaczeniu: właściwa) strona karty,
FACE OF THE DISC - górna powierzchnia dysku,
FACE UP - prawą stroną do góry,
FACEUP FEED - podawanie (kart) prawą stroną do góry,

FACILITATE - ułatwiać,
FACILITY - 1. urządzenie (pomocne, ułatwiające), aparatura, 2. sposoby,/możliwości, środki (łączności), 3. połączenie, łączność, 4. udogodnienie, ułatwienie,

FACSIMILE - faksymile (dokładna reprodukcja, podobizna, kopia), także: symiliografia,
FACSIMILE TELEGRAPHY - telegrafia kopiowa, symilografia,
FACSIMILE TRANSMISSION - nadawanie obrazów nieruchomych,
FACSIMILE TRANSMITTER - aparat telekopiowy, symilograf,

FACTOR - czynnik, współczynnik,
FACTORABLE - rozkład na czynniki,
FACTOR GROUP - grupa ilorazowa,
FACTORIAL - silnia,
FACTORING - rozkładanie na czynniki,
FACTORIZATION - patrz: FACTORING,
FACTOR OF AN INTEGER - dzielnik liczby całkowitej,

FADE - zanikać, także: tracić zabarwienie, blaknąć,
FADE-DOWN - obniżanie się poziomu mocy wyjściowej,
FADE-IN - wzmacnianie sygnału, także: rozjaśnianie obrazu,
FADE-OUT - zanikanie sygnału, także: ściemnianie obrazu,
FADING - zanik sygnału, także: ściemnianie lub rozjaśnianie obrazu.

FAIL - ulegać awarii, także: załamać się, zawieść,
FAILPROOF - niezawodny, także: zabezpieczony przed uszkodzeniami,
FAIL-SAFE - odporny na uszkodzenia, bezpieczny w razie uszkodzenia,
FAIL-SOFT - częściowo odporny na uszkodzenia (pozwalający na działanie systemu w ograniczonym zakresie, w razie uszkodzenia niektórych elementów),
FAIL-SOFT SYSTEM - system częściowo odporny na uszkodzenia,
FAIL TO COMPLY WITH ... - nie spełniać (warunków), nie odpowiadać (wymaganiom),
FAIL-TO-EXHAUSTION - błędy do usunięcia,

FAILURE - uszkodzenie, defekt, niezadziałanie,
FAILURE FLOW - strumień uszkodzeń,
FAILURE-FREE - bezawaryjny, bezusterkowy,
FAILURE RATE - wskaźnik awaryjności,
FAIRLY CLEAR CARBON - dość dobra kopia (odbitka),
FALL - spadek, zmniejszenie się, obniżenie, także: spadać, opadać, obniżać się, zmniejszać się.

FALLBACK - rezerwa awaryjna, przejście na awaryjny tryb pracy, zmiana trybu pracy lub konfiguracji sprzętu w celu uniknięcia niesprawności,
FALLOUT - znikanie, zanikanie, przepadanie,

FALSE - fałszywy, także: fałsz (pojęcie używane w logice),
FALSE ACCEPTANCE - przyjęcie fałszywej hipotezy,
FALSEHOOD - przekłamanie, fałszywy wniosek, fałszywa decyzja,
FALSE REJECTION - odrzucenie hipotezy fałszywej,
FALSE RETRIEVAL - uzyskanie niepotrzebnej informacji w automatycznym systemie wyszukiwania,
FALSE STATEMENT - twierdzenie fałszywe,

FAMILY - rodzina,
FAMILY OF CHARACTERISTICS - rodzina charakterystyk,
FAMILY OF CURVES - rodzina krzywych,
FANFOLD - druk ciągły składany w "harmonię",
FANFOLD FORM - formularze ciągłe złożone w postaci harmonii,

FAN-IN - obciążalność wejściowa (bramki logicznej),
FAN-OUT - obciążalność wyjściowa (bramki logicznej),
FAR CALL - wywołanie długie, wywołanie odległe,
FAR END CROSS TALK - przesłuch skrośny,
FAR PLANE - daleki plan (w grafice komputerowej określenie płaszczyzny ograniczającej przestrzeń zobrazowania na ekranie),
FAR-REACHING - dalekosiężny,

FAST - szybki, szybko,
FAST-ACCESS STORAGE - pamięć szybka, pamięć o dostępie szybkim,
FAST-ACTING - szybki, szybko działający,
FAST LINE PRINTER - szybka drukarka wierszowa,
FAST MEMORY - szybka pamięć (operacyjna),
FAT - patrz: FILE ALLOCATION TABLE,

FATAL ERROR - błąd uniemożliwiający kontynuację programu,
FATHER FILE - początkowa (wyjściowa) wersja pliku (zbioru),
FAULT - 1. błąd, 2. uszkodzenie, zakłócenie, niesprawność, 3. wada (fizyczna wada sprzętu),
FAULT CLEARING - likwidacja zakłóceń, usuwanie uszkodzeń,
FAULT DETECTOR - wskaźnik uszkodzeń,
FAULT FINDING - wykrywanie uszkodzeń, także: usuwanie zakłóceń,
FAULTLESS - bezusterkowy, bezbłędny,
FAULTLESS TAPE - taśma sprawdzona, taśma bezbłędna,
FAULT LOCATION - lokalizacja uszkodzeń,
FAULT SIMULATOR - symulator uszkodzeń, symulator zakłóceń,
FAULT TIME - czas przestoju z powodu uszkodzenia,
FAULT-TOLERANT SYSTEM - system odporny na uszkodzenia,
FAULT TRACING - wyszukiwanie uszkodzeń,
FAULT TREE GATE - element drzewa niesprawności (usterek),

FAULTY - 1. błędny, 2. uszkodzony, zakłócony, niesprawny, 3. wadliwy,
FAULTY OPERATION - działanie wadliwe, działanie błędne,
FAX - patrz: FACSIMILE,
FAY - przylegać, dokładnie pasować,
FC - patrz: FONT-CHANGE CHARACTER,
FCB - patrz: FILE CONTROL BLOCK,
FCFS - patrz: FIRST-COME FIRST-SERVED,
FCP - patrz: FILE CONTROL PROCESSOR,

FCS - patrz: FRAME CHECK SEQUENCE,
FD - patrz: FILE DESCRIPTION,
FDB - patrz: FILE DESCRIPTION BLOCK,
FDM - patrz: FREQUENCY DIVISION MULTIPLEXING,
FEASIBLE - możliwy (do wykonania), wykonalny, realny, dopuszczalny,
FEASIBLE SOLUTION - rozwiązanie dopuszczalne,
FEATURE - cecha, właściwość, także: możliwość, możność,
FEED - 1. magazynek podawczy kart, 2. zasilanie maszyny (np. kartami, drukami, papierem), 3. dostarczać, wprowadzać (np.dane do komputera),
FEEDBACK - sprzężenie zwrotne,
FEEDBACK CONTROL - sterowanie ze sprzężeniem zwrotnym, sterowanie w układzie zamkniętym,
FEED BACK ELEMENT - element sprzęgający,
FEEDBACK FACTOR - współczynnik sprzężenia zwrotnego,
FEEDBACK FILTER - filtr sprzężenia zwrotnego,
FEEDER - 1. zasilacz, 2. podajnik, 3. dozownik, 4. przewód zasilający,
FEEDFORWARD - uprzedzanie, poprzedzanie, zapobieganie, wyprzedzenie,
FEED HOLE - dziurka prowadząca, dziurka przesuwu,
FEED HOPPER - kaseta podająca,
FEEDING - zasilanie, podawanie, wprowadzanie,
FEEDING INFORMATION - dostarczanie danych, wprowadzanie danych,
FEED INTO A COMPUTER - wprowadzić do (pamięci) komputera,
FEED PITCH - odstęp linii,
FEED RACK - gąsienica przesuwu, traktor przesuwu,

FEED SPOOL - szpula, z której taśma się rozwija,

FEEDTHROUGH - bezpośrednie połączenie, bezpośrednie łącze, połączenie skrośne, łącze przelotowe,

FEED TRACK - ścieżka prowadząca,

FEELOW - członek towarzystwa naukowego,

FEP - patrz: FRONT-END PROCESSOR,

FERRITE - ferryt, także: ferrytowy,

FERRITE CORE - rdzeń ferrytowy,

FERRITE-CORE STORE - pamięć ferrytowa,

FERRITE SWITCHING CORE - rdzeń ferrytowy przełączający,

FERROELECTRIC - ferroelektryczny,

FERROGRAPHY - ferrografia,

FERROMAGNETIC - ferromagnetyczny,

FERROUS COATED TAPE - taśma z warstwą tlenkową,

FET - patrz: FIELD EFFECT TRANSISTOR,

FETCH - ładować do pamięci, pobierać, także: cykl maszynowy odczytu kodu rozkazu,

FETCH CYCLE - cykl maszynowy odczytu kodu rozkazu,

FETCHING - patrz: FETCH,

FETCH TIME - czas ładowania (wczytania) do pamięci,

F/F - patrz: FLIP-FLOP,

FF - patrz: FIRST FIT ALGORITHM,

FF - patrz: FORM FEED,

FFD - patrz: FIRST FIT DECREASING ALGORITHM,

FIB - patrz: FILE IDENTIFICATION BLOCK,

FIBERS - włókno szklane, także: światłowód,

FIBER-GLASS FILAMENT - włókno szklane,

FIBONACCI NUMBERS - liczby Fibonacciego,

FIBRE - patrz: FIBER,

FIBRE OPTICS - technika światłowodowa,

FICHE - mikrofilm, mikrofotokopia,

FIDELITY - dokładność, wierność (odtwarzania),

FIELD - pole, obszar w zapisie (rekordzie),

FIELD EFFECT TRANSISTOR - tranzystor polowy,

FIELD EXTRACTOR - czynność wyboru pola zapisu (rekordu),

FIELD MARK - cecha pola, znacznik pola,

FIELD OF KNOWLEDGE - dziedzina wiedzy,

FIELD-PROGRAMMABLE - programowany w warunkach eksploatacji,

FIELD-PROGRAMMABLE LOGIC ARRAY - tablica logiczna programowalna przez użytkownika,

FIELD-PROGRAMMABLE ROM - pamięć stała reprogramowalna polem,

FIELD SELECTION - wybór (selekcja) pola zapisu (rekordu),

FIFO - patrz: FIRST-IN FIRST-OUT,

FIFO LIST - kolejka typu FIFO,

FIFO PROCESSING - przetwarzanie (obsługa) według kolejności typu FIFO,

FIFO QUEUE - patrz: FIFO LIST,

FIFTH-GENERATION COMPUTER - komputer piątej generacji,

FIGURATIVE - obrazowy,

FIGURATIVE CONSTANT - stała symboliczna,

FIGURE - figura, także: cyfra,

FIGURE OUT (UP) - obliczać, wyliczać,

FIGURES KEY - klawisz cyfrowy,

FIGURES SHIFT - przełączanie na cyfry,

FIGURE STAMP - znacznik cyfrowy,

FIGURING - pismo cyfr, także: cyfry skali,

FILE - plik, zbiór,

FILE ADDRESSING - metoda dostępu,

FILE ALLOCATION - rozmieszczenie zapisów (rekordów) pliku (zbioru),

FILE ALLOCATION ROUTINES - programy rezerwacji pliku (zbioru),

FILE ALLOCATION TABLE - tablica rezerwacji plików (zbiorów),

FILE AREA - obszar pliku (zbioru),

FILE ATTRIBUTE - cecha pliku (zbioru),

FILE BACKUP - składowanie zbioru,

FILE CONTROL BLOCK - blok kontrolny pliku (zbioru),

FILE CONTROL PROCESSOR - program powtarzalny do otwierania i zamykania zbiorów danych,

FILE CREATION - zakładanie pliku (zbioru),

FILE DEFINITION AREA - opis pliku (zbioru),

FILE DESCRIPTION - opis pliku (zbioru),

FILE DESCRIPTION BLOCK - blok opisu pliku (zbioru),

FILE DEVICE - urządzenie plikowe, urządzenie o strukturze pliku,

FILE DIRECTORY - katalog plików (zbiorów), kartoteka plików (zbiorów),

FILE EVENT - zwrócenie się do pliku (zbioru),

FILE EXCHANGE UTILITY - program kopiowania plików (zbiorów) z możliwością zmiany struktury,

FILE FORMAT - struktura pliku (zbioru),

FILE HANDLER - manipulator zbiorów,

FILE HEADER LABEL - etykieta początku zbioru danych,

FILEID - identyfikator pliku (zbioru),

FILE IDENTIFICATION BLOCK - blok określający (identyfikujący) plik (zbiór),

FILE LABEL - etykieta (znacznik) pliku (zbioru),

FILE LAYOUT - rozplanowanie układu pliku (zbioru),

FILE LOCKING - zajęcie pliku (zbioru), zablokowanie pliku (zbioru),

FILE MAINTENANCE - aktualizacja zbioru danych,

FILE MANAGER - określenie części systemu operacyjnego odpowiedzialnego za wykonywanie operacji na plikach (zbiorach),

FILEMARK - znak ostatniego zapisu (rekordu) w pliku (zbiorze), także: znacznik sekcji,

FILE MERGING - scalanie pliku (zbioru),

FILE NAME - nazwa zbioru (pliku) danych,

FILE ORGANIZATION - organizacja pliku (zbioru),

FILE ORGANIZATION CODE - symbol organizacji pliku (zbioru),

FILE-ORIENTED - przeznaczony do pracy ze zbiorami,

FILE OVERFLOW AREA - obszar przepełnienia pliku (zbioru),

FILE-PROCESSING - przetwarzanie pliku (zbioru),

FILE PROTECTION - zabezpieczenie pliku (zbioru) danych,

FILE-PROTECTION RING - pierścień zezwalający na zapis, pierścień zapisu,

FILE QUALIFIER - kwalifikator pliku (zbioru), klucz pliku (zbioru),

FILER - patrz: FILE MANAGER,

FILE SECURITY - zabezpieczenie programowe pliku (zbioru), także: ograniczenie dostępu do pliku (zbioru),

FILE SEPARATOR - patrz: FILE SEPARATOR CHARACTER,

FILE SEPARATOR CHARACTER - znak rozdzielający pliki (zbiory), znak oddzielający pliki (zbiory),

FILE SEQUENCE - porządek kartoteki,

FILE SHARING - rozdział zasobów pliku (zbioru),

FILE SORTING - sortowanie pliku (zbioru),

FILE SPECIFIER - wyróżnik pliku (nazwa pliku i opcjonalnie oznacznik urządzenia),

FILESTORE - pamięć zbiorów systemu (np. w SO GEORGE-3), także: magazyn, skład plików (zbiorów),

FILE-STRUCTURED - o organizacji plikowej, o strukturze plikowej,

FILE SYSTEM - patrz: FILE MANAGER,

FILE TRAILER IDENTIFIER - etykieta końcowa,

FILETYPE - identyfikator typu pliku (zbioru),

FILE UNIT - zewnętrzna pamięć masowa,

FILE UPDATING - aktualizacja pliku (zbioru),

FILING - kartotekowanie, segregowanie dokumentu, ewidencjonowanie, także: zapis do pliku (zbioru),

FILL - dowolny symbol wypełniający lub uzupełniający,

FILL-AREA ATTRIBUTE - atrybut zamazywania, atrybut maskowania (w grafice sposób wydzielenia obszaru obrazu),

FILL CHARACTER - znak wypełniający, znak pusty,

FILLER - znak wypełniający, znak pusty, wypełniacz,

FILL THE FORM - wypełnić formularz,

FILL UP - dopełnić, uzupełnić,

FILO - patrz: FIRST-IN LAST-OUT ,

FILTER - filtr, także: filtrować,

FINAL - końcowy,

FINAL CONTROL ELEMENT - człon wykonawczy,

FINAL SEGMENT OF SET - reszta zbioru,

FIND - znaleźć, odkryć, wykryć,

FINDABLE - wykrywalny,

FIND APPLICATION - znajdować zastosowanie,

FINDER - szukacz, także: wyznacznik,

FINDING - wnioskowanie (w systemach ekspertowych),

FINDINGS - 1. wyniki poszukiwań, wyniki badań, 3. dane uzyskane, 4. wnioski,

FIND OUT - dowiedzieć się, wykryć,

FIND USE - znajdować zastosowanie,

FINE INDEX - indeks szczegółowy, indeks wtórny,

FINISH - kończyć, skończyć,

FINITE - określony, skończony,

FINITE-DIFFERENCE METHOD - metoda różnic skończonych,

FINITE-ELEMENT METHOD - metoda projekcyjna, także: metoda różnicowo-projekcyjna,

FINITELY - skończenie,

FINITE NUMBER - liczba skończona,

FINITE SET - zbiór skończony,

FINITE-STATE AUTOMATON - automat skończony, automat ze skończoną ilością stanów,

FINITE-STATE GRAMMAR - gramatyka o skończonej liczbie stanów, także: gramatyka automatów,

FINITE-STATE MACHINE - patrz: FINITE-STATE AUTOMATON,

FINUFO - patrz: FIRST-IN-NOT-USED-FIRST-OUT,

FIRM - firma, także: zwarty, zwięzły, mocny,

FIRMWARE - oprogramowanie układowe, oprogramowanie sprzętowe, oprogramowanie mikrokomputerowe, mikrooprogramowanie,

FIRST BUCKET IN FILE AREA MARKER - wskaźnik pierwszego bloku w obszarze pliku (zbioru),

LIGA MYŚLĄCYCH

ZADANIE 1

Pewien obywatel podejmuje z końcem każdego roku ze swego wkładu złożonego w kasie oszczędności na 3%, 90.— zł z każdego tysiąca wkładu. Po jakim czasie podejmie on cały wkład?

ZADANIE 2

Piłka gumowa odbija się na $^2/_3$ wysokości, z której była puszczona. Piłkę puszczono z wysokości 8,1 metra. Po ilu odbiciach piłka wzniesie się na wysokość tylko 1,6 metra?

ZADANIE 3

Przed jednym z bardziej okazałych budynków miasta znajdują się schody. Gdyby liczbę stopni schodów o wysokości 3,5 metra zwiększyć o 10, to wysokość każdego stopnia zmniejszyłaby się o 4 cm. Ile stopni mają schody?

ZADANIE 4

Znaleźć liczbę dwucyfrową, jeżeli iloraz z podzielenia tej liczby przez iloczyn jej cyfr wynosi $2^2/_3$, a różnica między szukaną liczbą i liczbą o tych samych cyfrach, lecz napisanych w porządku odwrotnym, równa się 18

ZADANIE 5

Pociąg A o prędkości V km/h wyjeżdża ze stacji po odejściu pociągu B, jadącego z prędkością V_1 km/h. Godzina odejścia pociągu A była tak dobrana, aby oba pociągi przyjechały jednocześnie do stacji końcowej. Pociąg B przejechawszy $^2/_3$ drogi zmuszony był zmniejszyć prędkość o połowę i wskutek tego pociągi spotkały się a kilometrów przed stacją końcową. Wyznaczyć odległość między tymi stacjami

Rozwiązanie zadań prosimy przesyłać do redakcji do końca września br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

Odwracanka *ZX Spectrum*

W „IKS-ie" (2/86) opublikowano program „odwracanka". Jest to gra na mikrokomputer Atari. Na podstawie podanego opisu opracowałem uproszczoną wersję programu na ZX Spectrum (wydruk 1). W zeszycie „IKS" — programy 2 zamieszczona została inna wersja tej gry („Rubik") na ZX Spectrum. Ta wersja jest wygodniejsza ze względu na inny sposób wyboru odwracanego pola. Prezentowany program w minimalnym stopniu wykorzystuje komputer. Wszelkie decyzje podjął programista. W czasie pracowitego wpisywania linii programu. Można ten program skrócić w sposób dość znaczny. Wystarczy pola planszy oznaczyć kolejnymi literami alfabetu (nie pomijając V). Współrzędne i kształt „Krzyża" może wyznaczyć komputer (wydruk 2)

Wiesław RYCHLICKI

```
   0>REM W.RYCHLICKI 1987
   5 BORDER 0: PAPER 0: INK 7
  10 FLASH 0: BRIGHT 0: OVER 1
  15 LET a$="████████████"
  20 CLS : GO SUB 1000
  25 POKE 23658,8
  30 LET w$=INKEY$
  40 IF w$="0" THEN GO TO 20
  50 IF w$<"A" OR w$>"Y" THEN
     GO TO 30
  60 LET x=CODE w$-65
  70 LET y=4*INT (x/5)
  80 LET x=4*x-5*y
  90 GO SUB 2000: BEEP .2,15
 100 GO TO 30
 110
1000 FOR i=0 TO 5
1010 PLOT i*32,175: DRAW 0,-160:
     PLOT 0,i*32+15: DRAW 160,0
1020 NEXT i
1030 PRINT #0;AT 0,0;
     "0  START GRY OD POCZATKU"
1040 FOR i=0 TO 4: FOR j=0 TO 4
1050 PRINT AT 4*i+1,4*j+1;
     CHR$ (65+5*i+j)
1060 NEXT j: NEXT i
1070 RETURN
1080
2000 LET a=y: LET b=x-(x>0)*4
2010 LET c=8+(x>0 AND x<16)*4
2020 GO SUB 2100
2030 LET b=x: LET c=4
2040 IF y>0 THEN LET a=y-4:
     GO SUB 2100
2050 IF y<16 THEN LET a=y+4:
     GO SUB 2100
2060 RETURN
2070
2100 FOR i=0 TO 3
2110 PRINT AT a+i,b;a$( TO c)
2120 NEXT i: RETURN
```

```
   0>REM WIESLAW RYCHLICKI 1986
  10 CLS : PRINT AT 0,5;" O D W
R A C A N I E "
  20 PRINT : PRINT "NA 25 POLOWE
J PLANSZY ZNAJDUJA  SIE KWADRATY
 Z JEDNEJ STRONY    BIALE A Z DR
UGIEJ CZARNE"
  30 PRINT "PODAJAC WSPOLRZEDNE
KWADRATU     ODWRACAMY GO NA DRUG
A STRONE     WRAZ Z ODWRACANYM KW
ADRATEM      ODWRACAJA SIE KWADRA
TY SASIEDNIE"
  40 PRINT "NALEZY ODWROCIC WSZY
STKIE        KWADRATY"
  50 LET A=12: LET B=12
  60 LET PKT=0: FOR I=1 TO 5: PR
INT AT A+I,B;I;"█████";AT A,B+I;
I: NEXT I
  70 INPUT "ODWRACANY KWADRAT"'"
WIERSZ "; LINE A$,"KOLUMNA "; LI
NE B$
  80 IF LEN A$<>1 OR LEN B$<>1 T
HEN GO TO 70
  90 IF A$<"1" OR A$>"5" OR B$<"
1" OR B$>"5" THEN GO TO 70
 100 LET X=VAL A$: LET Y=VAL B$
 110 PRINT AT A+X,B+Y; OVER 1; B
RIGHT 1;"█"
 120 IF X>1 THEN PRINT AT A+X-1,
B+Y; OVER 1; BRIGHT 1;"█"
 130 IF X<5 THEN PRINT AT A+X+1;
B+Y; OVER 1; BRIGHT 1;"█"
 140 IF Y<5 THEN PRINT AT A+X,B+
Y+1; OVER 1; BRIGHT 1;"█"
 150 IF Y>1 THEN PRINT AT A+X,B+
Y-1; OVER 1; BRIGHT 1;"█"
 160 LET PKT=PKT+1: PRINT AT 19,
0;"LICZBA RUCHOW : ";PKT
 170 GO TO 70
```

LOGOGRYF

1						
2 I	N	E	R	C	J	A
3						
4						
5						
6 L	O	T	E	R	I	A

Litery, które znajdą się w oznaczonych polach, czytane rzędami poziomymi, utworzą rozwiązanie zadania

1) głos podobny do gwizdu, 2) bezwładność, 3) domena transportu, 4) roślina z rodziny jaskrowatych o żółtych kwiatach, 5) pochlebca, 6) z losami

„MIREK"

1 S	A	W	A	N	N	A
2 N	A	D	A	W	C	A
3						
4						
5 K	A	S	K	A	D	A

Litery, które znajdują się w oznaczonych polach, czytane rzędami poziomymi, utworzą rozwiązanie zadania

1) zbiorowisko roślinne typowe dla Afryki, 2) o kimś, kto wysyła coś za pośrednictwem poczty, 3) gruczoł wydzielania wewnętrznego, 4) dodatkowa porcja płynu, 5) niewielki wodospad

„MIREK"

Magiczna krzyżówka dwuliterowa

ZNACZENIE WYRAZÓW: 1) artystka malarka, wyspecjalizowana w technice akwarelowej, 2) puszysta tkanina wełniana lub bawełniana, 3) pozostałość pochodząca z dawnych czasów ● naturalna lub zbożowa, 4) może być holownicza ● drąg używany przez linoskoczków, 5) gładziarka, 6) dom zajezdny na Wschodzie, miejsce postoju karawany.

„MIREK"

	1	2	3	4	5	6
1						
2					■	
3				■	KA	WA
4	LI	NA	■			
5		■				
6						

Pocztowa Giełda

Rozwiązanie krzyżówki z nr. 3/88 „IKS-a"

Nagrody książkowe otrzymują: Sławomir Bukowski — Pruszków, Barbara Kozina — Świdnica, Igor Zgoła — Gdynia, Wanda Roguzińska — Pisz, Jacek Pawlik — Pyskowice, Jan Mamajko — Kraków, Andrzej Kmiecicki — Gdynia, Irena Pietruszewicz — Poznań, Piotr Kaczmarek — Leszno, Leszek Knapczyk — Nowy Targ.

Rozwiązanie krzyżówki z nr. 4/88 „IKS-a"

Nagrody książkowe otrzymują: Sewer Nyckowski — Gostynin, Dorota Baranowska — Oleśnica, Dawid Płusa — Jelenia Góra, Joanna Hryniewicka — Białystok, Andrzej Herla — Głogówek, Andrzej Wawrzyński — Gdynia, Piotr Sinielewicz — Lublin, Piotr Rynkowski — Warszawa, Mariola Klimek — Gdańsk-Oliwa, Anna Cembrzyńska — Gorlice.

Rozwiązania zadań (przynajmniej jednego) prosimy nadsyłać pod adresem redakcji do końca września (naklejając kupon „IKS-a"). Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.



„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1658 Nr ind. 361682.

INFORMATYKA KOMPUTERY SYSTEMY



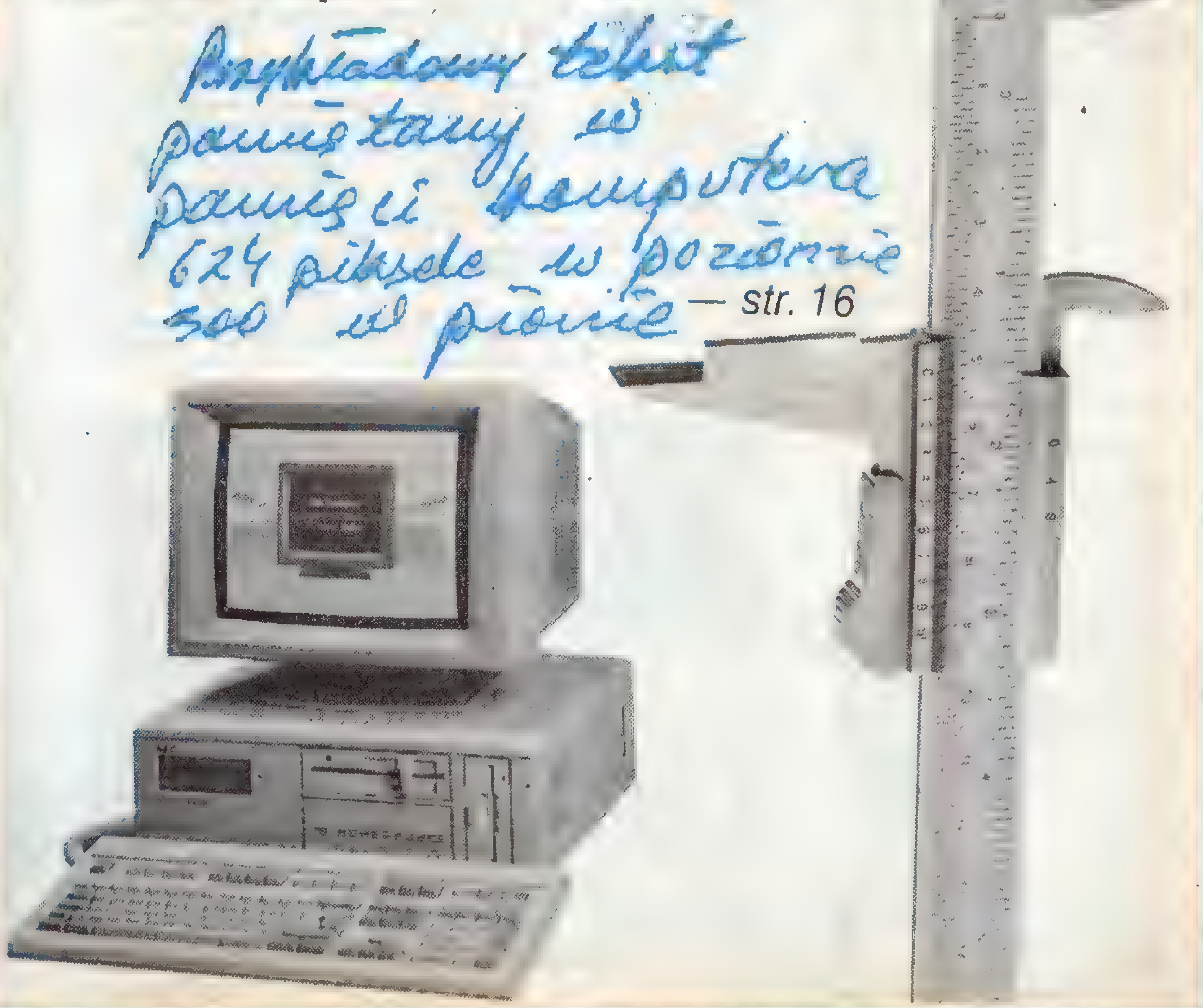
CENA 120 ZŁ

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" NR 9/1988 ISSN 0860 — 2794

Informatyka w szkole
— str. 14

Co widzi komputer

Przykładowy tekst pamiętany w pamięci komputera 624 piksele w poziomie 500 w pionie — str. 16



KOMPUTERY A PSYCHIKA CZŁOWIEKA

Coraz więcej ludzi na całym świecie na co dzień ma bezpośredni kontakt z nowoczesnymi urządzeniami telekomunikacyjnymi i wymiany informacji. W przedsiębiorstwach, biurach, placówkach służby zdrowia i innych instytucjach wykorzystywane są komputery. Automatyzacja i komputeryzacja wkracza także do domów. Pod względem fizycznym z pewnością urządzenia te coraz bardziej ułatwiają pracę i oszczędzają czas. Uczeni zaczynają się zastanawiać, jak nowa technika wpływa na sposób myślenia, postrzeganie otaczającej rzeczywistości oraz na zachowanie ludzi.

— *Przede wszystkim następuje ograniczenie wszystkich umiejętności człowieka, które nie łączą się bezpośrednio z tym, co nazywamy racjonalnym działaniem* — podkreślają **Fritz Boehle** oraz **Brigitte Milkau** z Instytutu Nauk Społecznych w Monachium. Ich zdaniem potwierdzają to badania osób zatrudnionych przy obrabiarkach sterowanych numerycznie. Obserwuje się u nich głównie ograniczenie postrzegania.

Tego rodzaju zjawisko występuje także w wypadku komputera, choć człowiek nie odczuwa tu, że ma pod ręką jakieś konkretne urządzenie. Mimo tego zaraz pojawia się grymas na jego twarzy, jak tylko nastąpi coś nieprzewidzianego. Według specjalistów, człowiek znajduje się wtedy jakby w „schizofrenicznej sytuacji", kiedy powinien zaraz zrobić to, czego się „wyuczył", ale nie jest w stanie temu sprostać.

Psycholog **Robert Schurz** oraz matematyk **Joerg Pfuegler** prowadząc badania osób które na co dzień posługują się komputerem, doszli do wniosku, że reprezentują oni „maszynowy" typ osobowości. Myślenie takiego człowieka charakteryzuje formalizm i operowanie jedynie wyuczonymi algorytmami. Ludzie o takiej osobowości nie znoszą wszelkich niespodziewanych sytuacji, w ogóle tego, co zakłóca ustalony porządek. Mają silną potrzebę kontrolowania wszystkiego, unikają jednocześnie kontaktów z innymi ludźmi, gdyż obawiają się zajścia czegoś, czego się nie spodziewają. W życiu codziennym u takich ludzi wszystko musi przebiegać tak jak w zaprogramowanej maszynie — musi być zaplanowane i systematyczne.

Zdaniem wielu uczonych, tego rodzaju rozdział sfery myślowej w psychice człowieka oraz zmysłowego i emocjonalnego kontaktu ze światem nie jest spowodowany wyłącznie przez automatyzację i komputeryzację. — *To niekorzystne zjawisko pojawiło się znacznie wcześniej* — podkreśla **Michael Wimmer**, politolog z Wiednia — *i związane jest z cywilizacją zachodnioeuropejską, w której liczy się wyłącznie abstrakcyjne myślenie i całkowicie zaprogramowana wraz z ludźmi produkcja.*

Spójrzmy teraz na relację komputer — psychika człowieka nieco z innej strony, a właściwie nawet perspektywy. Na początku lat sześćdziesiątych psycholog **Raymond Fowler** z University of Alabama był jednym z pierwszych, którzy do komputera wprowadzili program Minnesota Multiphasic Personality Inventory — wielofazowy wykaz osobowości. Gdy program dobrze się przyjął, na język cyfr zaczęto przekładać inne testy psychologiczne.

Dzisiaj, gdy ktoś czuje się przygnębiony, czy zestresowany zwraca się o pomoc do psychologa specjalisty. Pacjent sam odtwarza własną charakterystykę. Dokonuje tego siedząc przed komputerem i odpowiadając na bogatą listę pytań dotyczących własnej osoby. W ciągu kilku minut od naciśnięcia ostatniego klawisza komputer opracowuje i wydrukowuje ścisłą charakterystykę ankietowanego. Jak twierdzi profesor psychiatrii w University of Wisconsin-Madison, **John Greist**, w dziedzinie wywiadów z pacjentem komputery są szczególnie przydatne, gdyż bardzo często przewyższają klinicystów pod względem dokładności i staranności badań. Co więcej, osobisty komputer stojący na biurku terapeuty może błyskawicznie posortować i przeanalizować setki odpowiedzi, a następnie wydrukować opis osobowości pacjenta. Mankamentem jest jednak fakt, że komputery nie potrafią przeanalizować wielu spontanicznych wypowiedzi wyartykułowanych w potocznym języku, chociaż najnowsze eksperymentalne systemy są w stanie zrozumieć zdania w zakresie kilku tysięcy słów.

Najkorzystniejsze dla pacjenta byłoby niewątpliwie sprzęgnięcie komputera z intuicją doskonałego lekarza-naukowca. Komputery bowiem nie potrafią jeszcze samodzielnie przeprowadzać skomplikowanych zadań terapeutycznych. Największe zastosowanie znajduje na razie ich zdolność do przeprowadzania operacji statyczno-obliczeniowych. Progiem do dalszego rozwoju jest zdolność uczonych do formułowania reguł terapeutycznych, które będzie można wprowadzać do komputera. Wówczas w większym stopniu możliwa będzie penetracja duszy człowieka przez zaprogramowaną przez niego samego maszynę.

(Raj.)

W NUMERZE:

W szponach Atari (18) — str. 4
Edytor Basic'a — str. 8
Helios — monitorem — str. 10
Wyznaczanie rozwiązań układu równań różniczkowych zwyczajnych I-go rzędu — str. 12
PRIMUM N°N NOCERE, czyli jak chcemy uczyć Podstaw Informatyki — str. 14
Wprowadzenie do Podstaw Informatyki — str. 16
Nuty z komputera (4) — str. 19
Lilavati (8) — str. 24
Różne sposoby kopiowania ekranu mikrokomputera na drukarce — str. 26
Dodatkowe zestawy znaków graficznych do tworzenia tablic i ramek — str. 28
Słownik informatyczny — str. 29
Liga Myślących — str. 31



(Foto: J. Zelman)

Rady i porady programisty, czyli 3 × P (1)

Na początek podprogram, który znaleźć powinien się w każdym programie. Jego głównym zadaniem jest wyświetlenie na ekranie monitora tekstu zawierającego informacje o przeznaczeniu i funkcjach napisanego przez Ciebie programu.

Jednocześnie można go wykorzystać do wywoływania tzw. helpów w trakcie realizacji programu. Eliminuje to konieczność wprowadzania, czasami długich tekstów do ciała programu. Długie teksty w programie powodują bardzo szybkie „pożeranie" pamięci niejednokrotnie niezbędnej do załadowania programu.

Pamiętając o tym zapoznaj się z niżej przedstawionym programem. Zasadniczą jego część stanowi podprogram wyświetlania tekstu, jemu też poświęcę najwięcej uwagi. Aby skorzystać z tego podprogramu należy wcześniej korzystać z edytora tekstu np. TASWORD, AMSWORD itp. zredagować tekst, który chcemy wyświetlić w naszym programie. Trzeba pamiętać podczas redakcji tekstu o wstawieniu na początku każdej lini dowolnego znaku np. ilustruje to przykład:

```
 -----------------------------------------------
!                                               !
:               TEST PROGRAMU                   :
!                                               !
 -----------------------------------------------

          ą-ę-ń-ć-ź-ż-ó-ś-Ł-Ą-Ż
```

Jedynym ograniczeniem jest to, że nie możesz użyć w tekście przecinków i średników. Długość tekstu w zależności od potrzeb i możliwości użytego edytora tekstu. W zależności od trybu, w jakim będzie wyświetlany tekst ilość znaków w wierszu ustalony na 40 trybu 1 i 80 dla trybu 2. Jeżeli wykorzystujesz do prezentacji tekstu cały ekran.

Uwaga!

pominięcie znaku na początku linii spowoduje utratę porządku na ekranie — spróbuj, a przekonasz się sam.

Poniżej zamieszczono listing podprogramu wyświetlenia tekstu:

```
70 '$$$$ PODPROGRAM WYSWIETLANIA TEKSTU $$$
80 MODE 2
90 PEN 1
100 OPENIN a$
110 WHILE NOT EOF
120 LL=LL+1
130 INPUT£9,zx$
140 LOCATE 1,LL
150 PRINT MID$(zx$,2,LEN(zx$))
160 IF LL<=24 THEN WEND:GOTO 230
170 WINDOW£1,1,80,25,25:PEN£1,0:PAPER£1,1:CLS£1
180 LOCATE£1,15,1
190 PRINT£1,"KONTYNUACJA TEKSTU - NACISNIJ KLAWISZ <ENTER>"
200 IF INKEY(6)=-1 THEN 200
210 LL=0:CLS£1:CLS
220 GOTO 110
230 CLOSEIN
240 WINDOW£1,1,80,25,25:PEN£1,0:PAPER£1,1:CLS£1
250 LOCATE£1,15,1
260 PRINT£1,"POWROT DO POCZATKU POMOCY - <P> $$ POWROT DO PROGRAMU - <ENTER>"
270 CLEAR INPUT
280 IF INKEY(6)=0 THEN LL=0:CLEAR INPUT: RETURN
290 IF INKEY(27)=0 THEN LL=0:CLS:GOTO 100
300 GOTO 280
```

Jak widać jest on przeznaczony do wyświetlania w trybie 2 tekstu zawartego w uprzednio założonym zbiorze o nazwie a$. Ponieważ ekran w tym trybie pracy ma 25 linii, podprogram jednorazowo powoduje wyświetlenie 24 linii tekstu — w 25 linii zdefiniowano okno 1, przeznaczone na komunikaty. Jeśli tekst jest dłuższy niż 24 linie w oknie 1 pojawi się komunikat:

KONTYNUACJA TEKSTU — NACIŚNIJ KLAWISZ ENTER

W wypadku wyświetlenia na ekranie całej zawartości zbioru o nazwie umieszczonej w a$ w oknie 1 pojawi się komunikat:

POWRÓT DO POCZĄTKU TEKSTU — P
POWRÓT DO PROGRAMU ENTER

Zmiana trybu na 1 (mode 1) pociąga za sobą odpowiednią redakcję tekstu (40 znaków e lini), i zmianę definicji okna 1 na komunikaty.

Przykład wywołania tego podprogramu przedstawiono poniżej:

```
10 ' $$$ SEGMENT STERUJACY $$$
20 INPUT a$
30 GOSUB 310
40 GOSUB 70
50 END
60 STOP
```

Realizuje on podstawienie pod nazwę a$ nazwy zbioru, wywołanie podprogramu wyświetlania.

```
310 '$$$ DEFINICJA POLSKICH LITER $$$
320 SYMBOL AFTER 141
330 SYMBOL 166,254,134,12,126,48,98,254,0
340 SYMBOL 224,24,0,126,78,24,48,126,0
350 SYMBOL 152,240,102,108,120,98,102,126,0
360 SYMBOL 184,56,26,28,24,56,24,60,0
370 SYMBOL 173,0,0,120,12,124,204,118,6
380 SYMBOL 177,0,0,60,102,126,96,60,14
390 SYMBOL 187,6,12,60,126,102,102,60,0
400 SYMBOL 175,12,24,60,102,96,102,60,0
410 SYMBOL 186,6,12,220,102,102,102,102,0
420 SYMBOL 222,12,24,126,76,24,48,126,0
430 SYMBOL 191,12,24,60,96,60,6,124,0
440 RETURN
```

Powyższy podprogram definiuje polskie litery. Kody polskich liter muszą być zgodne z kodami używanymi w edytorze tekstu, za pomocą, którego redagowany był tekst. Jest to warunek, aby w wyświetlanym tekście nie zabrakło a, e. ż itd.

Kończąc mam nadzieję, że zaprezentowany wyżej program zainteresuje Czytelników i zastosują go w swoich programach, uatrakcyjniając go i przyczyniając się między innymi do ułatwienia wymiany programów.

Zdzisław ŁUKASZEWICZ



— A ile ty tato palisz na 100 km.

PROGRAM 56

COMMODORE

Stroiciel gitary

Przedstawiony niżej krótki program generuje dźwięki umożliwiające strojenie gitary. Dla zainteresowanych reszta wyjaśnień w programie.

```
10 PRINT '■STROICIEL GITARY':PRINT
20 FOR I=1 TO 6:READ L(I), H(I):NEXT
30 R=54272:FOR I=R TO R+23:POKE I,0:NEXT

40 POKE R+5,102:POKE R+6,102:POKE R+24,1
5
50 PRINT'■ NACISNIJ:':PRINT
60 PRINT'■DOWOLNY KLAWISZ ABY WLACZYC LU
B         WYLACZYC DZWIEK':PRINT
70 PRINT'■SPACJE ABY UZSKAC DZWIEK
          NASTEPNEJ STRUNY':PRINT
80 PRINT'■DOWOLNY KLAWISZ FUNKCYJNY
          ABY ZAKONCZYC PROGRAM'
90 GET A$:IF A$='' THEN 90
100 POKE R+4,0
110 IF A$=CHR$(32)THEN I=I+1:GOTO 140
120 IF A$>CHR$(132)AND A$<CHR$(141) THEN
 POKE R+24,0:END
130 IF X THEN X=0:GOTO 90
140 IF I>6 THEN I=1
150 POKE R,L(I):POKE R+1,H(I)
160 POKE R+4,33:X=I:GOTO 90
170 DATA 71,5,12,7,104,9,142,12,210,15,3
0,21
READY.
```

Opracował: Tadeusz CISEK

W szponach ATARI (18)

Okna — ciąg dalszy

Tomasz MROWIEC Ludwik PIELA

W poprzednim odcinku cyklu („IKS" nr 7—8/1988) przedstawiliśmy procedury umożliwiające tworzenie okien, w trybach tekstowych i graficznych, we własnych programach napisanych w języku BASIC. Mieliśmy pewne zastrzeżenia do procedur działających w trybach graficznych. Umieszczały one w oknach teksty kolorowe, ale tylko w trybach wielokolorowych. W trybach dwukolorowych występowały dodatkowe efekty uboczne. Między innymi mało czytelne tło, na jakim umieszczane były napisy oraz zbyt szerokie znaki.

Z tych powodów opracowaliśmy nową procedurę konwersji, przeznaczoną dla trybów dwukolorowych (tryby BASIC-u o numerach 4, 6, 8 i 14). Jej zadaniem jest przygotowanie reprezentacji bitowej znaków tekstu, w której każdy punkt elementarny ekranu odwzorowany jest w jednej pozycji binarnej (a nie dwóch, jak poprzednio). Procedura ta przechowywana jest w zmiennej tekstowej F$ i wywołujemy ją następująco:

X=USR(ADR(F$), ADR(A$), ADR (KOM$), LEN(KOM$)

gdzie: **F$** — zmienna tekstowa przechowująca procedurę konwersji,

A$ — pomocnicza zmienna tekstowa służąca do przechowywania reprezentacji bitowej znaków tekstu; musi pomieścić 8 razy więcej znaków niż jest w wyświetlanym tekście,

KOM$ — zmienna tekstowa przechowująca wyświetlany tekst.

Wyświetlenie okna na ekranie realizuje procedura przechowywana w zmiennej W$. Sposób jej wywołania jest następujący:

X=USR(ADR(W$), ADR(A$), O, W, S, L)

gdzie: **O** — wielkość przesunięcia lewego górnego rogu okna względem lewego górnego rogu ekranu (w bajtach),

W — wysokość okna (w liniach obrazu),

S — szerokość linii w bajtach (liczba znaków w linii),

L — przesunięcie między początkiem każdej kolejnej linii, jego wielkość zależy od używanego trybu graficznego.

W poprzednim odcinku nie napisaliśmy wyraźnie, w jaki sposób usunąć okno z ekranu. Ze względu na zasadę działania procedury tworzenia okna, polegającą na wymianie zawartości ciągu z wybranym obszarem pamięci ekranu, zamknięcie okna uzyskamy poprzez ponowne użycie procedury tworzenia okna. Należy jednak pamiętać, aby parametry użyte w wywołaniu były takie same.

Program nr 1 demonstruje wykorzystanie omawianego podprogramu. W zależności od wybranego trybu graficznego, dwu- lub czterokolorowego, używana jest odpowiednia procedura konwersji. Bądź opisana powyżej, bądź z poprzedniego odcinka. Tak się składa, że tryby dwukolorowe są parzyste (4, 6, 8, 14) a czterokolorowe — nieparzyste (3, 5, 7, 15). Dzięki temu w prosty sposób możemy sprawdzić rodzaj trybu i wywołać odpowiednią procedurę.

Efektywne wykorzystanie opisanych procedur wymaga dokładnego przeanalizowania programu nr I oraz opisów zamieszczonych w poprzednim cyklu.

Prezentowane dotychczas procedury tworzenia okien na ekranie wymagały podania szeregu parametrów. Wśród nich był również adres zmiennej tekstowej zawierającej wyświetlaną w oknie informację. Jak pamiętamy, wymagało to używania procedury konwersji znaków tekstu z kodu ATASCII na kod ekranu. Możliwe jest również inne rozwiązanie. Procedura tworzenia okna przygotowuje tylko obszar ekranu, w którym za pomocą instrukcji PRINT możemy umieścić stosowny komunikat. A zatem wykonuje ona te same funkcje co poprzednio, za wyjątkiem wyprowadzania informacji na ekran. Oczywiście może być używana w taki sposób wyłącznie w trybie tekstowym GR.0.

Poprzednio prezentowane procedury włączane były do programu. Początkowo występowały w liniach DATA, a następnie umie-

szczane były w zmiennych tekstowych. Poniżej prezentujemy inne rozwiązanie, zaczerpnięte z czasopisma „Analog Computing" November 1986, w którym podprogramy otwierania i zamykania okien stanowią jakby rozszerzenie systemu operacyjnego. Umieszczone są w pamięci komputera poza obszarem przeznaczonym dla programu w języku BASIC. Odpowiednie dla nich miejsce uzyskujemy poprzez przesunięcie, w górę pamięci, dolnej granicy obszaru BASIC'u.

Rozwiązanie to wydaje się być niezbyt wygodnym, lecz jego zalety ujawniają się podczas pracy ze stacją dysków. Umieszczenie procedur w zbiorze AUTORUN.SYS zapewnia automatyczne wprowadzenie ich do pamięci po włączeniu komputera.

Ze względu na interesujące rozwiązanie oraz fakt, że nie każdy z czytelników dysponuje stacją dysków, opracowaliśmy również wersję taśmową tych procedur. Dlatego procedury otwierania i zamykania prezentujemy w dwóch wersjach: dyskowej i taśmowej.

Program nr 2 automatycznie tworzy na dyskietce zbiór AUTORUN.SYS i wprowadza do niego oba podprogramy. **Program nr 3** tworzy zbiór na taśmie magnetycznej, który może być wprowadzony do pamięci po włączeniu komputera z wciśniętym klawiszem (START).

Aby uzyskać odpowiednie zbiory należy wprowadzić do pamięci komputera, najlepiej za pomocą Edytora BASIC'a, program nr 2, jeśli dysponujemy stacją dysków, lub program nr 3, jeśli możemy korzystać tylko z magnetofonu. Radzimy przechowywać w pamięci zewnętrznej kopie wprowadzonych programów, aby uniknąć ponownego ich wprowadzania w wypadku wystąpienia jakiegokolwiek błędu. Przygotowujemy dyskietkę zawierającą DOS (bez zbioru AUTORUN.SYS) lub czystą taśmę magnetyczną i uruchamiamy program przechowywany w pamięci komputera. Po odliczeniu w dół od 640 do zera program nr 2 utworzy zbiór AUTORUN.SYS i zapisze do niego odpowiednie procedury, a program nr 3 zasygnalizuje chęć zapisania na taśmie magnetycznej. Zapisze je po włączeniu magnetofonu na zapis i naciśnięciu dowolnego klawisza. Zbiór taśmowy jest typu BOOT, to znaczy samoładujący się.

W obu wypadkach załadowanie procedur wymaga uprzedniego wyłączenia komputera. W wypadku używania stacji dysków wchodzą one do pamięci RAM bezpośrednio po wprowadzeniu DOS-u (bez wyłączania BASIC-a). W wypadku wprowadzania ich za pomocą magnetofonu należy wcześniej przewinąć taśmę i ustawić na początku zbioru, a następnie włączyć komputer z wciśniętym klawiszem (START).

Obie wersje, taśmowa i dyskowa, różnią się tylko sposobem wprowadzania do pamięci komputera. Ich działanie jest identyczne.

W celu otwarcia okna należy podać sześć parametrów. Pierwszym jest adres procedury otwarcia okna — 7459. Następne dwa określają lewy górny róg okna w postaci współrzędnych X, Y. Wartości X mogą zmieniać się od 0 do 39, a wartości Y od 0 do 23.

Czwarty i piąty parametr określają wielkość okna. Czwarty to szerokość, która może zmieniać się od 1 do 40 znaków. Piąty określa wysokość, od 1 do 24 znaków.

Ostatni parametr jest wskaźnikiem ramki. Jeśli podana jest wartość 0, to nie ma ramki. Wartość między 1 a 127 powoduje wykreślenie normalnej ramki. Wartość powyżej 127 — ramki w negacji.

Poniższe wywołanie otworzy okno o wielkości 10 na 5 znaków, bez ramki, w lewym, górnym rogu ekranu:

X=USR(7459, 0, 0, 10, 5, 0)

Zamykanie okna jest zawsze łatwiejsze. Przekazywany jest tylko jeden parametr, adres procedury zamknięcia okna.

X=USR(7826)

zamknie ostatnio otwarte okno.

Używanie okien jest bardzo proste. Trzeba jednak pamiętać o kilku sprawach. Jeśli otwieramy okno, które jest mniejsze niż 3x3 znaki, nie powinno ono mieć ramki. W przeciwnym wypadku wystąpią dziwne zjawiska. Jeśli przechodzimy do DOS-u, procedury będą likwidowane, o ile nie użyjemy zbioru MEM.SAV. Podczas otwierania lub zamykania okna zalecane jest ustawianie kursora poza planowanym obszarem okna. Jeśli kursor jest w oknie podczas zamykania, istnieje szansa, że znak nie zostanie podmieniony. Jeśli kursor jest w pobliżu okna, które jest otwierane lub zamykane, możemy nie martwić się o niego. Wstawienie zera do komórki o adresie 8045 przekaże procedurom informację, że aktualnie żadne okno nie jest otwarte.

Podczas uruchamiania programu zalecane jest zerowanie licznika okien, zwłaszcza w wypadku, gdy jakieś okna pozostały otwarte w poprzednio działających programach.

Normalnie do kreślenia okna używane są negatywy spacji. Ten znak wypełniania może być zdefiniowany przez użytkownika. Wstawienie wybranej wartości do komórki 7625 zmieni znak używany do kreślenia tła okna.

Procedura otwierania okna wyznacza adres lewego górnego rogu na podstawie dostarczonych współrzędnych X i Y. Znaki schowane poza oknem przechowywane są na stosie, który rozpoczyna się od góry pamięci i rozrasta w dół, w kierunku programu w BASIC-u. Oprócz znaków na stosie umieszczane są również parametry. Podczas zamykania okna następuje proces odwrotny. Najpierw zdejmowane są ze stosu parametry. Następnie znaki, które zapamiętywane są w obszarze pamięci ekranu, odtwarzając pierwotny stan ekranu.

Podczas otwierania lub zamykania okna zwracaną wartością jest adres dołu stosu. Powinniśmy śledzić tę wartość, aby uniknąć konfliktu z naszym programem, którego górny adres przechowywany jest w komórkach 14 i 15 następująco:

GORPRO=PEEK (14)+256*PEEK (15)

Przed otwarciem okna powinniśmy sprawdzić, czy stos nie zejdzie poniżej górnej granicy programu.

Poprzez użycie prezentowanych procedur, ich wybór pozostawiamy czytelnikom, tworzone amatorsko programy mogą uzyskać profesjonalny wygląd.

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Prosram nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
SH 100 GRAPHICS 0:POKE 752,1:POKE
    82,4
AD 110 ? :? "        OKNA TEKSTOW
   E "
RL 120 ? :? "      W TRYBACH GRAFI
   CZNYCH"
KR 130 DIM W$(116),KOM$(11),ROB$(
   22*16),E$(263),A$(11*16),F$(20
   0)
BN 140 ? :? "WCZYTANIE W$"
ME 150 FOR I=1 TO 116:READ A:W$(I
   ,I)=CHR$(A):NEXT I
ND 160 ? :? "WCZYTANIE E$"
VL 170 FOR I=1 TO 263:READ A:E$(I
   ,I)=CHR$(A):NEXT I
OD 171 ? :? "WCZYTANIE F$"
XO 172 FOR I=1 TO 147:READ A:F$(I
   ,I)=CHR$(A):NEXT I
EB 175 G=15:W=40:GRAPHICS G+16:GO
   SUB 260
CY 176 G=14:W=20:GRAPHICS G+16:GO
   SUB 260
WR 180 G=8:W=40:GRAPHICS G+16:GOS
   UB 260
WM 190 G=7:W=40:GRAPHICS G+16:GOS
   UB 260
US 200 G=6:W=20:GRAPHICS G+16:GOS
   UB 260
UN 210 G=5:W=20:GRAPHICS G+16:GOS
   UB 260
TX 220 G=4:W=10:GRAPHICS G+16:GOS
   UB 260
TS 230 G=3:W=10:GRAPHICS G+16:GOS
   UB 260
TW 240 GRAPHICS 0:POKE 82,2:? :?
   "KONIEC PROGRAMU":END
NL 250 REM PODPROGRAM WYSWIETLANI
   A KOMUNIKATOW
LI 260 SETCOLOR 0,9,10:SETCOLOR 1
   ,3,6:SETCOLOR 2,9,0:SETCOLOR 4
   ,1,2:C=(INT(G/2)=(G/2))
II 270 REM ZEROWANIE CIAGU
UZ 275 COLOR 1:PLOT W*3,0:FOR I=1
    TO 40:X=RND(0)*W*4:Y=RND(0)*W
   *2:DRAWTO INT(X),INT(Y):NEXT I
VW 280 ROB$=CHR$(0):ROB$(22*16)=R
   OB$:ROB$(2)=ROB$:KOM$=ROB$
OB 290 FOR L=1 TO 3
IZ 300 REM TWORZENIE OKNA
LX 305 A$=CHR$(0):A$(11*16)=A$:A$
   (2)=A$
MP 310 KOM$="Gr   ":KOM$(6-LEN(STR
   $(G)))=STR$(G)
HS 315 IF C=0 THEN ASM=USR(ADR(E$
   ),ADR(A$),ADR(KOM$),LEN(KOM$),
   L)
ML 317 IF C=1 THEN ASM=USR(ADR(F$
   ),ADR(A$),ADR(KOM$),LEN(KOM$))
KM 320 ROB$=A$
YW 325 IF C=0 THEN WSK=(LEN(KOM$)
   -0)*16+1
IM 326 IF C=1 THEN WSK=(LEN(KOM$)
   -0)*8+1
HT 330 KOM$="TYP  ":KOM$(5)=STR$(L
   )
MD 335 A$=CHR$(0):A$(11*16)=A$:A$
   (2)=A$
HJ 340 IF C=0 THEN ASM=USR(ADR(E$
   ),ADR(A$),ADR(KOM$),LEN(KOM$),
   L)
MC 342 IF C=1 THEN ASM=USR(ADR(F$
   ),ADR(A$),ADR(KOM$),LEN(KOM$))
XX 345 ROB$(WSK)=A$
XC 360 REM UMIESZCZANIE OKNA
OD 370 FOR I=1 TO 5
```

```
KQ 380 IF C=0 THEN ASM=USR(ADR(W$
   ),ADR(ROB$),W*8+W/2-5,16,LEN(K
   OM$)*2,W)
AC 382 IF C=1 THEN ASM=USR(ADR(W$
   ),ADR(ROB$),W*8+W/2-5,16,LEN(K
   OM$),W)
WK 385 FOR J=15 TO 0 STEP -15/100
AG 386 SOUND 0,L*75,14,J
HO 387 NEXT J
CE 390 NEXT I:NEXT L:RETURN
BV 400 REM PODPROGRAM UMIESZCZANI
   A OKNA
OG 410 DATA 104,104,133,206,104,1
   33,205,104,141,1,6,104,141,0,6
NZ 430 DATA 104,133,208,104,133,2
   07,104,104,133,209,104,141,3,6
   ,104,141,2,6
AT 450 DATA 165,88,133,203,165,89
   ,133,204,24,165,203,109,0,6,13
   3
JA 470 DATA 203,165,204,109,1,6,1
   33,204,160,0,177,203,170,177,2
   05,145
OC 490 DATA 203,138,145,205,200,1
   96,209,208,241,169,0,198,207,1
   97,207,208
MU 510 DATA 6,197,208,240,30,198,
   208,24,152,101,205,133,205,144
   ,3,230
XF 530 DATA 206,24,165,203,109,2,
   6,133,203,165,204,109,3,6,133,
   204,24,144,197,96
CR 560 REM PODPROGRAM KONWERSJI Z
   NAKOW NA FORMAT CZTEROKOLOROWY

BK 570 DATA 104,104,133,204,104,1
   33,203,104,133,212,104,133,211
   ,104,104,133
MK 590 DATA 216,10,133,217,104,10
   4,41,3,24,106,106,106,41,192,1
   33,207
XX 610 DATA 133,208,169,0,133,219
   ,169,4,133,220,169,0,168,145,2
   19,200
GL 630 DATA 192,16,208,249,169,25
   5,133,215,160,0,177,211,201,12
   8,144,4
TV 650 DATA 132,215,73,128,201,96
   ,144,4,160,3,176,19,201,64,144
   ,4
RS 670 DATA 160,1,176,11,201,32,1
   44,5,176,5,24,144,205,160,2,41
JA 690 DATA 31,10,10,10,133,213,2
   4,152,109,244,2,133,214,160,0,
   132
RU 710 DATA 209,132,210,162,0,134
   ,205,164,210,177,213,133,206,1
   64,209,165
BJ 730 DATA 206,48,6,177,219,5,20
   7,145,219,24,102,207,24,102,20
   7,6
IZ 750 DATA 206,232,224,4,208,231
   ,165,215,240,5,56,241,219,145,
   219,165
HE 770 DATA 208,133,207,230,209,2
   30,205,165,205,201,2,208,208,2
   30,210,164
UG 790 DATA 210,192,8,208,190,240
   ,3,24,144,160,165,203,72,165,2
   04,72
EN 810 DATA 160,0,177,219,145,203
   ,200,192,2,208,247,177,219,170
   ,200,177
XA 830 DATA 219,72,200,132,218,16
   4,217,138,145,203,104,200,145,
   203,24,165
LA 850 DATA 203,101,217,133,203,1
   44,2,230,204,164,218,192,16,20
   8,220,104
JB 870 DATA 133,204,24,104,105,2,
   133,203,144,2,230,204,230,211,
   208,2
KP 890 DATA 230,212,198,216,208,1
   77,96
WE 999 REM PODPROGRAM KONWERSJI Z
   NAKOW NA FORMAT DWUKOLOROWY
ZN 1000 DATA 104,104,133,206,104,
   133,205,104,133,204,104,133,20
   3,104,104,133,207,133,208,160,
   0,177,203,41,127
CK 1010 DATA 168,42,42,42,42,41,3
   ,170,152,41,159,29,73,251
HN 1020 DATA 133,212,169,0,162,3,
   24,10,6,212,144,2,105,0,202,20
   8,245,24,109,244,2,133,213
OM 1030 DATA 165,205,72,165,206,7
   2,162,0,160,0,169,255,133,209,
   177,203,41,128,208,4,169,0,133
   ,209,177,212,69,209
UR 1035 DATA 145,205,232,224,8,24
   0,23
QA 1040 DATA 165,212,24,105,1,133
   ,212,144,2,230,213,165,208,24,
   101,205,133,205,144,2,230,206,
   24,144,204
UH 1050 DATA 104,133,206,104,24,1
   05,1,133,205,144,2,230,206,24,
   230,203,208,2,230,204,198,207,
   208,129,96,-1


FK 1 REM ***************
FJ 2 REM *             *
XU 3 REM * Program nr 2 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
HY 7 REM Wersja dyskowa
EY 8 REM procedur
NO 9 REM
CV 10 DIM A$(641):GRAPHICS 0:POKE
    752,1
ZY 15 POSITION 12,10:? "Prosze cz
   ekac...";
RM 20 FOR I=1 TO 641
EB 25 POSITION 19,14:? 641-I;"
   ";
QF 30 READ A:IF A=-1 THEN 70
YY 40 A$(I,I)=CHR$(A)
IU 60 NEXT I
ZN 70 IF A<>-1 THEN ? "BLAD W DAN
   YCH":END
BM 80 OPEN #1,8,0,"D:AUTORUN.SYS"
VV 90 POKE 766,1:? #1;A$;:POKE 76
   6,0
KX 100 CLOSE #1
NS 110 END
[illegible] 1000 DATA 255,255,0,29,111,31
NP 1005 DATA 165,12,141,14,29,165
   ,13,141,15,29
GW 1010 DATA 76,16,29,32,255,255,
   169,0,141,231,2,169,32,141,232
   ,2
SU 1020 DATA 169,13,133,12,169,29
   ,133,13,96,104,104,104,141,100
   ,31,104
IU 1030 DATA 104,141,99,31,104,10
   4,141,101,31,141,103,31,104,10
   4,141,102
VV 1040 DATA 31,141,104,31,104,10
   4,141,98,31,41,128,141,106,31,
   173,99
DD 1050 DATA 31,133,203,169,0,133
   ,204,24,6,203,38,204,24,6,203,
   38
AG 1060 DATA 204,24,6,203,38,204,
   165,203,141,96,31,165,204,141,
   97,31
XR 1070 DATA 24,6,203,38,204,24,6
   ,203,38,204,24,165,203,109,96,
   31
OH 1080 DATA 133,203,165,204,109,
   97,31,133,204,24,173,100,31,10
   1,203,133
GI 1090 DATA 203,165,204,105,0,13
   3,204,24,165,203,101,88,133,20
   3,141,107
YP 1100 DATA 31,165,204,101,89,13
   3,204,141,108,31,56,169,40,237
   ,101,31
RA 1110 DATA 141,105,31,173,109,3
   1,208,13,173,229,2,133,205,173
   ,230,2
XK 1120 DATA 133,206,32,51,31,238
   ,109,31,160,0,177,203,145,205,
   169,128
OZ 1130 DATA 145,203,32,9,31,32,5
   1,31,206,103,31,173,103,31,208
   ,232
UE 1140 DATA 206,104,31,173,104,3
   1,240,23,24,165,203,109,105,31
   ,133,203
ZK 1150 DATA 165,204,105,0,133,20
   4,173,101,31,141,103,31,76,194
   ,29,32
SW 1160 DATA 37,31,165,203,145,20
   5,32,51,31,165,204,145,205,32,
   51,31
FB 1170 DATA 173,105,31,145,205,3
   2,51,31,173,102,31,145,205,32,
   51,31
TL 1180 DATA 173,101,31,145,205,3
   2,51,31,173,98,31,208,3,76,0,3
   1
VP 1190 DATA 174,101,31,202,202,1
   42,110,31,172,102,31,136,136,1
   40,111,31
QI 1200 DATA 173,107,31,133,203,1
   73,108,31,133,204,169,81,13,10
   6,31,160
ZA 1210 DATA 0,145,203,32,65,31,1
   72,101,31,136,169,69,13,106,31
   ,145
XX 1220 DATA 203,32,82,31,169,124
   ,13,106,31,160,0,145,203,172,1
   01,31
YK 1230 DATA 136,145,203,32,82,31
   ,206,111,31,173,111,31,208,230
   ,169,90
CA 1240 DATA 13,106,31,160,0,145,
   203,32,65,31,169,67,13,106,31,
   172
YJ 1250 DATA 101,31,136,145,203,7
   6,0,31,104,173,109,31,208,3,76
   ,0
QU 1260 DATA 31,32,23,31,206,109,
   31,160,0,177,205,141,101,31,14
   1,103
PY 1270 DATA 31,32,23,31,177,205,
   141,102,31,141,104,31,32,23,31
   ,177
TD 1280 DATA 205,141,105,31,32,23
   ,31,177,205,133,204,32,23,31,1
   77,205
WV 1290 DATA 133,203,32,23,31,177
   ,205,145,203,32,23,31,32,37,31
   ,206
HV 1300 DATA 103,31,173,103,31,20
   8,238,173,101,31,141,103,31,56
   ,165,203
ST 1310 DATA 237,105,31,133,203,1
   65,204,233,0,133,204,206,104,3
   1,173,104
FK 1320 DATA 31,208,210,32,51,31,
   165,205,133,212,165,206,133,21
   3,96,24
VP 1330 DATA 165,203,105,1,133,20
   3,165,204,105,0,133,204,96,24,
   165,205
IB 1340 DATA 105,1,133,205,165,20
   6,105,0,133,206,96,56,165,203,
   233,1
XP 1350 DATA 133,203,165,204,233,
   0,133,204,96,56,165,205,233,1,
   133,205
LA 1360 DATA 165,206,233,0,133,20
   6,96,160,1,174,110,31,169,82,1
   3,106
QP 1370 DATA 31,145,203,200,202,2
   08,250,96,24,165,203,105,40,13
   3,203,165
QE 1380 DATA 204,105,0,133,204,96
   ,0,0,0,0,0,0,0,0,0,0
EI 1390 DATA 0,0,0,0,0,0,255,255,
   224,2,227,2,0,29,34,29,-1


FK 1 REM ***************
FJ 2 REM *             *
YO 3 REM * Program nr 3 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
EA 7 REM Wersja tasmowa
EY 8 REM procedur
NO 9 REM
CK 10 DIM A$(622):GRAPHICS 0:POKE
    752,1
```

```
ZY 15 POSITION 12,10:? "Prosze cz
   ekac...";
QX 20 FOR I=1 TO 622
DG 25 POSITION 19,14:? 622-I;"
   ";
QF 30 READ A:IF A=-1 THEN 70
YY 40 A$(I,I)=CHR$(A)
IU 60 NEXT I
ZN 70 IF A<>-1 THEN ? "BLAD W DAN
   YCH":END
CI 80 OPEN #1,8,128,"C:"
VV 90 POKE 766,1:? #1;A$;:POKE 76
   6,0
KX 100 CLOSE #1
NS 110 END
DL 1000 DATA 0,5,3,29,9,29,169,60
   ,141,2,211,169,0
MY 1005 DATA 141,231,2,169,32,141
   ,232,2,169,34
CK 1010 DATA 133,10,169,29,133,11
   ,24,96,96
EO 1020 DATA 104,104,104,141,100,
   31,104
IU 1030 DATA 104,141,99,31,104,10
   4,141,101,31,141,103,31,104,10
   4,141,102
VV 1040 DATA 31,141,104,31,104,10
   4,141,98,31,41,128,141,106,31,
   173,99
DD 1050 DATA 31,133,203,169,0,133
   ,204,24,6,203,38,204,24,6,203,
   38
AG 1060 DATA 204,24,6,203,38,204,
   165,203,141,96,31,165,204,141,
   97,31
XR 1070 DATA 24,6,203,38,204,24,6
   ,203,38,204,24,165,203,109,96,
   31
OH 1080 DATA 133,203,165,204,109,
   97,31,133,204,24,173,100,31,10
   1,203,133
GI 1090 DATA 203,165,204,105,0,13
   3,204,24,165,203,101,88,133,20
   3,141,107
YP 1100 DATA 31,165,204,101,89,13
   3,204,141,108,31,56,169,40,237
   ,101,31
RA 1110 DATA 141,105,31,173,109,3
   1,208,13,173,229,2,133,205,173
   ,230,2
XK 1120 DATA 133,206,32,51,31,238
   ,109,31,160,0,177,203,145,205,
   169,128
OZ 1130 DATA 145,203,32,9,31,32,5
   1,31,206,103,31,173,103,31,208
   ,232
UE 1140 DATA 206,104,31,173,104,3
   1,240,23,24,165,203,109,105,31
   ,133,203
ZK 1150 DATA 165,204,105,0,133,20
   4,173,101,31,141,103,31,76,194
   ,29,32
SW 1160 DATA 37,31,165,203,145,20
   5,32,51,31,165,204,145,205,32,
   51,31
FB 1170 DATA 173,105,31,145,205,3
   2,51,31,173,102,31,145,205,32,
   51,31
TL 1180 DATA 173,101,31,145,205,3
   2,51,31,173,98,31,208,3,76,0,3
   1
VP 1190 DATA 174,101,31,202,202,1
   42,110,31,172,102,31,136,136,1
   40,111,31
QI 1200 DATA 173,107,31,133,203,1
   73,108,31,133,204,169,81,13,10
   6,31,160
ZA 1210 DATA 0,145,203,32,65,31,1
   72,101,31,136,169,69,13,106,31
   ,145
XX 1220 DATA 203,32,82,31,169,124
   ,13,106,31,160,0,145,203,172,1
   01,31
YK 1230 DATA 136,145,203,32,82,31
   ,206,111,31,173,111,31,208,230
   ,169,90
CA 1240 DATA 13,106,31,160,0,145,
   203,32,65,31,169,67,13,106,31,
   172
YJ 1250 DATA 101,31,136,145,203,7
   6,0,31,104,173,109,31,208,3,76
   ,0
QU 1260 DATA 31,32,23,31,206,109,
   31,160,0,177,205,141,101,31,14
   1,103
PY 1270 DATA 31,32,23,31,177,205,
   141,102,31,141,104,31,32,23,31
   ,177
TD 1280 DATA 205,141,105,31,32,23
   ,31,177,205,133,204,32,23,31,1
   77,205
WV 1290 DATA 133,203,32,23,31,177
   ,205,145,203,32,23,31,32,37,31
   ,206
HV 1300 DATA 103,31,173,103,31,20
   8,238,173,101,31,141,103,31,56
   ,165,203
ST 1310 DATA 237,105,31,133,203,1
   65,204,233,0,133,204,206,104,3
   1,173,104
FK 1320 DATA 31,208,210,32,51,31,
   165,205,133,212,165,206,133,21
   3,96,24
VP 1330 DATA 165,203,105,1,133,20
   3,165,204,105,0,133,204,96,24,
   165,205
IB 1340 DATA 105,1,133,205,165,20
   6,105,0,133,206,96,56,165,203,
   233,1
XP 1350 DATA 133,203,165,204,233,
   0,133,204,96,56,165,205,233,1,
   133,205
LA 1360 DATA 165,206,233,0,133,20
   6,96,160,1,174,110,31,169,82,1
   3,106
QP 1370 DATA 31,145,203,200,202,2
   08,250,96,24,165,203,105,40,13
   3,203,165
QE 1380 DATA 204,105,0,133,204,96
   ,0,0,0,0,0,0,0,0,0,0
IR 1390 DATA 0,0,0,0,0,0,-1
```

```
FK 1 REM ***************
FJ 2 REM *             *
ZI 3 REM * Prosram nr 4 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
EX 100 REM Prosram demonstracyjny
WN 110 WOTW=7459:WZAM=7826
AG 120 WYP=7625:LICZ=8045
ZW 130 POKE LICZ,0:POKE WYP,128
BL 140 DIM TEKST$(38):POKE 712,PE
   EK(710):POKE 752,1:? CHR$(125)
   ;:LIST 310,510
UU 150 REM TYTUL
NY 160 RESTORE :PX=2:PY=1:SX=36:S
   Y=5:RAMKA=1:GOSUB 460:POSITION
    PX+8,PY+2:GOSUB 500
WF 170 PX=2:PY=7:SX=36:SY=12:RAMK
   A=1:GOSUB 460:POSITION PX+13,P
   Y+1:GOSUB 500
GM 180 REM WLASNOSCI
FW 190 FOR X=3 TO 5:POSITION PX+3
   ,PY+X:GOSUB 500:NEXT X
FR 200 REM ROZNE WIELKOSCI
CY 210 POSITION PX+3,PY+6:GOSUB 5
   00
UT 220 FOR X=3 TO 7:PX=X-2:PY=10+
   X:SX=X:SY=X:RAMKA=128:GOSUB 46
   0:PX=42-X-X:GOSUB 460:NEXT X
MB 230 FOR X=1 TO 400:NEXT X:FOR
   X=1 TO 10:GOSUB 480:NEXT X
QC 240 REM WIELE KOLOROW
ZF 250 PX=2:PY=7:POSITION PX+3,PY
   +7:GOSUB 500
CU 260 POKE WYP,0:PX=6:PY=16:SX=9
   :SY=5:RAMKA=1:GOSUB 460
ZJ 270 POSITION PX+2,PY+2:GOSUB 5
   00
RR 280 POKE WYP,128:PX=25:RAMKA=1
   28:GOSUB 460
GC 290 POSITION PX+2,PY+2:GOSUB 5
   00:FOR X=1 TO 400:NEXT X:GOSUB
    480:GOSUB 480
BM 300 REM RAMKI
AE 310 PX=2:PY=7:POSITION PX+3,PY
   +8:GOSUB 500
PK 320 PX=2:PY=20:SX=36:SY=3:RAMK
   A=0:GOSUB 460:POSITION PX+5,PY
   +1:GOSUB 500
QA 330 RAMKA=1:GOSUB 460:POSITION
    PX+5,PY+1:GOSUB 500
CM 340 RAMKA=128:GOSUB 460:POSITI
   ON PX+5,PY+1:GOSUB 500
ZL 350 FOR X=1 TO 3:GOSUB 480:FOR
    Y=1 TO 100:NEXT Y:NEXT X
NG 360 REM ROZWIJANE MENU
BW 370 PX=2:PY=7:POSITION PX+3,PY
   +9:GOSUB 500
QJ 380 FOR Z=2 TO 36 STEP 9
SG 390 SX=9:SY=1:RAMKA=0:FOR Y=19
    TO 23
IR 400 PX=Z:PY=Y:GOSUB 460:NEXT Y
   :POSITION PX,20:GOSUB 500:POSI
   TION PX+4,22:GOSUB 500
IN 410 FOR X=19 TO 23:GOSUB 480:N
   EXT X:NEXT Z
OK 420 REM ZAMYKANIE WSZYSTKICH O
   KIEN
KC 430 FOR X=1 TO 400:NEXT X
TJ 440 FOR Y=1 TO 2:X=USR(WZAM):F
   OR X=1 TO 200:NEXT X:NEXT Y:GO
   TO 160
IW 450 REM OTWIERANIE OKNA
FZ 460 WINDOW=USR(WOTW,PX,PY,SX,S
   Y,RAMKA):RETURN
VX 470 REM ZAMYKANIE OKNA
FP 480 WINDOW=USR(WZAM):RETURN
WS 490 REM WYSWIETLANIE TEKSTU I
   OCZEKIWANIE
HU 500 READ TEKST$:? TEKST$:POSIT
   ION 0,0:? "";:FOR W=1 TO 300
   :NEXT W:RETURN
MF 510 REM TEKST
KR 520 DATA     O K N A   Wersj
   a 1.1,Wlasnosci:,-Caly kod w j
   ezyku maszynowym
WY 530 DATA -Do 255 okien otwarty
   ch naraz  ,-Odporne na RESET
   ,-Okna mosa miec rozne rozmiar
   y
DP 540 DATA -Okna mosa byc wielok
   olorowe,Czarne,Biale,-Ramki sa
    opcjonalne ,    Okno bez ramk
   i
KZ 550 DATA Okno z normalna ramka
       ,Okno z ramka w negacji
    ,-Rozwijane menu , M e n u,1
MY 560 DATA  M e n u ,2, M e n u
   ,3, M e n u ,4
```

```
FK 1 REM ***************
FJ 2 REM *             *
AC 3 REM * Prosram nr 5 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
EX 100 REM Prosram demonstracyjny
FO 110 WOPEN=7459:WCLOSE=7826
TM 120 POKE 8045,0:POKE 752,1:TRA
   P 120
PG 130 ? CHR$(125);"ILE OKIEN UTW
   ORZYC"
HG 140 INPUT N:N=INT(ABS(N))
FE 150 IF N=0 OR N>255 THEN 130
QJ 160 POKE 752,1:? CHR$(125);:LI
   ST
YN 170 FOR C=1 TO N:POSITION 0,0:
   ? CHR$(30);CHR$(31);
GZ 180 SX=INT(RND(0)*20)+4
GO 190 SY=INT(RND(0)*10)+4
PY 200 PX=INT(RND(0)*(39-SX))+1
LO 210 PY=INT(RND(0)*(23-SY))+1
BS 220 Z=USR(WOPEN,PX,PY,SX,SY,12
   8)
YM 230 POSITION PX+1,PY+1:? C
BQ 240 PROG=PEEK(14)+256*PEEK(15)
AG 250 IF PROG+256>Z THEN 270
DW 260 NEXT C
YO 270 FOR C=1 TO N:POSITION 0,0:
   ? CHR$(30);CHR$(31);
VP 280 Z=USR(WCLOSE):NEXT C:GOTO
   170
```

Tomasz MROWIEC,
Ludwik PIELA

Edytor Basic'a

Ludwik PIELA

Otrzymujemy sporo listów i telefonów w sprawie programu „Edytor BASIC'a" zamieszczonego w numerze „IKS-a" z 1987 roku. Przyznajemy, jest w nim jeden niewidoczny znak (nie z naszej winy) oraz jeden znak trudny do interpretacji. Poza tym program nie zawiera błędów. Większość problemów z jego uruchomieniem wynika z niedokładnego (lub błędnego) wprowadzenia liczb zawartych w liniach DATA. Ze względu na to, a także na fakt, że w przeciągu minionego półtora roku przybyło posiadaczy komputerów ATARI (i mamy nadzieję naszych czytelników), ośmielamy się zamieścić, głównie dla nich, wydruk programu „Edytor BASIC'a".

Program zachowuje wszystkie własności edycyjne interpretera, dopuszcza także stosowanie skrótów instrukcji. Umożliwia także sprawdzenie całego programu, jeśli został on wprowadzony w sposób konwencjonalny, z wykorzystaniem wbudowanego edytora.

Wprowadzenie edytora

W celu utworzenia własnej kopii Edytora BASIC'a należy dokładnie wykonać jedną z poniższych instrukcji.

Wersja dyskowa

1) Wprowadzić Wydruk nr 1 starannie i bez błędów.
2) Przechować program na dysku za pomocą SAVE „D:EDYTORW1. BAS".
3) Oczyścić pamięć komputera rozkazem NEW.
4) Wprowadzić Wydruk nr 2.
5) Dla bezpieczeństwa przechować na dysku za pomocą „D:EDYTORW2. BAS".
6) Uruchomić program instrukcją RUN i wykonać wyświetlone na ekranie polecenia. Zbiór danych o nazwie ML.DAT zapisany zostanie na dysku.
7) Załadować do pamięci pierwszy program rozkazem LOAD „D:EDYTORW1 BAS".
8) Dołączyć zbiór utworzony przez program nr 2 instrukcją ENTER „D:ML.DAT".
9) Przechować program wynikowy rozkazem LIST „D:EDYTOR.LST".

Wersja taśmowa

1) Wprowadzić Wydruk nr 1.
2) Przechować program na kasecie za pomocą CSAVE (nie przewijać taśmy).
3) Oczyścić pamięć komputera rozkazem NEW.
4) Wprowadzić Wydruk nr 2.
5) Dla bezpieczeństwa przechować program na taśmie.

6) Uruchomić program rozkazem RUN i wykonać wyświetlone na ekranie polecenia. Zbiór zostanie zapisany na kasecie.

7) Przewinąć taśmę i wprowadzić pierwszy program rozkazem CLOAD.

8) Dołączyć zbiór utworzony przez program nr 2 rozkazem ENTER „C:".

9) Na innej kasecie przechować program wynikowy rozkazem LIST „C:".

Używanie Edytora

Spójrzmy na wydruki programów zamieszczanych w ramach cyklu „W szponach ATARI". Zauważmy, że każda linia programu poprzedzona jest przez kod dwuliterowy. Jest to suma kontrolna tej linii, a nie część programu.

W celu wprowadzenia programu załaduj Edytor rozkazem ENTER i uruchom go. Zostaniesz zapytany, czy chcesz używać skrótów instrukcji. Jeśli tak, to wprowadź T i naciśnij RETURN. W przeciwnym razie podaj N. Ekran telewizora (monitora) zostanie podzielony na dwa „okna". W górnym wyświetlana będzie każda linia po przetworzeniu przez Edytor oraz suma kontrolna tej linii. W dolnym oknie odbywa się wprowadzanie i poprawianie linii.

Gdy program oczekuje na wprowadzenie, kursor pojawia się przy lewym marginesie okna wprowadzania. Wprowadź linię programu i naciśnij RETURN. Nastąpi sprawdzenie jej poprawności oraz obliczenie sumy kontrolnej. Linia programu pojawi się w oknie komunikatu wraz z wyznaczoną sumą kontrolną. Jeśli wyświetlony kod zgadza się z zamieszczonym w wydruku, można przejść do wprowadzania następnej linii programu. W przeciwnym wypadku należy podać rozkaz E (Edit) i nacisnąć RETURN. Ostatnio wprowadzona linia pojawi się w dolnym oknie, gdzie może zostać poprawiona. Po wprowadzeniu poprawek naciśnij RETURN.

Można wywołać dowolną linię programu, wcześniej wprowadzoną, którą chcemy poprawić, za pomocą rozkazu E z numerem linii. Na przykład, E23Ø wyświetli linię 23Ø w oknie wprowadzania.

Jeśli używamy skrótów instrukcji, obie wersje rozkazu Edit działają różnie. Rozkaz E bez numeru linii wyświetli ją w takiej postaci, w jakiej została wprowadzona. Po dodaniu numeru będzie ona wyświetlona w postaci rozwiniętej (bez skrótów).

W dowolnej chwili można wyjść z Edytora poprzez naciśnięcie klawisza B (Basic). Można teraz przejrzeć wykonaną pracę. Zauważmy, że linie o numerach 326ØØ i wyżej, należą do Edytora. Nasz program ma niższe numery. Aby powrócić do Edytora, należy podać GOTO 326ØØ.

W czasie wprowadzania dłuższych programów możemy mieć ochotę na robienie przerw, oczywiście po uprzednim zabezpieczeniu naszej pracy. W tym celu wprowadzamy L i RETURN, a następnie przechowujemy program w wybranej pamięci zewnętrznej w postaci umożliwiającej późniejsze łączenie.

Jeśli ktoś nie chce używać Edytora BASIC'a, preferując wbudowany edytor Atari, także może sprawdzić i ewentualnie poprawić wprowadzony już i przechowany w pamięci zewnętrznej program. W tym celu należy załadować go do pamięci, następnie dołączyć Edytor BASIC'a rozkazem ENTER i uruchomić go (GOTO 326ØØ). Na propozycję używania skrótów musimy odpowiedzieć N. Gdy Edytor pojawi się na ekranie, należy podać rozkaz P i pierwsza linia programu pojawi się w oknie wprowadzania. W celu jej przetworzenia należy nacisnąć RETURN. Linia programu, wraz z sumą kontrolną, pojawi się w górnym oknie. Jeżeli suma kontrolna zgadza się z podaną w wydruku, należy dwa razy nacisnąć RETURN i przetwarzaną będzie następna linia programu.

Po stwierdzeniu, że musimy poprawić aktualną linię, podajemy rozkaz E i zostanie ona przesunięta do okna wprowadzania.

Może się zdarzyć, że po wprowadzeniu programu za pomocą Edytora BASIC'a nie działa on właściwie. Możliwe są dwie przyczyny takiej sytuacji. Po pierwsze, mogło się zdarzyć, że nie wykonaliśmy dokładnie instrukcji wprowadzania programu. Dlatego zawsze należy wcześniej przeczytać artykuł dotyczący programu. Pod drugie, chociaż możemy zaufać Edytorowi, że unikniemy błędów wprowadzania linii programu, nie może on nam powiedzieć, czy nie opuściliśmy pewnych linii. Jeśli program nie chce działać, należy upewnić się, czy wszystkie linie zostały wprowadzone.

Edytor BASIC'a zgodny jest z podobnymi programami używanymi aktualnie przez „ANALOG Computing" i „ANTIC". Opracowany został na podstawie następujących programów: BASIC Editor, BASIC Editor II i Typo II. Dzięki temu można go wykorzystać do wprowadzania programów zamieszczanych w tych czasopismach, a także w „Bajtku".

```
32595 REM Edytor BASIC'u
32597 REM Wydruk nr 1
32598 REM
32600 IF FL THEN 32616
32602 DIM L$(115),SV$(115),C2$
(2),B$(115),M$(119),S$(98),E$(
69),A$(1):FL=1:STMTAB=PEEK(136
)+PEEK(137)*256
32604 GRAPHICS 0:POKE 710,0:P=
0:ABR=0:? "CZY CHCESZ UZYWAC S
KROTOW INSTRUKCJI";
32605 INPUT A$:IF A$="T" OR A$
="t" THEN ABR=1
32606 B$(1)=" ":B$(115)=" ":B$
(2)=B$
32616 OPEN #17,4,0,"E:":L$=" "
:GOSUB 32662:START=0
32618 POKE 766,1:POKE 83,39:PO
SITION 1,3:IF LEN(L$)<39 THEN
? L$:GOTO 32624
32620 IF LEN(L$)<77 THEN ? L$(
1,38):? L$(39,LEN(L$)):GOTO 32
624
32622 ? L$(1,38):? L$(39,76):?
 L$(77,LEN(L$))
32624 POKE 752,0:POKE 766,0:PO
KE 559,34:POKE 82,1:POKE 83,38
:POSITION 0,10:? " ";:INPUT #1
7,L$:POKE 766,1
32626 IF (L$="P" OR L$="p") AN
D START=0 THEN P=1:L$=""
32628 IF L$="E" OR L$="e" THEN
 E=1:POSITION 1,10:? SV$:GOTO
32624
32630 IF L$="L" OR L$="l" THEN
 GOSUB 32704
32632 IF L$="" AND P=1 THEN 32
686
32634 IF L$="" THEN 32624
32636 IF L$="B" OR L$="b" THEN
 GRAPHICS 0:? "WPROWADZ 'GOTO
32600' W CELU KONTYNUACJI":END
32638 IF L$(1,1)="E" OR L$(1,1
)="e" THEN E=1:TRAP 32624:EL=V
AL(L$(2)):POSITION 1,9:LIST EL
:GOTO 32624
32640 SV$=L$:TRAP 32624:X=VAL(
L$)
32642 START=1:IF P AND  NOT E
THEN 32652
32644 GOSUB 32674:IF  NOT ABR
OR P THEN 32652
32646 POKE 766,0:? CHR$(125):P
OSITION 0,3:L=VAL(L$):LIST L:?
 :? :? "CONT":L$=B$
32648 POSITION 0,0:POKE 842,13
:STOP
32650 POKE 842,12:A=USR(ADR(S$
),ADR(L$),4):L$=L$(1,A)
32652 CHKSUM=USR(ADR(M$),ADR(L
$),LEN(L$)):CHKSUM=CHKSUM+PEEK
(1542)*65536

32654 CHK=CHKSUM-(INT(CHKSUM/6
76)*676):HI=INT(CHK/26):LO=CHK
-(HI*26):C2$(1)=CHR$(HI+65):C2
$(2)=CHR$(LO+65)
32656 IF  NOT P OR E THEN E=0:
GOSUB 32662:IF  NOT P THEN 326
58
32658 POKE 83,39:POKE 752,1:FO
R X=3 TO 5:POSITION 1,X:? B$(1
,38):POSITION 1,X+7:? B$(1,38)
:NEXT X:POKE 83,38
32660 POKE 766,1:POKE 83,38:PO
SITION 7,7:? C2$:POKE 752,0:GO
TO 32618
32662 GOSUB 32702:POKE 766,0:P
OKE 752,1:? CHR$(125):POKE 82,
1:DL=PEEK(560)+256*PEEK(561)+4
32664 POKE DL-1,70:POKE DL+2,6
:POKE DL+3,112:POKE DL+4,112:P
OKE DL+5,112:POKE DL+13,112:PO
KE DL+14,112
32666 POKE DL+22,112:POKE DL+2
3,112:POKE DL+24,65:POKE DL+25
,PEEK(560):POKE DL+26,PEEK(561
):POKE 83,39
32668 POSITION 20,0:? "     edy
tor basic       ":POSITION 0,7:?
 "          OKNO WPROWADZANIA
     "
32670 POSITION 0,1:? "
   OKNO KOMUNIKATU
   ":POSITION 1,7:? "KOD  ";
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:PO
KE 766,1:POKE 82,0:POKE 83,39:
POSITION 0,3:? L$:? :? :? "
CONT":POSITION 0,0
32676 POKE 842,13:STOP
32678 POKE 842,12:TRAP 32682:A
=USR(ADR(E$),VAL(L$)):IF A=4 T
HEN POP :GOTO 32682
32680 RETURN
32682 GOSUB 32662:SOUND 0,75,1
0,8:FOR X=1 TO 20:NEXT X:SOUND
 0,0,0,0:POSITION 1,3:? "BLAD
SKLADNI!":POKE 766,1
32684 POKE 83,38:POSITION 1,10
:? SV$:GOTO 32624
32686 LINE=PEEK(STMTAB)+PEEK(S
TMTAB+1)*256:IF LINE>32599 THE
N 32690
32688 OFS=PEEK(STMTAB+2):STMTA
B=STMTAB+OFS:POSITION 1,9:LIST
 LINE:GOTO 32624
32690 POSITION 1,10:? "Numer l
inii nie moze byc wiekszy od
 32599 !"
32692 FOR I=1 TO 1000:NEXT I:G
OTO 32624

32702 POKE 16,112:POKE 53774,1
12:RETURN
32704 POSITION 5,12:? "KASETA
CZY DYSKIETKA";:INPUT L$:IF L$
<>"K" AND L$<>"D" THEN 32704
32706 IF L$="D" THEN 32714
32708 POSITION 1,12:? "Przysot
uj kasete i nacisnij RETURN":G
OSUB 32720
32710 LIST "C:",1,32599:GOTO 3
2718
32714 POSITION 0,12:? "NAZWA Z
BIORU (D:FILENAME.EXT)    ";CHR
$(30);CHR$(30);CHR$(30);:INPUT
 L$
32715 TRAP 32716:OPEN #1,4,0,L
$:CLOSE #1:GOTO 32722
32716 LIST L$,1,32599
32718 POSITION 0,12:? "    <36
znakow spacji>               "
:RETURN
32720 GET #17,A:RETURN
32722 ? CHR$(125);"ZBIOR JUZ I
STNIEJE!":? "CZY WYKASOWAC GO"
;:INPUT A$:TRAP 32722:IF A$="T
" THEN 32716
32724 FOR X=12 TO 15:POSITION
0,X:? B$:NEXT X:GOTO 32714
```

```
2 REM Edytor BASIC'u
3 REM Wydruk nr 2
4 REM
10 DIM L$(120),ML$(119),A$(1)
20 GRAPHICS 0:POKE 710,0:? "Dy
sk czy Kaseta";:INPUT A$:IF A$
<>"K" AND A$<>"D" THEN 20
30 IF A$="K" THEN 50
40 ? "Umiesc sformatowana dysk
ietke w stacji dyskow, ";:? "n
astepnie nacisnij RETURN":INPU
T L$
45 INPUT L$:OPEN #1,8,0,"D:ML.
DAT":GOTO 60
50 ? :? "Przysotuj kasete, nac
isnij RETURN";:INPUT L$:OPEN #
1,8,0,"C:"
60 L$="32608 M$(1)=":L$(13)=CH
R$(34)
70 N=119:GOSUB 130:L$(14)=ML$(
1,58):L$(LEN(L$)+1)=CHR$(34):?
 #1;L$
80 L$(1)="32610 M$(59)=":L$(14
)=CHR$(34):L$(15)=ML$(59):L$(L
EN(L$)+1)=CHR$(34):? #1;L$
90 L$(1)="32612 S$=":L$(10)=CH
R$(34)
100 ML$="":N=98:GOSUB 130:L$(1
1)=ML$:L$(LEN(L$)+1)=CHR$(34):
? #1;L$
110 L$(1)="32614 E$=":L$(10)=C
HR$(34)
120 ML$="":N=69:GOSUB 130:L$(1
1)=ML$:L$(LEN(L$)+1)=CHR$(34):
? #1;L$:END
130 FOR X=1 TO N:READ A:ML$(X)
=CHR$(A):NEXT X:RETURN
140 DATA 104,104,133,204,104,1
33,203,104,104,133,205,169,0,1
41,3,6,141,2,6,141,4,6,141,5,6
150 DATA 141,6,6,238,3,6,32,68
,218,172,2,6,177,203,133,212,3
2,170,217,32,182,221,32,68,218
160 DATA 173,3,6,133,212,32,17
0,217,32,219,218,32,210,217,16
5,212,141,0,6,165,213,141,1,6,
24
170 DATA 173,0,6,109,4,6,141,4
,6,173,1,6,109,5,6,141,5,6,144
,3,238,6,6,238,2
180 DATA 6,172,2,6,196,205,208
,176,173,4,6,133,212,173,5,6,1
33,213,96
190 DATA 104,104,133,204,104,1
33,203,104,104,141,255,6,169,0
,133,213,216,165,88,133,205,16
5,89,133,206
200 DATA 174,255,6,24,165,205,
105,40,133,205,144,2,230,206,2
02,208,242,160,0,177,205,201,6
4,144,18

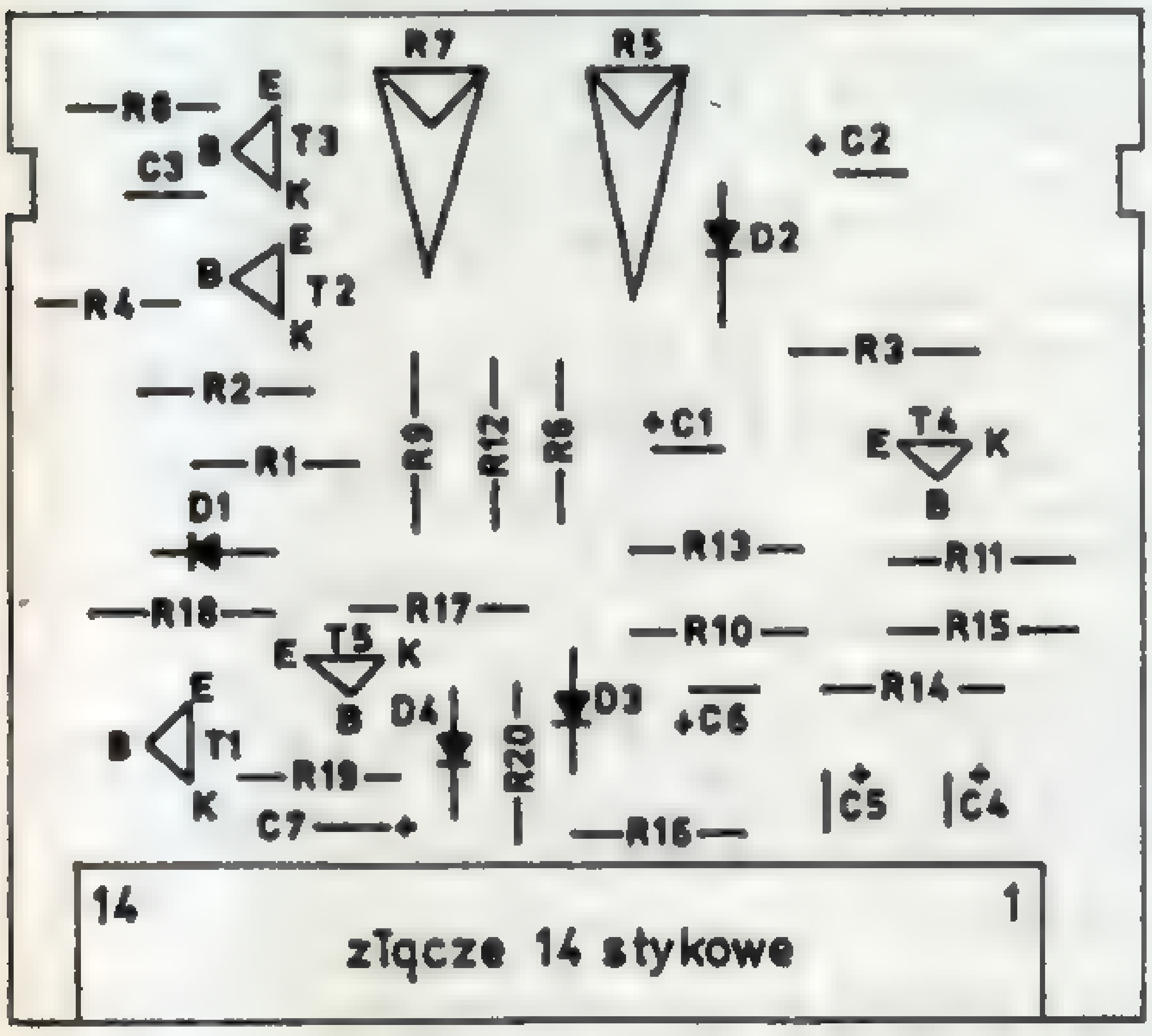
210 DATA 201,96,144,19,201,128
,144,10,201,192,144,6,201,224,
144,7,176,8,24,105,32,144,3,56
,233
220 DATA 64,145,203,200,192,11
4,240,2,208,215,177,203,201,32
,208,3,136,208,247,200,132,212
,96
230 DATA 104,104,141,254,6,104
,141,253,6,169,0,133,213,216,1
65,136,133,205,165,137,133,206
,160,0,177
240 DATA 205,205,253,6,208,8,2
00,177,205,205,254,6,240,15,16
0,2,177,205,24,101,205,133,205
,144,228
250 DATA 230,206,176,224,160,4
,177,205,201,55,240,4,160,0,24
0,0,132,212,96
```

Wobec częstych pytań informujemy, że tylko autorzy dysponują kopiami (na taśmie magnetycznej) wszystkich programów zamieszczonych w ramach cyklu „W szponach Atari". Zainteresowanych kopiami tych programów (i leniwych) prosimy o bezpośrednie skontaktowanie się z jednym z autorów.

Podajemy adres: **Ludwik PIELA**
Ul. Topograficzna 7/2
00-910 WARSZAWA

Helios — monitorem

Wielu użytkowników mikrokomputerów korzysta z telewizora połączonego z komputerem poprzez wejście antenowe. Jakość uzyskiwanego tak obrazu nie jest zadowalająca i najczęściej nie ma dźwięku. Rezygnacja z toru wysokiej częstotliwości telewizora (i komputera) daje znaczną poprawę jakości obrazu oraz umożliwia uzyskanie fonii. Opisany w artykule układ — moduł monitora umożliwia wykorzystanie jako monitora telewizora kolorowego HELIOS TC 500. OTV HELIOS jest fabrycznie przygotowany do zainstalowania tego modułu i w związku z tym wymaga niewielu przeróbek. Należy w tym miejscu zaznaczyć, że układ ten nie zapewnia uzyskania kolorowego obrazu z komputera, jeżeli telewizor nie posiada dekodera PAL. Przy wyposażeniu tylko w dekoder SECAM obraz będzie czarno-biały. Moduł monitora umożliwia współpracę telewizora z dowolnym mikrokomputerem mającym wyjścia kompletnego sygnału wizji (ang. composite video) i fonii. W rozwiązaniu modelowym z Heliosem współpracuje Atarii 130XE. Możliwa jest również współpraca telewizora z magnetowidem.

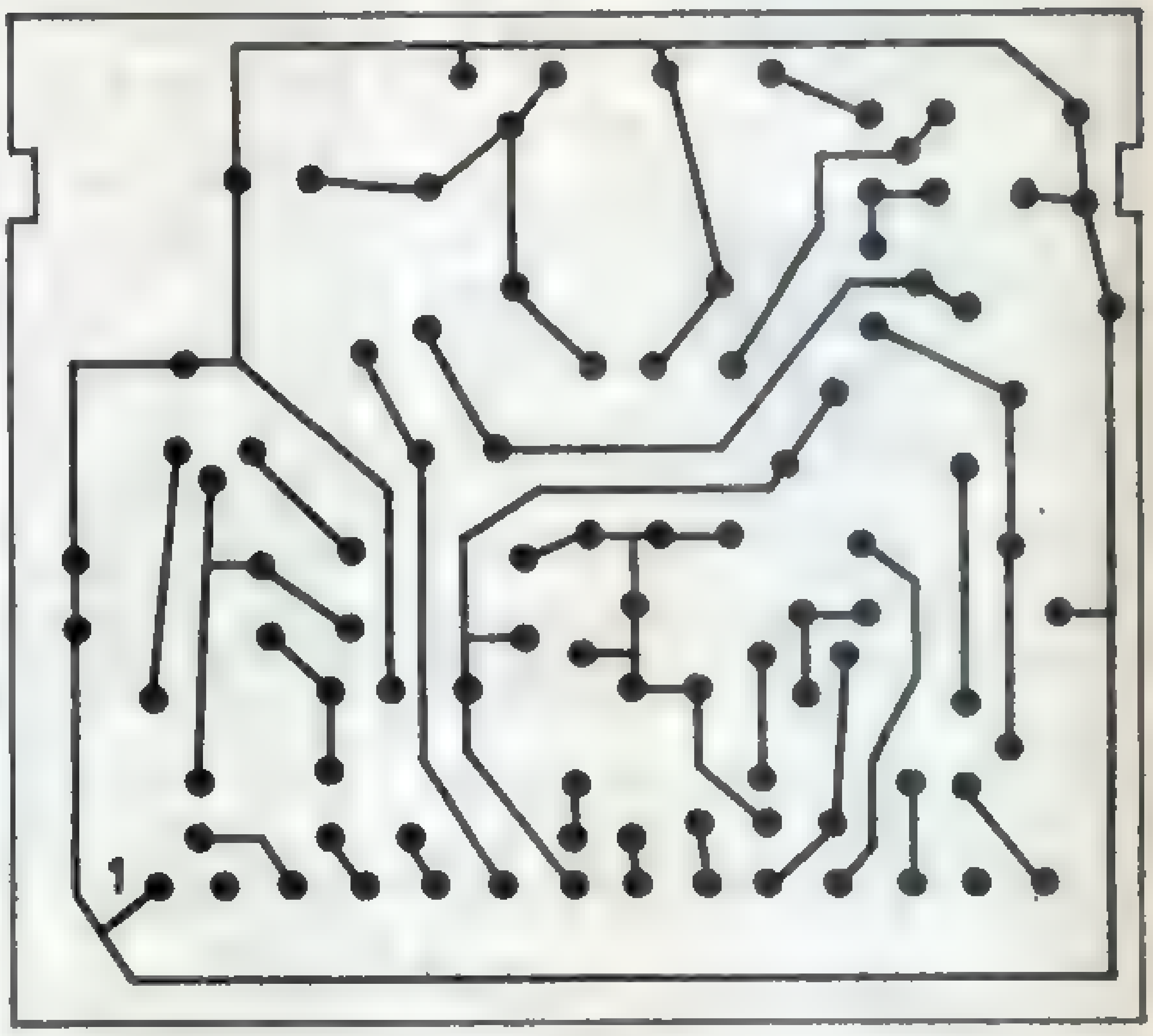


rys. 1

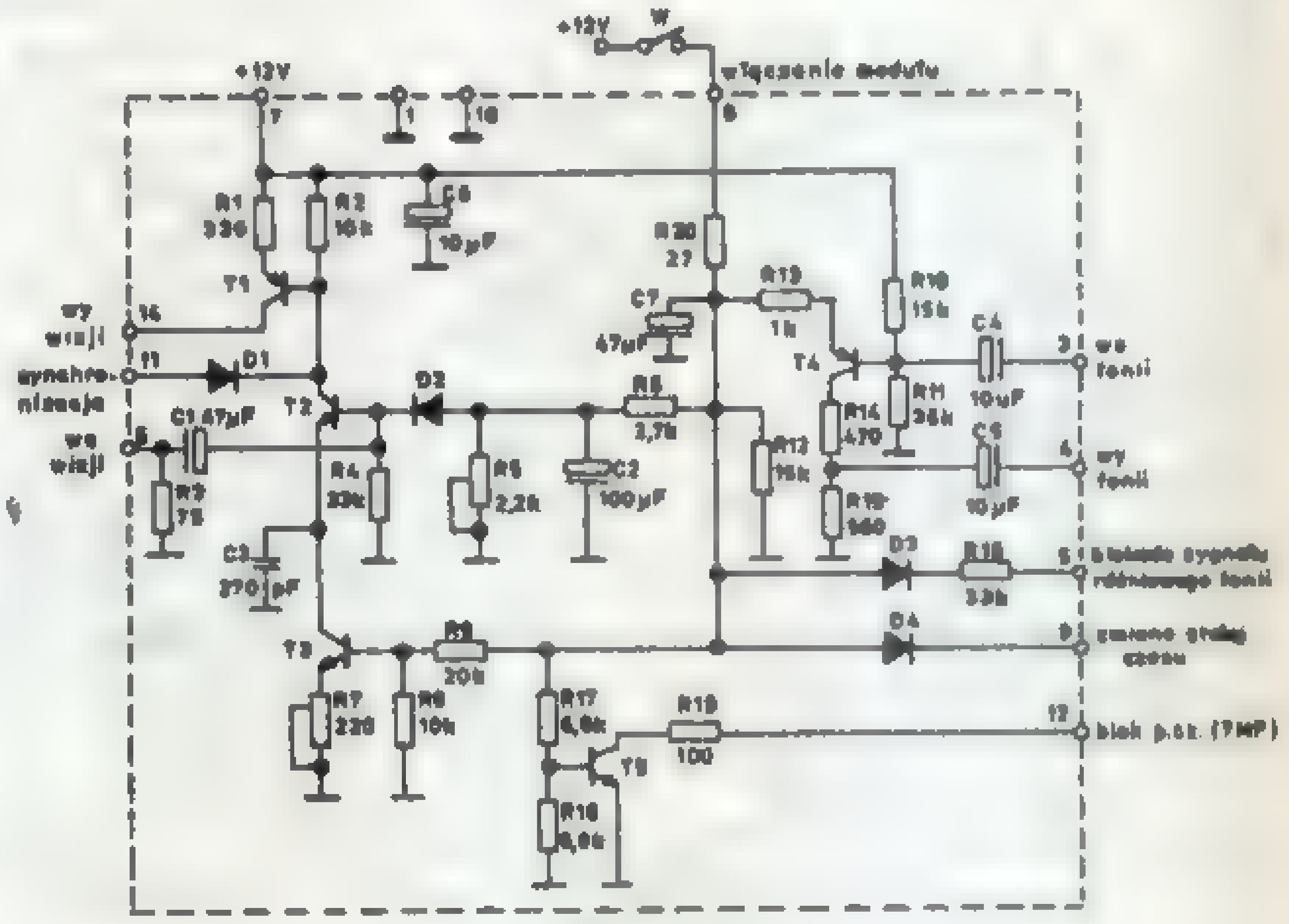
Schemat źródłowy modułu przedstawiono na **rysunku nr 1**. Zadaniem układu jest dopasowanie sygnałów wizji i fonii z komputera do poziomu wymaganego przez układy telewizora oraz wyłączenie z pracy części w.cz. odbiornika. Przełączenie telewizora w tryb pracy monitora odbywa się przełącznikiem „W". Przez jego styki podawane jest napięcie +12 V zapewniające pracę wzmacniacza wizji (T1, T2, T3) i fonii (T4) oraz powodujące blokowanie części w.cz. telewizora: modułu pośredniej częstotliwości MP, sygnału różnicowego fonii w module MF oraz zmianę stałej czasu w module synchronizacji MH. Część w.cz. musi być wyłączona, aby umożliwić współpracę z zewnętrznym źródłem sygnału wizji i fonii. Powrót do pracy w trybie odbiornika TV odbywa się poprzez ustawienie przełącznika „W" w przeciwne położenie.

Układ zamontowano na jednostronnie drukowanej płycie (rys. 2 i 3). Moduł wyposażono w 14-stykowe gniazdo (tego samego typu, jak pozostałe moduły telewizora), a w bloku BS 2030 w miejscu oznaczonym „MM" wlutowano 14 kołeczków stykowych. Ponadto konieczne są jeszcze przeróbki bloku BS 2030 (rys. 4):

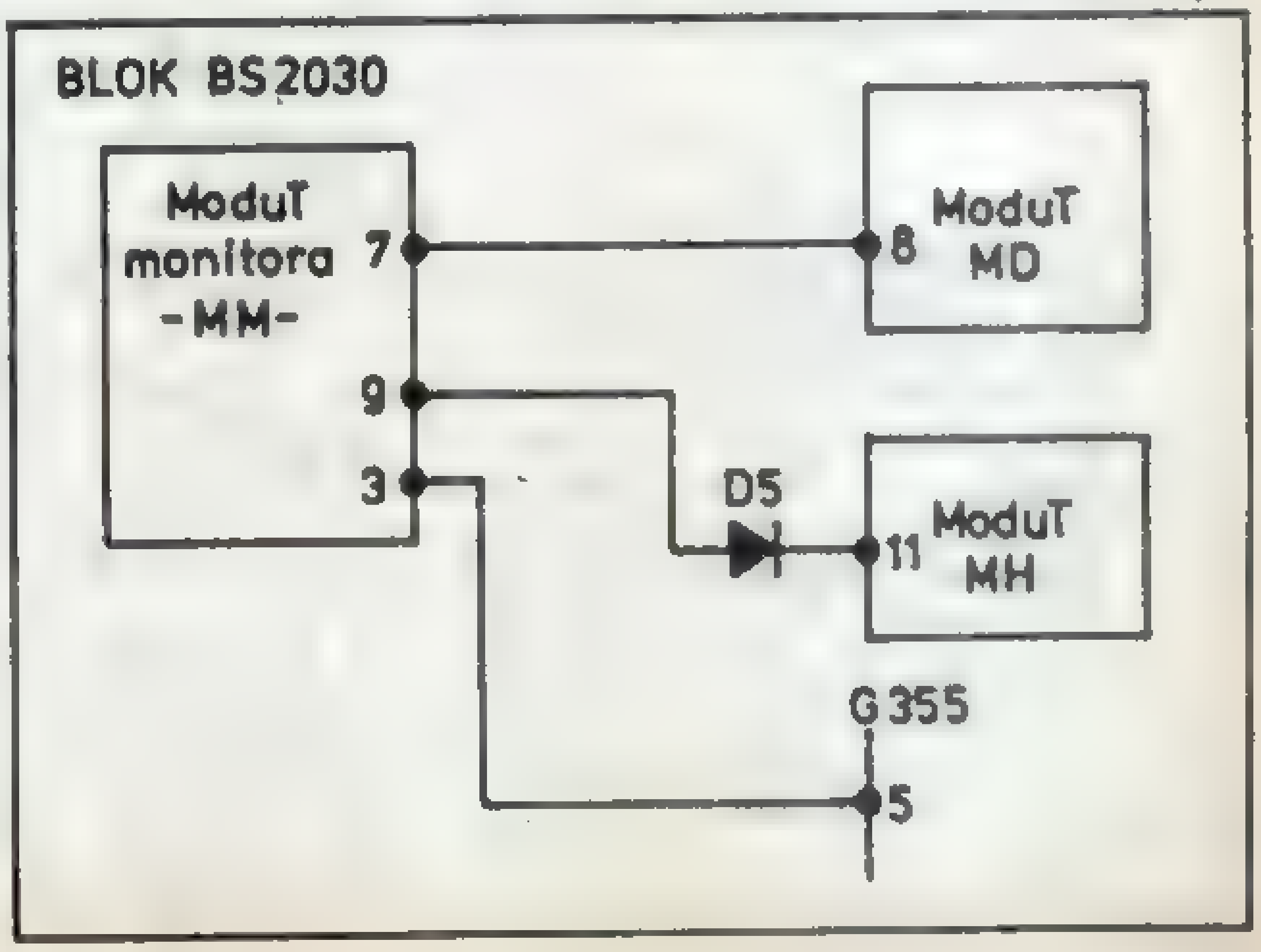
- połączenie punktu 3 złącza z punktem 5 gniazda G 355,
- rozcięcie ścieżki dochodzącej do punktu 11 (ST.CZASU) złącza MH i połączenie rozciętej ścieżki diodą D5,
- doprowadzenie napięcia +12 V z punktu 8 złącza MD do punktu 7 złącza MM.



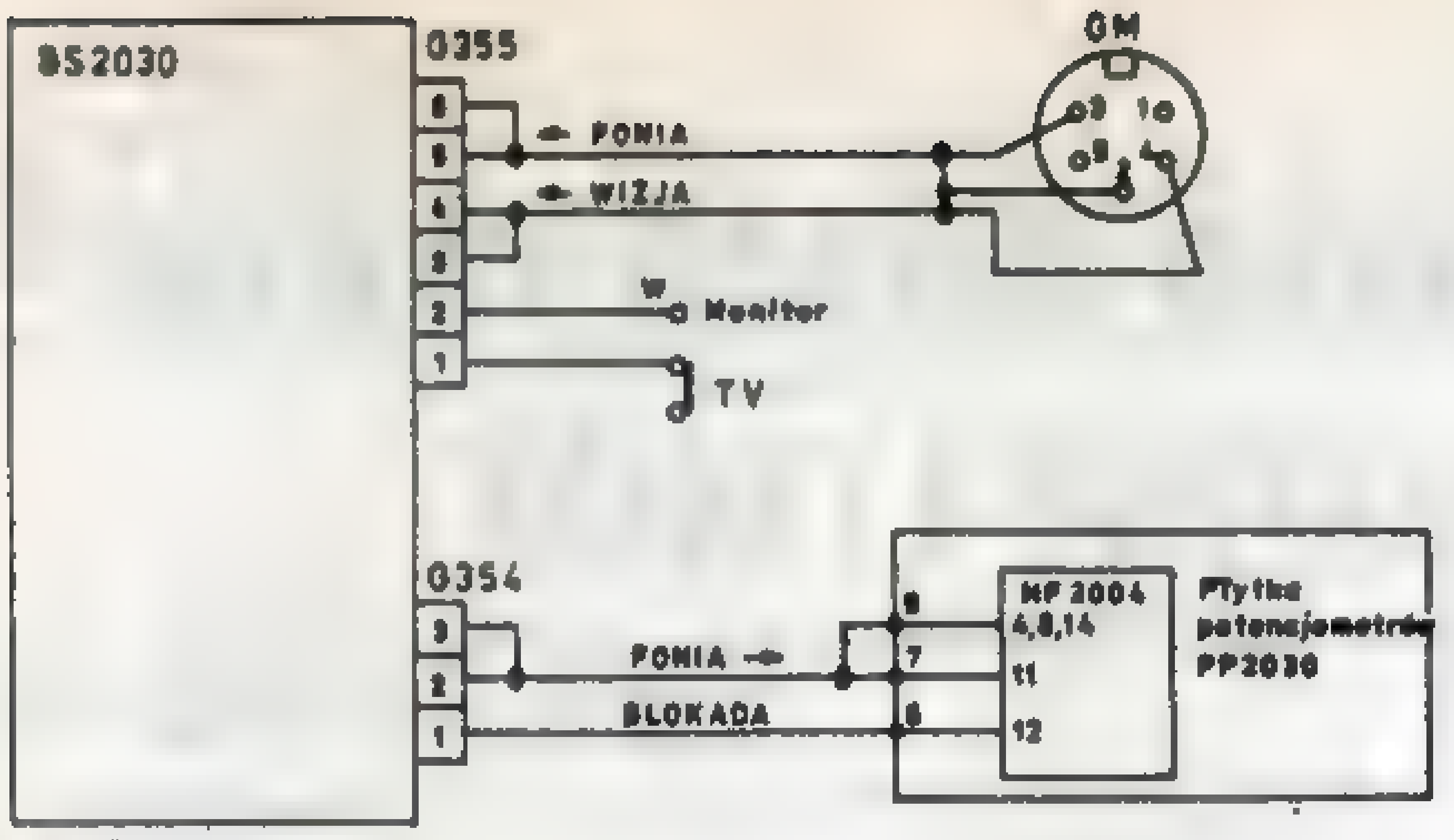
rys. 2



rys. 3



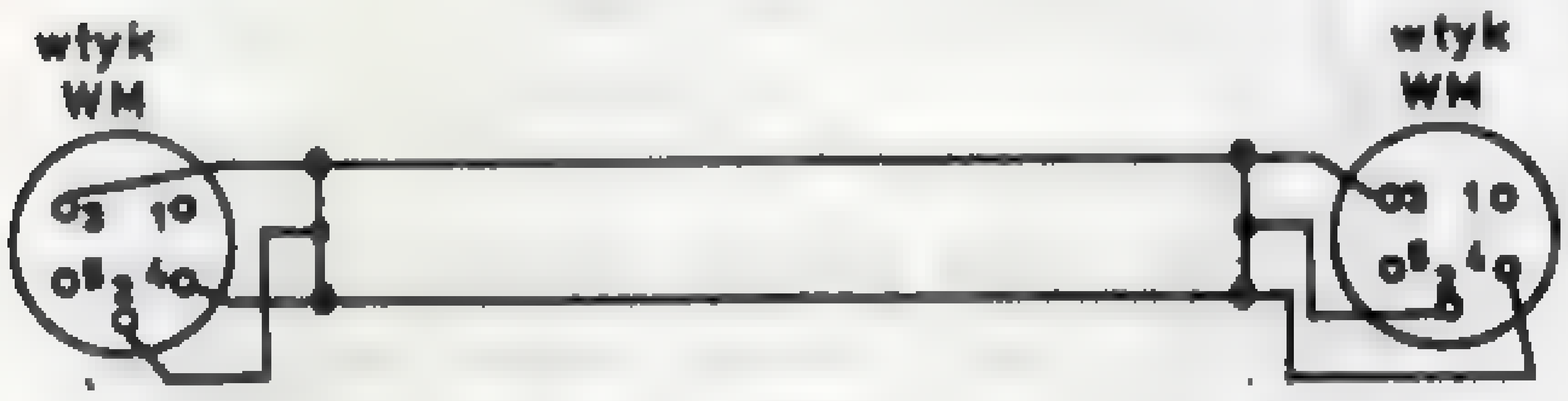
rys. 4



rys. 5

Doprowadzenie zewnętrznych sygnałów wizji i fonii do modułu zrealizowano poprzez 5-stykowe gniazdo magnetofonowe zamontowane w miejscu firmowego gniazda służącego do nagrywania na magnetofon (firmowe połączenia z tym gniazdem należy usunąć). Połączeń między gniazdem, płytką potencjometrów PP 2030 a blokiem BS 2030 należy dokonać ekranowym przewodem (rys. 5). Włącznik „W" umieszczono z tyłu telewizora przy gniazdku antenowym.

Do połączenia komputera z telewizorem wykorzystano ekranowany przewód dwużyłowy zakończony z obu stron 5-stykowymi wtykami magnetofonowymi (rys. 6).



rys. 6

Po wykonaniu powyższych czynności oraz sprawdzeniu ich poprawności umieszczamy moduł w złączu MM w bloku BS 2030 i przystępujemy do regulacji. Potencjometry montażowe R5 i R7 ustawiamy w środkowym położeniu. Włączamy telewizor i przyłączamy w tryb pracy monitora (włącznikiem „W").

Do gniazda Gm doprowadzamy sygnał z komputera. Regulację najlepiej jest przeprowadzać gdy wytwarzany przez komputer obraz jest nieruchomy i kolorowy.

Może to być np.: program 256 kolorów z programu demonstracyjnego ATARI. Regulujemy potencjometrem R5 tak, aby uzyskać na ekranie wytwarzany przez komputer obraz. Następnie ustawiamy potencjometr R7 tak, aby obraz był stabilny a odtwarzanie kolorów właściwe. Ewentualnej korekty nasycenia barw możemy dokonać zmieniając ponownie ustawienie R5. Na tym regulację kończymy. W wypadku współpracy z innym komputerem (lub magnetowidem) może zajść konieczność zmiany wartości rezystora R15 w torze fonii modułu. Należy w tym wypadku kierować się zasadą, aby przy takim samym położeniu suwaka regulacji głośności uzyskać podobne natężenie dźwięku w obu trybach pracy odbiornika.

Bieżącej korekcji jasności, nasycenia, kontrastu i głośności dokonujemy elementami regulacyjnymi telewizora.

Spis elementów

Tranzystory
T1, T4 — BC 308, BC 178
T2, T3, T5 — BC 238, BC 108

Diody
D1÷D5 — BAVP 17, 18, 19

Rezystory
Typu MŁT 0,125 W
o wartościach jak na schemacie,
R5, R7 — miniaturowe potencjometry montażowe (poziome).

Kondensatory
Elektrolityczne napięcie 16 V
C3 — ceramiczny.

Inne
W — przełącznik suwakowy miniaturowy dowolnego typu.
Gniazdo 14-stykowe.
Kołeczki stykowe — 14 szt.
Gniazdo magnetofonowe 5-stykowe typu GM.
Wtyki magnetofonowe 5-stykowe typu WM.

LITERATURA:
[1] Instrukcja serwisowa HELIOS TC 500 — Wema 1985 Warszawa,
[2] M.DEC. Monitory, monitory... — Komputer 9/1986

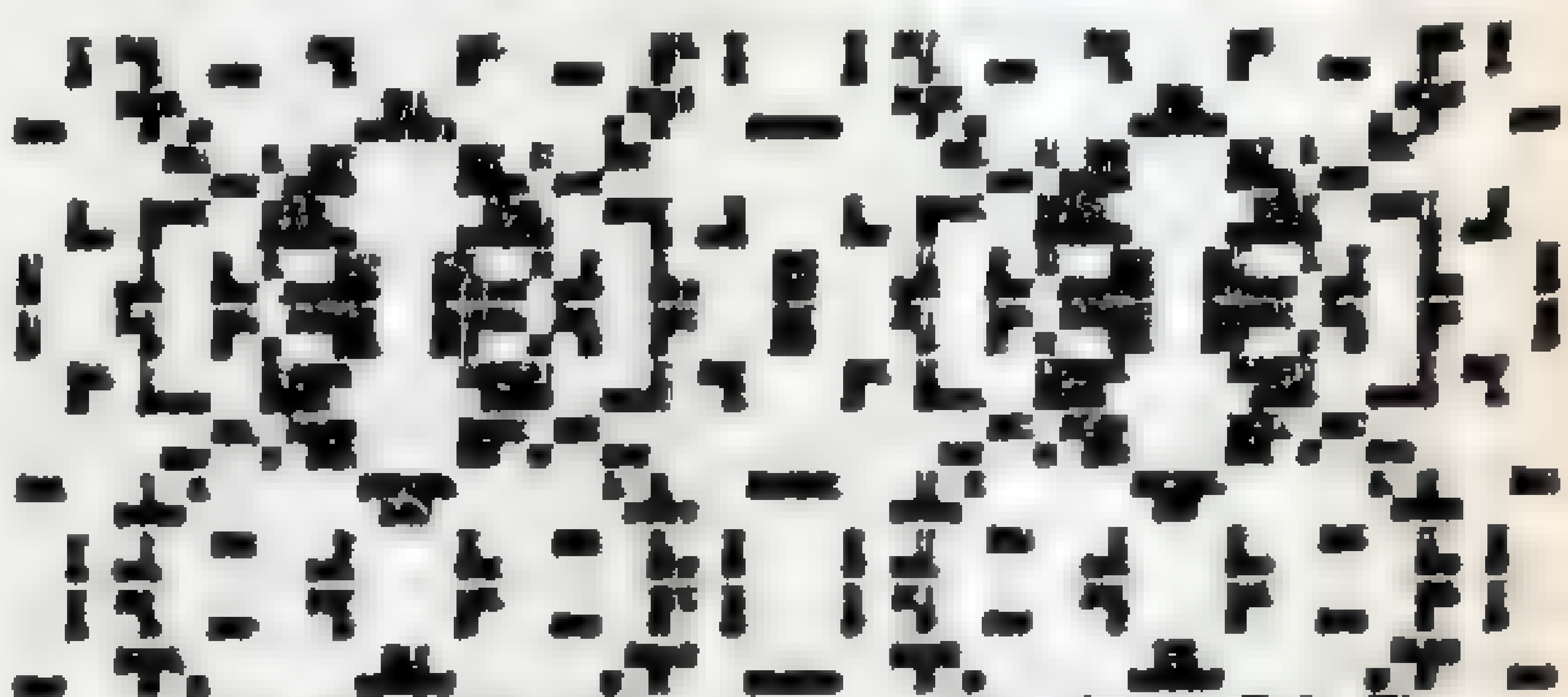
Bogusław LIPCZYŃSKI

PROGRAM 57

ZX SPECTRUM

Makatka

```
  1 BORDER 0: CLS : DIM b$(64): PRINT PAPER 0;b$: LET j=1: LET d=2: LET t=3: LET cz=4: LET e=11: LET f=12: LET g=15: LET h=16: LET i=21: LET k=31: LET l=13: LET p=5: LET o=8: DIM a$(80): LET a$="[illegible]"
  2 PAPER INT (4+RND*4): INK INT (RND*4): LET a=cz*INT (RND*9): LET b=INT (RND*p): LET c=INT (RND*o): PRINT AT b+d,c;a$(a+j);AT e-b,c;a$(a+t);AT b+d,g-c;a$(a+d);AT e-b,g-c,a$(a+cz);AT b+d,c+h;a$(a+j);AT e-b,c+h;a$(a+t);AT b+d,k-c;a$(a+d);AT e-b,k-c;a$(a+cz),AT b+f,c;a$(a+j);AT i-b,c;a$(a+t);AT b+f,g-c;a$(a+d);AT i-b,g-c;a$(a+cz);AT b+f,c+h;a$(a+j);AT i-b,c+h;a$(a+t);AT b+f,k-c;a$(a+d);AT i-b,k-c;a$(a+cz): GO TO d
  3 SAVE "Makatka" LINE 1
```



J. Rubinowicz

Wyznaczanie rozwiązania układu równań różniczkowych zwyczajnych I rzędu

Wojciech KOZERA

Program służy do graficznego przedstawienia rozwiązania układu równań różniczkowych, opisujących, w wypadku ogólnym, wielowymiarowe obiekty dynamiczne nieliniowe, w oparciu o metodę Rungego-Kutty czwartego rzędu (1) (str. 325). Treść programu w języku BASIC na komputer Amstrad/Schneider została opracowana na podstawie prac (2), (3).

Dane początkowe

Przed uruchomieniem programu należy wstawić do programu głównego instrukcje określające równania wyjścia (zgodnie z komentarzem w liniach 110—126) oraz instrukcje określające prawe strony równań stanu obiektu dynamicznego (zgodnie z komentarzem w liniach 286—296).

W trybie konwersacyjnym zadawane są wartości następujących danych:

- *n* — liczba zmiennych stanu obiektu,
- $x^0(m)$ — wartość i — tego elementu wektora stanu w chwili początkowej (m = 1...n),
- *tp* — chwila początkowa,
- *tk* — chwila końcowa,
- *kr* — krok wydruku,
- *lw* — liczba wykresów (patrz komentarz w liniach 110—126 programu głównego,
- *mik* — najmniejszy dopuszczalny krok,
- *eps* — względna dokładność rozwiązania,
- *eta* — oczekiwany rząd odpowiedzi.

Ponadto, na pytanie o wydruk wyników na drukarce, należy nacisnąć klawisz (T) dla akceptacji, lub (N) — w przeciwnym wypadku.

Wyniki końcowe

Efektem działania programu jest wyświetlenie na ekranie monitora, a przy wybranej opcji druku — także wydruk na drukarce, przebiegów w czasie wybranych zmiennych oznaczonych kolejno małymi literami alfabetu (literze a odpowiada przebieg pierwszej zmiennej).

Przykład testowy

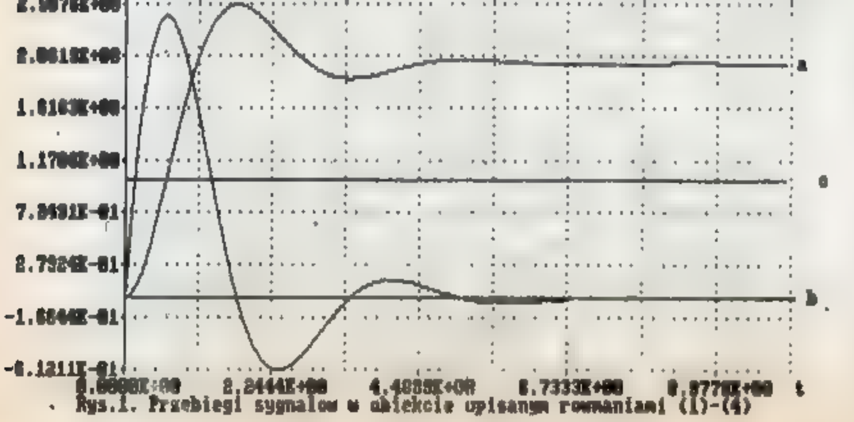
Program sprawdzono, między innymi, na przykładzie obiektu jednowymiarowego opisanego równaniami:

$$x_1(t) = x_2(t), \qquad (1)$$

$$x_2(t) = -4 \cdot x_1(t) - 1.6 \cdot x_2(t) + 8 \cdot u(t), \qquad (2)$$

$$y(t) = x_1(t), \qquad (3)$$

$$u(t) = 1. \qquad (4)$$



Rys.1. Przebiegi sygnalow w obiekcie opisanym rownaniami (1)-(4)

Instrukcje, określające prawe strony równań (1) i (2), mają postać:

$$d(1) = yw(2), \qquad (5)$$

$$d(2) = 4 * yw(1) - 1.6 * yw(2) + 8 \qquad (6)$$

a instrukcje, określające równania wyjścia, przyjęto w postaci:

- w(le,1) = y0(1) — przebieg odpowiedzi y(t) obiektu, oznaczony na wykresie literą *a*,
- w(le,2) = y0(2) — przebieg składowej $x_2(t)$ wektora stanu *x(t)* obiektu, oznaczony na wykresie literą *b*,
- w(le,3) = 1 — przebieg wymuszenia *u(t) = 1(t)*, oznaczony na wykresie literą *c*.

Dla *n* = 2, *x0(1)* = *x0(2)* = 0, *tp* = 0, *tk* = 10, *kr* = 0,02, *lw* = 3, *mik* = *eps* = *eta* = 1E-6 oraz wybranej opcji druku uzyskano przebiegi przedstawione na rys. 1.

```
2 REM  ************************************************
4 REM  *  WYZNACZANIE ROZWIAZYWANIA UKLADU ROWNAN   *
6 REM  *  ROZNICZKOWYCH  ZWYCZAJNYCH  I-GO  RZEDU   *
8 REM  *       (C): Wojciech Kozera, Szczecin 1988   *
10 REM ************************************************
12 DEFINT i-n
14 MEMORY &9FFF
16 GOSUB 480
18 MODE 2:WINDOW #1,42,80,15,16
20 PRINT SPC(16);
22 PRINT CHR$(24);" PROGRAM  WYZNACZANIA  ROZWIAZYWANIA  UKLADU ";CHR$(24)
24 PRINT SPC(16);
26 PRINT CHR$(24);" ROWNAN ROZNICZKOWYCH ZWYCZAJNYCH I-GO RZEDU ";CHR$(24)
28 ir=-1
30 LOCATE 20,5:PRINT CHR$(7);:INPUT "Liczba zmiennych stanu ........  n=",n
32 DIM d(n),y(n),y0(n),y1(n),y2(n),y3(n),yh(n),yw(n)
34 LOCATE 25,6:PRINT "Warunki poczatkowe :  x0(";
36 FOR m=1 TO n
38   LOCATE 50,6:PRINT SPACE$(20):LOCATE 50,6
40   PRINT USING("#");m;:PRINT CHR$(7);")";
42   INPUT "=",y0(m)
44 NEXT m
46 LOCATE 20,6:PRINT SPACE$(59);
48 LOCATE 20,6:PRINT CHR$(7);:INPUT "Chwila poczatkowa .............. tp=",tp
50 x0=tp
52 LOCATE 20,7:PRINT CHR$(7);:INPUT "Chwila koncowa ................. tk=",tk
54 LOCATE 20,8:PRINT CHR$(7);:INPUT "Krok wydruku ................... kr=",rkr
56 LOCATE 20,9:PRINT CHR$(7);:INPUT "Liczba wykresow ................ lw=",lw
58 LOCATE 20,10:PRINT CHR$(7);"Czy drukowac wykresy  <T/N> ? :  ";
60 s$=UPPER$(INKEY$):IF (s$="") THEN GOTO 60
62 IF (s$="T") THEN PRINT "Tak" ELSE PRINT "Nie"
64 LOCATE 18,12:PRINT CHR$(7);
66 INPUT "Najmniejszy krok dopuszczalny ....  mik=",rmh
68 LOCATE 18,13:PRINT CHR$(7);
70 INPUT "Wzgledna dokladnosc rozwiazania ..  eps=",eps
72 LOCATE 18,14:PRINT CHR$(7);
74 INPUT "Oczekiwany rzad odpowiedzi .......  eta=",eta
76 LOCATE 18,15:PRINT CHR$(7);"Podpis pod rysunkiem :";
78 INPUT #1,"",t$:t$=" "+t$+" "
80 IF (s$<>"T") THEN GOTO 94
82 i=-1
84 WHILE i
86   IF (INP(&F500) AND 64)=0 THEN i=0
88   IF i THEN LOCATE 30,24:PRINT CHR$(7);"Wlacz drukarke !";
90 WEND
92 PRINT #8,CHR$(27);"1";CHR$(10);
94 LOCATE 24,24:PRINT CHR$(24);" Chwila aktualna ......  t=";CHR$(24);
96 mle=4000/lw
98 DIM w(mle,lw)
100 lt=LEN(t$)/2:lt=40-lt:le=0
102 WHILE (x0<tk)
104   LOCATE 51,24:PRINT SPACE$(28):LOCATE 51,24
106   PRINT CHR$(24);INT(x0*1000000+0.5)*0.000001;CHR$(24);
108   le=le+1

110 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
112 '                  Podaj rownania wyjscia o postaci:
114 '                                                                     '
116 '          w(le,m)=fm(y0(1),y0(2),...,y0(n),x0) ;  m=1..lw           '
118 '                                                                     '
120 '        Uwaga:  Na rysunku  wykresy kolejnych odpowiedzi beda
122 '        -----  oznaczane odpowiednio malymi literami alfabetu
124 '                                                                     '
126 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
128   w(le,1)=y0(1)                                                       '
130   w(le,2)=y0(2)                                                       '
132   w(le,3)=1
134 '                                                                     '
136 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
138  x1=x0+rkr
140  eps1=0.025*eps:w1=rkr:sh=0
142  IF (w1<0) THEN sh=-1
144  IF (w1>0) THEN sh=1
146  nt=0
148  IF ir THEN h=w1:prh=h:ir=0:GOTO 156
150    h=prh
152    IF (ABS(h)<rmah) THEN h=rmah*sh:prh=h
154    IF ((x0+1.25*h-x1)*sh)=0) THEN h=w1 ELSE nt=-1
156  na=-1
158  FOR i=1 TO n
160    y(i)=y0(i)
162  NEXT i
164  x=x0:hp=h
166  GOSUB 274
168  FOR i=1 TO n
170    y1(i)=yh(i)
172  NEXT i
174  hh=0.5*h
176  FOR i=1 TO n
178    y(i)=y0(i)
180  NEXT i
182  x=x0:hp=hh
184  GOSUB 274
186  FOR i=1 TO n
188    y2(i)=yh(i)
190  NEXT i
192  FOR i=1 TO n
194    y(i)=y2(i)
196  NEXT i
198  x=x0+hh:hp=hh
200  GOSUB 274
202  FOR i=1 TO n
204    y3(i)=yh(i)
206  NEXT i
208  w4=0
210  FOR k=1 TO n
212    w1=y1(k):w3=y3(k):w2=ABS(w1-w3):w1=ABS(w1):w3=ABS(w3)
214    IF (w1<w3) THEN w1=w3
216    IF (w1<eta) THEN w1=eta
218    w1=w2/w1
220    IF (w1>w4) THEN w4=w1
222  NEXT k
224  IF (w4<=eps) THEN GOTO 246
226    na=0
228    IF (ABS(hh)>=rmah) THEN GOTO 240
230    CLS
232    PRINT CHR$(7);"** Nie mozna osiagnac zadanej dokladnosci  eps =";eps;
234    PRINT "  **"
236    PRINT " - nawet przy zmniejszeniu kroku do wartosci  kr =";rmah;
238    END
240    h=hh:prh=h:nt=-1
242    FOR k=1 TO n:y1(k)=y2(k):NEXT k
244    GOTO 174
246  FOR k=1 TO n
248    w3=y3(k):y0(k)=w3+6.66666667E-02*(w3-y1(k))
250  NEXT k
252  IF (nt=0) THEN GOTO 262
254    x0=x0+h
256    IF ((w4<eps) AND na) THEN h=h+h:prh=h
258    IF ((x0+1.25*h-x1)*sh)=0) THEN h=x1-x0:nt=0
260    GOTO 156
262  x0=x1
264  IF (le>=mle) THEN GOTO 268
266 WEND
268 GOSUB 320
270 END
272 REM
274 w4=0.5*hp:w1=w4:w2=x
276 FOR k=1 TO n:yw(k)=y(k):yh(k)=0:NEXT k
278 FOR i=1 TO 4
280   ON i GOTO 282,284,284,282
282     w3=w4:GOTO 286
284     w3=hp
286 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
288 ' Przypisz zmiennym d(i) wartosci pochodnych zgodnie z ukladem rownan:   '
290 '                                                                         '
292 '            d(i)=fi(yw(1),yw(2),...,yw(n),w2) ;  i=1..n                 '
294 '                                                                         '
296 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
298     d(1)=yw(2)                                                            '
300     d(2)=-4*yw(1)-1.6*yw(2)+8                                             '
302 '                                                                         '
304 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
306   FOR k=1 TO n
308     w2=d(k):yh(k)=yh(k)+w3*w2:yw(k)=y(k)+w1*w2
310   NEXT k
312   w2=x+w1:w1=w3
314 NEXT i
316 FOR k=1 TO n:yh(k)=y(k)+0.333333333*yh(k):NEXT k
318 RETURN
320 REM ==================================
322 REM =  PROCEDURA SPORZADZANIA WYKRESU  =
324 REM ==================================
326 kk=le MOD 506
328 IF (kk)0) THEN kk=1 ELSE kk=0
330 kk=le\506+kk
332 ymin=1E+10:ymax=-1E+10
334 FOR i=1 TO lw
336   FOR j=1 TO le
338     t=w(j,i)
340     IF (t>ymax) THEN ymax=t
342     IF (t<ymin) THEN ymin=t
344   NEXT j
346 NEXT i
348 ysk=336/(ymax-ymin)
350 FOR k=1 TO kk
352   CLS
354   IF (s$="T") THEN GRAPHICS PAPER 0:GRAPHICS PEN 1:GOTO 360
356   WINDOW 12,75,2,23:PAPER 1:PEN 0:CLS
358   GRAPHICS PAPER 1:GRAPHICS PEN 0
360   WINDOW 1,80,1,25:PAPER 0:PEN 1
362   ia=-8
364   FOR i=1 TO 8
366     ia=ia+48:PLOT 90,ia:DRAWR 2,0:ib=92
368     FOR j=1 TO 63
370       ib=ib+8:PLOT ib,ia
372     NEXT j
374   NEXT i
376   ia=-20
378   FOR i=1 TO 5
380     ia=ia+112:PLOT ia,36:DRAWR 0,2
382   NEXT i
384   ia=36
386   FOR i=1 TO 10
388     ia=ia+56:ib=40
390     FOR j=1 TO 42
392       ib=ib+8:PLOT ia,ib
394     NEXT j
396   NEXT i
398   ia=36
400   FOR i=1 TO 5
402     ia=ia+112:PLOT ia,378:DRAWR 0,2
404   NEXT i
406   tp1=506*(k-1)*rkr+tp:tk1=tp1+505*rkr
408   LOCATE 12-2*lw,1
410   FOR i=1 TO lw:PRINT CHR$(96+i);",";:NEXT i
412   ysk1=(ymax-ymin)/7:t=ymax
414   FOR i=1 TO 8
416     LOCATE 1,3*i-1:PRINT USING("##.####^^^^");t:t=t-ysk1
418   NEXT i
420   LOCATE 76,24:PRINT "t";
422   xsk=(tk1-tp1)/9:t=tp1+xsk:LOCATE 11,1
424   FOR i=1 TO 5
426     PRINT "   ";USING("##.####^^^^");t;:t=t+2*xsk
428   NEXT i
430   t=tp1:LOCATE 4,24
432   FOR i=1 TO 5
434     PRINT "   ";USING("##.####^^^^");t;:t=t+2*xsk
436   NEXT i
438   ia=506*(k-1)+1:ib=ia+505:IF (ib>le) THEN ib=le
440   FOR i=1 TO lw
442     l=1:PLOT 92,INT(ysk*(w(ia,i)-ymin)+0.5)+40
444     FOR j=ia+1 TO ib
446       DRAW l+92,INT(ysk*(w(j,i)-ymin)+0.5)+40
448       l=l+1
450     NEXT j
452     l=YPOS+8:MOVE 593+i*8,l
454     TAG:PRINT CHR$(96+i);:TAGOFF
456   NEXT i
458   IF (tp1*tk1>0) THEN GOTO 462
460     PLOT INT(0.5-1/rkr*tp1)+92,40:DRAWR 0,344
462   IF (ymin*ymax>0) THEN GOTO 466
464     PLOT 92,INT(0.5-ysk*ymin)+40:DRAWR 507,0
466   LOCATE lt,25
468   IF (s$="T") THEN PRINT t$;:CALL &A000 ELSE PRINT CHR$(24);t$;CHR$(24);
470   PRINT CHR$(7);
472   WHILE INKEY$="":WEND
474 NEXT k
476 CLS
478 RETURN
480 REM ==========================
482 REM =  PROCEDURA KOPIOWANIA  =
484 REM =  EKRANU  NA  DRUKARKE  =
486 REM ==========================
488 FOR i=&A000 TO &A0BF
490 READ k:POKE i,k:l=l+k:NEXT
492 DATA &cd,&ba,&bb,&cd,&e7,&bb,&32,&bd
494 DATA &a0,&cd,&bc,&a0,&21,&8f,&01,&22
496 DATA &be,&a0,&11,&00,&00,&3e,&07,&32
498 DATA &c0,&a0,&cd,&7c,&a0,&0e,&00,&3a
500 DATA &c0,&a0,&47,&e5,&d5,&c5,&cd,&f0
502 DATA &bb,&c1,&d1,&21,&bd,&a0,&be,&e1
504 DATA &37,&20,&01,&a7,&cb,&11,&2b,&2b
506 DATA &10,&e9,&cd,&af,&a0,&79,&cd,&a6
508 DATA &a0,&13,&e5,&21,&7f,&02,&37,&ed
510 DATA &52,&e1,&38,&05,&2a,&be,&a0,&18
512 DATA &cc,&23,&7c,&b5,&c8,&2b,&11,&00
514 DATA &00,&22,&be,&a0,&3e,&07,&bd,&20
516 DATA &b9,&7c,&b4,&20,&b5,&3e,&04,&32
518 DATA &c0,&a0,&18,&ae,&3e,&1b,&cd,&a6
520 DATA &a0,&3e,&31,&cd,&a6,&a0,&00,&00
522 DATA &00,&00,&00,&c9,&e5,&3e,&42,&cd
524 DATA &1e,&bb,&e1,&28,&02,&e1,&c9,&3e
526 DATA &0d,&cd,&a6,&a0,&3e,&0a,&cd,&a6
528 DATA &a0,&3e,&1b,&cd,&a6,&a0,&3e,&4c
530 DATA &cd,&a6,&a0,&3e,&7f,&cd,&a6,&a0
532 DATA &3e,&02,&cd,&a6,&a0,&c9,&cd,&2e
534 DATA &bd,&38,&fb,&cd,&2b,&bd,&c9,&3a
536 DATA &c0,&a0,&fe,&07,&c8,&af,&cb,&11
538 DATA &cb,&11,&cb,&11,&c9,&00,&00,&00
540 IF l<>23151 THEN PRINT CHR$(7);"**  Blad w DATA !  **";END
542 RETURN
544 REM ==========================
```

Wojciech KOZERA

LITERATURA:

(1) *H. Górecki. Praca zbiorowa: Algorytmy i programy sterowania, WNT, Warszawa 1980.*

(2) *ELWRO: ODRA—Algol. Procedury. Opis 1204—VIII—11, Wrocław 1970.*

(3) *Schneider CPC - 6128. Benutzerhandbuch.*

PRIMUM NON NOCERE, czyli jak chcemy uczyć podstaw informatyki

Włodzimierz GOGOŁEK

Primum non nocere, zasada ta obowiązuje także nauczycieli. Mają oni bowiem ogromne możliwości kształtowania naszej wiedzy. Pozwala nam ona dalej żyć godnie, dostrzegać dobro i zło, ułatwia poruszanie się w otaczającym świecie. A nie jest to proste u schyłku wieku, który wraz z poprzedzającym go tworzą najburzliwszy okres ze stu wieków, mijających od początku, gdy na Ziemi rozpoczął swoje panowanie homo sapiens.

Po sile pary, potędze elektryczności, poznajemy teraz moc informacji. Stało się to za przyczyną stosunkowo młodziutkiej gałęzi wiedzy, jaką jest informatyka. Fundamenty jej tworzono od bardzo dawna, jednak dopiero (już?) 50 lat temu pierwszy komputer wspomógł intelektualną pracę człowieka. Pozwoliło to dostrzec czubek przysłowiowej góry lodowej. Tworzy ją tutaj znaczenie efektywnego (szybkiego!) wykorzystania i przetworzenia informacji. Dzień dzisiejszy pozwala nam zauważyć zarysy, wcześniej niewyobrażalnego ogromu problematyki związanej z informacją i jej przetwarzaniem lub inaczej z INFORMATYKĄ, prościej — z komputerami.

Jednak mimo ogromnej rangi, jaką ostatnio zdobywa sobie informatyka, należy pamiętać, że jest to dziedzina wiedzy dająca nam narzędzia, które mogą wspomagać każde działanie ludzkie. Poza specjalistycznymi kierunkami studiów, informatyka nie powinna występować zatem samodzielnie, jako cel działania — mamy w tym zakresie duże i jednocześnie smutne doświadczenia — piękne ośrodki obliczeniowe — duma dyrekcji..., senne zajęcia o komputerach, programy pisane za resztki stypendium.

Przystępując zatem do nauczania podstaw informatyki (PI), należy pamiętać — „po pierwsze nie szkodzić", przez zniechęcanie do informatyki. Przyczynia się temu między innymi bardzo ryzykowne traktowanie informatyki jako samodzielnej nauki — przedmiot ten staje się wówczas abstrakcyjny, mało użyteczny. Motywacja do nauki gwałtownie topnieje (widać to po twarzach uczących się, cień zainteresowania pozostaje tylko u nielicznych entuzjastów.

Pierwszym zatem postulatem „by nie szkodzić" jest ścisłe powiązanie nauczania podstaw informatyki z jej praktycznymi zastosowaniami. Nie historia komputerów, system dwójkowy, a atrakcyjne (spektakularne) zastosowania komputerów powinny być tematem pierwszych zajęć. Gwarantuje to, iż już podczas pierwszych zajęć nie stracimy „kredytu zainteresowania" naszych uczniów. Kredyt ten, mimo pozorów, jest bardzo niewielki — ukształtowany zazwyczaj „modą" lub traktowaniem komputerów jako urządzeń do zabawy. Warto w tym miejscu podkreślić pozytywne efekty nieograniczonego dostępu do komputerów w celu wykorzystania ich do zabawy. Okazuje się, że zabawy te zmniejszają okres poznawania podstawowych czynności manualnych, które należy wykonywać podczas obsługi komputera.

Drugim postulatem w myśl zasady „nie szkodzić" jest wyeliminowanie zajęć z podstaw informatyki w szkołach, gdzie nie można zapewnić swobodnego dostępu wszystkim uczącym się tego przedmiotu (dwie osoby na jedno stanowisko). Przy małej liczbie komputerów jest to oczywiście kwestia odpowiedniego podziału na grupy i liczby nauczycieli...

Trzeci postulat — na każdych zajęciach nauczyciel powinien wykorzystywać komputer. Wiąże się to z koniecznością wcześniejszego przygotowania odpowiednich programów demonstracyjnych, także obrazów komputerowych. Konkretne propozycje tego typu pomocy dydaktycznych zawarte będą w cyklicznych odcinkach stanowiących rozwinięcie zaproponowanego dalej programu zajęć z PI.

Pominięte zostały założenia zerowe. Jednym z nich jest to, by uczący się nie uczyli nauczycieli. Innym założeniem zerowym jest przygotowanie odpowiedniej literatury. Oczywiście kolejne odcinki zawierały będą propozycje uzupełniającej literatury, jednak konieczne jest bazowanie na jednym, sprawdzonym zwartym wydaniu, jest nim np. „Komputery osobiste" D. Madeja i innych.

Zgodnie z naszą deklaracją, z wakacyjnego numeru „IKS-a", prezentujemy własną (zweryfikowaną) propozycję programu „Podstaw informatyki". Zakłada się w nim, że minimalna liczba godzin zajęć powinna wynosić 50, przy proporcji 2 do 3, na korzyść ćwiczeń. Znając naszą „rzeczywistość informatyczną" należy założyć, że owe ćwiczenia odbywać się będą w połowie z wykorzystaniem sprzętu (w sali komputerowej) i w połowie „na sucho" bez komputera. Przeplot ten pozwoli, by na przykład sala z 10 komputerami była wystarczająca dla 40 uczących się (dwie grupy po 20 osób).

Przedstawiony poniżej program będzie cyklicznie uzupełniany wspomnianymi odcinkami, które w miarę dokładnie prezentowały będą treści kolejnych zajęć. Zdając sobie jednak sprawę z tego, że nasza propozycja może być dyskusyjna, przy każdym odcinku PI, będziemy zamieszczali opinie państwa, ale tylko te, które poza krytyką zawierały będą konkretne propozycje. Uzupełnieniem każdego odcinka będą także materiały naszych współpracowników — rozszerzające proponowane treści nauczania, w tym także wspomniane obrazy dydaktyczne i programy. Dopełnienie całości tworzyć będzie lista źródeł materiałów do zajęć — głównie wybrane artykuły z krajowej prasy informatycznej.

PODSTAWY INFORMATYKI
(Tematy wykładów wraz z tezami)

1. Wprowadzenie do podstaw informatyki. 2 godz.
   Omówienie niektórych (atrakcyjnych) zastosowań komputerów.
   Proponowany jest przykład przetwarzania obrazów, a w tym: analiza obrazów uzyskiwanych z kosmosu, komputerowe wspomaganie zabiegów konserwatorskich obrazów na przykładzie Mony Lisy.
   Zastosowanie maszyn do identyfikacji autorów tekstów.
   Krótka historia rozwoju technologii budowy komputerów (generacje).
2. Sprzęt informatyczny. 2 godz.
   Reprezentacja informacji w komputerze — system binarny, a cyfry dziesiętne, litery i obrazy (kod ASCII, atrybuty obrazów).
   Budowa mikrokomputera, funkcje, podstawowe parametry jego podzespołów (mikroprocesor, pamięć RAM, ROM).
3. Programowanie. 4 godz.
   Algorytm — jego istota i przeznaczenie.
   Interpretacja terminu „instrukcja".
   Cechy charakterystyczne niektórych języków programowania: Basic, Pascal, Logo, C i Assembler.
   Istota funkcji kompilatora (i interpretera) — „szybkość programu".
   Wybrane instrukcje Basica.

UWAGA Po trzecim temacie, powinien nastąpić początek zajęć praktycznych, zgodnie z odrębnym planem. Celowe jest w tym czasie polecić wykonanie uczącym się (indywidualnie lub w grupach) opracowania nt. „Najciekawsze zastosowania komputerów".

4. Sprzęt informatyczny — urządzenia zewnętrzne. 2 godz.
   Przeznaczenie, zalety i wady urządzeń wejściowych, wyjściowych i pamięci zewnętrznych (magnetycznych). Klawiatura, monitor, drukarka, pióro świetlne, joystick, mysz, OCRy. Przeznaczenie, zalety i wady pamięci typu: dyski elastyczne i sztywne oraz dyski optyczne i streamery.
5. Profesjonalne komputery. 2 godz.
   Zastosowania, zalety i wady. IBM PC jako przykład unifikacji podzespołów, urządzeń zewnętrznych i standaryzacji oprogramowania.
   Informacja o dużych komputerach serii ODRA, RIAD oraz superkomputerach — CRAY, ETA.
6. Oprogramowanie systemowe. 1 godz.
   Podstawowe cechy systemów operacyjnych na przykładzie CPM'a i DOS'a.
7. Oprogramowanie użytkowe. 2 godz.
   Omówienie oprogramowania („życzliwości komputerów") na przykładzie:
   — redagowanie tekstów (WordStar, Chiwriter, Fontasy),
   — bazy danych (dBase III Plus),
   — programy kalkulacyjne (VisiCalc),
   — programy graficzne (Paintbrush),
   — gry komputerowe.
   Proponowane przykłady powinny być zastąpione odpowiednim oprogramowaniem, które jest dostępne w szkolnych komputerach.
8. Oprogramowanie własne, szczególnego przeznaczenia.
   Omówienie możliwości praktycznego wykorzystania wcześniej przygotowanych programów. Np. programy: sortujący, zbierający i przetwarzający dane z dziennika lekcyjnego (pensum nauczycieli, obecności, średnie oceny, itp.).
9. Sieci komputerowe, wielodostęp, teletransmisja informacji. 1 godz.
   Międzykomputerowa łączność — przesyłanie wszelkiej postaci informacji, także obrazów, przykłady. Szybki dostęp (przekazywanie) do superaktualnych informacji. Zasady dostępu do światowych baz danych na przykładzie International Computing Centre.
10. Zastosowanie komputerów w „szkolnej branży", np.: w muzyce, sporcie, pedagogice, wojsku. 2 godz.
    Wykład gościnny profesjonalisty (z „darem dydaktycznym") lub wycieczka do firmy świadczącej informatyczne usługi.
11. Problematyka dialogu człowieka z komputerem. 2 godz.
    Niepowodzenia sztucznej inteligencji. Ekspert-systemy. Porozumiewanie się z komputerami w języku naturalnym — głosem i za pośrednictwem informacji pisanych. Życzliwość komputerów.
12. Prezentacja najciekawszych opracowań nt.: „Najciekawsze zastosowanie komputerów". 2 godz.

PODSTAWY INFORMATYKI
(Tematy ćwiczeń wraz z tezami)

Uwagi wstępne

Założono, że dwugodzinne ćwiczenia odbywać się będą jednocześnie w dwóch grupach — przemiennie: jedna godzina zajęć teoretycznych (T) i w sali komputerów godzina zajęć praktycznych (P).

1. (T) Reprezentacja informacji w komputerze. 2×1 godz.
   System binarny a cyfry dziesiętne, litery i obrazy, kod ASCII.
   (P) Obsługa mikrokomputera na przykładzie dostępnego sprzętu. 2×1 godz.
   Włączenie i wyłączenie zestawu mikrokomputera. Rozmieszczenie klawiszy QWERTY. Klawisze funkcyjne (systemowe) itp.
2. (T) Budowa mikrokomputera. 2×1 godz.
   Wprowadzanie liczb do pamięci komputera, obliczanie sumy/różnicy/ilorazu/iloczynu, wyświetlanie wyniku.
   Mikroprocesor, pamięci półprzewodnikowe, układy współpracy z urządzeniami zewnętrznymi.
   (T) Przykłady algorytmizacji zadań. 2×1 godz.
   Znaczenie konwersacyjnego wprowadzania danych. Kontrola zakresów danych podczas ich wprowadzania oraz wyświetlanie komentarzy.
   (P) Sprawdzenie działania wybranych instrukcji Basica. 4×1 godz.
   PRINT (PRINT „X",a), GOTO, FOR, NEXT, DIM, INPUT (INPUT „X",a) i w dalszej kolejności IF, THEN.
3. (T) Języki programowania. 2×1 godz.
   Podstawowe cechy charakterystyczne języków: Basic, Pascal, LOGO, C, Plan, Cobol. Zalety i wady Assemblera (szybkość, unikalność). Kompilatory i interpretery.
   (T) Przeznaczenie, zalety i wady urządzeń zewnętrznych komputerów różnej skali miniaturyzacji.
   Urządzenia wejściowe i wyjściowe, pamięci magnetyczne.
   Analiza różnic między komputerami: ODRA, RIAD, PC, superkomputery.
   (P) Budowanie własnych, krótkich programów w języku Basic. 6 godz.
   Zwrócenie uwagi na poprawną konstrukcję komentarzy generowanych przez komputer po uruchomieniu własnego programu — teksty wyświetlane przy wykorzystaniu instrukcji INPUT"" i PRINT"".
4. (T) Analiza dostępnego oprogramowania na komputery szkolne. 4×1 godz.
   Nauka korzystania z dokumentacji, określania poprawnych parametrów.
   (P) Nauka i ćwiczenie samodzielnego uruchamiania wybranych programów dostępnych na szkolne komputery. 4×1 godz.



— Pan chce mi wmówić, że myśmy to zaprogramowali?

INFORMATYKA

WPROWADZENIE DO PODSTAW INFORMATYKI

(Materiały do Podstaw Informatyki cz. 1)

Rozpoczynamy, zapowiadane już wcześniej, drukowanie materiałów, które stanowią naszą propozycję uzupełnienia treści nauczania podstaw informatyki (PI). Wydaje się, że w proponowanym zakresie przedmiot ten może być prowadzony w klasie maturalnej szkoły średniej. Zważywszy na to, iż tematyka ta nie ma zbyt dużych tradycji, sądzimy, że może także znaleźć się w programie szkół pomaturalnych.

Pragnę podkreślić, że materiały poświęcone PI, adresowane są głównie do uczących się, stąd „beletrystyczna" forma ich prezentacji.

Zacznijmy od takiego wstępu, który nie powinien zmniejszyć potencjalnego zainteresowania informatyką naszych słuchaczy. Nie straszymy ich zatem procedurami i algorytmami, bajtami i pojemnością pamięci — przynajmniej na początku. Celowe wydaje się natomiast zwrócenie uwagi na to, że już dzisiaj praktycznie nie ma sfery życia, której nie może wspomagać komputer. „Jutro komputer będzie wszędzie tam, gdzie dzisiaj jest długopis", wiedza informatyczna stanie się niezbędna podobnie jak umiejętność pisania.

Zawsze frapowało mnie, jak jest faktycznie z XIII księgą „Pana Tadeusza". Mickiewicz, Fredro, a może jeszcze ktoś inny jest jej autorem? Problem ten, dla laika w tej dziedzinie, wydawał się nie do pokonania. Okazuje się jednak, że rozwiązywanie tego typu zagadek ma już wieloletnią tradycję i to nie bez sukcesów. Wystarczy bowiem policzyć częstotliwość używania przez autora określonej długości słów i otrzymujemy krzywą (oś X tworzą długości słów, Y — częstotliwość ich występowania) identyfikującą jego „styl". Można by dyskutować czy jest to „styl" czy nie — fakt pozostaje bezsporny, że krzywe te bezbłędnie rozróżniają dzieła napisane np.: Szekspira i Bacona. Niestety okazało się, że identyfikator ten jest kiepski (podobny) dla dzieł Marlowa i twórcy Makbeta. Szukano zatem dalej innych identyfikatorów jak np.: częstotliwość występowania określonych słów, pojedynczych liter. Wszystkie te metody, po setkach godzin mozolnej pracy okazywały się **prawie** dobre. Wydawało się, że sprawa jest przegrana. Do chwili, gdy z pomocą przyszedł komputer. Czasochłonne okazało się tylko wprowadzanie tekstu do komputera — dalej rachowała już samodzielnie maszyna. Dzięki jej pomocy zauważono, że częstotliwość używania nie pojedynczych liter, a ich par lub nawet triad jest niemal jednoznacznym identyfikatorem „stylu" autora. „Ręczne" liczenie częstotliwości używania par i triad liter jest praktycznie niewykonalne. Komputer natomiast jest bardzo szybki i z łatwością policzył wspomniane krzywe częstotliwości par liter bezbłędnie identyfikując, że autorem „Mrowiska" jest ...?

Maszyny cyfrowe są szybkie i często wykonują te zadania, które dotychczas były odkładane na półki — wymagały bowiem zbyt dużo czasu — trwającej dziesiątki lat „pracy ręcznej". Komputery zmieniły ten stan rzeczy — przedsięwzięcia niewykonalne, z powodzeniem przekazano do realizacji.

Prawdopodobnie jednym z najsłynniejszych obrazów jest Mona Lisa (La Gioconda) Leonarda da Vinci. Obecnie portret ten pozostał ledwo cieniem swej oryginalnej postaci. Jest to efekt działania czasu i niestety zabiegów wielu pokoleń konserwatorów dzieł sztuki.

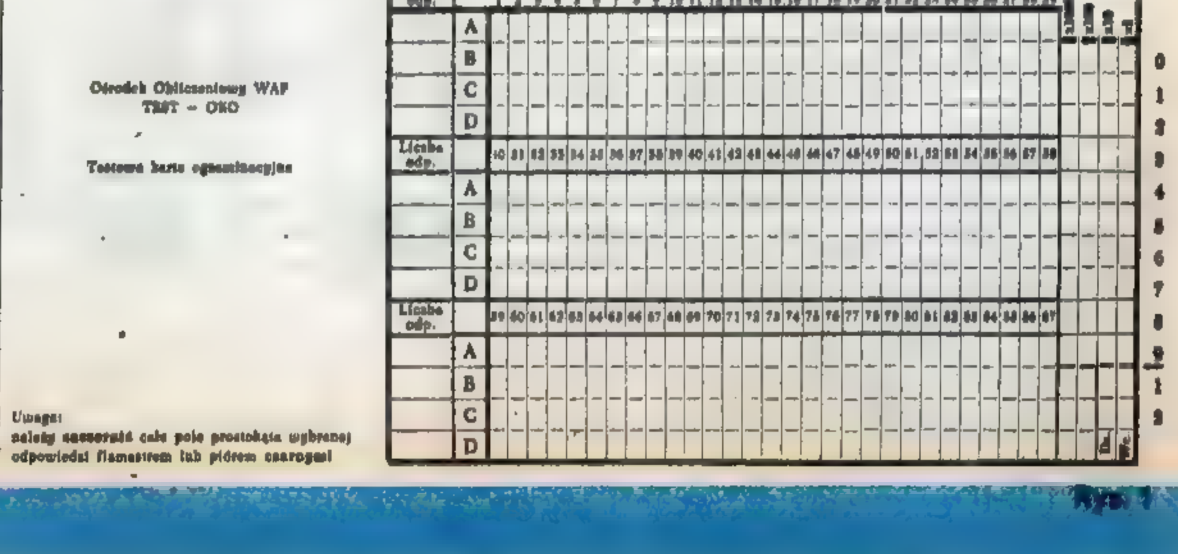
Nadal jednak zabiegi konserwatorskie trzeba wykonywać — powietrze bowiem nie staje się wcale mniej aktywne chemicznie („żrące"), a czas płynie nieubłaganie. Pozostaje zatem wiara w uczciwość (artyzm?) konserwatorów, dobrą jakość farb i innych chemikaliów mających bezpośredni kontakt z płótnem.

Przystępując do odtwarzania obrazu — zabiegów konserwatorskich — powstaje pytanie podstawowe: jak wyglądał oryginał, chwilę po tym, gdy Leonardo da Vinci odłożył pędzel. Zważywszy na to, że fotografii wówczas nie było, pozostaje intuicja, wyobraźnia — jednym słowem zabawa w głuchy telefon. Usiłując zmienić ten stan rzeczy grupa amerykańskich entuzjastów postanowiła wykorzystać do tego celu komputer.

Okazało się to stosunkowo proste. Przyjęto, że portret La Giocondy składa się z 20 milionów punktów (każdy z nich, to fragment obrazu o wielkości 0,02 mm). Czarny punkt (piksel) oznaczono jedynką, biały natomiast zerem — stąd tylko krok do komputera, a ściślej do jego pamięci. Wiadomo bowiem, że komputer, co jak co, ale zera i jedynki doskonale zapamiętuje. Pozostało zatem jedynki i zera odzwierciedlające obraz przenieść do komputera, by maszyna miała w swej pamięci nasz portret.

Można wyrazić wątpliwość co do wierności zapamiętanego w ten sposób obrazu. Liczba pikseli nie budzi wątpliwości — jest wystarczająca, by wzrok nasz nie dostrzegł „ziarnistości obrazu", ale zagubiona została w ten sposób subtelność barw, cieni. Pomyślano i o tym, bowiem każdy piksel portretu naszej niewiasty zapisywany był nie jedynką lub zerem, ale liczbą od zera (czerń) do 255 (biel) dla każdego z trzech kolorów standardu RGB. Wielkości te dowodzą ogromnej precyzji „cyfrowej reprezentacji" obrazu w pamięci komputera. Odzwierciedla ona wszystkie dostrzegalne okiem subtelności dzieła. Spełniony został w ten sposób podstawowy warunek, wykorzystania potencjału obliczeniowego maszyn do przetwarzania obrazu. Aby zatem rozjaśnić obraz, wystarczy by komputer dodał np. 10 do każdej z wartości opisujących treść wyróżnialnych pikseli obrazu. Podobnie, zmniejszenie jasności obrazu lub jego fragmentów jest bardzo proste — wystarczy odejmować odpowiednie wartości.

W profesjonalnej praktyce wykorzystywane są bardziej wyrafinowane metody przekształcania obrazu, pozwoliły one uzyskać na ekranie monitora komputerowego obraz Mony Lisy w formie niemal identycznej z oglądaną pięć wieków temu. Ukazał się, twórcom tej metody, portret, na którym kolor nieba z brą-



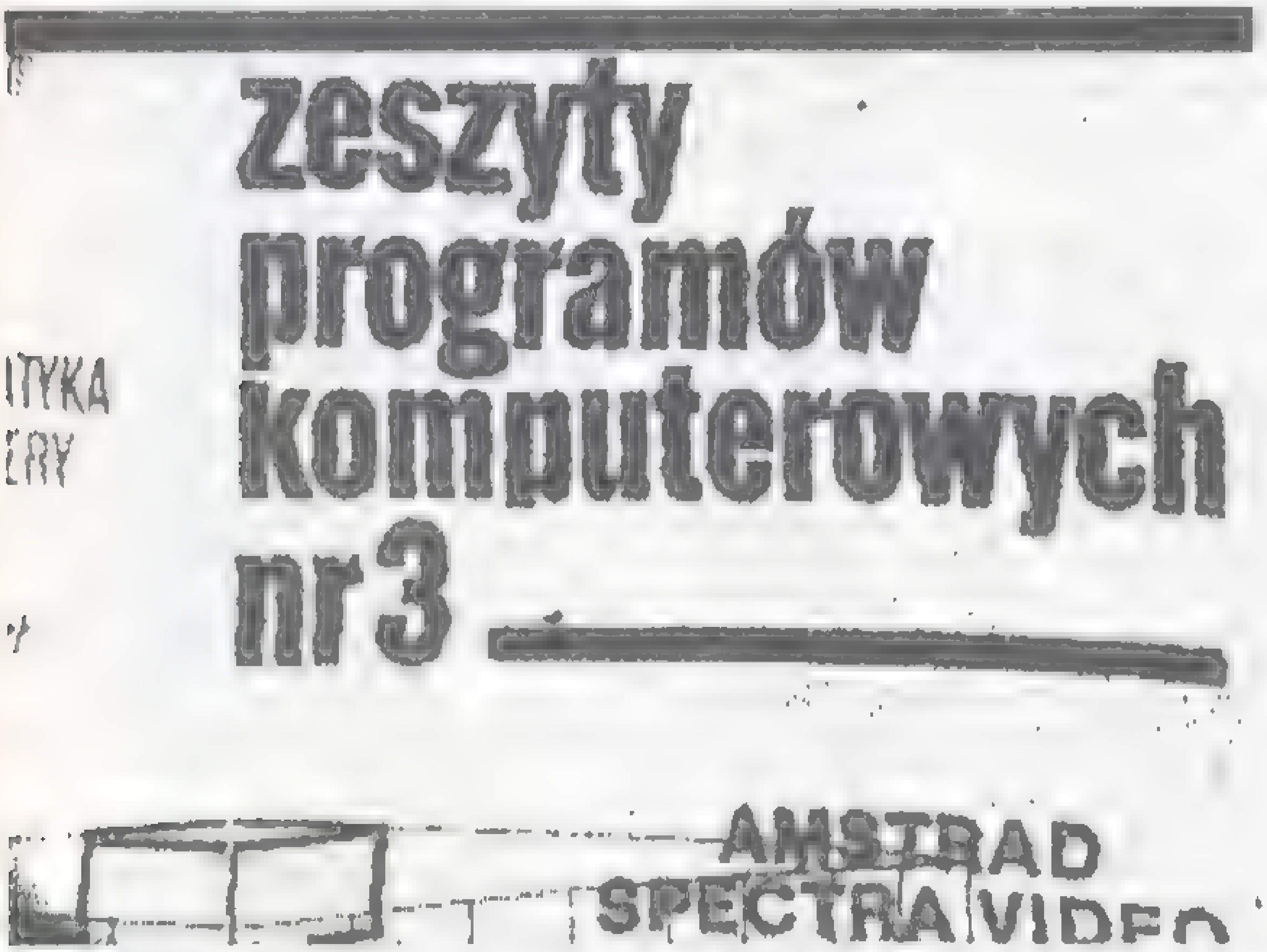
Rys. 1

a) Fotografia strony tytułowej zeszytu programów komputerowych nr 3
b) Fotografia wydruku z komputera fragmentu strony



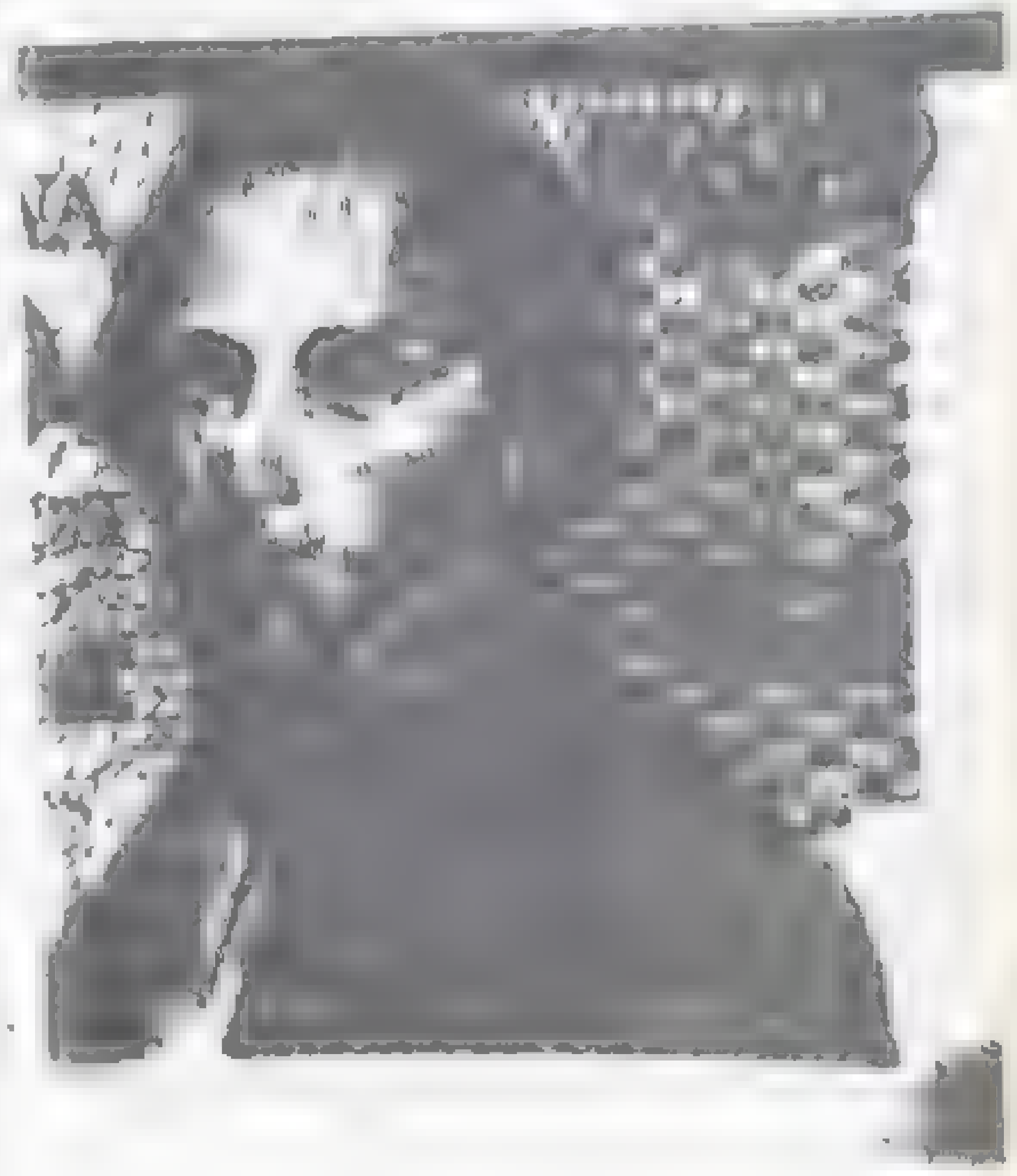
Rys. nr 2 Oryginał (a)

i „komputerowa kopia" (b) fragmentu strony jednego z naszych zeszytów.

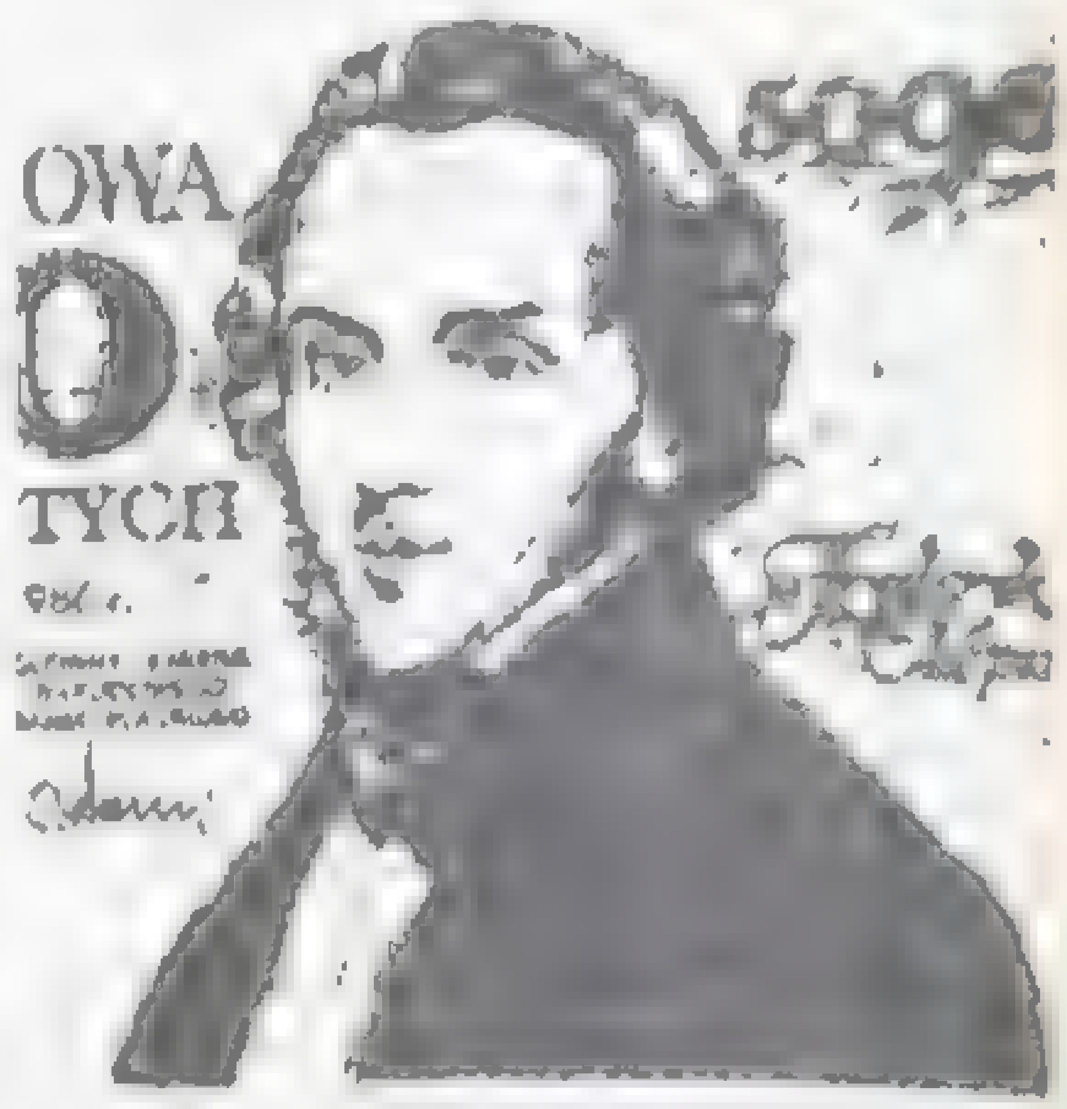


a) Fotografia „Chopina" zaciemnionego
b) Fotografia „Chopina" rozjaśnionego

Rys. nr 3 Komputerowe odwzorowanie fragmentu banknotu,



a) po dodaniu wartości 4 do każdego piksela oryginału,



b) po odjęciu wartości 4 od każdego piksela oryginału.

zowego stał się niebieski, skóry dziewczyny z żółtej na alabastrowy, a z sukni bohaterki wydobyto odcień pięknej zieleni.

Oczywiście proces wprowadzania informacji o treści każdego piksela obrazu do pamięci komputera odbywa się automatycznie. Służą temu tak zwane skanery. Urządzenia te wyposażone są w kamery TV. Kamera zamienia światło odbite np. od portretu lub od testowej karty egzaminacyjnej (zob. rys. nr 1) na sygnał elektryczny (trwa to 1/50 sek.), ten z kolei zostaje zmieniony na liczby, które po chwili „układane" są po kolei w pamięci komputera.

Nie przypadkiem wspomniałem tutaj o karcie egzaminacyjnej, mając bowiem jej obraz w pamięci maszyny, nie człowiek, a komputer (jego na „kawę" nie zaprosisz) może sprawdzić jej treść. Gdy treść stanowią zakreślone kratki (egzamin testowy), komputer natychmiast rozpozna czy jedynki (zakreślone pole karty testowej) są w odpowiednim miejscu analizowanego dokumentu.

Urządzenie takie zostało zbudowane i z powodzeniem wykorzystane cztery lata temu w WAP. Odczyt i analiza 87 odpowiedzi trwa około 2 sek. Należy w tym miejscu podkreślić, że współczesny sprzęt informatyczny raczej się „nie męczy", pozwoliło to m.in. wczytać 10 000 kart bez przerwy.

INFORMATYKA W SZKOLE

Zasygnalizowane możliwości komputerów stały się realne dzięki bardzo szybkiemu rozwojowi, najpierw sprzętu, krótko potem oprogramowania.

Początek informatyki (umowny), to maszyny elektromechaniczne. Tworzą one tak zwaną **zerową generację** komputerów. Budowane były głównie z przekaźników elektromagnetycznych. Płynący przez ich uzwojenie prąd powodował, iż „kowadełko" było przyciągane — **jedynka**. Brak prądu — kowadełko odciągała sprężynka — **zero**. (Podobnie jak w dzwonku nad drzwiami — 0,1,0,1,0.... dzwoni). Szybkość, gabaryty, pobór prądu komputerów tej klasy była gorsza niż zła. Niezadowolenie z tego stanu rzeczy spowodowało szybki postęp. Pojawiły się komputery **pierwszej generacji** — lampowe. W większości przekaźniki zastąpiono układami lampowymi. Miniaturyzacji nie osiągnięto. Na przykład reprezentant tej generacji ENIAC ważył 30 ton, ale szybkość działania komputera zwiększono znacznie. W czasie jednej sekundy wykonywał około 5000 operacji dodawania. Jednak moc pobierał ogromną (150 KW), co w połączeniu z ograniczoną żywotnością lamp stanowiło o tym, iż lepiej tych komputerów (gdy pracowały dobrze) nie wyłączać. Ponowne bowiem włączenie niosło z sobą ryzyko (niemal pewność), że parę lamp w tym momencie dokona swego żywota.

Rewolucyjną zmianę w technice cyfrowej spowodowały dopiero półprzewodniki — osławiony (dziś już czterdziestoletni) tranzystor, dioda i inne podobne elementy. Zbudowane z nich komputery — **druga generacja** pracują zapewne do dzisiaj — ODRA 1304. Urządzenia te wykonywały niemal 30 000 prostych operacji arytmetycznych w czasie jednej sekundy. Wielkość ta w połączeniu ze stosunkowo małymi gabarytami oraz dużą niezawodnością pozwoliła by sprzęt tej generacji wysłano w kosmos. To zastosowanie maszyn cyfrowych przyczyniło się do dalszej **miniaturyzacji sprzętu**, tym samym zmniejszenia **poboru mocy, zwiększenia niezawodności** przy jednoczesnym zwiększaniu **szybkości działania**. Są to (były) w zasadzie cztery podstawowe kryteria pozwalające w miarę jednoznacznie klasyfikować komputer do określonej generacji.

Kolejny etap doskonalenia sprzętu informatycznego — **trzecia generacja** — komputery budowane z układów scalonych. Motorem tego postępu był (i jest) nadal kosmos. Sputniki wymagały bowiem coraz doskonalszych komputerów o małym ciężarze, lepszej niezawodności, a źródłem prądu pozostały nadal mało wydajne baterie słoneczne.

Spowodowało to, iż „upakowano" wiele elementów w jeden układ scalony, tak zwaną „kość". W zależności od stopnia upakowania („integracji") mieści w sobie ona różną liczbę elementów (zazwyczaj są to: tranzystory, diody, oporniki, kondensatory). Układ małej skali integracji (SSI) zawiera do 100 elementów, średniej (MSI) do 1000 elementów, dużej skali integracji (LSI) ma aż do 100 tysięcy elementów, a nie jest to ostatnie słowo „upakowywania").

Przykładem komputerów III generacji są nasze ODRY, 1305 i 1325. Te nasze „staruszki" stanowią jeszcze dzisiaj (niestety) podstawowy potencjał obliczeniowy zainstalowany w naszym kraju. Sądzę, że stwierdzenie to wywoła u części czytelników oburzenie — przecież mikrokomputery. Owszem są i to w dużej ilości (biorąc na wagę), ale to już odrębny problem...

Faktem jest natomiast, że mikrokomputery reprezentują już **IV generację**. Wykorzystano w nich dwa charakterystyczne dla tej generacji typy układów, które poprawiają wszystkie parametry sprzętu informatycznego, są to: **mikroprocesor i pamięć półprzewodnikowa**.

Mikroprocesor, to kolejny krok integracji, tym razem nie elementów, a całych układów. Jest to już bardzo duża skala integracji (VLSI). Współczesne mikroprocesory (jeden układ) zbudowane są już z milionów elementów, a szybkość wykonywanych przez nie mikrooperacji mierzona jest w milionach. Ta sama technologia pozwala nam budować pamięci komputerów z układów scalonych (jeszcze niedawno potrzebne były do tego rdzenie ferrytowe), które mają pojemności sięgające setek tysięcy znaków w jednej obudowie.

Czwarta generacja stworzyła warunki do budowy mikrokomputerów przeznaczonych do domu, wysyłanych na Marsa, a także stawianych na biurkach. Budowane są w tym samym czasie superkomputery, nazywane czasami hiperkomputerami (jest ich na świecie mniej niż 300), które dzięki specjalnej metodzie przetwarzania informacji (procesory macierzowe) wykonują np.: dodawanie w czasie, gdy światło przebiega w latarce drogę od żarówki do chroniącego ją szkiełka (3 cm!).

Wydawać by się mogło, że osiągnięto szczyt technologiczny doskonalenia komputerów (bariera szybkości światła), prowadzone są jednak nadal badania, słyszymy o komputerach optycznych, pracujących w warunkach nadprzewodnictwa, hybrydach łączących urządzenia z żywymi tkankami.

Sądzę jednak, że najbardziej obiecującym kierunkiem doskonalenia komputerów jest „zrównoleglanie" operacji wykonywanych przez maszynę — na wzór najdoskonalszego modelu, jakim jest mózg ludzki.

Wracając do naszych generacji — kolejną jest **piąta**. W tym względzie zdania uczonych są podzielone. Mało kto kwestionuje fakt, że nie jest to kolejny etap doskonalenia sprzętowych parametrów komputera, a doskonalenie oprogramowania. Wydaje się, że piąta generacja komputerów to **„życzliwość maszyn"**. Zdaję sobie sprawę, że jest to dosyć ryzykowne określenie, kryje jednak ono w sobie ideę sztucznej inteligencji, ekspert-systemów i analizy języka naturalnego (mówionego i pisanego). Wszystko to zmierza do zaspokojenia naszych oczekiwań już w momencie kupienia komputera — bez względu na planowaną sferę jego zastosowań.

Komputer piątej generacji przede wszystkim z nieograniczoną cierpliwością, także przystępnie, nauczy nas korzystać z jego możliwości, są one ogromne. Dziennikarzowi pomagają w redagowaniu tekstów, (ortografia, synonimy, tłumaczenia itp.). Maszyna staje się wysokiej klasy partnerem w fachowej dyskusji na wybrane, aktualne tematy. O aktualizację wiedzy naszego komputera dbają inni, nasza maszyna korzysta bowiem (via sieć telefoniczna) z innych baz danych, rozsianych po całym świecie, ale to już odrębny temat naszych podstaw informatyki.

Włodzimierz GOGOŁEK

Tezy do pierwszych zajęć z podstaw informatyki

Temat: WPROWADZENIE DO PODSTAW INFORMATYKI

1. **Omówienie przykładów ilustrujących potrzeby (i możliwości) praktycznych zastosowań komputerów**
   - znaczenie szybkości przetwarzania informacji i pojemności pamięci komputerów;
   - możliwości wykorzystywania komputerów w różnych dziedzinach życia,
   - cyfrowa reprezentacja obrazu — wstęp do systemu dwójkowego, pojęcie „pojemność pamięci".
2. **Podstawowe etapy rozwoju informatyki**
   Generacje komputerów. System binarny a elementy pamiętające (nie logiczne) np. w zerowej generacji. Pojęcie „szybkość" komputera. Podstawowe parametry charakteryzujące generacje (miniaturyzacja, szybkość, moc pobierana, niezawodność).

Propozycja materiałów pomocniczych:

1. Na początku był ENIAC, „IKS" nr 6/86
2. Krótka historia mikroprocesorów, „IKS" nr 3/86
3. Komputer i stylistyczne linie papilarne, „IKS", nr 4/87.
4. Komputer u kompozytora, „Komputer", nr 2, 3, 4/87
5. Timeworks Publisher, „Komputer", 5/88
6. Kosmos, medycyna i komputer, „IKS" nr 12/87
7. Mona Lisa i komputer, „IKS" nr 11/87.

Włodzimierz GOGOŁEK

UWAGA.
Ilustracje „obrazów komputerowych" mogliśmy wykonać dzięki uprzejmości Dyrektora Zakładu Projektowo-Wdrożeniowego ATM..N, który udostępnił nam, zbudowany tam skaner sygnału telewizyjnego. Serdecznie dziękujemy.

Nuty z komputera (4)

Krzysztof POŹNIAK, Jarosław ZIEMBICKI

Zbliżamy się do końca pracy nad edytorem sekwencera. Nie oznacza to niestety zakończenia naszego programu. Pozostaje praktycznie nie omówiony do tej pory problem współpracy z instrumentami muzycznymi. O tym w dalszej części naszego artykułu.

Zamieszczony obecnie program wypełnia ostatnią „dziurę" w oprogramowaniu edytora sekwencera. Jest to edytor utworu. Dzięki niemu z utworzonych do tej pory taktów będziemy mogli budować całe kompozycje, korygować ich struktury itp. Od strony programowej utwór jest ciągiem taktów wykonywanych w ściśle określonej kolejności. Przewidzieliśmy stosowanie pętli (podobnych do pętli FOR—NEXT w Basicu) pozwalających objąć „klamrą" powtarzające się fragmenty utworu.

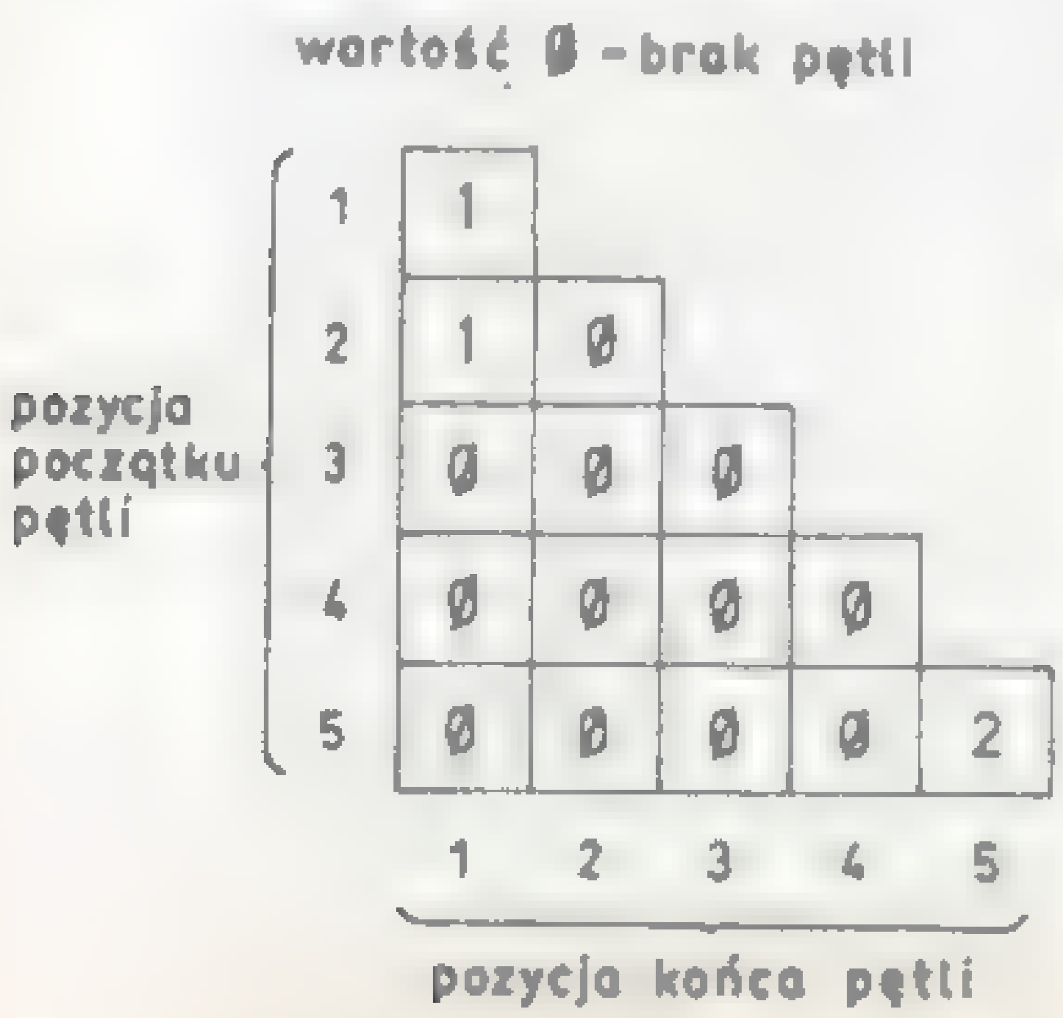
Przypominamy: *zamieszczony wydruk należy dołączyć do opublikowanych poprzednio programów. Istniejące już linie o tych samych numerach należy zamienić na obecnie prezentowane.*

EDYTOR GŁÓWNY — nowe funkcje

Klawisz **W** — wywołanie utworu zapisanego w pamięci (zmiennej tekstowej u$)
parametry: 1) numer utworu.
Klawisz **U** — utworzenie utworu — wywołanie edytora utworu
parametry: 1) nazwa utworu (zasady takie same, jak przy tworzeniu nazwy taktu)
Klawisz **K** — kasowanie utworu zapisanego w pamięci (zmiennej tekstowej u$)
parametry: 1) numer utworu
Klawisz **L** — lista utworów
parametry: brak

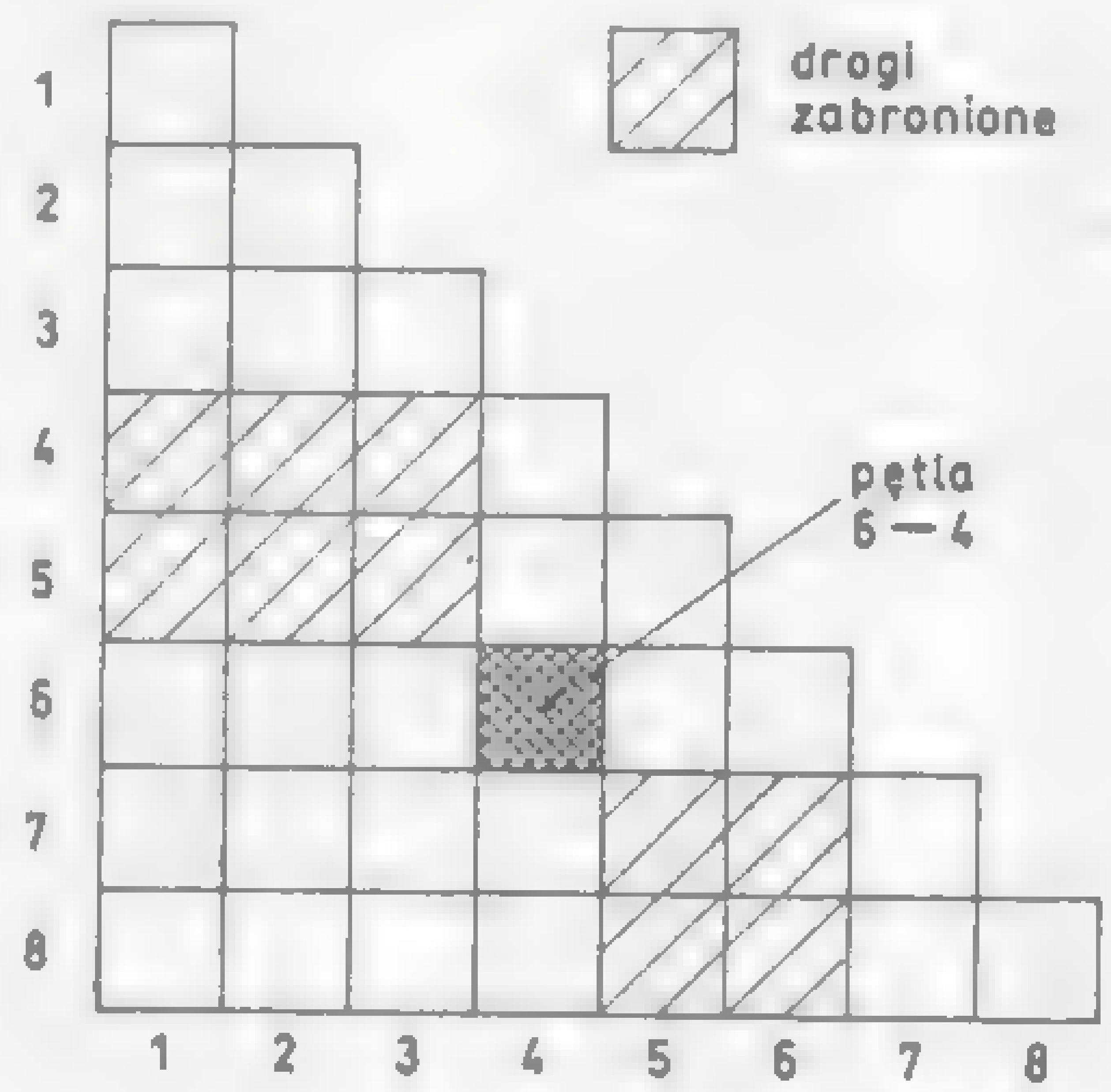
EDYTOR UTWORU

Klawisz **d** — dołączenie taktu do utworu
parametry: 1) numer pozycji, na którą będzie wprowadzony takt, 2) numer taktu.



Rys. 1

uwaga: *wprowadzenie taktu na istniejącą już pozycję powoduje przesunięcie taktów umieszczonych poniżej o jedną pozycję w dół. Podanie pozycji o jedną większą od już istniejących pozwala na umieszczenie nowego taktu na końcu utworu*



Rys. 2

Klawisz **l** — wywołanie listy taktów
parametry: brak
Klawisz **w** — wywołanie taktu z pamięci — wejście do edytora taktu
parametry: 1) numer taktu
Klawisz **k** — kasowanie taktu w utworze
parametry: 1) numer pozycji w utworze, na której znajduje się dany takt

uwaga: *następuje kasowanie wszystkich pętli, które bezpośrednio zaczynały się lub kończyły na danej pozycji*

Klawisz **p** — utworzenie pętli powtórzeń
parametry: 1) numer pozycji, od której rozpocznie się pętla powtórzenia,
2) numer pozycji, przed którą pętla się zakończy,
3) ilość powtórzeń taktów w pętli (maksym. do 99).

uwaga: *pętle nie mogą się wzajemnie przecinać, mogą się natomiast zagłębiać w sobie. Prawidłowo poprowadzone pętle prezentuje rysunek nr 3*

Klawisz **k** — kasowanie pętli powtórzeń
parametry: 1) numer pozycji początku pętli,
2) numer pozycji końca pętli
Klawisz **S** — wyświetlanie aktualnej struktury utworu
parametry: brak
Klawisz **z** — zapis utworu do pamięci (zmiennej tekstowej u$)
parametry: brak

uwaga — *stara wersja danego utworu zostaje skasowana*

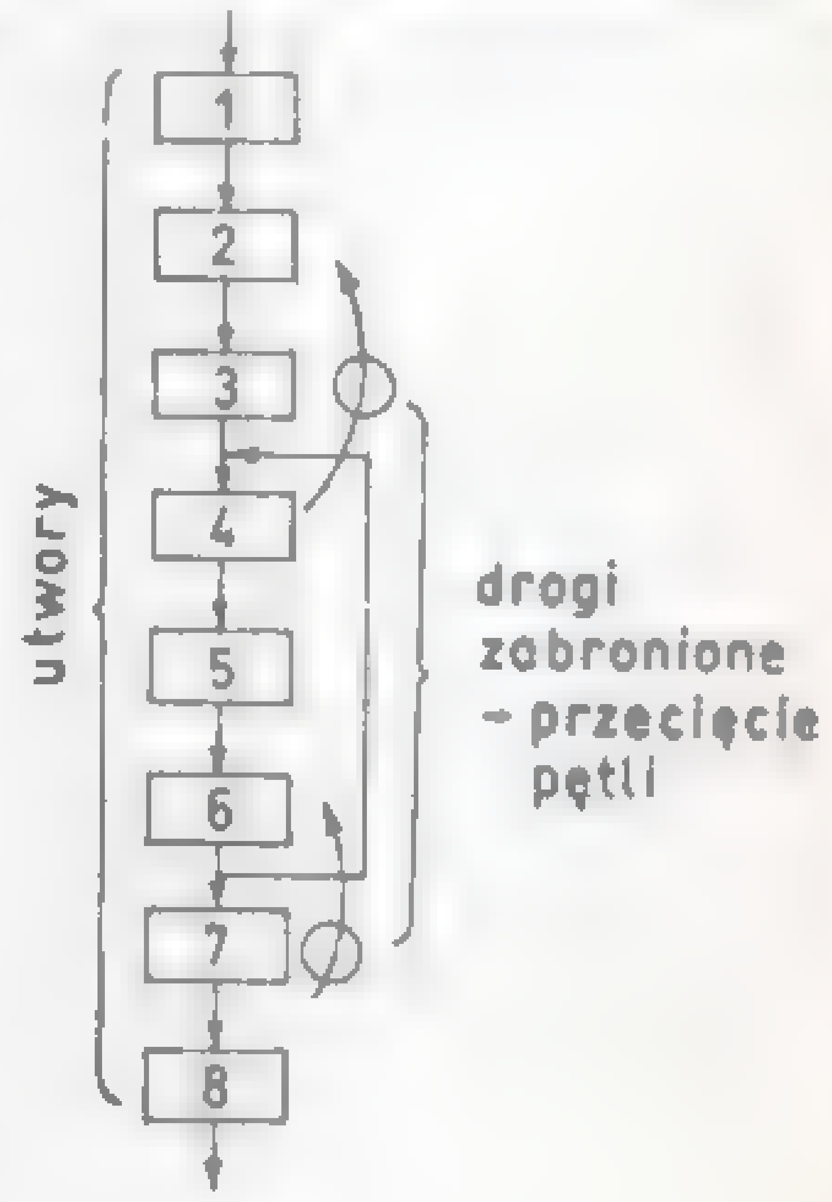
Klawisz **e** — powrót do edytora głównego

Na rysunku nr 3 przedstawiliśmy przykładowy utwór utworzony za pomocą edytora utworu

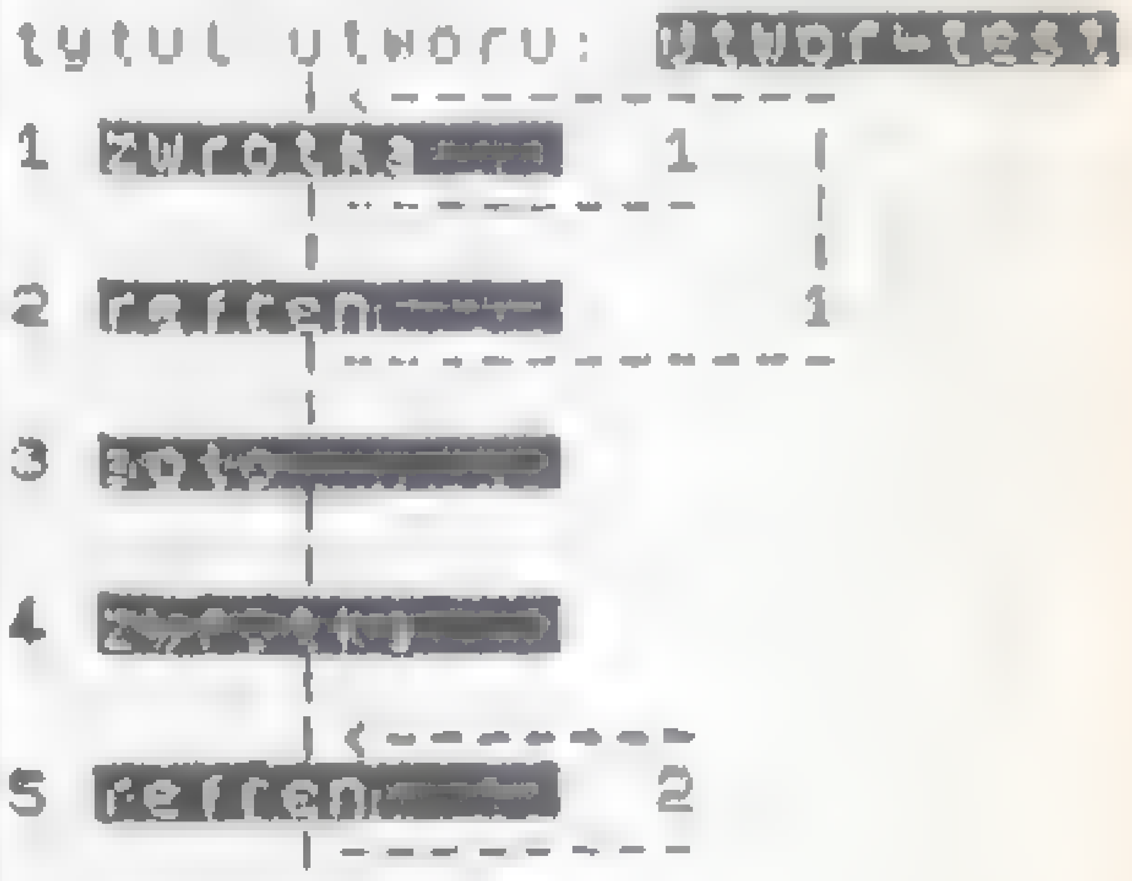
Dla dociekawskich

Utwór aktualnie tworzony za pomocą edytora utworu jest trzymany w zmiennej tekstowej r. Strukturę utworu wygodniej jest przedstawić za pomocą „trójkąta pętli". Na rysunku nr 1 przedstawiliśmy strukturę utworu z rysunku nr 3. Proponujemy, aby czytelnicy sami znaleźli powiązania obu rysunków.

Dzięki takiej formie zapisu bardzo wygodnie kontroluje się warunki nieprzecinania się pętli. Każda nowo wprowadzana pętla



podlega w programie tego typu kontroli. Rysunek nr 2 przedstawia sytuację wprowadzania pętli od pozycji nr 6 do pozycji nr 4 i warunki jej umieszczenia. Zakreskowane pola muszą posiadać wartość 0



EDYTOR UTWORU

Pora zastanowić się, w jaki sposób połączymy komputer z elektronicznym instrumentem muzycznym i co za jego pomocą będziemy sterować. Zanim weźmiemy się do solidnej pracy konstruktorskiej warto poznać trochę historię i budowę syntezatorów.

Krótka historia

Historia elektronicznych urządzeń muzycznych zaczęła się na przełomie XIX i XX wieku. Kiedy to zbudowano pierwsze organy elektromechaniczne. W pewnym sensie były one powtórzeniem koncepcji organów piszczałkowych, ponieważ każdy klawisz miał przydzielony własny generator drgań elek-

trycznych o odpowiedniej częstotliwości. Koncepcja ta z niewielkimi zmianami, dominowała w konstrukcjach instrumentów przez kilkadziesiąt lat. Konsekwencją stosowania dużej liczby generatorów o dosyć prostej konstrukcji było uzyskiwanie mało zróżnicowanych brzmień.

Pierwsze próby rozbudowy syntezatorów umożliwiające kształtowanie dźwięku z dużą dowolnością zaczęły się już w latach pięćdziesiątych, ale dopiero konstrukcje **dr. Roberta Mooga**, bazujące na elementach półprzewodnikowych, spowodowały rewolucję w elektronice muzycznej. Syntezator Mooga składał się z modułów, z których każdy uczestniczył w kształtowaniu innego parametru dźwięku. Ponieważ koncepcja ta jest stosowana do dzisiaj, omówimy ją nieco szerzej.

Zasadniczym blokiem jest **generator** drgań akustycznych. Określa on **wysokość** dźwięku. Sygnał z generatora jest poddawany **filtrowaniu** (precyzyjnej regulacji barwy): pewne częstotliwości składowe danego dźwięku mogą zostać stłumione lub wzmocnione, końcowym blokiem jest **wzmacniacz**, który wpływa na głośność dźwięku. Wymienione moduły mogą być sterowane zewnętrznie (np. za pomocą napięcia) przez tzw. **moduły sterujące**. Typowym modułem sterującym jest klawiatura na skutek naciśnięcia klawisza wysyła ona sygnały sterujące powodując generację odpowiedniego dźwięku. Innym źródłem sterującym jest generator drgań wolnozmiennych (LFO), który umożliwia wibrację dźwięku (uzyskujemy efekty typu: vibrato, tremolo). Generator obwiedni kontroluje amplitudę dźwięku w czasie: ustala szybkość narastania, trwania i wybrzmiewania dźwięku. Uniwersalnym modułem jest generator szumu. Może on być źródłem dźwięku, może także tym dźwiękiem sterować wprowadzając przypadkową modulację amplitudy lub częstotliwości. Jest bardzo przydatny przy imitacji naturalnych źródeł dźwięku.

Omówiony układ stanowi tor syntezy pojedynczego dźwięku, stąd pierwsze (najprostsze) syntezatory były jednogłosowe. Okazało się, że podłączenie do każdego klawisza po jednym torze syntezy jest bardzo trudne technicznie i nieekonomiczne. Przecież tak naprawdę podczas gry uruchamiamy nie więcej jak kilka (w wyjątkowych wypadkach kilkanaście) klawiszy naraz. Rozsądnym wyjściem jest zastosowanie sterownika, który na podstawie stanu klawiatury uruchamiałby odpowiednią ilość torów syntezy. Na skonstruowanie takiego sterownika trzeba było poczekać do lat siedemdziesiątych. Wtedy pojawił się pierwszy wielogłosowy syntezator — **Polymoog** — zawierający wspomniany sterownik klawiatury. Sterownikiem był zazwyczaj mikroprocesor. Przeglądał on stan klawiatury i za pomocą przetworników cyfrowo/analogowych sterował analogowymi modułami dźwiękowymi. Wkrótce pojawiły się syntezatory całkowicie cyfrowe o bardzo różnorodnych koncepcjach syntezy dźwięku.

Odrębną grupę elektronicznych instrumentów muzycznych stanowią urządzenia perkusyjne. Początkowo były realizowane techniką analogową, obecnie prymat wiodą urządzenia wyłącznie cyfrowe. Najczęściej generują one dźwięk w ich pamięci.

Ubogą namiastką wymienionych urządzeń mogą być generatory dźwięku zainstalowane w naszych komputerach.

Znamy już urządzenia, którymi będziemy sterować. Warto zastanowić się nad kolejnym problemem.

Jak będziemy sterować instrumentami elektronicznymi
(koncepcja kanałów)

Koncepcja kanałów jest bardzo wygodną i wypróbowaną metodą współpracy z urządzeniami zewnętrznymi. Zastanówmy się w tym momencie nad prostym przykładem. posiadamy w pamięci zbiór tekstowy (np. program). Możemy wyświetlić go na ekranie, wydrukować na drukarce, zapisać na dysku, nagrać na magnetofonie, wręcz przesłać do innego komputera. Ponieważ każdemu z tych urządzeń przypisany jest określony numer kanału, stąd o tym, gdzie treść zbioru zostanie przekazana decyduje podany przez użytkownika numer kanału. Na tej podstawie zostaje uruchomiona odpowiednia procedura transmitująca. Uniwersalność tej metody jest oczywista:

po pierwsze — możemy dowolnie przypisywać urządzenia do kanałów w zależności od ich ilości

po drugie — możemy współpracować z dowolnymi urządzeniami; wystarczy dopisać procedurę transmisyjną i przypisać ją odpowiedniemu kanałowi.

Dla sekwencera urządzeniami zewnętrznymi będą instrumenty elektroniczne. Każdy z nich wymaga odpowiedniego standardu transmisji i formatu przesyłanych danych. Ten problem rozwiążą procedury transmitujące. Użytkownik będzie je dobierał w zależności od typów wykorzystywanych instrumentów. Dobór odpowiedniego kanału nastąpi automatycznie, ponieważ każda ścieżka w takcie jest przyporządkowana konkretnemu instrumentowi.

Sposoby przesyłania danych

W programie wykorzystamy trzy najbardziej rozpowszechnione sposoby transmisji:

a) *transmisja wewnętrzna* — używana przy przesyłaniu danych w obrębie jednego obszaru pamięci. Wykorzystamy ją sterując generatorami w komputerze

b) *transmisja równoległa* — pozwala na jednoczesne wysłanie całego bajtu. Często jest wykorzystywana do sterowania instrumentami perkusji elektronicznych.

c) *transmisja szeregowa* — bit po bicie. Standard wykorzystywany przez system MIDI.

System MIDI

MIDI to skrót angielskiej nazwy: Musical Instrument Digital Interface (cyfrowy interfejs instrumentów muzycznych). Oznacza ogólnie przyjęty w świecie standard przesyłania informacji między urządzeniami muzycznymi. Pozwala na łączenie instrumentów w większe zestawy, umożliwia współpracę z komputerem. Łączenie instrumentów w jednolitą całość zwiększa możliwości brzmieniowe, pozwala na grę wielogłosową. W epoce syntezatorów analogowych eksperymenty tego typu były bardzo kłopotliwe w realizacji i zmuszały do prowadzenia licznych przewodów sterujących między instrumentami.

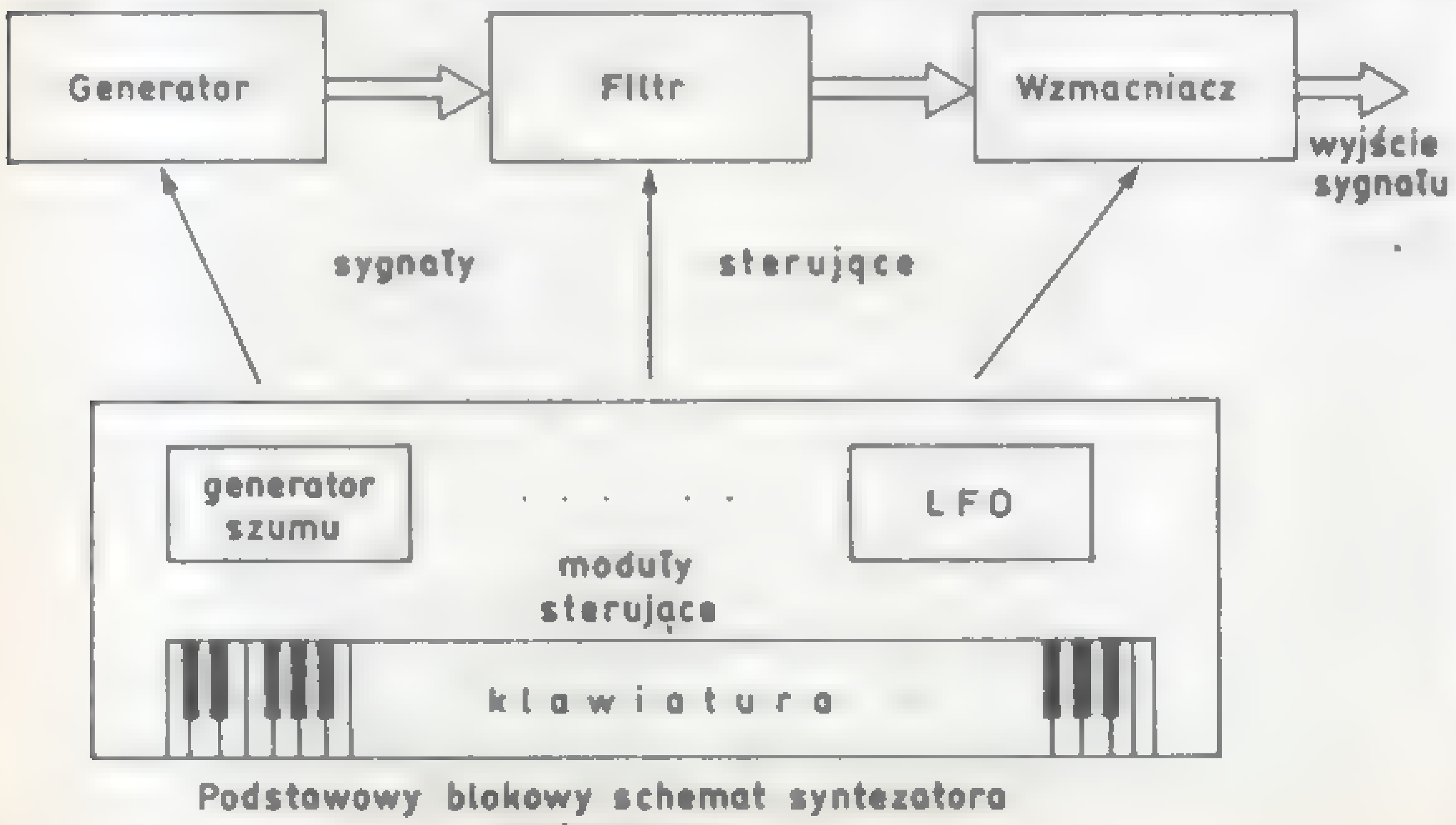
Standard MIDI zaproponowała firma *Sequential Circuits* i jest on dzisiaj jedynym tego typu standardem liczącym się na świecie.

Pojedyncze złącze MIDI składa się z nadajnika, linii łączącej (dwa przewody) i odbiornika. Stanem biernym linii jest brak przepływu prądu — oznacza to logiczną jedynkę. Logiczne zero jest prezentowane przez płynący w linii prąd (standardowo 5 mA).

Najmniejszą jednostką informacji jest bajt. Transmisja bajtu polega na wysłaniu kolejno dziesięciu bitów: bitu startu równego 0, ośmiu bitów przesyłanego bajtu (od najmłodszego do najstarszego) oraz bitu stopu równego 1.

Przestawienie każdego bitu trwa 32 mikrosekundy bitów. Bit stopu wprowadza linię MIDI w stan bajtu. Taki sposób przesyłania danych nosi nazwę *transmisji szeregowej asynchronicznej*.

Mimo że wiele komputerów posiada złącza transmisji szeregowej (np. do współpracy z drukarkami), najczęściej ich parametry czasowe i elektryczne nie odpowiadają standardowi MIDI. W najbliższym czasie opublikujemy schemat prostego nadajnika MIDI, który będzie można podłączyć do



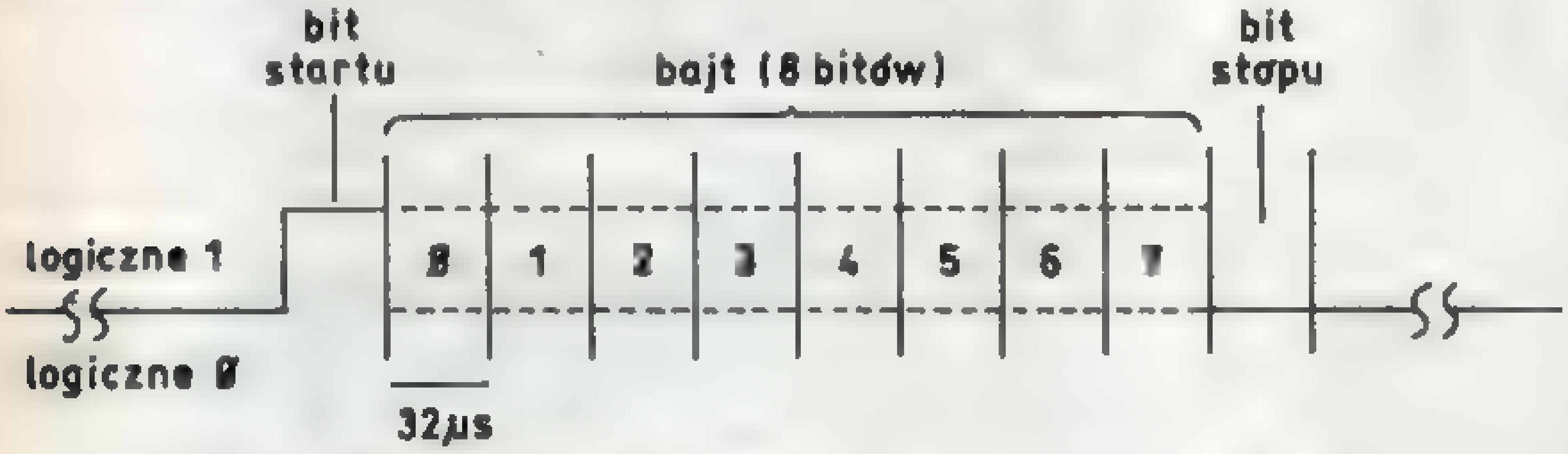
Podstawowy blokowy schemat syntezatora analogowego

Rys. 4

komputera niezależnie od pozostałych jego interfejsów.

Podstawowe konfiguracje urządzeń w systemie MIDI

Do łączenia urządzeń standard przewiduje trzy gniazda:



Forma transmisji MIDI dla pojedyńczego bajtu

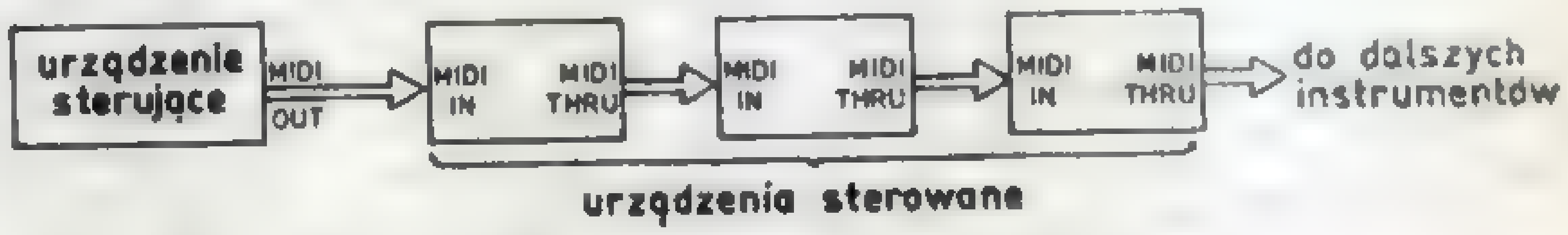
Rys. 5

1) *MIDI IN* — wejście do przyjmowania danych przesyłanych z innych urządzeń,
2) *MIDI OUT* — wyjście z urządzenia wysyłające dane z tego urządzenia,
3) *MIDI THRU* — („przelotka"), wyjście odwzorowujące dane przychodzące do gniazda MIDI IN pozwala na szeregowe łączenie instrumentów w większe zestawy



Rys. 6

Przykłady połączeń:
1) najczęściej spotykany zestaw składa się z dwóch urządzeń: nadajnika (urządzenia A) i odbiornika (urządzenie B). W tym wypadku wyjście MIDI OUT urządzenia A łączymy z wejściem MIDI IN urządzenia B,
2) jedno urządzenie sterujące (np. syntezator, sekwencer, komputer) steruje kilkoma urządzeniami odbiorczymi (mogą być inne syntezatory, ekspandery, itp.).



Rys. 7

Urządzenia podporządkowane łączymy szeregowo za pomocą gniazd MIDI IN — MIDI THRU.

Komunikaty transmisji MIDI

informacje podstawowe

Wszystkie informacje w systemie MIDI są przesyłane w formie **komunikatów**. Najważniejsze dla naszego programu komunikaty przedstawia tabela 1.

Każde z urządzeń w systemie MIDI posiada przypisany mu **numer kanału**. Stwarza to możliwość wybiórczego sterowania instrumentami. Istnieją także komunikaty nie wymagające numeru kanału — dotyczą wszystkich urządzeń w systemie.

Nazwa komunikatu	1 BAJT bajt stanu	2 BAJT dane	3 BAJT dane
klawisz wciśnięty	1001nnnn	0kkkkkkk	0ddddddd
klawisz puszczony	1000nnnn	0kkkkkkk	00000000
Przełączenie brzmienia	1100nnnn	0bbbbbbb	- - - - - - - -

tabela 1

wyjaśnienia:

1) **nnnn** — jest numerem kanału MIDI, dla którego komunikat jest przeznaczony (zakres: 1—16). Np.: nnnn = 0000 — kanał 1, nnnn = 1111 — kanał 16,
2) **kkkkkkk** — jest binarnym zapisem numeru klawisza (zakres: 0—127). Klawisze są ponumerowane w kolejności od tonów niskich do wysokich. Np. typowa klawiatura pięciooktawowa (od C do c⁴) obejmuje klawisze o numerach od 36 do 96,
3) **ddddddd** — dotyczy siły uderzenia w klawisz (zakres: 0—127). Jest to parametr wykorzystywany przez syntezatory z tzw. klawiaturą dynamiczną. Pozostałe syntezatory ignorują ten parametr ustawiając maksymalną dynamikę (ddddddd = 1111111) lub jej połowę (ddddddd = 1000000),
4) **bbbbbbb** — numer brzmienia (zakres: 0—127). Jeżeli instrument posiada mniejszą ilość brzmień, to bierze pod uwagę odpowiednią ilość młodszych bitów wartości bbbbbbb

```
   9 DEF FN e(y,x)=CODE r$((y-1)
*y/2+x): DEF FN u(y)=256*CODE u$
(y+1)+CODE u$(y)
  55 LET u$=CHR$ 0
1001 LET it=CODE m$(1): LET iu=C
ODE u$(1)
1010 GO SUB 9198: PRINT AT 19,0;
"EDYTOR GLOWNY"
1013 IF x$="W" AND iu>0 THEN GO
SUB 1165: GO TO 1005
1015 IF x$="U" AND it>0 THEN GO
SUB 3000: GO TO 1005
1025 IF x$="K" AND iu>0 THEN GO
SUB 1135: GO TO 1010
1032 IF x$="L" AND iu>0 THEN GO
SUB 1190: GO TO 1010
1122 LET z$="podaj nr utworu:":
LET md=iu: LET q=0: GO TO 9190
1131 LET zapis=0: IF q=1 THEN GO
 SUB 8540: RETURN
1135 GO SUB 1122: IF q=0 THEN RE
TURN
1136 LET y=q: GO TO 3820
1165 GO SUB 1122: IF q=0 THEN RE
TURN
1166 LET nu=q: LET l=FN u(2*q):
LET itu=CODE u$(l): LET b$=u$(l+
1 TO l+10+itu): LET r$=u$(l+11+i
tu TO l+10+itu+itu*(itu+1)/2): G
O TO 3010
1190 CLS : PRINT "nr   nazwa
   it": PRINT
1191 FOR t=1 TO iu: LET n=FN u(2
*t): PRINT t;TAB 3; INVERSE 1;u$
(n+1 TO n+10); INVERSE 0;TAB 17;
CODE u$(n)
1193 NEXT t: RETURN
1913 PRINT TAB 5;"W - wywolanie
utworu"
1915 PRINT TAB 5;"U - utworzenie
 utworu"
1919 PRINT TAB 5;"K - kasowanie
utworu"
1933 PRINT TAB 5;"L - lista utwo
row"
3000 REM edytor utworu
3005 LET k=1: GO SUB 3800: IF k<
=iu OR q$="" THEN RETURN
3006 LET b$=q$: LET nu=iu+1
3007 LET itu=0: LET r$=""
3010 GO SUB 3900
3020 GO SUB 9198: PRINT AT 19,0;
"EDYTOR UTWORU"
3025 LET x$=INKEY$: IF x$="" THE
N GO TO 3025
```

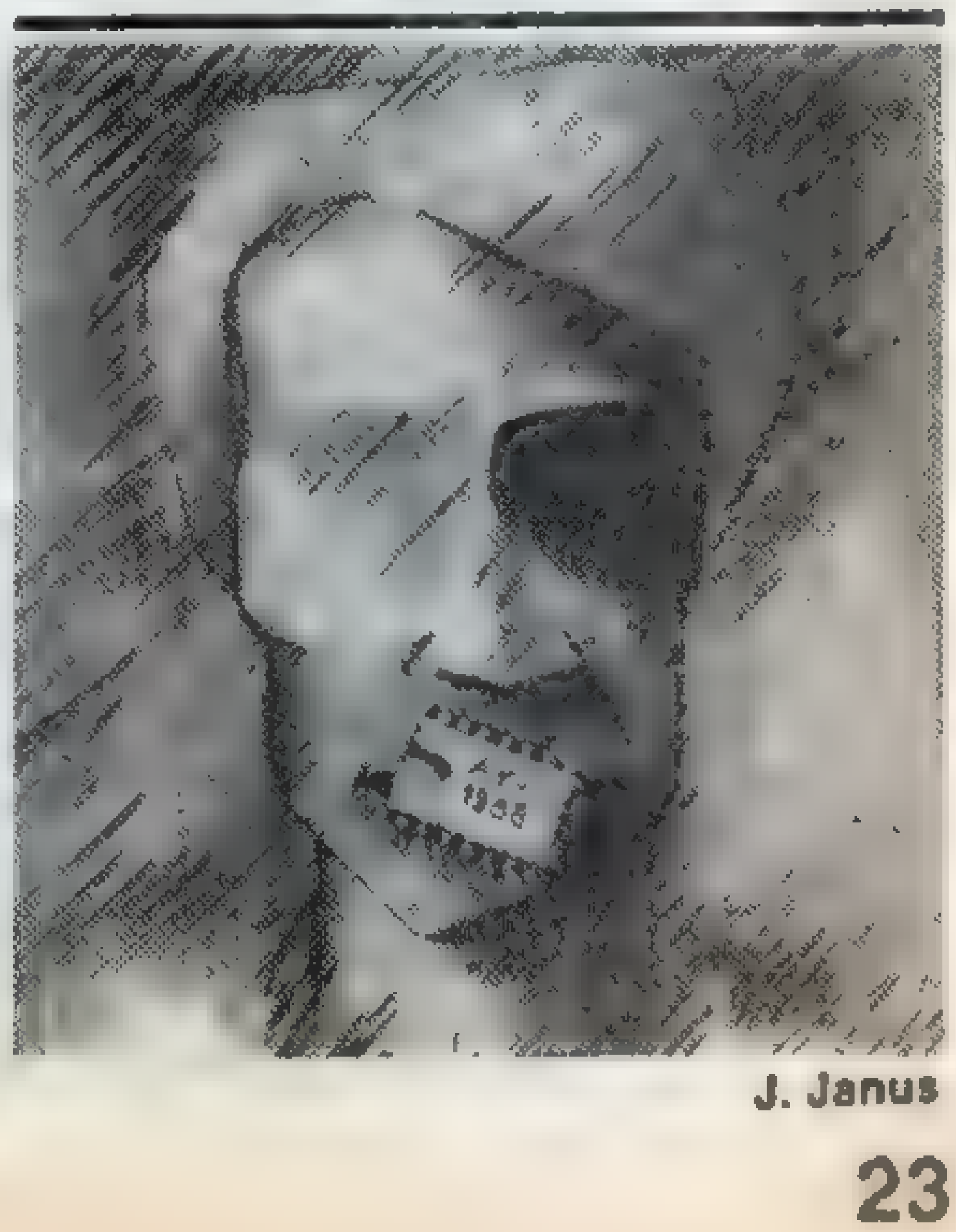
```
3030 IF x$="d" AND itu<6 THEN GO
 SUB 3100: GO TO 3020
3035 IF x$="t" THEN GO SUB 1180:
 GO TO 3020
3040 IF x$="s" THEN GO SUB 3900:
 GO TO 3020
3045 IF x$="p" AND itu>0 THEN GO
 SUB 3200: GO TO 3020
3050 IF x$="k" AND itu>0 THEN GO
 SUB 3300: GO TO 3020
3055 IF x$="K" AND itu>0 THEN GO
 SUB 3400: GO TO 3020
3060 IF x$="z" AND itu>0 THEN GO
 TO 3500
3065 IF x$="e" THEN GO TO 3600
3070 IF x$="w" THEN GO SUB 1115:
 GO SUB 3900: GO TO 3020
3080 GO TO 3025
3100 LET z$="dol. taktu na poz.
nr": LET md=itu+1: LET q=0: GO S
UB 9190: IF q=0 THEN RETURN
3115 LET k=q: GO SUB 1120: IF q=
0 THEN RETURN
3120 LET z$="": FOR l=1 TO k: LE
T z$=z$+CHR$ 0: NEXT l: LET j=k*
(k-1)/2+1: LET r$=r$( TO j-1)+z$
+r$(j TO )
3121 FOR l=k+1 TO itu+1: LET j=l
*(l-1)/2+k: LET r$=r$( TO j-1)+C
HR$ 0+r$(j TO ): NEXT l: LET b$=
b$( TO 9+k)+CHR$ q+b$(10+k TO ):
 LET itu=itu+1: GO TO 3900
3200 LET z$="powt. od pozycji nr
": LET md=itu: LET q=0: GO SUB 9
190: IF q=0 THEN RETURN
3210 LET k=q: LET z$="powt. do p
ozycji nr": LET md=k: LET q=0: G
O SUB 9190: IF q=0 THEN RETURN
3220 FOR y=q TO k-1: FOR x=1 TO
q-1: IF FN e(y,x)<>0 THEN GO TO
3250
3225 NEXT x: NEXT y
3230 FOR y=k+1 TO itu: FOR x=q+1
 TO k: IF FN e(y,x)<>0 THEN GO T
O 3250
3235 NEXT x: NEXT y
3240 LET l=q: LET z$="ile powtor
zen": LET q=0: LET md=99: GO SUB
 9190: IF q=0 THEN RETURN
3245 LET r$((k-1)*k/2+l)=CHR$ q:
 GO TO 3900
3250 PRINT AT 19,0; FLASH 1;" BL
AD STRUKTURALNY   -   Enter  ":
PAUSE 1: PAUSE 0: RETURN
3300 LET z$="kas. powt. od poz.
nr": LET md=itu: LET q=0: GO SUB
 9190: IF q=0 THEN RETURN
3310 LET k=q: LET z$="kas. powt.
 do poz. nr": LET md=k: LET q=0:
 GO SUB 9190: IF q=0 THEN RETURN

3320 LET r$((k-1)*k/2+q)=CHR$ 0:
 GO TO 3900
3400 LET z$="kasowanie pozycji n
r": LET md=itu: LET q=0: GO SUB
9190: IF q=0 THEN RETURN
3410 LET k=(q-1)*q/2: LET r$=r$(
 TO k)+r$(k+q+1 TO ): FOR l=1 TO
 itu-q: LET k=q+l: LET k=(k-1)*k
/2-l: LET r$=r$( TO k)+r$(k+2 TO
 ): NEXT l
3420 LET b$=b$( TO 9+q)+b$(11+q
TO ): LET itu=itu-1: GO TO 3900
3500 LET q=1: LET z$="powrot z z
apisem? t=1.": GO SUB 9190: IF q
=0 THEN GO SUB 9198: GO TO 3010
3502 LET v=nu: IF v<=iu THEN GO
SUB 3820
3505 GO TO 3810
3600 LET z$="powrot bez zapisu 1
-tak": LET md=1: LET q=1: GO SUB
 9190: IF q=1 THEN RETURN
3610 GO TO 3020
3800 REM [inverse text]
3802 LET md=10: LET z$="nazwa ut
woru: ": LET q$="": GO SUB 8980:
 IF q$="" THEN RETURN
3805 FOR k=1 TO iu: LET l=FN u(2
*k)+1: IF u$(l TO l+9)=q$ THEN G
O TO 8979
3807 NEXT k: RETURN
3810 REM [inverse text]
3812 LET b$=CHR$ itu+b$
3814 IF v>iu THEN FOR l=2 TO 2*i
u STEP 2: LET u$(l TO l+1)=FN z$
(FN u(l)+2): NEXT l: LET u$=u$(
TO 2*iu+1)+FN z$(LEN u$+3)+u$(2*
iu+2 TO )+b$+r$: GO TO 3817
3815 LET n=FN u(2*v): LET k=LEN
b$+LEN r$: FOR l=2 TO 2*iu STEP
2: LET u$(l TO l+1)=FN z$(FN u(l
)+k*(l>=2*v)+2): NEXT l
3816 LET u$=u$( TO 2*v-1)+FN z$(
n+2)+u$(2*v TO n-1)+b$+r$+u$(n T
O )
3817 LET iu=iu+1: LET u$(1)=CHR$
 iu: RETURN
3820 REM [inverse text]
3822 LET u=FN u(2*v): LET ur=FN
u(2+2*v)*(v<iu)+(LEN u$+1)*(v=iu
)-FN u(2*v): FOR l=2 TO 2*iu STE
P 2: LET u$(l TO l+1)=FN z$(FN u
(l)-ur*(l>=(2+2*v))-2): NEXT l
3824 LET u$=u$( TO 2*v-1)+u$(2+2
*v TO u-1)+u$(u+ur TO )
3826 LET iu=iu-1: LET u$(1)=CHR$
 iu: RETURN
3830 REM [inverse text]
3831 LET v=1
3832 FOR v=v TO iu: LET c=FN u(2
*v): LET ai=CODE u$(c): LET p=1
3833 FOR p=p TO ai: LET ad=CODE
u$(p+10+c): IF ad>t THEN LET u$(
p+10+c)=CHR$ (ad-1): GO TO 3835
3834 IF ad=t THEN GO TO 3840
3835 LET p=p+1: GO TO 3833
3837 NEXT p: LET v=v+1: GO TO 38
32
3838 NEXT v: RETURN
3840 IF ai=1 THEN GO SUB 3820: G
O TO 3832
3841 FOR k=v+1 TO iu: LET l=2*k:
 LET u$(l TO l+1)=FN z$(FN u(2*k
)-ai-1): NEXT k
3843 LET k=c+10+ai+(p-1)*p/2: LE
T u$=u$( TO k)+u$(k+p+1 TO ): FO
R l=1 TO ai-p: LET k=p+l: LET k=
c+10+ai+(k-1)*k/2-l: LET u$=u$(
TO k)+u$(k+2 TO ): NEXT l
3845 LET u$=u$( TO c+9+p)+u$(c+1
1+p TO ): LET ai=ai-1: LET u$(c)
=CHR$ ai: GO TO 3833
3900 REM [inverse text]
3910 CLS : PRINT "tytul utworu:
"; INVERSE 1;b$(1 TO 10)
3920 FOR k=1 TO itu: LET l=3+FN
z(2*CODE b$(10+k)): PRINT TAB 6;
"|";TAB 0;k;TAB 8; INVERSE 1;m$(
l TO l+9); INVERSE 0;TAB 6;"|":
NEXT k
3930 FOR k=0 TO itu-1: FOR x=1 T
O itu-k: LET y=x+k: LET q=FN e(y
,x): IF q<>0 THEN GO SUB 3940
3935 NEXT x: NEXT k: RETURN
3940 PRINT AT 3*x-2,7;"(-------"
;: FOR l=1 TO k: PRINT "---";: N
EXT l
3942 FOR l=3*x-1 TO 3*y-2: PRINT
 AT l,14+3*k;"|": NEXT l
3943 PRINT AT l,13+3*k+(q<10);q
3945 PRINT AT 3*y,7;"--------";:
 FOR l=1 TO k: PRINT "---";: NEX
T l
3950 RETURN
8828 LET it=it-1: LET m$(1)=CHR$
 it: LET w$=w$( TO t-1)+w$(t+1 T
O ): IF zapis=0 THEN GO SUB 3830
8829 RETURN
8905 GO SUB 9198: PRINT AT 19,0;
"EDYTOR POLA AKTYWNOSCI"
9025 GO SUB 9198: PRINT AT 19,0;
"EDYTOR TAKTU": GO SUB 150
9225 LET zapis=1: LET q=nt: IF q
<=it THEN LET t=q: GO SUB 8925
```

Nowe litery

J. RUBINOWICZ

```
110 REM [illegible]
120 CLEAR 55615: INVERSE 0: BRIGHT 0: CLS : BORDER 6: LET z$="NOWE LITERY ": DIM v$(96): GO SUB 230
130 POKE 23609,50
140 PRINT AT 6,0; PAPER 6;v$
150 FOR d=1 TO 13: PRINT PAPER 2; INK 6;AT 7,2*d+2;z$(d): NEXT d
160 LET a$="ABCDEFGHIJklmnopqrst1234567890"
170 GO SUB 220: BEEP .2,26: BEEP .2,14: BEEP .2,2
200 IF PEEK 56001<>124 THEN GO SUB 240
210 PRINT AT 3,0;: FOR n=32 TO 143: GO SUB 220: PRINT " "; BRIGHT 1;CHR$ n; BRIGHT 0;: GO SUB 230: PRINT CHR$ n; BRIGHT 0;" ";: NEXT n
214 LET inv=0: LET poz=(32-LEN a$)/2: PRINT AT 16,0; INVERSE 0;v$
215 FOR n=1 TO 8: LET inv=ABS inv: INVERSE inv: GO SUB 220: PAUSE 50: BEEP .05,26: PRINT AT 19,poz; BRIGHT 1;a$: GO SUB 230: PAUSE 50: BEEP .05,23: PRINT AT 19,poz; BRIGHT 1;a$: LET inv=inv-1: NEXT n
216 INPUT "tekst (max. 32 znaki) "; LINE a$: IF LEN a$>32 THEN BEEP .2,-10: GO TO 216
217 GO TO 214
220 POKE 23606,64: POKE 23607,216: RETURN : REM NOWE LITERY
230 POKE 23606,0: POKE 23607,60: RETURN : REM STARE LITERY
240 RESTORE : FOR n=55616 TO 56383 STEP 8: LET su=0: FOR m=0 TO 7: READ bt: POKE m+n,bt: LET su=su+bt: NEXT m: READ sm: IF sm<>su THEN PRINT "blad w wierszu ";((n-55360)/8)+1000:
250 PRINT AT 7,14; BRIGHT 1;CHR$ ((n-55360)/8): NEXT n: RETURN
1032 DATA 0,0,0,0,0,0,0,0,0
1033 DATA 0,24,24,24,24,0,24,0,120
1034 DATA 0,102,34,0,0,0,0,0,136
1035 DATA 24,66,102,116,116,102,99,0,640
1036 DATA 0,24,126,88,126,26,126,24,540
1037 DATA 0,98,100,8,16,38,70,0,330
1038 DATA 0,0,60,102,124,96,60,6,448
1039 DATA 0,24,8,0,0,0,0,0,32
1040 DATA 0,12,24,24,24,24,12,0,[illegible]
1041 DATA 0,48,24,24,24,24,48,0,192
1042 DATA 0,0,102,24,126,24,102,0,378
1043 DATA 0,0,24,24,126,24,24,0,222
1044 DATA 0,0,0,0,0,24,24,8,56
1045 DATA 0,0,0,0,62,0,0,0,62
1046 DATA 0,0,0,0,0,24,24,0,48
1047 DATA 0,0,6,12,24,48,96,0,186
1048 DATA 0,60,102,110,118,102,60,0,552
1049 DATA 0,28,44,12,12,12,30,0,138
1050 DATA 0,60,102,6,60,96,126,0,450
1051 DATA 0,60,70,12,6,70,60,0,278
1052 DATA 0,12,28,44,76,126,12,0,298
1053 DATA 0,126,96,124,6,70,60,0,482
1054 DATA 0,60,96,124,102,102,60,0,544
1055 DATA 0,126,6,12,24,48,96,0,312
1056 DATA 0,60,102,60,102,102,60,0,486
1057 DATA 0,60,102,102,62,6,60,0,392
1058 DATA 0,0,24,24,0,24,24,0,96
1059 DATA 0,0,24,24,0,24,24,8,104
1060 DATA 0,0,12,24,48,24,12,0,120
1061 DATA 0,0,0,62,0,62,0,0,124
1062 DATA 0,0,48,24,12,24,48,0,156
1063 DATA 0,60,102,12,24,0,24,0,222
1064 DATA 0,0,60,6,62,102,62,3,295
1065 DATA 0,24,60,102,102,126,102,0,516
1066 DATA 0,124,102,124,102,102,124,0,678
1067 DATA 0,60,102,96,96,102,60,0,516
1068 DATA 0,120,108,102,102,108,120,0,660
1069 DATA 0,126,96,124,96,96,126,0,684
1070 DATA 0,126,96,124,96,96,96,0,634
1071 DATA 0,60,102,96,110,102,60,0,530
1072 DATA 0,102,102,126,102,102,102,0,636
1073 DATA 0,126,24,24,24,24,126,0,348
1074 DATA 0,6,6,6,102,102,60,0,282
1075 DATA 0,102,108,120,108,102,99,0,639
1076 DATA 0,96,96,96,96,96,126,0,606
1077 DATA 0,66,102,126,102,102,102,0,600
1078 DATA 0,70,102,118,110,102,98,0,600
1079 DATA 0,60,102,102,102,102,60,0,528
1080 DATA 0,124,102,102,124,96,96,0,644
1081 DATA 0,60,102,102,102,102,60,6,534
1082 DATA 0,124,102,102,124,108,102,0,662
1083 DATA 0,60,96,60,6,102,60,0,384
1084 DATA 0,126,24,24,24,24,24,0,246
1085 DATA 0,102,102,102,102,102,60,0,570
1086 DATA 0,102,102,102,102,60,24,0,492
1087 DATA 0,102,102,102,102,126,36,0,570
1088 DATA 0,102,60,24,24,60,102,0,372
1089 DATA 0,195,102,60,24,24,24,0,429
1090 DATA 0,126,12,24,24,48,126,0,360
1091 DATA 0,126,96,124,96,96,126,3,667
1092 DATA 0,24,26,28,24,56,20,0,166
1093 DATA 0,24,60,102,102,126,102,3,519
1094 DATA 0,24,60,90,24,24,24,0,246
1095 DATA 0,0,0,0,0,0,0,255,255
1096 DATA 0,96,108,120,112,96,126,0,658
1097 DATA 0,0,62,6,62,102,62,0,294
1098 DATA 0,96,96,124,102,102,124,0,644
1099 DATA 0,0,60,96,96,96,60,0,412
1100 DATA 0,6,6,62,102,102,62,0,340
1101 DATA 0,0,60,102,124,96,60,0,442
1102 DATA 0,28,48,56,48,48,48,0,276
1103 DATA 0,0,62,102,102,62,6,60,394
1104 DATA 0,96,96,124,102,102,102,0,622
1105 DATA 0,24,0,56,24,24,60,0,188
1106 DATA 0,12,0,12,12,12,108,56,212
1107 DATA 0,96,108,120,120,108,102,0,654
1108 DATA 0,56,24,24,24,24,28,0,180
1109 DATA 0,0,108,86,86,86,86,0,452
1110 DATA 0,0,124,102,102,102,102,0,532
1111 DATA 0,0,60,102,102,102,60,0,426
1112 DATA 0,0,124,102,102,124,96,96,644
1113 DATA 0,0,62,102,102,62,6,7,341
1114 DATA 0,0,60,102,96,96,96,0,450
1115 DATA 0,0,60,96,60,6,124,0,346
1116 DATA 0,24,60,24,24,24,28,0,184
1117 DATA 0,0,102,102,102,102,62,0,470
1118 DATA 0,0,99,99,54,54,28,0,334
1119 DATA 0,0,70,86,86,86,44,0,372
1120 DATA 0,0,102,60,24,60,102,0,348
1121 DATA 0,0,102,102,102,62,6,60,434
1122 DATA 0,0,126,12,24,48,126,0,336
1123 DATA 6,4,60,98,96,98,60,0,422
1124 DATA 24,0,126,12,24,48,126,0,360
1125 DATA 6,4,62,96,60,6,124,0,358
1126 DATA 24,126,12,24,24,48,126,0,384
1127 DATA 12,6,124,102,102,102,102,0,552
4000 FOR n=55616 TO 56383 STEP 8: POKE 23692,255: PRINT CHR$ ((n-55360)/8);" ";: LET su=0: BEEP .05,26: FOR m=0 TO 7: PRINT PEEK (n+m);",";: LET su=su+PEEK (n+m): NEXT m: PRINT BRIGHT 1;su: PRINT : NEXT n
4010 GO TO 4000
9998 REM save "NOWELITERY"
9999 CLS : SAVE "NOWELITERY" LINE 1: BEEP .2,2: SAVE "nowelitery"CODE 55616,768: PRINT INK 2; BRIGHT 1; FLASH 1;AT 11,12;" VERIFY ": PRINT AT 0,0;: BEEP .5,-10: VERIFY "": VERIFY ""CODE : RESTORE : GO TO 160
```



J. Janus

Lilavati (8)

Wilk i owce

Danuta KWASIŻUR, Mieczysław SKONIECZNY

W dzisiejszym odcinku mikrokomputerowych anegdot matematycznych chcemy zaproponować grę pod ogólnie znanym tytułem „Wilk i owce". Jest to gra o nierównych szansach, przeznaczona dla 2 graczy: jeżeli grający owcami nie popełni rażącej pomyłki, powinien wygrać.

Gra jest realizowana na standardowej szachownicy o 64 polach. Jeden z graczy ma 4 owce (ponumerowane 1—4), drugi natomiast jednego wilka. Zarówno wilk, jak i owce mogą się przesuwać na skos o jedno pole tylko tej samej barwy. Owce mogą się przesuwać jedynie do przodu, gdy tymczasem wilk ma również prawo do ruchów do tyłu.

Na początku gry należy odpowiedzieć na 2 pytania wyświetlane w prawym dolnym rogu ekranu:

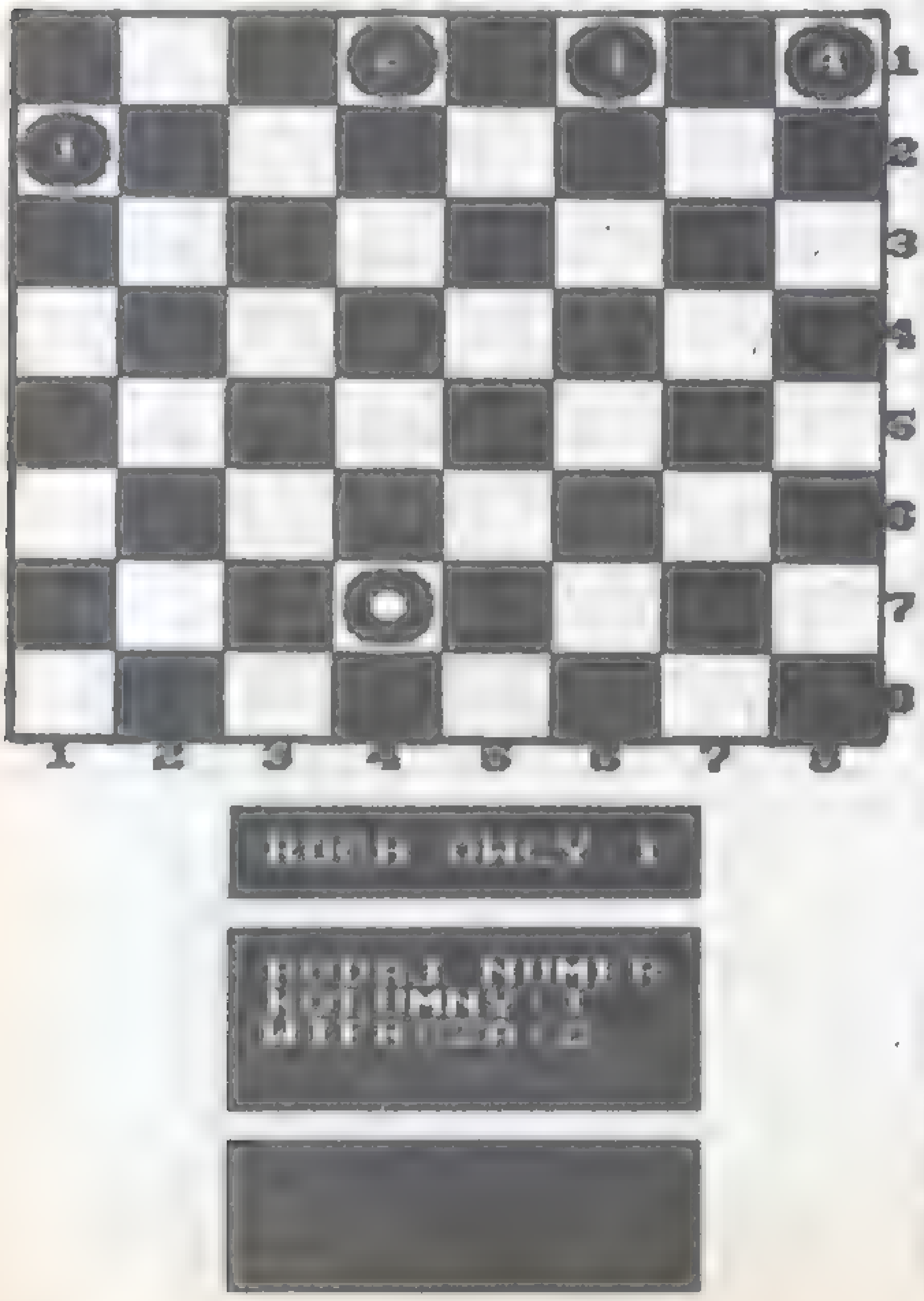
Na jakich polach gramy
B — białe
C — czarne
Podaj kod literowy:

oraz

Kto pierwszy robi ruch:
O — owce
W — wilk
Podaj kod literowy:

wciskając odpowiednie klucze literowe.

Następnie rozpoczyna się gra wg parametrów wprowadzonych uprzednio. Początkową sytuację po 2 ruchach prezentuje poniższy rysunek



Przebiegu gry nie komentujemy, gdyż jest on oczywisty i nie powinien sprawiać Czytelnikom problemów.

Trzecie dolne okienko po prawej stronie ekranu jest przeznaczone na komunikaty o sytuacjach błędnych, które są następujące

TO POLE JEST ZAJĘTE
TAK NIE WOLNO SIĘ PRZESUWAĆ

Gra kończy się wygraną wilka, gdy przedostanie się poza szereg owiec. Owce wygrywają wtedy, gdy osaczą wilka tak że nie może on dokonać żadnego ruchu.

Wersja źródłowa programu jest przedstawiona na załączniku 1

```
10 SYMBOL AFTER 247
20 SYMBOL 247,0,0,0,3,7,15,31,31
30 SYMBOL 248,0,0,126,255,255,255,255,255
40 SYMBOL 249,0,0,0,192,224,240,248,248
50 SYMBOL 250,31,63,63,63,63,63,63,31
60 SYMBOL 251,195,129,0,0,0,0,129,195
70 SYMBOL 252,248,252,252,252,252,252,252,248
80 SYMBOL 253,31,31,15,7,3
90 SYMBOL 254,255,255,255,255,255,126
100 SYMBOL 255,248,248,240,224,192
110 'Ekran informacyjny
120 MODE 1:INK 0,26:INK 1,0:INK 2,6:INK 3,19:BORDER 0:PAPER 1:CLS:PEN 0
130 WINDOW £1,1,40,1,3:PAPER £1,0:PEN £1,1:CLS£1
140 WINDOW £2,1,40,23,25:PAPER £2,0:PEN £2,1:CLS£2
150 LOCATE £1,10,2:PRINT£1,"PROGRAM - WILK I OWCE"
160 LOCATE £2,8,2:PRINT£2,"NACISNIJ DOWOLNY KLAWISZ"
170 CLEAR INPUT:A$=INKEY$
180 IF A$="" GOTO 170
190 'Deklaracja kolorow i okien w trybie 1
200 MODE 1:INK 0,26:INK 1,0:INK 2,6:INK 3,19
210 PAPER 2:BORDER 6:CLS
220 WINDOW £1,28,40,1,3:PEN £1,1:PAPER £1,0:CLS£1
230 WINDOW £2,28,40,5,10:PEN £2,1:PAPER £2,0:CLS£2
240 WINDOW £3,28,40,12,16:PEN £3,1:PAPER £3,0:CLS£3
250 'Rysowanie szachownicy
260 PAPER 0:PEN 1:FOR i=0 TO 6 STEP 2
270 FOR i1=6*i+1 TO 6*i+3
280 FOR i2=1 TO 4
290 PRINT CHR$(143);CHR$(143);CHR$(143);"   ";
300 NEXT i2:PRINT: NEXT i1
310 FOR i1=6*i+4 TO 6*i+6
320 FOR i2=1 TO 4
330 PRINT "   ";CHR$(143);CHR$(143);CHR$(143);
340 NEXT i2:PRINT: NEXT i1:NEXT i
350  PAPER 2:PRINT" 1  2  3  4  5  6  7  8"
360 FOR i=1 TO 8 :LOCATE 25,i*3-1:PRINT USING"£";i:NEXT
370 'Wybor koloru pol na ktorych gramy
380 LOCATE 28,19:PRINT"Na jakich":LOCATE 28,20:PRINT"polach gramy:";
390 LOCATE 30,21:PRINT"B-biale":LOCATE 30,22:PRINT"C-czarne
400 LOCATE 28,23:PRINT"Podaj kod":LOCATE 28,24:INPUT"literowy:",kolor$
410 kolor$=UPPER$(kolor$)
420 IF kolor$<>"B" GOTO 450
430 FOR i=1 TO 4:owce(i,1)=2*i:owce(i,2)=1:NEXT
440 wilk(1)=2*(1+INT(RND*4))-1:wilk(2)=8:GOTO 520
450 IF kolor$<>"C" GOTO 480
460 FOR i=1 TO 4:owce(i,1)=2*i-1:owce(i,2)=1:NEXT
470 wilk(1)=2*(1+INT(RND*4)):wilk(2)=8:GOTO 520
480 LOCATE 28,25:PRINT"BLAD !!!"
490 FOR t=1 TO 2000:NEXT
500 LOCATE 28,24:PRINT SPC(12):LOCATE 28,25:PRINT SPC(9)
```

```
510 GOTO 400
520 'Wybor kto zaczyna
530 FOR i=19 TO 24:LOCATE 28,i:PRINT SPC(13);:NEXT
540 LOCATE 28,19:PRINT"Kto pierwszy":LOCATE 28,20:PRINT"robi ruch:"
550 LOCATE 30,21:PRINT"O-owce":LOCATE 30,22:PRINT"W-wilk"
560 LOCATE 28,23:PRINT"Podaj kod":LOCATE 28,24:INPUT"literowy:",ruch$
570 ruch$=UPPER$(ruch$)
580 IF ruch$="O" OR ruch$="W" GOTO 630
590 LOCATE 28,25:PRINT"BLAD !!!"
600 FOR t=1 TO 2000:NEXT
610 LOCATE 28,24:PRINT SPC(12):LOCATE 28,25:PRINT SPC(9)
620 GOTO 560
630 FOR i=19 TO 24:LOCATE 28,i:PRINT SPC(13);:NEXT
640 'Poczatkoweustawienie wilka na szachownicy
650 IF kolor$="B" THEN PAPER o:PEN 1 ELSE PAPER 1:PEN 0
660 x=3*wilk(1)-2:LOCATE x,22:PRINT CHR$(247);CHR$(248);CHR$(249)
670 LOCATE x,23:PRINT CHR$(250);CHR$(251);CHR$(252)
680 LOCATE x,24:PRINT CHR$(253);CHR$(254);CHR$(255)
```

W trakcie realizacji gry wielokrotnie są wywoływane podprogramy, których rola jest następująca:

1. Podprogram podawania współrzędnych ruchu

```
690 'Poczatkowe ustawienie owiec na szachownicy
700 FOR i=1 TO 4
710 x=3*owce(i,1)-2:LOCATE x,1:PRINT CHR$(247);CHR$(248);CHR$(249)
720 LOCATE x,2:PRINT CHR$(250);CHR$(251);CHR$(252)
730 LOCATE x,3:PRINT CHR$(253);CHR$(254);CHR$(255)
740 IF kolor$="C" THEN PAPER 0:PEN 1 ELSE PAPER 1:PEN 0
750 LOCATE x+1,2:PRINT USING "£";i
760 IF kolor$="C" THEN PAPER 1:PEN 0 ELSE PAPER 0:PEN 1
770 NEXT
780 'Realizacja kolejnego ruchu
790 GOSUB 860
800 blad=0:GOSUB 1150
810 IF blad=1 GOTO 790
820 GOSUB 1420
830 GOSUB 1670
840 IF ruch$="O" THEN ruch$="W" ELSE ruch$="O"
850 GOTO 780
```

2. Podprogram badania dopuszczalności ruchu

```
860 'Podprogram podawania wspolrzednych ruchu
870 IF ruch$="W" GOTO 1040
880 CLS £1:LOCATE £1,2,2
890 PRINT£1,"RUCH OWCY "
900 CLS £2:LOCATE £2,2,2:PRINT£2,"PODAJ NUMER"
910 LOCATE £2,2,3:INPUT £2,"OWCY:",nrowcy
920 IF nrowcy>0 AND nrowcy<5 GOTO 950
930 a=4:GOSUB 1090
940 LOCATE £2,2,3:PRINT£2,SPC(12):GOTO 910
950 LOCATE £1,11,2:PRINT £1,nrowcy
960 LOCATE £2,2,3:INPUT £2,"KOLUMNY:",nrkol
970 IF nrkol>0 AND nrkol<9 GOTO 1000
980 a=8:GOSUB 1090
990 LOCATE £2,2,3:PRINT£2,SPC(12):GOTO 960
1000 LOCATE £2,2,4:INPUT £2,"WIERSZA:",nrwier
1010 IF nrwier>0 AND nrwier<9 GOTO 1080
1020 a=8:GOSUB 1090
1030 LOCATE £2,2,4:PRINT£2,SPC(12):GOTO 1000
1040 CLS £1:LOCATE £1,2,2
1050 PRINT£1,"RUCH WILKA"
1060 CLS £2:LOCATE £2,2,2:PRINT£2,"PODAJ NUMER"
1070 GOTO 960
1080 RETURN
1090 'Komunikat o bledzie
1100 LOCATE £3,3,2:PRINT£3,"TEN NUMER"
1110 LOCATE £3,3,3:PRINT£3,"MUSI BYC"
1120 LOCATE £3,3,4:PRINT£3,"OD 1 DO ";USING "£";a
1130 FOR t=1 TO 2000:NEXT
1140 CLS £3:RETURN
```

3. Podprogram realizacji ruchu

```
1150 'Podprogram badania dopuszczalnosci ruchu
1160 'Badanie zajetosci pola
1170 IF nrkol=wilk(1) AND nrwier=wilk(2) GOTO 1220
1180 FOR i=1 TO 4
1190 IF nrkol=owce(i,1) AND nrwier=owce(i,2) GOTO 1220
1200 NEXT:GOTO 1270
1210 'Komunikat o bledzie
1220 LOCATE £3,4,2:PRINT£3,"TO POLE"
1230 LOCATE £3,5,3:PRINT£3,"JEST"
1240 LOCATE £3,4,4:PRINT£3,"ZAJETE"
1250 FOR t=1 TO 2000:NEXT
1260 CLS £3:blad=1:GOTO 1410
1270 IF ruch$="W" GOTO 1330
1280 'Badanie ruchu owcy
1290 IF owce(nrowcy,2)+1<>nrwier GOTO 1360
1300 IF owce(nrowcy,1)-1<>nrkol AND owce(nrowcy,1)+1<>nrkol GOTO 1360
1310 GOTO 1410
1320 'Badanie ruchu wilka
1330 IF wilk(1)<>nrkol-1 AND wilk(1)<>nrkol+1 GOTO 1360
1340 IF wilk(2)<>nrwier-1 AND wilk(2)<>nrwier+1 GOTO 1360
1350 GOTO 1410
1360 'Komunikat o bledzie
1370 LOCATE £3,4,2:PRINT£3,"TAK NIE"
1380 LOCATE £3,3,3:PRINT£3,"WOLNO SIE"
1390 LOCATE £3,3,4:PRINT£3,"PRZESUWAC"
1400 FOR t=1 TO 2000:NEXT:blad=1
1410 CLS £3:RETURN
```

4. Podprogram badania i realizacji końca gry

```
1420 'Podprogram realizacji ruchu
1430 IF ruch$="O" GOTO 1540
1440 'Przesuniecie wilka
1450 x=3*wilk(1)-2:y=3*wilk(2)-2
1460 FOR i=0 TO 2:LOCATE x,y+i
1470 PRINT SPC(3):NEXT
1480 x=3*nrkol-2:y=3*nrwier-2
1490 FOR i=0 TO 2:LOCATE x,y+i
1500 FOR j=0 TO 2:PRINT CHR$(247+3*i+j);:NEXT
1510 NEXT
1520 wilk(1)=nrkol:wilk(2)=nrwier
1530 GOTO 1660
1540 'Przesuniecie owcy
1550 x=3*owce(nrowcy,1)-2:y=3*owce(nrowcy,2)-2
1560 FOR i=0 TO 2:LOCATE x,y+i
1570 PRINT SPC(3):NEXT
1580 x=3*nrkol-2:y=3*nrwier-2
1590 FOR i=0 TO 2:LOCATE x,y+i
1600 FOR j=0 TO 2:PRINT CHR$(247+3*i+j);:NEXT
1610 NEXT
1620 IF kolor$="C" THEN PAPER 0:PEN 1 ELSE PAPER 1:PEN 0
1630 LOCATE x+1,y+1:PRINT USING "£";nrowcy
1640 IF kolor$="C" THEN PAPER 1:PEN 0 ELSE PAPER 0:PEN 1
1650 owce(nrowcy,1)=nrkol:owce(nrowcy,2)=nrwier
1660 RETURN

1670 'Podprogram badania i realizacji konca gry
1680 'Wygrana wilka
1690 IF wilk(2)=1 THEN a=1:GOTO 1960
1700 FOR i=1 TO 4
1710 IF wilk(2)>owce(i,2) GOTO 1750
1720 NEXT
```

```
1730 a=1:GOTO 1960
1740 'Wygrana owiec
1750 IF wilk(1)<>1 GOTO 1810
1760 FOR b=wilk(2)-1 TO wilk(2)+1 STEP 2
1770 FOR i=1 TO 4
1780 IF owce(i,1)=2 AND owce(i,2)=b GOTO 1800
1790 NEXT i:GOTO 2080
1800 NEXT b:GOTO 1950
1810 IF wilk(1)<>8 GOTO 1870
1820 FOR b=wilk(2)-1 TO wilk(2)+1 STEP 2
1830 FOR i=1 TO 4
1840 IF owce(i,1)=7 AND owce(i,2)=b GOTO 1860
1850 NEXT i:GOTO 2080
1860 NEXT b:GOTO 1950
1870 FOR b=wilk(2)-1 TO wilk(2)+1 STEP 2
1880 FOR c=wilk(1)-1 TO wilk(1)+1 STEP 2
1890 FOR i=1 TO 4
1900 IF owce(i,1)=c AND owce(i,2)=b GOTO 1930
1910 NEXT
1920 GOTO 2080
1930 NEXT:NEXT
1940 'Czy koniec gry
1950 a=2
1960 CLS £1:LOCATE £1,2,2:PRINT£1,"KONIEC GRY"
1970 CLS £2:LOCATE £2,4,2
1980 IF a=1 THEN PRINT£2,"WYGRAL" ELSE PRINT£2,"WYGRALY"
1990 LOCATE £2,5,4
2000 IF a=1 THEN PRINT£2,"WILK" ELSE PRINT£2,"OWCE"
2010 LOCATE £3,4,2:PRINT£3,"CZY GRE":LOCATE £3,3,3:PRINT£3,"POWTORZYC"
2020 LOCATE £3,5,4:INPUT £3, "(T/N):",dalej$
2030 dalej$=UPPER$(dalej$)
2040 IF dalej$="T" GOTO 190
2050 PAPER 2:CLS
2060 LOCATE 16,12:PRINT"KONIEC GRY"
2070 GOTO 2070
2080 RETURN
```

Danuta KWASIŻUR
Mieczysław SKONIECZNY

Różne sposoby kopiowania ekranu mikrokomputera na drukarce

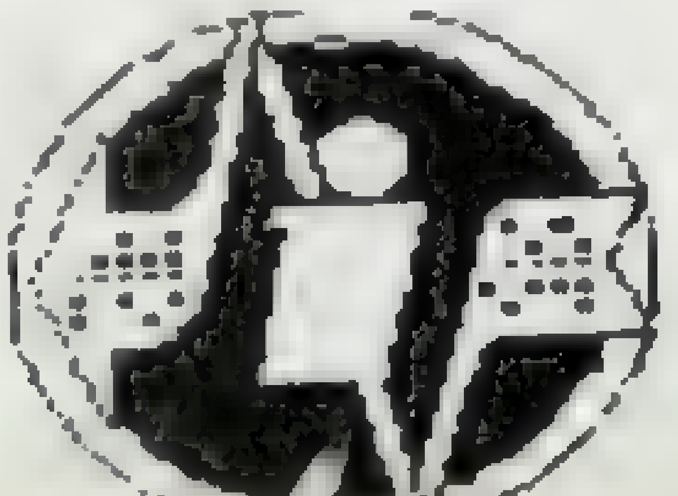
Wśród użytkowników mikrokomputerów znanych jest sporo programów kopiujących zawartość ekranu na drukarkę. Stanowią one wygodną formę uzyskiwania ciekawych rysunków. Technika współpracy z takim programem jest najczęściej następująca:

1. Przygotowanie rysunku jako zbioru na dysku. Realizuje się to instrukcją: SAVE „rysunek", b, &c000, &4000, wydrukowany zbiór zajmuje 17 kB pamięci i ma dopełnienie. BIN.
2. Wprowadzenie do pamięci mikrokomputera programu kopiującego, zainicjowanie jego pracy i wprowadzenie niezbędnych parametrów (jest to najczęściej nazwa zapisana na dysku zbioru).
3. Wydruk rysunku.

Chcemy zaproponować Czytelnikom wygodny w użyciu program, który wykonuje kopie ekranu dwoma różnymi sposobami:

1. **Normalnym** — ścisłe odwzorowanie ekranu na papierze. Wydruk powstaje przez kolejne kopiowanie wierszy od góry do dołu.
2. **Powiększonym** — odwzorowanie ekranu na papierze w skali 1.2. Wydruk powstaje przez kolejne kopiowanie kolumn od strony prawej do lewej.

Poniższe 2 rysunki pokazują kopie tego samego rysunku wykonane ww. sposobami.





rys. 2

Jak widać, pierwszy sposób nieznacznie spłaszcza obiekt, natomiast drugi go wydłuża. Mamy jednak nadzieję, że te drobne mankamenty nie będą przeszkadzały użytkownikom tego programu. Program pracuje w trybie konwersacyjnym, a więc na kolejne pytania należy udzielać następujących odpowiedzi:

1. PODAJ WARIANT KOPIOWANIA:
   Wprowadzamy liczbę 1 lub 2 w zależności od sposobu, którym chcemy kopiować.
2. PODAJ NAZWĘ ZBIORU:
   Wprowadzamy nazwę zeskładowanego na dysku zbioru bez dopełnienia (program sam uzupełnia ją o dopełnienie. BIN)
3. PODAJ TRYB, W KTÓRYM UTWORZONO OBRAZ:
   Wprowadzamy liczbę całkowitą z przedziału (0,2).

Na bieżąco jest przeprowadzana kontrola formalna wprowadzanych danych. W wypadku próby podania błędnego parametru program sygnalizuje to jednym z komunikatów:

BŁĘDNY WARIANT
ZA DŁUGA NAZWA ZBIORU
BŁĘDNY TRYB

i żąda ponownego wprowadzenia tej danej. Następnie po włożeniu w stację odpowiedniej dyskietki i przygotowaniu drukarki wykonuje się kopia ekranu. Po wykonaniu rysunku program wraca do stanu początkowego, co pozwala wykonać jednorazowo więcej niż 1 rysunek.

Segment sterujący programu opracowano w języku BASIC mikrokomputera AMSTRAD CPC 6128 i ma on następującą postać:

```
10 REM 'Program kopiowania ekranu mikrokomputera AMSTRAD CPC 6128'
20 MODE 1:INK 0,0:INK 1,14:INK 2,26:INK 3,6:PAPER 0:CLS:BORDER 0
30 WINDOW £1,7,33,1,3:PAPER £1,1:CLS £1:PEN £1,0
40 REM 'Mozliwosci programu'
50 LOCATE £1,2,2:PRINT £1,"PROGRAM KOPIOWANIA EKRANU"
60 PEN 1:LOCATE 2,7:PRINT "1. KOPIOWANIE NORMALNE":PRINT:PRINT " 2. KO
PIOWANIE POWIEKSZONE (Skala 2:1)"
70 WINDOW £2,1,40,13,20:PAPER £2,2
80 REM 'Wprowadzanie wariantu kopiowania'
90 CLS £2:PEN £2,0:LOCATE £2,2,2:INPUT £2,"PODAJ WARIANT KOPIOWANIA: "
,WK
100 IF WK<1 OR WK>2 THEN PEN 3:LOCATE £2,2,4:PRINT £2,"BLEDNY WARIANT"
:FOR T=1 TO 1000:NEXT:GOTO 90
110 REM 'Wprowadzanie nazwy zbioru'
120 LOCATE £2,2,4:INPUT £2,"PODAJ NAZWE ZBIORU: ",A$
130 IF LEN(A$)>8 THEN PEN 3:LOCATE £2,2,6:PRINT £2,"ZA DLUGA NAZWA ZBI
ORU":FOR T=1 TO 1000:NEXT:LOCATE £2,2,6:PRINT £2,SPC(30):PEN 0
:LOCATE £2,21,4:PRINT £2,SPC(19);:GOTO 120
140 A$=A$+".BIN"
150 REM 'Wprowadzanie trybu obrazu'
160 LOCATE £2,2,6:INPUT £2,"PODAJ TRYB W KTORYM UTWORZONO OBRAZ: ",TRY
B
170 IF TRYB<0 OR TRYB>2 THEN PEN 3:LOCATE £2,2,8:PRINT £2,"BLEDNY TRYB
":FOR T=1 TO 1000:NEXT:LOCATE £2,2,8:PRINT £2,SPC(11):PEN 0:LO
CATE £2,39,6:PRINT £2," ":GOTO 160
180 PAPER 0:PEN 1:LOCATE 1,22:PRINT "Jezeli masz w stacji dysku dyskie
tke z  obrazem i drukarka jest gotowa do pracy to nacisnij dow
olny klawisz."
190 CLEAR INPUT
200 b$=INKEY$
210 IF b$="" THEN GOTO 200
220 REM 'wydruk obrazu na drukarce'
230 MODE tryb
240 LOAD a$
250 MEMORY 39999
260 ON wk GOSUB 470,280
270 GOTO 10
```

Poszczególne sposoby kopiowania są realizowane przez następujące podprogramy, opracowane w języku wewnętrznym:

1. **Kopiowanie powiększone**
   Podprogram ten zajmuje obszar 415 bajtów o adresach od 40000 do 40414. Jego postać poszerzona o instrukcje ładowania go do pamięci i wywoływania jest następująca:

```
280 REM 'Podprogram kopiowania powiekszonego'
290 RESTORE 370
300 FOR a=40000 TO 40414
310 READ x
320 POKE a,x
330 NEXT a
340 CALL 40000
350 :COPY
360 RETURN
370 DATA 1,96,156,33,92,156,205,209,188,201,164,74,46,87,46,47,49,57,5
6,55,47,87,73,73,67,79,80,217,0,0,0,0,88,156,237,115,223,157,2
05,190,157,62,27,205,184,157,62,49,205,184,157
380 DATA 62,255,50,113,157,33,79,192,62,81,50,225,157,62,8,50,226,157,
205,203,156,58,225,157,61,40,35,50,225,157,205,189,156,58,226,
157,71,61,50,226,157,62,8,144,71,205,43,157,58,226
390 DATA 157,61,183,32,219,6,1,205,193,156,24,207,62,240,50,113,157,20
5,189,156,205,158,157,201,58,226,157,71,205,43,157,205,93,157,
205,131,157,201,229,253,33,174,158,6,25,17,0,8,229,14
400 DATA 8,126,205,239,156,253,119,0,253,35,25,13,32,243,225,17,80,0,2
5,16,230,225,43,201,197,6,0,79,230,85,183,40,2,6,15,121,230,17
0,183,40,2,62,240,176,193,201,213,197,79,6
410 DATA 4,175,203,1,203,23,203,1,203,1,203,1,203,1,203,23,87,203,15,2
03,19,203,15,203,19,122,179,95,16,227,193,209,201,201,4,5,200,
221,33,230,157,253,33,174,158,175,14,200,197,253
420 DATA 203,0,14,221,203,0,22,221,182,0,16,243,221,35,253,35,193,13,3
2,234,71,58,227,157,79,120,50,227,157,177,50,228,157,201,58,22
8,157,183,200,6,200,221,33,230,157,205,167,157,221,126
430 DATA 0,203,47,230,255,205,184,157,205,184,157,205,184,157,205,184,
157,221,35,16,233,201,62,13,205,184,157,62,10,205,184,157,221,
229,253,229,229,205,9,187,218,215,157,225,253,225,221,225,201,62
440 DATA 27,205,184,157,62,64,24,17,62,27,205,184,157,62,76,205,184,15
7,62,32,205,184,157,62,3,205,43,189,48,251,201,33,209,157,205,
17,188,135,79,6,0,9,94,35,86,237,83,218,156,201
450 DATA 239,156,5,157,42,157,205,158,157,237,123,223,157,201
```

2. **Kopiowanie normalne**
   Podprogram ten zajmuje obszar 177 bajtów o adresach od 40960 do 41136. Jego postać poszerzona o instrukcje ładowania go do pamięci i wywoływania jest następująca

```
460 REM 'Podprogram kopiowania normalnego'
470 RESTORE 530
480 FOR i=&A000 TO &A0B0
490 READ v:POKE i,v
500 NEXT i
510 CALL &A000
520 RETURN
530 DATA 205,166,160,62,27,205,157,160
540 DATA 62,49,205,157,160,205,186,187
550 DATA 205,231,187,50,180,160,17,0
560 DATA 0,33,143,1,34,178,160,62
570 DATA 7,50,177,160,62,10,205,157
580 DATA 160,62,13,205,157,160,62,27
590 DATA 205,157,160,62,76,205,157,160
600 DATA 62,127,205,157,160,62,2,205
610 DATA 157,160,14,0,58,177,160,71
620 DATA 229,197,213,205,240,187,209,193
630 DATA 33,180,160,190,225,55,32,1
640 DATA 167,203,17,43,43,16,233,58
650 DATA 177,160,254,7,40,7,175,203
660 DATA 17,203,17,203,17,121,205,157
670 DATA 160,19,229,33,127,2,55,237
680 DATA 82,225,56,5,42,178,160,24
690 DATA 193,35,124,181,40,32,43,17
700 DATA 0,0,34,178,160,62,7,189
710 DATA 32,146,124,180,32,142,62,4
720 DATA 50,177,160,24,135,205,46,189
730 DATA 56,251,205,43,189,201,62,27
740 DATA 205,157,160,62,64,205,157,160,201
```

Jeżeli Czytelnicy dysponują innymi programami kopiowania ekranu, to w prosty sposób mogą je dołączyć do tego programu wykonując następujące modyfikacje.

1. Poszerzyć liczbę wariantów i zmienić sposób kontroli (instrukcje 60—100).
2. Zmienić instrukcję 260 dopisując do skoku z przełącznikiem etykiety dołączonych podprogramów
3. Dopisać nowy podprogram kopiujący wraz z jego wywołaniem.



K. WALCZAK

Dodatkowe zestawy znaków graficznych do tworzenia ramek i tablic

Danuta KWASIŻUR Mieczysław SKONIECZNY

Mikrokomputer AMSTRAD daje w ręce użytkownika 256 symboli graficznych. Symbole o nr. 0—127, to standardowe znaki kodu ASCII, których nie powinno się zmieniać. Pozostałe symbole (nr. 128—255), to dodatkowe możliwości graficzne, które można zdefiniować instrukcjami SYMBOL AFTER i SYMBOL.

Sporo programów użytkowych wyprowadza wyniki obliczeń w postaci tabelarycznej. Ważne, z punktu widzenia użytkownika, jest otrzymanie tabel w czytelnej i estetycznej postaci. Spełnienie tego postulatu niejednokrotnie warunkuje pozytywną ocenę programu.

Chcemy zaproponować zestaw 3 typów linii do modelowania ramek i tabelek. Dwa pierwsze zestawy, to symbole zdefiniowane (numery 234—244 oraz 245—255), natomiast trzeci zestaw wykorzystuje istniejące symbole (numery: 147, 149, 150, 151, 153—159).

Program pokazowy przedstawia wszystkie 3 dostępne zestawy symboli graficznych w następującej kolejności:

1. Linie pojedyncze
2. Linie podwójne
3. Linie pogrubione

Po przedstawieniu tytułu tabeli oraz jej zawartości akcja programu została wstrzymana do momentu interwencji użytkownika — naciśnięcia dowolnego klawisza.

Wszystkie zestawy linii są prezentowane w trybie 1. Można rozszerzyć działanie programu tak, aby również wyświetlał te same treści w trybie 0 i 2. Tę modyfikację pozostawiamy zainteresowanym Czytelnikom.

Program pokazowy opracowano w języku BASIC, a jego postać jest następująca:

```
10 [illegible]
20 [illegible]
30 MODE 1:BORDER 0:INK 0,[illegible]:INK 1,[illegible]:PAPER 0:PEN 1
40 GOSUB 1000
50 DIM rodz$(3)
60 rodz$(1)="POJEDYNCZE"
70 rodz$(2)="PODWOJNE"
80 rodz$(3)="POGRUBIONE"
85 'Wyswietlenie 3 zestawow znakow w petli
90 FOR nrz=1 TO 3
[illegible]
[illegible]
[illegible]
[illegible]
140 [illegible] PRINT"LINIE ";rodz$(nrz)
150 'Zawieszenie akcji programu
160 CLEAR INPUT:a$=INKEY$
170 IF a$=""GOTO 160
180 'Wyswietlenie tabeli
190 CLS
200 'Naglowek tabeli
210 LOCATE 1,1:PRINT CHR$(z(nrz,1));
220 FOR i=1 TO 3[illegible]:PRINT CHR$(z(nrz,4));:NEXT
225 LOCATE 14,1:PRINT CHR$(z(nrz,7))
230 LOCATE 25,1:PRINT CHR$(z(nrz,7)):LOCATE 34,1
230 PRINT CHR$(z(nrz,6)):PRINT CHR$(z(nrz,2));" El.tablicy ";
240 PRINT CHR$(z(nrz,2));" Nr.znaku ";
250 PRINT CHR$(z(nrz,2));" Postac ";CHR$(z(nrz,2))
260 PRINT CHR$(z(nrz,9));
270 FOR i=1 TO 32:PRINT CHR$(z(nrz,4));:NEXT
280 PRINT CHR$(z(nrz,10))
290 LOCATE 14,3:PRINT CHR$(z(nrz,11))
300 LOCATE 25,3:PRINT CHR$(z(nrz,11))
310 'Wyswietlanie w petli kolejnych wierszy tabeli
320 FOR w=0 TO 10
330 LOCATE 1,2*w+4:PRINT CHR$(z(nrz,2));" z(";
340 PRINT USING "#";nrz;:PRINT",";:USING "##";w+1;
350 PRINT")  ";CHR$(z(nrz,2));"  ";:USING "###";[illegible](nrz,w+1);
360 PRINT"  ";CHR$(z(nrz,2));"    ";CHR$(z(nrz,w+1));
370 PRINT"  ";CHR$(z(nrz,2))
380 PRINT CHR$(z(nrz,9));:FOR i=1 TO 32:PRINT CHR$(z(nrz,4));:NEXT
390 PRINT CHR$(z(nrz,10)):LOCATE 14,2*w+5:PRINT CHR$(z(nrz,11))
400 LOCATE 25,2*w+5:PRINT CHR$(z(nrz,11))
410 NEXT w
420 LOCATE 14,25:PRINT CHR$(z(nrz,8));:LOCATE 1,25:PRINT CHR$(z(nrz,3))
430 LOCATE 25,25:PRINT CHR$(z(nrz,8));:LOCATE 34,25:PRINT CHR$(z(nrz,5));
440 'Zawieszenie akcji programu
450 CLEAR INPUT:a$=INKEY$
460 IF a$="" GOTO 450
470 NEXT nrz
480 'Zakonczenie pracy
490 CLS:nrz=2:x=7:y=9:sze=29:wys=7:GOSUB 2000:x=8:y=10:sze=25:wys=3:GOSUB 2000
500 LOCATE 10,12:PRINT"Koniec pracy programu"
510 GOTO 510
```

Do tego programu należy dołączyć podprogram, który zrealizuje następujące czynności:

— definiowanie dwóch pierwszych zestawów linii:

a) symbole 234—244, to 1 zestaw znaków (linie pojedyncze);

b) symbole 245—255, to 2 zestaw znaków (linie podwójne).

— wprowadzenie numerów symboli wszystkich 3 zestawów do dwuwymiarowej tablicy Z (3,11).

Projektując tabele z wykorzystaniem prezentowanych 3 typów linii można postępować w dwojaki sposób:

— korzystać z elementów tablicy Z (i, j) gdzie:

i to nr zestawu (1—3);

j to nr znaku (1—11);

przykład: PRINT CHR$ (z (2,10)

Tak postępowano w programie pokazowym.

— bezpośrednio używać nr. symbolu; przykład: PRINT CHR$ (254)

Efekt uzyskany na ekranie w obu przykładach będzie oczywiście identyczny.

Wersja źródłowa tego podprogramu jest następująca:

```
1000 SYMBOL AFTER 234
1010 'Pierwszy zestaw znakow
1020 SYMBOL 234,0,0,0,31,16,16,16,16
1030 SYMBOL 235,16,16,16,16,16,16,16,16
1040 SYMBOL 236,16,16,16,31
1050 SYMBOL 237,0,0,0,255
1060 SYMBOL 238,16,16,16,240
1070 SYMBOL 239,0,0,0,240,16,16,16,16
1080 SYMBOL 240,0,0,0,255,16,16,16,16
1090 SYMBOL 241,16,16,16,255
1100 SYMBOL 242,16,16,16,31,16,16,16,16
1110 SYMBOL 243,16,16,16,240,16,16,16,16
1120 SYMBOL 244,16,16,16,255,16,16,16,16
1130 'Drugi zestaw znakow
1140 SYMBOL 245,0,0,63,32,47,40,40,40
1150 SYMBOL 246,40,40,40,40,40,40,40,40
1160 SYMBOL 247,40,40,47,32,63
1170 SYMBOL 248,0,0,255,0,255
1180 SYMBOL 249,40,40,232,8,248
1190 SYMBOL 250,0,0,248,8,232,40,40,40
1200 SYMBOL 251,0,0,255,0,239,40,40,40
1210 SYMBOL 252,40,40,239,0,255
1220 SYMBOL 253,40,40,47,32,47,40,40,40
1230 SYMBOL 254,40,40,232,8,232,40,40,40
1240 SYMBOL 255,40,40,239,0,239,40,40,40
1250 'Wprowadzanie numerow symboli do tablicy
1260 DIM z(3,11)
1320 FOR i=1 TO 2
1330 FOR j=1 TO 11
1340 z(i,j)=233+j+(i-1)*11
1350 NEXT j:NEXT i
1360 FOR j=1 TO 11
1370 READ z(3,j):NEXT j
1380 DATA 150,149,147,154,153,156,158,155,151,157,159
1390 RETURN
```

Dodatkowo chcemy zaproponować podprogram rysowania ramki wykorzystujący wcześniej zdefiniowane symbole graficzne. Jego wykorzystanie w programach użytkowych wymaga nadania przed wywołaniem wartości następującym parametrom:

NRZ — numer zestawu znaków (1—3);

X,Y — współrzędne znakowe lewego, górnego rogu ramki (dopuszczalne przedziały dla X są uzależnione od trybu, a dla Y = <1,25>);

SZE — szerokość ramki w znakach;

WYS — wysokość ramki w znakach.

Wersja źródłowa tego podprogramu jest następująca:

```
2000 'Podprogram rysowania ramki
2010 'Parametry wejsciowe
2020 '  nrz-nr zestawu znakow
2030 '  x,y-wspolrzedne lewego, gornego                rogu okna
2040 '  sze-szerokosc okna
2050 '  wys-wysokosc okna
2060 LOCATE x,y:PRINT CHR$(z(nrz,1));
2070 FOR p=1 TO sze-2
2080 PRINT CHR$(z(nrz,4));:NEXT p
2090 PRINT CHR$(z(nrz,6))
2100 FOR p=1 TO wys-2
2110 LOCATE x,y+p:PRINT CHR$(z(nrz,2))
2120 LOCATE x+sze-1,y+p:PRINT CHR$(z(nrz,2))
2130 NEXT p
2140 LOCATE x,y+wys-1:PRINT CHR$(z(nrz,3));
2150 FOR p=1 TO sze-2:PRINT CHR$(z(nrz,4));:NEXT p
2160 PRINT CHR$(z(nrz,5));
2170 RETURN
```

Oczywiście, proponowane symbole graficzne można także wykorzystać do innych celów — budowa schematów organizacyjnych, plansz czy chociażby labiryntów. Na zakończenie chcemy przedstawić 3 tabele utworzone tymi symbolami, a przedstawiające:

— element tablicy zawierający dany symbol;

— numer znaku;

— postać znaku.

El.tablicy	Nr.znaku	Postac
z(3, 1)	150	┏
z(3, 2)	149	┃
z(3, 3)	147	┗
z(3, 4)	154	━
z(3, 5)	153	┛
z(3, 6)	156	┓
z(3, 7)	158	┳
z(3, 8)	155	┻
z(3, 9)	151	┣
z(3,10)	157	┫
z(3,11)	159	╋

El.tablicy	Nr.znaku	Postac
z(2, 1)	245	╔
z(2, 2)	246	║
z(2, 3)	247	╚
z(2, 4)	248	═
z(2, 5)	249	╝
z(2, 6)	250	╗
z(2, 7)	251	╦
z(2, 8)	252	╩
z(2, 9)	253	╠
z(2,10)	254	╣
z(2,11)	255	╬

El.tablicy	Nr.znaku	Postac
z(1, 1)	234	┌
z(1, 2)	235	│
z(1, 3)	236	└
z(1, 4)	237	─
z(1, 5)	238	┘
z(1, 6)	239	┐
z(1, 7)	240	┬
z(1, 8)	241	┴
z(1, 9)	242	├
z(1,10)	243	┤
z(1,11)	244	┼

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

FIRST-COME FIRST-SERVED – pierwszy zgłoszony pierwszy obsłużony (w teorii kolejek),
FIRST-COME LAST-SERVED – pierwszy zgłoszony ostatni obsłużony (w teorii kolejek),
FIRST FIT – metoda pierwszego odpowiadającego (metoda podziału pamięci),
FIRST FIT ALGORITHM – algorytm dopasowywania,
FIRST FIT DECREASING ALGORITHM – algorytm dopasowania uporządkowanych zadań,
FIRST-IN FIRST-OUT – pierwszy na wejściu pierwszy na wyjściu (w teorii kolejek),
FIRST-IN LAST-OUT – pierwszy na wejściu, ostatni na wyjściu (w teorii kolejek),
FIRST-IN-NOT-USED-FIRST-OUT – metoda wymiany stronic (segmentów) pamięci w systemach z pamięcią wirtualną, polegająca na usuwaniu stronic do których przez długi czas nie było odwołań,
FIRST-ITEM – pierwsza pozycja,
FIRST-LEVEL ADDRESS – adres pierwszego poziomu,
FIRST LEVEL INDEX TABLE – tablica indeksów pierwszego stopnia,
FIRST LEVEL OVERFLOW BUCKET – porcja (blok programowy) w obszarze przepełnień pierwszego stopnia,
FIRST LEVEL OVERFLOW ROUTINES – procedury przepełnień pierwszego poziomu,
FIRST-ORDER LOGIC – logika pierwszego rzędu,
FIRST-ORDER SUBROUTINE – podprogram pierwszego rzędu,
FIRST RULES OF ARITHMETIC – działania arytmetyczne podstawowe,
FITNESS FOR USE – zdatność użytkowa,
FITTING – dopasowanie, montaż,
FIT TOGETHER – pasować do siebie,
FIVE-LEVEL CODE – kod pięciościeżkowy,
FIX – poprawianie, sprostowanie, ustalanie, także: poprawiać, sprostować, ustalać, umacniać,
FIXED – stały, ustalony, nieruchomy,
FIXED ADDRESS – adres stały,
FIXED AREA – obszar ustalony,
FIXED DISK – dysk stały, dysk wymienny,
FIXED HEAD DISC – dysk z nieruchomą głowicą,
FIXED INFORMATION LENGTH – stała długość informacji,
FIXED INSTRUCTION LENGTH – stała długość rozkazu,
FIXED-LENGTH ADDRESSING – adresacja ze stałą długością adresu (dot. sposobu adresacji w sieci komputerowej),
FIXED-LENGTH COMPUTER – komputer ze słowem o stałej długości,
FIXED-LENGTH FIELD – pole stałej długości,
FIXED-LENGTH RECORDS – zapisy (rekordy) o stałej długości,
FIXED LINE – łącze stałe,
FIXED POINT – przecinek stały,
FIXED-POINT ARITHMETIC – arytmetyka stałopozycyjna, arytmetyka stałoprzecinkowa,
FIXED-POINT CALCULATION – rachunek stałoprzecinkowy,
FIXED-POINT INSTRUCTION – rozkaz stałoprzecinkowy,
FIXED-POINT NOTATION – zapis stałoprzecinkowy,
FIXED-POINT NUMBER – liczba stałoprzecinkowa,
FIXED-POINT OPERATION – operacja stałoprzecinkowa,
FIXED-POINT REPRESENTATION – przedstawienie liczb stałoprzecinkowych,
FIXED-PROGRAM COMPUTER – komputer o stałym programie (mający na stałe, ustaloną kolejność wykonywanych rozkazów),
FIXED RADIX NUMERATION SYSTEM – system liczbowy o stałej podstawie,
FIXED RETENTION – stałe członkostwo, ustalone członkostwo,
FIXED ROUTING – ustalona trasa (droga),
FIXED STORAGE – pamięć stała, pamięć wielkości stałych, pamięć martwa,
FIXED WIRE – sznur do łączenia na stałe, tablicy połączeń,
FIXED WORD LENGHT – ustalona długość słowa,
FIXING – ustalanie, utrwalanie,
FLAD BED PLOTTER – ploter stołowy (płaski),
FLAG – znacznik, flaga, sygnalizator, znak sygnalizujący, także: bit oznaczający granice pola,
FLAG BIT – znacznik,
FLAG FIELD – znacznik pola,
FLAG PAGE – strona tytułowa,
FLAG REGISTER – rejestr flagowy,
FLAG SEQUENCE – znacznik transmisji oddzielający obrazy przesyłane na monitor w sieci komputerowej, także: znacznik wielobitowy, ramka,
FLASHING – podświetlanie szablonu (jako tła obrazu),
FLASH PHOTOGRAPHIC PRINTER – drukarka fotograficzna błyskowa,
FLAT – płaski, także: spłaszczać, równać,
FLAT ADDRESSING – adresacja prosta,
FLATBED PLOTTER – pisak planszetowy, ploter planszetowy,
FLAT FILE – plik (zbiór) składający się z zapisów (rekordów) tego samego typu,
FLATPACK – płaska obudowa typu krab (mikroukładu),
FLAT SUBASSEMBLY – pakiet,
FLATTEN – równać, spłaszczać,
FLEXIBILITY – giętkość, podatność, elastyczność,
FLEXIBLE – giętki, podatny, elastyczny,
FLEXIBLE ARRAY – tablica o zmiennym rozmiarze,
FLEXION – druga pochodna,
FLEXOWRITER – kodopis,
FLEX POINT – punkt przegięcia,
FLICKER – niepożądane migotanie, miganie, mruganie,
FLICKER NOISE – szum migotania,
FLIGHT PATH COMPUTER – przelicznik toru lotu,
FLIGTH COMPUTER – komputer do celów nawigacji (lotniczej lub kosmicznej),
FLIMSY – rozdzielająca karta dziurkowana,
FLIP – przerzucać (z jednego położenia w drugie),
FLIP-FLOP – przerzutnik, multiwibrator bistabilny,
FLIP OFF – wyłączyć, zgasić (np. światło),
FLIP ON – włączyć, zapalić (np. światło),
FLIPPY DISK – dyskietka dwustronna z niezależnie używanymi stronami,
FLOAT – płynny,
FLOATING – wahający się, zmienny, pływający,
FLOATING ADDRESS – adres symboliczny, adres pływający, adres zmienny,
FLOATING GATE – pływająca bramka,
FLOATING POINT – zmienny przecinek,
FLOATING-POINT ARITHMETIC – arytmetyka zmiennopozycyjna, arytmetyka zmiennoprzecinkowa,
FLOATING-POINT BAR – podstawa liczenia ze zmiennym przecinkiem,
FLOATING-POINT CALCULATION – rachunek zmiennoprzecinkowy,
FLOATING POINT CHARACTERISTIC – cecha liczby zmiennoprzecinkowej,
FLOATING POINT COMPUTATION – rachunek zmiennoprzecinkowy,
FLOATING-POINT INSTRUCTION – rozkaz zmiennoprzecinkowy,
FLOATING-POINT NOTATION – zapis zmiennoprzecinkowy,
FLOATING-POINT NUMBER – liczba zmiennoprzecinkowa,
FLOATING-POINT OPERATION – operacja zmiennoprzecinkowa,
FLOATING-POINT OPERATIONS PER SECOND – liczba operacji zmiennoprzecinkowych na sekundę (określenie szybkości działania procesora),
FLOATING-POINT PACKAGE – program zmiennoprzecinkowy, program zmiennopozycyjny,
FLOATING POINT REPRESENTATION – przedstawienie liczb zmiennoprzecinkowych,
FLOATING-POINT WORD – słowo zmiennoprzecinkowe,
FLOODING – kaskadowy wybór trasy, także: rozpływowy,
FLOPOVER – przesuwanie się obrazu w pionie (na ekranie monitora),
FLOPPY-BASED – urządzenie pamięciowe z dyskami elastycznymi,
FLOPPY DISK – dysk elastyczny, minidysk, dyskietka,
FLOPPY DISK DRIVE – napęd dyskietkowy,
FLOPS – patrz: FLOATING-POINT OPERATIONS PER SECOND,
FLOW – przepływ, także: sterowanie,
FLOW AUGMENTING PATH – droga powiększająca przepływ,
FLOW CHART – schemat czynności, diagram przebiegu, sieć działań, schemat działania programu, schemat blokowy programu,
FLOWCHARTER – program sporządzający schemat blokowy,
FLOWCHARTING – sporządzanie schematu blokowego,
FLOW CHART SYMBOL – element schematu blokowego,
FLOWCHART TEMPLATE – szablon do projektowania,
FLOW CONTROL – sterowanie strumieniem danych, sterowanie przepływem (w sieciach komputerowych),
FLOW DIAGRAM – schemat działania programu, schemat blokowy programu, sieć działań,
FLOW DIRECTION – kierunek przebiegu (przepływu) na schemacie blokowym,
FLOWLINE – linia obiegu, linia łącząca dwa symbole na schemacie blokowym,
FLOW OF CURRENT – przepływ prądu elektrycznego,
FLOW OF INFORMATION – strumień informacji,
FLOW SHOP PROBLEM – przepływowy problem obsługi,
FLOW TIME – czas przepływu (dot. przebywania zadania w systemie obsługi),
FLUCTUATION – wariacja,
FLUID – płyn, także: płynny,
FLUID COMPUTER – komputer na elementach strumieniowych,
FLUTTER – drżenie dźwięku,
FLUX – strumień,

FORMAL LOGIC - logika matematyczna, logika symboliczna, logistyka,
FORMAL PARAMETER - parametr formalny,
FORMAL SPECIFICATION - opis formalny,
FORMAT - format (sposób przedstawienia danych na nośniku),
FORMAT BUFFER - pamięć formatu formularza,
FORMAT CHARACTER - znak sterujący formatem,
FORMAT EFFECTOR - formatyzator,
FORMAT ERROR - błąd formalny, błąd formatu,
FORMATION - tworzenie się, powstawanie,
FORMAT LABEL - etykieta formatu,
FORMATLESS - bezformatowy,
FORMATTED - formatowany, sformatowany,
FORMATTED CAPACITY - pojemność nośnika danych, pojemność użytkowa,
FORMATTER - program edycji tekstów tworzący formaty,
FORMATTING - formatowanie (rozplanowywanie tekstów na ekranie monitora),
FORM FEED - przesunięcie strony, przesuw formularza, nowa strona,
FORMFEED - patrz: FORM FEED,
FORM FEED CHARACTER - znak zasilenia (drukarki) formularzem, także: znak przesuwu strony,
FORM FEEDING - przesuw formularza,
FORMING - kształtowanie, formowanie, tworzenie,
FORM OVERLAY - szablon (tabela, siatka lub mapa stanowiąca element tła obrazu),
FORM SET - zestaw formularzy,
FORMS FLASH - podświetlanie szablonu, (jako tła obrazu),
FORMULA - wzór, wyrażenie,
FORMULA LANGUAGE - język formuł,
FORMULATION - sformułowanie, wyrażenie za pomocą wzoru,
FORMULA TRANSLATION - patrz: FORTRAN,
FOR TECHNICAL PURPOSES - do celów technicznych,
FORTH - wysokopoziomowy język programowania,
FOR THE TIME BEING - na pewien czas, tymczasowo,
FOR THIS PURPOSE - w tym celu,
FORTRAN - język proceduralny do zastosowań naukowych i technicznych,
FORWARD - nadawać, transmitować, komunikować,
FORWARD CHAINING - także: wnioskowanie uprzedzające (w systemach ekspertowych),
FORWARD CHANNEL - kanał docelowy,
FORWARD-COMPATIBLE - zgodny, kompatybilny (o programie, który może pracować na innym typie komputera),
FORWARD DECLARATION - wstępny (częściowy) opis obiektu programu,
FORWARD DIRECTION - kierunek przewodzenia,
FORWARDER - urządzenie przesyłania danych (w sieci),
FORWARDING - przesyłanie danych (do miejsca wykorzystania),
FORWARD REFERENCE - odsyłacz, odnośnik użytkowany wstępnie,
FORWARDS - do przodu,
FORWARD SPACING - przesunięcie do przodu,
FORWARD SUPERVISION - sterowanie wprost,
FOUNT - patrz: FONT,
FOUR-ADDRESS - czteroadresowy,
FOUR-ADDRESS INSTRUCTION - rozkaz czteroadresowy,
FOURTH-GENERATION COMPUTER - komputer czwartej generacji,
FOUR-WIRE COMMUNICATION - przesyłanie czteroprzewodowe,
FOUR-WIRE LINE - łącze czteroprzewodowe,
FOX MESSAGE - standardowy komunikat kontroli poprawności transmisji, zawierający wszystkie litery alfabetu łacińskiego (postać komunikatu stanowi zdanie: "A quick brown fox jumps over the lazy dog 123456789 0"),
FPLA - patrz: FIELD-PROGRAMMABLE LOGIC ARRAY,
FPROM - patrz: FIELD-PROGRAMMABLE ROM,
FRACTION - ułamek, także: frakcja, część,
FRACTIONAL - ułamkowy, także: frakcyjny, częściowy,
FRACTIONAL DIGIT - cyfra po przecinku,
FRACTIONAL EXPONENT - wykładnik ułamkowy,
FRACTIONAL NUMBER - liczba ułamkowa,
FRACTIONAL PART - mantysa,
FRACTION BAR - kreska ułamkowa,
FRAGMENT - fragment, odłamek, także: fragmentować,
FRAGMENTARY - częściowy, fragmentaryczny,
FRAGMENTATION - fragmentacja, rozdrabnianie,
FRAME - 1. grupa (blok) danych, 2. rządek bitów położonych w poprzek taśmy, 3.system odczytu, 4. kadr (w sieciach - przesyłana porcja danych), 5. karkas, korpus, ramka,
FRAME BUFFER - bufor obrazu, bufor ramki,
FRAME CHECK SEQUENCE - ciąg kontrolny ramki,
FRAME DELETION - skasowanie obrazu, skasowanie ramki (fragmentu lub całego ekranu monitora),
FRAME GRABBER - urządzenie wprowadzania i zapisywania obrazów (fragmentu lub całego ekranu monitora),

FLUXING - upłynnianie (nadawanie płynności), także: uplastycznianie, zmiękczanie,
FLX - patrz: FILE EXCHANGE UTILITY,
FLYBACK - szybki ruch powrotny, powrót,
FLYING HEAD - głowica magnetyczna zapisu/odczytu nie dotykająca nośnika danych lecz unosząca się nad nim na cienkiej warstwie powietrza,
FLYING LEAD - cienki drut, łączący pole kontaktowe z nóżką obudowy np. tranzystora, układu scalonego,
FLYING SPOT - ruchoma plamka świetlna,
FLYING SPOT SCAN - skaning (system kodowania obrazu za pomocą techniki omiatania rastrowego, uwzględniający jaskrawość obrazu),
FLYING-SPOT SCANNING - wybieranie ruchomą plamką świetlną,
FM - patrz: FREQUENCY MODULATION,
FM-RECORDING - patrz: FREQUENCY MODULATION RECORDING,
FMT - patrz: FORMAT,
FOCAL POINTS - punkty skupiające,
FOLD - składać (np. dane w celu zwartego przedstawienia),
FOLDABILITY - podatność na zginanie,
FOLD DISTANCE - odstęp zagięcia,
FOLDER - określenie kartoteki plików (zbiorów) tekstowych w systemach,
FOLDING - składany,
FOLIATE - rozwarstwiać się,
FOLLOWED BY ... - także: zakończone ...
FOLLOWER - powtarzacz, także: człon bierny, człon napędzany,
FOLLOWING - kolejny, następny,
FOLLOW INSTRUCTIONS - postępować według wskazówek,
FOLLOW UP - nadążać,
FOLLOW-UP - dalsza obróbka, dalsze opracowanie,
FOLLOW-UP FILE - plik uporządkowany według klucza, inicjowanego określonym zdarzeniem w systemie,
FONT - zbiór znaków graficznych o określonym kroju, krój pisma,
FONT-CHANGE CHARACTER - znak zmiany kroju czcionki,
FONT RECTICLE - miejsce na papierze lub ekranie, na jeden znak,
FOOLPROOF - bezpieczny przy nieostrożnym lub nieumiejętnym obchodzeniu się, także: zabezpieczać urządzenie przed skutkami nieostrożnego lub nieumiejętnego obchodzenia się,
FOOL-TOLERANCE - odporność (wytrzymałość) na nieumiejętne obchodzenie się,
FOOTAGE COUNTER - licznik długości taśmy,
FOOTER - stopka (opis powtarzający się na dole każdej strony tekstu),
FOOTING - zsumowanie,
FORBIDDEN - zakazany, niedopuszczalny, wzbroniony,
FORBIDDEN TERM - askryptor, nondeskryptor, termin zakazany,
FORCE - siła, także: wciskać, wtłaczać, wymuszać,
FORCED - wymuszony,
FORCE OUT - wypychać,
FORCING INPUT - wejście przełączające (przerzutnika)
FORECAST - prognoza,
FORECAST DEMAND - zapotrzebowanie przewidywane,
FORECASTING - prognozowanie, przewidywanie, spekulacja,
FOREDESIGN - projekt wstępny,

FOREGROUND-BACKGROUND SCHEDULING - szeregowanie wielopoziomowe,
FOREGROUND COLOR - kolor przedniego planu obrazu,
FOREGROUND IMAGE - przedni plan obrazu,
FOREGROUNDING - wykonywanie zadań o wyższym priorytecie, obsługa priorytetowa,
FOREGROUND MUSIC - muzyka pierwszoplanowa,
FOREGROUND (PRIORITY) - o wyższym piorytecie, pierwszoplanowy,
FOREGROUND PROCESSING - przetwarzanie pierwszoplanowe,
FOREGROUND PROGRAM - program pierwszoplanowy,
FOREIGN KEY - klucz zewnętrzny,
FOREIGN MATTER - obce ciało,
FORESEE - przewidywać,
FORESEEABLE - przewidywalny, możliwy do przewidzenia, dający się przewidzieć,
FOREST - las, wielka liczba drzew (w teorii grafów),
FORGETFULLNESS - brak działania następczego, wtórnego (w teorii masowej obsługi),
FORK - rozgałęzienie, rozwidlenie,
FORM - formularz, druk, sformatować, także: druk, formularz,
FORMAL - formalny,
FORMALIZATION - formalizacja (opisu, problemu),
FORMALIZE - formalizować,
FORMAL LANGUAGE - język formalny,

LIGA MYŚLĄCYCH

ZADANIE 1

Od dwóch kawałków stopu, o różnej zawartości procentowej miedzi, ważących m kilogramów i n kilogramów, odcięto jednakowe wagowo kawałki. Każdy z odciętych kawałków stopiono z pozostałą resztą drugiego kawałka stopu i okazało się, że procentowa zawartość miedzi w otrzymanych stopach jest jednakowa. Ile ważył każdy z odciętych kawałków?

ZADANIE 2

Łódź motorowa płynąc z prądem rzeki przebywa odległość między dwoma miastami w czasie a = 4 h, a płynąc pod prąd odległość tę przebywa w czasie b = 12 h. W jakim czasie belka niesiona prądem rzeki przebędzie tę odległość?

ZADANIE 3

Zboże zawiera k % zanieczyszczeń. Po każdorazowym oczyszczeniu ilość zanieczyszczeń zmniejsza się o q %, przy czym ilość czystego ziarna nie ulega zmianie. Ile razy trzeba powtórzyć oczyszczenie zboża, aby jego zanieczyszczenie zmniejszyło się do f % (f L q)?

ZADANIE 4

Dziesięć litrów wody o temperaturze 13°C zmieszano z 12,5 litrami wody o takiej temperaturze, że po ustaleniu się temperatury mieszanina miała temperaturę wyższą od 25°C. Lecz mniejszą od 30°C. Jaką temperaturę miała woda dolana?

ZADANIE 5

Mamy 24 gramy radioaktywnego pierwiastka o czasie połowicznego rozpadu T_1 = 7 lat i 3 gramy innego radioaktywnego pierwiastka. Jaki jest czas połowicznego rozpadu T_2 drugiego pierwiastka, jeżeli po 21 latach ilości obu pierwiastków są równe?

Rozwiązania zadań prosimy przesyłać do redakcji do końca listopada br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

Tylko dla informatyków

JĘZYK D

Zdecydowana większość oprogramowania napisana została indywidualnie na określone typy komputerów. Natomiast niewiele jest programów, które pracują na wszystkich komputerach.

Obecnie oprogramowanie tworzone jest głównie na komputery „marki" VAX. Jest to zapewne przyszłościowy komputer, w każdym razie w naszych warunkach. Stanowi on ogniwo pośrednie między sprzętem PC, a superkomputerami. Stąd też często nazywa się ten sprzęt — minikomputery, cena wywoławcza (umowna?) około 100 (stu) milionów zł.

Wśród 25 pakietów przetwarzania tekstów tylko LEX (stworzony przez Ace Microsystem) pracuje na wszystkich VAXach, PDP-11 i systemach Rainbow'a. Stało się to możliwe dzięki temu, iż LEX napisany jest w języku D.

LEX był pierwszym zintegrowanym systemem przetwarzania tekstów z jednoczesną obsługą bazy danych. Oryginalnie (w 1979 roku) napisano go na PDP-11. Komputer ten święcił wówczas swe najlepsze dni.

LEX napisany został w języku RPL. Język ten miał pewne ograniczenia, które dosyć szybko dały o sobie znać. Wynikało to z tego, iż kompilator RPLa został napisany w MACRO asemblerze. Decydowało to o tym, że tworzone oprogramowanie mogło wyłącznie pracować na PDP-11. By pokonać ten problem stworzono właśnie język D.

D jest językiem unikalnym, opartym o elementy, które jego twórcy (bardzo młodzi — D. Medford i T. Barnard) nazywają — „tabele decyzyjne". Język ten rozpowszechniany jest przez National Computing Centre jako Filetab-B.

Języki: RPL i D mają podobną syntaktykę, która pozwala przekładać programy z RPLa na D.

Wydaje się, że najważniejsza jest ta możliwość kompilatora języka D, która polega na tym, iż tworzy on kod maszynowy dla różnych (wybranych) maszyn.

Należy jednak pamiętać, że D nie jest narzędziem na zmianę istniejącego oprogramowania napisanego na PDP-11 na programy pracujące na VAXie (za wyjątkiem gdy użytkownik napisał swoje oprogramowanie na PDP-11 w jednym z języków: D lub RPL).

Omawiana własność języka D stwarza całkiem nowe warunki dla nowego oprogramowania, które bez żadnych zmian może pracować nie tylko na PDP, VAXie, ale na wielu innych komputerach. Podstawową zaletą języka D okazuje się właśnie to, że użytkownik (posługujący się nim) nie jest skazany na PDP lub VAXa. Ilustracją tego może być przykład przeniesienia programu źródłowego z RIADa na ODRĘ...

Program w języku D składa się ze zbioru tabel decyzyjnych. Zawierają one szereg warunków tworzących kolejne „strony" programu.

Jak już zostało wspomniane, język D był projektowany z zamiarem stworzenia możliwości jego implementacji na różne komputery.

Zbiór dostępnych komend i operatorów oraz odpowiadające im kody maszynowe, które muszą być generowane, są przechowywane jako modyfikowalny zbiór danych — TMC (wybrany kod maszynowy) — zbiór ten wykorzystywany jest przez kompilator.

TMC tworzony jest ze zbioru danych ASCII przez program nazywany D Compiler Compiler. Zbiór ten nazywany jest CGT. Liczba zbiorów tego typu jest równa liczbie procesorów, na które język D może być implementowany.

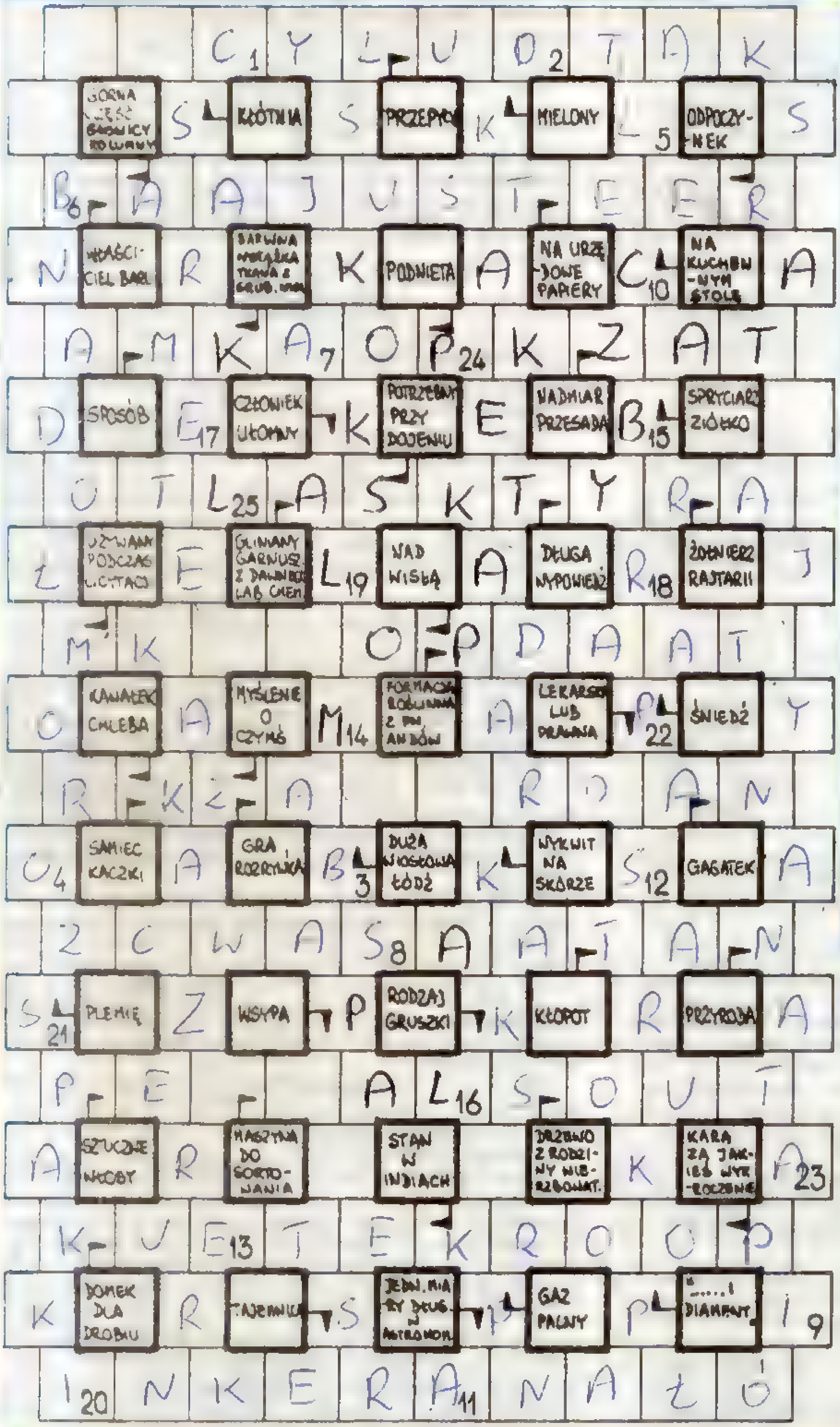
Specyfikacja zbiorów CGT przewiduje wymagania odpowiedniego procesora — jego architekturę, specjalistyczne zadania itp. Zatem, gdy na rynku pojawia się nowy procesor, tworzony jest odpowiedni zbiór CGT. Następnie używając Compiler Compiler powstaje zbiór TMC — kompilator języka D na nowy procesor jest gotów. Możliwe staje się tworzenie kodu maszynowego dla komputera z nowym typem procesora, przy użyciu do prac programowych oczywiście języka D.

Należy równocześnie dokonać pewnych zabiegów przystosowujących eksploatowany system operacyjny do programów napisanych w języku D. Obecnie dostępne są wersje LEXa na 28 różnych systemów operacyjnych.

Język D okazał się bardzo użyteczny, także na popularny u nas sprzęt mikrokomputerowy oraz na minikomputery (od PDP, VAXa do sprzętu budowanego na Motoroli 68000 i Intelu 8086). Wydaje się, że język D staje się bardziej użyteczny, niż jakikolwiek inny, taki jak na przykład język C.

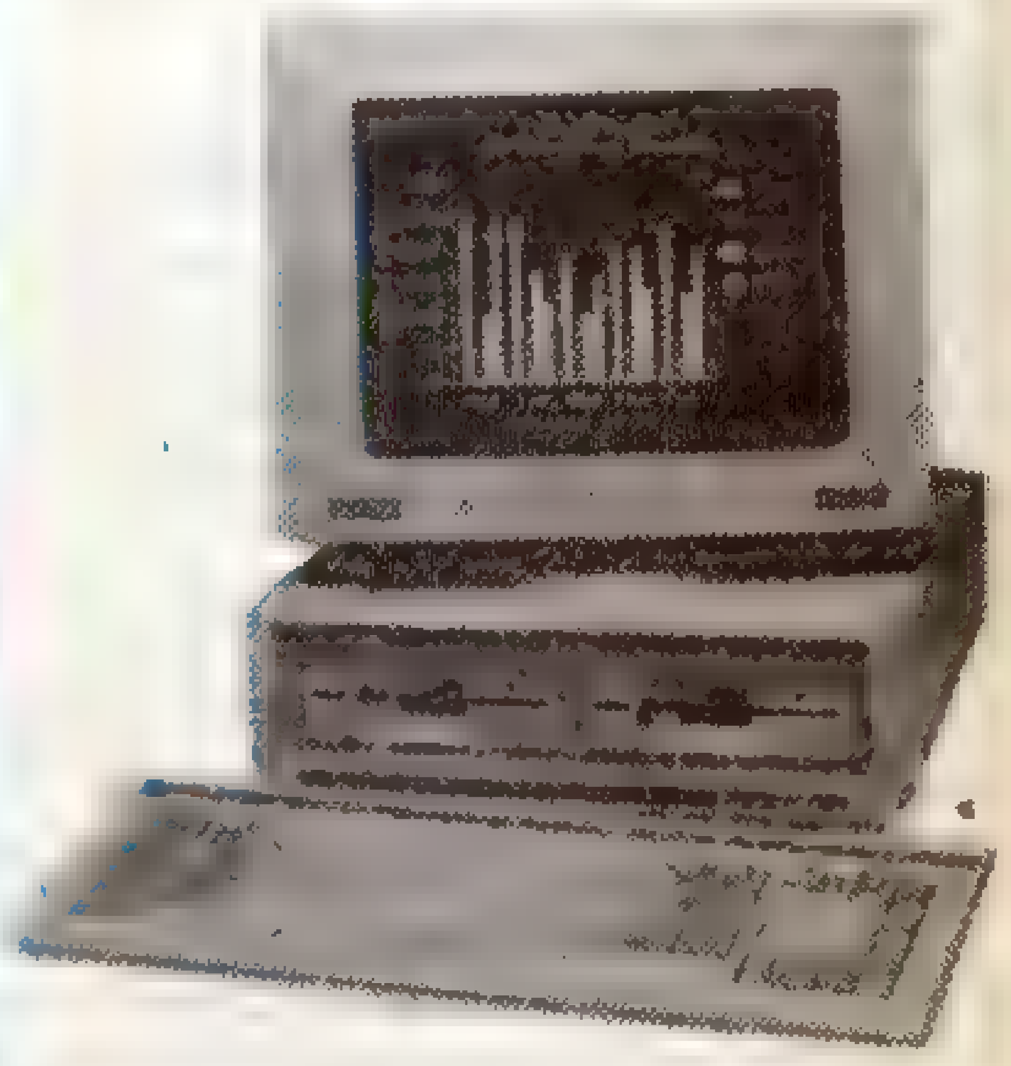
Opracowane na podstawie:
D. Medford, A Language Created for Riding the Range, dec/computing, 20 stycznia 1988.

KRZYŻÓWKA NR 7



AMSTRAD

PC 1640 CD



CPC – grafika

```
10 REM ******** GRAFIKA **********
20 DEFINT b,c,x
30 MODE 2
40 b=RND*5 + 1:c=RND*5 + 1
50 ORIGIN 320,200
60 FOR a=0 TO 1000 STEP PI/30
70 x=100*COS(a):MOVE x,x
80 DRAW 200*COS(a/b),200*SIN(a/c)
90 IF INKEY$<>"" THEN 30
100 NEXT: GOTO 30
```

Litery z ponumerowanych kratek od 1 do 25 utworzą 5 terminów związanych z komputerami, które wystarczy podać jako rozwiązanie zadania przesyłając pod adresem redakcji na kartach pocztowych w terminie do końca listopada naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

I.K.S.

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1775. Nr ind. 361682. U-21

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 ZŁ

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" NR 10/1988 ISSN 0860 — 2794

Informatyka w szkole - 16



SF 27



4

prenumerata tylko do 10 listopada

W NUMERZE:

W szponach Atari (18) — str. 5

Gra w tysiąca — str. 11

Rady i porady programisty (cz. 2) — str. 13

Schowek — str. 14

Informatyka w szkole — str. 16

Projektant — znaki — str. 20

Programowanie nie jest trudne, czyli MINI-GEN — str. 22

Dwa głosy na Spectrum — str. 22

Wydruk ekranu — str. 23

Lilavati (8) — str. 24

POLSCRIPT — edytor tekstu z polskimi literami dla C-64 — str. 25

SF: Zagadka — str. 27

Giełda Pomysłów — str. 28

Słownik informatyczny — str. 29

Liga Myślących — str. 31

Trałowanie — str. 31

Grafika komputerowa — str. 32

Komputerowa rewolucja

— Wkrótce uzyskamy możliwość rozmowy z komputerami, później zaś zbudujemy człekopodobne komputery z inteligencją nadczłowieka — oświadczył Clive Sinclair. To, co się dzieje w tej dziedzinie, potwierdza tę opinię. Niemal każdy dzień przynosi dowody na błyskawiczny rozwój komputeryzacji.

I tak amerykańska firma „Cray" wprowadziła na rynek urządzenie oznaczone symbolem Y-MP. Ten superkomputer kosztujący 20 mln dolarów może przeprowadzić 2 do 4 mld operacji arytmetycznych w ciągu sekundy! To cudo techniki skonstruował Steven Chen, który nie pracuje już w „Cray'u", a tworzy obecnie w International Business Machines.

Z kolei ów koncern elektroniczny ogłosił niedawno, że jego konstruktorzy dokonali nowego przełomu, budując chip — podstawowy element pamięci komputerowej — zdolny do przyjęcia bitu informacji w ciągu 20 miliardowych części sekundy, czyli trzykrotnie szybszy od chipów stosowanych dotychczas. IBM wprowadza też w swych wyrobach nową architekturę budowy komputerów pozwalającą znacznie zwiększyć zasób informacji gromadzonych w pamięci maszyny oraz ułatwiający dostęp do nich. Taka konstrukcja daje możliwości około 8 tys. razy większe niż dotychczasowe systemy rozszerzonej pamięci, stosowane w komputerach IBM. Umożliwi to użytkownikom dużych komputerów dostęp do prawie 16 bilionów jednostek informacyjnych. Tego rodzaju konstrukcje, o czym są przekonani specjaliści, odegrają decydującą rolę w informatyce lat dziewięćdziesiątych.

Tymczasem także w Japonii trwają intensywne badania nad komputerem nowej generacji. Maszyna ta będzie mogła rozmawiać i myśleć podobnie jak człowiek, czyli podejmować samodzielne logiczne decyzje na podstawie posiadanych danych, a nie jak obecnie pracować tylko zgodnie z wprowadzonymi programami.

Opracowany został już w tym kraju typ komputera optycznego, który jest w stanie uczyć się, jak odróżniać obrazy. Jego budowa oparta jest na elektronicznych obwodach imitujących strukturę neuronów ludzkiego mózgu. W rezultacie ten „neuro-komputer" może sam siebie zaprogramować ucząc się rozpoznawania znaków. Jest to na razie konstrukcja w początkowym stadium rozwoju. Komputer w trakcie uczenia się odtwarza znaki czy kształty w sposób niedokładny, co wynika z niedokładnej pamięci komputera. Jednakże po przeanalizowaniu różnic między znakami „na wejściu" i „na wyjściu" koryguje on swą pamięć i odwzorowuje właściwe znaki, nawet jeśli są one trochę zniekształcone. Takie kojarzenie oznacza, że w przyszłości komputery optyczne będą czytały znaki pisane ręcznie.

W niezwykle szybko rozwijającej się komputeryzacji nie zamierzają próżnować specjaliści RWPG. Rozwiązali oni główne problemy techniczne i technologiczne budowy komputera, który potrafi przeprowadzić do miliarda operacji na sekundę. Obecnie w dalszych badaniach najważniejszą sprawą jest zapewnienie niezawodności środków techniki obliczeniowej i jakości ich obsługi technicznej.

(RAJ)

POMOC — AMSTRAD 6128

Jeżeli chcesz usprawnić proces uruchamiania własnych programów i ustrzec się przed nieprzewidzianymi skutkami jego działania, skorzystaj z POMOCY.

Program umożliwia, w dowolnej chwili, skopiowanie 64 k bajtów pamięci do drugiego bloku tej samej wielkości. Po tej operacji można wprowadzić do testowanego programu dowolne zmiany. Jeżeli uznasz, że są niepoprawne, to możesz odzyskać poprzednią postać programu.

POMOC działa poprawnie tylko na Amstradzie 6128 lub 464/664 z rozszerzoną pamięcią pod warunkiem, że drugi blok 64 k bajtów, nie podłączony aktualnie do szyny adresowej, nie będzie przez program wykorzystywany.

Wprowadź do komputera zamieszczony program. Jeśli nie będzie w nim błędów, to kod maszynowy zostanie automatycznie nagrany na dyskietce.

W celu skopiowania pamięci należy (w dowolnej chwili) wgrać POMOC. BIN i wykonać CALL&BFOO, natomiast aby ją odzyskać — CTRL—SHIFT, CALL&BFOO, 1.

Kod maszynowy został umieszczony od adresu &BFOO, ponieważ obszar &BFOO—&BFFF jest stosem i podczas zerowania pamięci przez CTRL—SHIFT—ESC nie jest niszczony.

```
10 'POMOC - AMSTRAD 6128 !
20 'ZACHOWANIE ZAWARTOSCI PAMIECI
30 '
40 adr=&BF00: ln=200
50 FOR i=0 TO 5
60 sum=0: READ a$,d
70 FOR j=1 TO LEN(a$) STEP 2
80 bajt=VAL("&"+MID$(a$,j,2))
90 POKE adr,bajt:sum=sum+bajt
100 adr=adr+1
110 NEXT
120 IF sum<>d THEN PRINT "Zla dana w linii ";ln:END
130 ln=ln+10
140 NEXT
150 SAVE "!POMOC.BIN",b,&BF00,&5E
160 PRINT "Przeslanie pamieci: CALL &BF00"
170 PRINT:PRINT "Odzyskanie pamieci: CALL &BF00,1":PRINT
200 DATA A72006214ABFCD1CBF2154BFCD1CBF3E,&6B9
210 DATA C001007FED79CD11BCC31CBD0605C5E5,&791
220 DATA CD2ABFE12323C110F5C97EE60FC6C001,&866
230 DATA 007FED797EE6F057237EE60FC63F477E,&7F0
240 DATA E6F0670E00596BEDB0C947C146804401,&788
250 DATA C04145C1C54140C1C741864004410000,&621
```

Rafał FOLTYNIEWICZ

Konkurs rozstrzygnięty

Do ogłoszonego na łamach „Żołnierza Wolności" i „IKS-a" konkursu na scenariusz programu komputerowego, którego tematem była historia i tradycje ludowego Wojska Polskiego zgłoszono kilkadziesiąt prac. Zadanie, jak się okazało było trudne, wymagało bowiem nie tylko inwencji twórczej, ale także umiejętności nadania scenariuszowi takiej formy, która pozwoliłaby na jego podstawie napisać program komputerowy.

Jury nie miało łatwego zadania. Spośród ocenianych prac za najlepszy uznano scenariusz zatytułowany „Ucieczka Orła". Autorem tej pracy i zarazem zdobywcą pierwszego miejsca jest płk dr inż. Andrzej Grzesiak.

Drugie miejsce zajął zespół programistów z Wyższej Oficerskiej Szkoły Wojsk Pancernych w Poznaniu w składzie: Danuta Kwasiżur, kpt. Mieczysław Skonieczny i Robert Zugehort, za zbiór scenariuszy i pakiet programów przeznaczony na mikrokomputer Amstrad CPC 6128.



Redaktor naczelny „Żołnierza Wolności" płk Zdzisław Janoś wręcza nagrodę (mikrokomputer TIMEX) zwycięzcy.

INFOGRYF '88

W zasadzie zrezygnowałem z planowanego wyjazdu na Infogryf'88 — koszt uczestnictwa w tej konferencji okazał się bowiem zbyt duży. Szczęśliwie dyrektor okazał się jednak hojny i opłata w wysokości około 55 tys. zł została dokonana.

Tradycyjnie już otwarcia Infogryfu dokonał prof. dr hab. **Tadeusz Wierzbicki**. Stwierdził, iż jest to dziewiąte spotkanie informatyków na tego rodzaju imprezie. Liczba uczestników wynosiła około 300 osób (stosunkowo mało — cena!) w tym goście z Bułgarii, Francji, NRD, RFN i ZSRR.

Na konferencję nadesłano 115 referatów, które zostały opublikowane w czterech tomach. Każda sekcja to oddzielny tom. Są tam referaty „perełki" (mało) i referaty „nie perełki" (dużo), w sumie zamiast zadrukowania 239+297+269+215=1020 stron można by „całkiem spokojnie" zużyć 100 może 105 kartek (np.: streszczając część materiałów). Gwoli ścisłości należy wymienić wyróżnione sekcje/tytuły tomów:

— sekcja pierwsza: Informatyka w zarządzaniu przedsiębiorstwem (nowe generacje systemów), Systemy Informatyczne zarządzania (struktury — funkcjonowanie — rozwój),

— sekcja druga: Informatyka w zarządzaniu przedsiębiorstwem (nowe generacje systemów), Systemy wspomagania decyzji (istota, zasady, możliwości zastosowań),

— sekcja trzecia: Informatyka w zarządzaniu przedsiębiorstwem (nowe generacje systemów), Metody tworzenia i rozwoju systemów (stan i perspektywy rozwoju — praktyczne zastosowanie),

— sekcja czwarta: Mikroskala'88 (wersje i kontrowersje zastosowań mikrokomputerów), Informatyka w zarządzaniu przedsiębiorstwem (nowe generacje systemów).

Prawda, że z tych tytułów „emanuje" precyzja informatyczna?

Referaty uzyskano z tak zwanego wolnego naboru, to znaczy wynagradzając PT Autorów satysfakcją, iż ich materiały zostały przepisane (nie bez błędów) i powielone w 320 plus 3 egzemplarzach (wszystkie!). Jest także tom piąty (stron: 451, czyli łącznie 1471 kartek razy 323), a raczej pierwszy, zawierający (na dużo lepszym papierze) referaty programowe konferencji w następujących językach: angielskim, niemieckim, polskim i rosyjskim (dla informatyków poliglotów?).

Zważywszy jednak na fakt, iż większość, to MY Polacy — tematy poruszane na konferencji, według słów profesora, dotyczyły będą naszych problemów — znad Wisły. Potwierdził te nadzieje (życzenia?), w sposób niezmiernie barwny w swym wystąpieniu, a konkretnie w „publicystycznym wykładzie inauguracyjnym" doc. dr hab. Wojciech Olejniczak — sekretarz naukowy konferencji. Wystąpienie owo było zresztą „awaryjne", planowany bowiem wykład nie mógł się odbyć bo prelegent nie dojechał.

Pan docent „obśmiał" chyba wszystkie przywary naszej informatyki, może nawet trochę więcej, zresztą wstęp wystąpienia wiele wyjaśnia. Oto cytat: „Referat ma charakter publicystyczny; w tym sensie rozmyślnie dokonano w nim przejaskrawień i aby było weselej, dla prezentacji obrazu użyto kilku krzywych zwierciadeł. Innymi słowy w referacie nie zawsze mówi się prawdę (choć nie są to kłamstwa), nie zawsze mówi się wszystko, lecz zawsze mówi się dokładnie to, co się chce powiedzieć", koniec cytatu.

Prelegent mówił jeszcze o pociągach: o pędzącym superekspresie i pociągu osobowym, który stoi przed semaforem..., korzystał także jeszcze z wielu innych metafor. Po tej, w sumie trochę przydługiej inauguracji, szefowie kolejnych sekcji zarządzili, by autorzy referatów (115) zgłosili się do nich i „się zobaczy", kto będzie występował, a kto nie. Pozostawiam tę konkurencję bez komentarza.

Równolegle odbywały się także imprezy towarzyszące, znacząco zwróciły one na siebie uwagę, pomijam oczywiście spotkanie towarzyskie (a 1500 zł), chodzi mi o firmy wystawiające sprzęt, a szczególnie kompetencje fachowców tam pracujących. Nadzwyczaj interesujące było dla mnie spotkanie z dr. Andrzejem Niemcem, reprezentującym ELWRO. Rozmowa z nim (na tematy zastosowań mikro do obliczeń statystycznych) była nieporównywalnie wartościowsza od szeregu referatów. Jaką mamy bowiem korzyść z wysłuchania np.: referatu, który dotyczył kompleksowego komputeryzowania stoczni na bazie sprzętu i oprogramowania rodem z VANGa? Z założeniem, iż stocznia ma dolary — dostatecznie dużo dolarów.

Spotkanie towarzyskie minęło bez ekscesów — jednak informatyka zdominowana jest przez mężczyzn, a szkoda...

O świcie następnego dnia rozpoczęła się Mikroskala równolegle z jeszcze inną sekcją. Zważywszy na nasze („IKS-a") zainteresowania — oczywiście wybrałem Mikroskalę. Na wstępie dzielenia się wrażeniami z tego uczestnictwa pragnę zwrócić uwagę na głośną sprawę sądową, o której opowiedział jeden z organizatorów. Oskarżonym była nasza rodzima mikroinformatyka, po wysłuchaniu obu stron — prokuratora i obrońców, sąd orzekł odroczenie sprawy — oskarżona jest po prostu nieletnia (to wiele tłumaczy).

Jako pierwszy wystąpił pan doktor z Politechniki Warszawskiej, mówiąc o trendach w rozwoju systemów operacyjnych dla mikrokomputerów. Był to w zasadzie wykład o systemach operacyjnych od „początku świata" plus totalna negacja wszelkich krajowych osiągnięć. Prelegent mówił o tysiącach osobolat, które spędzono na budowie systemów operacyjnych dla dużych maszyn typu ODRA 1305, o CPM-ie, DOS-ach (i ich popularności — 10 milionów instalacji), o UNIX-ie i oczywiście o OS2, który obecnie jest w trakcie bu-

dowy. Od trzech lat pracuje nad nim 35 osób. Wydaje się, że PS2 — jest nowym głównym kierunkiem „rynkowym" IBM-a, który między innymi przez „nałożone" patenty pragnie odciąć się od kłopotliwej konkurencji Dalekiego Wschodu. Niemniej za DOS-em opowiada się 50 proc. użytkowników, 25 proc. za OS2, pozostali za mutacjami UNIX-a. Wykładowca wspomniał o problemie komunikacji użytkownika z komputerem (ikony, ekrany dotykowe). W tym miejscu nie mogę sobie podarować uwagi co do znacznej zbieżności przedstawionych treści o zawartości ostatnich BAYT-ów. Pan doktor mówiąc o problemach krajowych nie powiedział dokładnie nic.

Pan Kajkowski jako pierwszy zabrał głos w dyskusji i zadeklarował kompetentną pomoc w sprzężeniu DOS-a z UNIX-em...

Kolejnym problemem był **WIRUS** — czarny strach padł na większość uczestników konferencji, w tym na piszącego te słowa. „Zasady" są podobne do AIDS-a, przekładając je na język użytkowników mikro należy:
— korzystać z zablokowanych (na zapis) dyskietek,
— skopiować (natychmiast!) wszystkie programy i zbiory,
— nowe programy powinny przechodzić kwarantannę na wyróżnionym w firmie komputerze, sprzęt ten trzeba odciąć od sieci — wirus biega także po drutach!

Oto kilka fragmentów głosów w tej kwestii: „czy jest szczepionka", „widziałem w akcji wirusa z Austrii", „północna Polska jest zakażona", „nie ma skutecznej ochrony", „znalazłem wirusa w COM", „porządne firmy zachodnie nie dają wirusa", itp.

Cóż na pewno należy się wystrzegać przypadkowych kontaktów naszego komputera z obcymi... dyskietkami (asceza?). Choroba ta przenosi się natychmiast na inne programy, BIOS-a i wylęga się. Trwa to pewien czas „bezobjawowo", a potem tragedia — twardy trzeba sformatować, bez wcześniejszej możliwości skopiowania czegokolwiek.

Po tych informacjach dostrzegłem za oknem piękne słońce — to był wspaniały spacer brzegiem morza.

Po obiedzie była giełda systemów i kolejne wystąpienia, między innymi, jak wykorzystać mikro do optymalizacji wykrojów (blachy). Temat „dyżurny", bo już w 1972 roku na komputerze ODRA 1013 w ZETO Bydgoszcz został ten problem rozwiązany. Na marginesie — ZETO Bydgoszcz ma doskonale pracującą sieć na bazie R32 i trzech JSG zainstalowanych we Włocławku (do 8 monitorów) w Grudziądzu (do 8 monitorów) i w Bydgoszczy tyle samo. Informacje „biegają" między miastami z szybkością 4800 bodów! Efekty tego systemu mogą ocenić niektórzy klienci ZUS.

Infogryf nabierał rozmachu. Wieczorem o 20.00 (to już maraton) rozpoczęła się giełda firm komputerowych. Czego tam nie było? Komputery z Singapuru, Tajwanu, wielodostęp, galanteria (komputerowa), Junior, 801AT („made in ELWRO" — „kości" z Tajwanu, obudowa nasza, 512 KB, 6 MHz, Hercules, DOS kupiony formalnie i twardy 20 MB — 60 ms!, cena 3,8 miliona zł, oczekiwanie 6 miesięcy), OA Link (bardzo ciekawa sieć — do 6 terminali będących samodzielnymi komputerami, które w postaci kart zainstalowane są w gniazdach płyty głównej centralnego AT), Winchestery 380 MB, plotery, streamery, pamięci taśmowe, 32-bitowe PCty z zegarem 25 MHz.

Była także dyskusja, czasami zabawna, między panami dyrektorami firm komputerowych, a uczestnikami konferencji. Powstał między innymi problem, czy system finansowo-księgowy instaluje się dwa dni czy rok...? Zdania były podzielone, pan prowadzący („po amerykańsku" z ręką w kieszeni) ledwo zapanował nad emocjami dyskutantów.

Dowiedzieliśmy się także, że miesięcznik „Mikroklan" będący blisko bankructwa wykupiło ELWRO. Redaktorem naczelnym został dyrektor fabryki. Działem programowania kieruje dr A. Niemiec, który zadeklarował tematykę miesięcznika: „powyżej XT", promocja polskich programów, sieci i wielodostęp oraz szeroko pojęta edukacja komputerowa.

Następnego dnia, także o świcie, zważywszy na to, iż Mikroskala dobiegła końca, udałem się do sali IKARa, by posłuchać nowości w zakresie systemów wspomagania decyzji (DSS). „Wysokie loty" (IKAR) prelegentów przypominały mi tragiczną historię Ikara, im wyżej leciał tym słabsze były jego skrzydła... Dużo mówiono o DSS-ach sensu stricto i largo, osiągnięciach Zachodu i kraju za oceanem, różnicach między bazami danych i bazami wiedzy, systemach ekspertowych (eksperckich? doradczych?). Dopiero podczas wystąpienia gościa bułgarskiego powiało nieco praktyką (programy: gromadzące wiedzę ekspertów, tworzące bazy danych i przetwarzające te dane i wiedzę). Ale już za chwilę, po pewnym akcencie futurologicznym, pojawił się temat: „Zastosowanie teorii zbiorów rozmytych do symulacyjnego wspomagania decyzji" — natychmiast się „zmyłem". Słońce pięknie świeciło, mimo to zacząłem spisywać te oto słowa, bo termin składania mija za kilkanaście godzin.

Kończąc te na gorąco spisane wrażenia z INFOGRYFU'88, wydaje mi się konieczne podkreślenie faktu, iż większość luminarzy nauki związanych z informatyką nie docenia (wręcz nie dostrzega) fachowości tych na „dole" — autentycznych specjalistów (praktyków) wdrażania komputerów we wszystkich dziedzinach naszego życia codziennego. Trzeba obecnie tworzyć język, nowy język, porozumiewania się wielkiej nauki z praktyką komputerową nad WISŁĄ.

Kołobrzeg 20.10.88

UCZESTNIK

NAUKA ● TECHNIKA

Trzy placówki badawcze w USA poinformowały ostatnio o opracowaniu bardzo zaawansowanych komputerowych systemów rozpoznawania ludzkiej mowy. Mogą one rozróżnić do tysiąca słów wypowiadanych stosunkowo szybko, bez długich pauz między nimi. Również firma IBM jest bliska zakończenia prac nad tego rodzaju systemem o nazwie Tangora. Współpracowałby on z komputerami personalnymi.

Najdoskonalszy spośród trzech wspomnianych systemów opracowany został w Carnegie Mellon University, w Pensylwanii przez Kai-Fu-Lee. System o nazwie Sphinx wykorzystuje w swym działaniu tzw. statystyczny model rozpoznawania i generowania mowy. Dźwięk w tym systemie przedstawiany jest w postaci cyfrowej. Najważniejsze jest jednak to, że jest to system rozpoznający słowa niezależnie od tego, kto je wypowiada. Dotychczasowe natomiast wymagały zaprogramowania na określonego mówcę.

Dokładność rozpoznawania słów wynosi 96 proc. (gdy zdania są proste). Odsetek błędów rośnie do 10,1 proc., gdy zdania są bardziej skomplikowane. Nieco gorszy pod tym względem jest drugi system opracowany w laboratoriach MIT, gdyż odsetek błędów wynosi 13,6 proc.

Trzeci system rozpoznający ok. 1 tys. słów powstał w prywatnym laboratorium Bolt, Deranek and Newman w Cambridge, Massachusetts. Ten system, którego autorem jest John Makhoul określa się jako „adaptujący się do mówcy". Przez 30 minut przygotowuje się on do rozpoznawania określonego głosu.

Mimo tych niewątpliwych osiągnięć nadal systemy rozpoznawania mowy znajdują się w początkowej fazie rozwoju. Specjaliści m.in. z Carnegie Mellon University są zdania, że na urządzenie, które będzie w stanie rozpoznawać spontanicznie wypowiadane zdania, a więc nie będzie stwarzało żadnych ograniczeń, mówiącemu poczekać trzeba będzie co najmniej 10, a może nawet 30 lat.

* * *

Japońskie firmy komputerowe: NEC, Hitachi i Fujitsu opracowały własne konstrukcje superkomputerów, nie ustępujące produkcji światowego potentata w tej dziedzinie — amerykańskiej firmie Cray Research Corp.

Obecnie funkcjonuje na świecie ok. 260 superkomputerów. Najszybsze z nich potrafią wykonać ponad miliard działań w ciągu sekundy, zaś każdy z nich może zastąpić przeszło sto tysięcy komputerów osobistych.

Z ostatnich nowości: Cray Research wprowadził niedawno nowy superkomputer, który umożliwia trójwymiarowe modelowanie. Maszyna nazwana Y-MP — kosztująca 20 mln dolarów — wykorzystuje jednocześnie osiem mikroprocesorów.

* * *

Roboty wielkości ... komara, które mogłyby np. przedostawać się naczyniami krwionośnymi do serca, aby usunąć zagrażające życiu ludzkiemu schorzenia tego organu, powstają w laboratoriach amerykańskich w wyniku wytrawiania zaprojektowanych wzorów na płytkach krzemowych, a więc dzięki technologiom stosowanym w mikroelektronice. Tą samą techniką wytwarzane są mikroskopijne mechanizmy, trybiki, turbinki i silniczki. W laboratoriach amerykańskiego koncernu ATT otrzymano już kółka zębate, których ząbki mają zaledwie 15 mikronów szerokości!

Animacja w trybie znakowym

Tomasz MROWIEC, Ludwik PIELA

Jedną z mocniejszych stron 8-bitowych komputerów Atari była zawsze animacja, czyli ruchome rysunki. Podstawą większości animacji jest grafika znakowa, prosty i efektywny sposób tworzenia własnych kształtów poprzez składanie odpowiednio dobranych znaków, którymi mogą być litery, cyfry i znaki interpunkcyjne.

Atari dostarcza także pewnych „specjalnych" znaków, które mogą być używane do kreślenia rysunków. Gdy uda się je pomysłowo połączyć, można otrzymać interesujące kształty (patrz **program nr 13**).

Rysunki w trybie znakowym możemy tworzyć za pomocą instrukcji PRINT i POSITION, w taki sam sposób jak tekst. Grafika znakowa jest bardzo efektywna pod względem zajętości pamięci. Na przykład ekran w trybie o wysokiej rozdzielczości (GRAPHICS 8) wymaga około 8000 komórek pamięci do przechowania wszystkich liczb reprezentacji bitowej. Używając znaków w zwykłym trybie tekstowym możemy zmieścić ten sam ekran w 960 bajtach pamięci (40 kolumn razy 24 wiersze).

Może wydawać się, że jesteśmy ograniczeni przez dysponowanie tylko dostępnymi w Atari znakami. Możliwe jest jednak tworzenie własnych znaków, dostosowanych do naszych aktualnych potrzeb. Zwykle nie musimy przerabiać całego zestawu znaków. Często kilka zmian wystarczy do utworzenia takiego obrazu, jaki chcemy.

Sposób tworzenia i wyświetlania znaków na ekranie opisaliśmy w drugim odcinku naszego cyklu zamieszczonym w „IKS-ie" nr 3 z 1987 roku. Dlatego postaramy się uniknąć powtarzania ograniczając się tylko do przypomnienia pewnych istotnych informacji.

Dobrze znanym sposobem umieszczania znaków na ekranie są instrukcje PRINT oraz POSITION. Poniżej przedstawiamy dwa mniej znane sposoby. Pierwszym z nich jest umieszczanie (za pomocą POKE) kodów znaków bezpośrednio w obszarze pamięci ekranu. W tym celu najpierw musimy znaleźć adres początku pamięci ekranu. Robimy to następująco:

```
10 PAMEKR = PEEK (88) + 256*PEEK (89)
```

Znając adres początku pamięci ekranu umieścimy w nim literę. W tym celu wprowadźmy następującą linię i RUN dla programu:

```
20 POKE PAMEKR, 33
```

W lewym górnym rogu ekranu pojawi się litera **A.** Kod wewnętrzny umieszczony w odpowiednim miejscu pamięci spowoduje wyświetlenie znaku na ekranie.

Musimy pamiętać, że podczas ustawiania do pamięci używamy kodów wewnętrznych, a nie ATASCII. Obejrzymy teraz cały zestaw znaków komputera. W tym celu wprowadźmy następujące linie i RUN dla programu.

```
5 GRAPHICS 0
20 POSITION 2, 15:PRINT „Wstawianie" (POKE)"
30 FOR X=0 TO 255
40 POKE PAMEKR+X,X
50 NEXT X
```

Na ekranie otrzymamy wszystkie 128 znaków, a następnie negatywy każdego z nich. W taki sposób możemy umieścić znaki bezpośrednio gdzieś w pamięci.

Innym sposobem umieszczania znaków na ekranie jest użycie instrukcji PLOT w trybie tekstowym.

```
10 GRAPHICS 1
20 COLOR (33+160)
30 PLOT 5,7
```

Powyższe instrukcje umieszczą niebieską literę A w miejscu o współrzędnych 5, 7. Nic nie stoi na przeszkodzie aby następną instrukcją było

```
40 DRAWTO 15,7
```

Teraz jesteśmy już przygotowani do animacji pewnych znaków. Wystarczy użyć jedną z powyższych technik do zmiany znaków w tym samym miejscu ekranu. Mały program, przedstawiony poniżej, będzie animował litery A B C.

Wprowadźmy NEW, a następnie poniższe linie:

```
10 GRAPHICS 0
20 POSITION 10,10
30 PRINT „A"
40 POSITION 10,10
50 PRINT „B"
60 POSITION 10,10
70 PRINT „C"
80 GOTO 20
```

Odbywa się to dosyć szybko. Szybkość ą możemy sterować za pomocą pętli opóźniającej. Wykona to prosta pętla FOR—NEXT. Dodajmy zatem poniższe linie

```
5 OPO=100
35 GOSUB 100
55 GOSUB 100
55 GOSUB 100
100 FOR I-1 TO OPO: NEXT I:RETURN
```

Sterowanie szybkością uzyskamy poprzez zmianę wartości zmiennej OPO w linii 5

Powyższy efekt, z wykorzystaniem liter ABC, nie jest zbyt interesujący. Aby uzyskać naprawdę ciekawe efekty, konieczne jest dysponowanie znakami o odpowiednich kształtach, które musimy utworzyć sami. Wiadomo że cały zestaw znaków zajmuje 1024 bajty pamięci. Za pomocą instrukcji POKE można zmienić kształt dowolnego znaku. Pewne kłopoty sprawia fakt umieszczenia ich w pamięci ROM, której zawartości nie można zmienić.

Zestaw znaków jest zbiorem wzorów, które definiuje kształt każdego znaku. Standardowy zestaw znaków Atari przechowywany jest w pamięci ROM. W celu utworzenia własnych znaków musimy umieścić nowy ich zestaw gdzieś w pamięci, a następnie poinformować komputer, aby używał nowego zestawu w miejsce starego. Podstawową sprawą jest określenie gdzie umieścimy nowy zestaw znaków.

Jeden ze sposobów rozwiązania powyższego problemu sugerowany jest w Dodatku 12 książki „Mapping the Atari". Specjalny program kopiuje system operacyjny z ROM do RAM i wyłącza ROM. Zatem komputer „widzi" ukryty obszar RAM. Po wykonaniu tego możemy, za pomocą POKE, umieścić własny zestaw znaków w obszarze zajmowanym początkowo przez zestaw międzynarodowy. Zaletą tej techniki jest to, że nowe znaki nie zmniejszają wielkości pamięci dostępnej dla naszego programu. W podobny sposób możemy zmienić kształt dowolnych znaków w standardowym zestawie.

Inny sposób polega na utworzeniu własnych znaków w wybranym obszarze RAM i poinformowanie o tym systemu operacyjnego. Jest to czynność pracochłonna i wykonywana bardzo rzadko, bowiem wymaga zdefiniowania całego zestawu znaków. Łatwiejsze i mniej pracochłonne jest modyfikowanie istniejącego zestawu, po uprzednim przepisaniu go z pamięci ROM do RAM.

W obu wymienionych ostatnio sposobach istotne jest wybranie bezpiecznego miejsca w pamięci dla przechowania zestawu znaków. Zawsze musimy chronić dane poprzez umieszczenie ich w takim miejscu, gdzie nie będą zniszczone. A system operacyjny, podobnie jak BASIC, stale przemieszcza swój kod. Na szczęście istnieje łatwy sposób tworzenia bezpiecznego miejsca.

Granice używanej pamięci sterowane są przez RAMTOP w komórce 106 i MENTOP w komórkach 741 i 742. Komórki te przechowują wartości reprezentujące 256-bajtowe stronice pamięci. Poprzez zmianę ich wartości możemy „przekonać" komputer, że pewien nieużywany obszar pamięci jest już zajęty.

RAMTOP przechowuje najwyższy numer stronicy wolnej pamięci RAM dostępnej do użycia. System operacyjny nie chce używać pamięci powyżej tej stronicy. Poprzez zmniejszenie jej numeru możemy używać RAM między oryginalną granicą i tą nową bez zakłóceń.

Po przesunięciu RAMTOP w dół o 2 KB (8 razy 256) otrzymamy bezpieczny obszar pamięci. Czasami pierwsze 1024 bajty są wciąż niepewne, dlatego będziemy używać drugiego 1 KB dla naszego zestawu znaków.

Komputer wymaga, aby zestaw znaków w trybie graficznym 0 zaczynał się od adresu

podzielnego przez 1024 (dla GR.1 lub 2 podzielny przez 5120. Oznacza to dwie, albo cztery stronice naraz.

Jesteśmy już gotowi do rozpoczęcia animacji. Musimy zmodyfikować trzy znaki. Lecz najpierw musimy przesunąć standardowy zestaw znaków z ROM do RAM. Możemy to zrobić następująco

```
100 NOWYZ=PEEK (106)
110 NOWYZ=NOWYZ-8:POKE 106,
    NOWYZ
120 GRAPHICS 0
130 CHSET=57344
140 FOR BAJT=0 TO 1023
150 POKE NOWYZ*256+BAJT, PEEK
    (CHSET+BAJT)
160 NEXT BAJT
```

Wykonanie powyższego programu zajmuje około 15 s. Lepiej będzie wykorzystać do tego procedurę w języku maszynowym. Nie musimy wiedzieć jak ona działa, wystarczy umieć ją użyć. To znaczy, jak włączyć ją do programu w BASIC-u oraz jak wywołać i z jakimi parametrami.

```
 10 DIM MM $ (41)
 20 FOR ZNAK=1 TO 41
 30 READ KOD
 40 MM $ (ZNAK, ZNAK)=CHR$ (KOD)
 50 NEXT ZNAK
 60 DATA 104, 104, 133, 207, 104, 133,
    206, 104, 133, 209, 104,
 70 DATA 133, 208, 104, 170, 160, 255,
    138, 208, 2, 104, 168, 177
 80 DATA 206, 145, 208, 136, 192,
    255, 208, 247, 230, 207, 230
 90 DATA 206, 202, 224, 255, 208, 233,
    96
130 CHSET=57344
140 X=USR (ADR) MM$, (CHSET,
    NOWYZ*256, 1024)
```

Teraz zestaw znaków Atari jest już w chronionym obszarze RAM. Możemy to sprawdzić zmieniając wartość CHBAS:

```
POKE 756,NOWYZ
```

Następnie należy podjąć decyzję, którym znakom chcemy nadać nowe kształty. Nie mamy potrzeby używania którejkolwiek z liter, dlatego podmienimy znaki CONTROL-A do CONTROL-C. Normalnie są to znaki graficzne Atari, ich kody ATASCII mają odpowiednio wartości od 1 do 3.

Jak już wspomnieliśmy, zestaw znaków ma inną kolejność niż ATASCII. W podręczniku języka BASIC możemy sprawdzić, że znaki te mają kody wewnętrzne 65 do 67

Poniżej prezentujemy trzy różne kształty robota, a także linie programu, które to realizują:

```
  ###    = 00111000 = 56
 # ###   = 01011100 = 92
#######  = 11111110 = 254
 #####   = 01111100 = 124
 #   #   = 01000100 = 68
 #  # #  = 01001010 = 74
 #       = 01000000 = 64
# #      = 10100000 = 160

  ###    = 00111000 = 56
 ## ##   = 01101100 = 108
#######  = 11111110 = 254
 #####   = 01111100 = 124
 #   #   = 01000100 = 68
 #   #   = 01000100 = 68
 #   #   = 01000100 = 68
# # # #  = 10101010 = 170

  ###    = 00111000 = 56
 ### #   = 01110100 = 116
#######  = 11111110 = 254
 #####   = 01111100 = 124
 #   #   = 01000100 = 68
# #  #   = 10100100 = 164
     #   = 00000100 = 4
    # #  = 00001010 = 10
```

```
170 ZES=NOWYZ*256
180 FOR ZNAK=65 TO 67
190 FOR BAJT=0 TO 7
200 READ DANE
210 POKE ZES+ZNAK*8+BAJT, DANE
220 NEXT BAJT
230 NEXT ZNAK
1000 DATA 56, 92, 254, 124, 68, 74, 64, 160
1010 DATA 56, 108, 254, 124, 68, 68, 68, 170
1020 DATA 56, 116, 254, 124, 68, 164, 4, 10
```

Nasz zestaw znaków jest już gotowy do działania, poinformujemy komputer, gdzie on jest:

```
240 POKE 756, NOWYZ
```

Wprowadźmy teraz kod poruszający robota:

```
 250 POKE 752,1
 260 POSITION 10,4
 270 PRINT CHR$(4)
 280 RESTORE 1100
 290 FOR LP=1 TO 3
 300 READ ZNAK
 310 POSITION 10,2
 320 PRINT CHR$ (ZNAK)
 330 FOR CZEK=1 TO 2
 340 NEXT CZEK
 350 NEXT LP
 360 GOTO 270
1100 DATA 1,2,3
```

Zauważmy, że animacja robota polega na wyprowadzaniu różnych wersji, jednej po drugiej, w tym samym miejscu ekranu, z małym opóźnieniem między nimi, w celu umożliwienia obejrzenia każdego z obrazów. Podobna technika używana jest przez część programów do realizacji prostej animacji.

Jeśli chcemy wypróbować zestaw znaków w trybach GR.1 i 2, dokonujemy następujących zmian w programie:

```
120 GRAPHICS 1:REM LUB 2
240 POKE 756,NOWYZ+2
270 PRINT#6 CHR $ (4)
320 PRINT #6 CHR $ (ZNAK)
```

Serca możemy usunąć z ekranu poprzez przedefiniowanie znaku serca na znak spacji. Jeśli potrzebne nam są wielkie i małe litery, możemy to uzyskać poprzez przedefiniowanie zestawu znaków tak, aby znaki graficzne stały się wielkimi literami. W celu szybkiego dokonania modyfikacji możemy użyć procedury przechowywania w zmiennej MMs. Następująca instrukcja przesunie wielkie litery w miejsce znaków graficznych w pamięci

```
X=USR (ADR (MM$), 57377, NOWYZ *
256+65,26)
```

Teraz naciśnięcie CTRL-A da nam wielkie A.

Jeśli chcemy szybko przełączyć kilka zestawów znaków możemy wykorzystać jedną z poniższych technik:

1. Możemy wprowadzić kilka kształtów znaków do RAM i przełączać wskaźnik początku zestawu na odpowiedni krój. Jest to nieefektywne, ponieważ musimy przechowywać w pamięci wiele zestawów znaków.

2. Możemy wykorzystać procedurę MMs. Przesunie ona nasz zestaw znaków ze zmiennej tekstowej do RAM, a wskaźnik początku zestawu pozostaje niezmieniony. Potrzebny jest tylko jeden obszar dla wszystkich danych

Po tej części opisowej wprowadźmy do komputera **program nr 2** i połączmy go z programem nr 1. **Program nr 3** przygotuje nam linie 110 i 150, które również należy dołączyć

Program wynikowy wykonuje animację poprzez wyprowadzanie zmodyfikowanych znaków w tym samym miejscu ekranu.

Otrzymana sylwetka jest bardzo mała i wykonuje ruchy w miejscu. Można ją powiększyć poprzez użycie do kreślenia kilku znaków. W tym celu wprowadźmy **program nr 4** i przechowajmy go w pamięci zewnętrznej. **Program nr 5**, po wprowadzeniu i uruchomieniu, przygotuje nam linie 2110—2120, które należy dołączyć (za pomocą ENTER) do programu nr 4.

Program ten dodaje pewne nowe linie do poprzedniego, w celu powiększenia sylwetki robota, i usuwa inne. Dlatego wprowadźmy wszystkie instrukcje REM, które usuwają stare linie. Połączmy go z programem poprzednim.

W programie nr 4 występują dwa interesujące chwyty. Pierwszym jest przechowanie kształtu znaków w zmiennej tekstowej. Pozwala to zaoszczędzić pamięć, ponieważ każda liczba (jak np.: 240) przechowywana wcześniej w instrukcji DATA, teraz jest zapamiętana w jednym bajcie. Drugi chwyt, w liniach 200 i 210, powoduje szybkie przesłanie nowych kształtów do nowego zestawu znaków

Prostym sposobem sprawdzenia poprawności utworzonych znaków jest zatrzymanie komputera po dokonanu przez niego zmiany znaków. Teraz wystarczy nacisnąć odpowiednie klawisze, aby zobaczyć na ekranie nowe kształty. W naszym wypadku naciśnijmy następujące klawisze:

abc def ghi jkl mno pqr

Powinno to wyświetlić różne części nowo utworzonych kształtów. W ten sposób wygodniej jest obejrzeć nowe znaki niż podczas działania programu (są one nieruchome).

Jak widać z powyższego spisu, tworzenie własnych znaków nie jest zadaniem trudnym, lecz jakże pracochłonnym. Dlatego w celu ułatwienia pracy proponujemy wykorzystać prezentowany edytor znaków. Oprócz normalnych własności posiada on dwie dodatkowe: animację wybranego ciągu znaków oraz łączenie czterech znaków w jedną większą jednostkę.

W celu uzyskania własnej kopii edytora należy wprowadzić **program nr 6** i przechować go w pamięci zewnętrznej. **Program nr 7** przygotowuje pewne linie dla programu nr 6, które ciężko byłoby wprowadzić z wydruku. Dlatego po wprowadzeniu programu nr 7 należy uruchomić go i otrzymane w pamięci zewnętrznej linie dołączyć do programu nr 6.

Po uruchomieniu programu wynikowego zobaczymy ekran podzielony na pięć podstawowych obszarów roboczych.

1. Siatka edycji — każdy z 64 kwadratów w siatce reprezentuje bit — każdy poziomy wiersz bitów reprezentuje bajt. Jest to podstawowy obszar dla edycji lub animacji znaków.

2. Obraz znaku — podczas edycji widzimy, jak będzie wyglądał znak w trzech różnych trybach graficznych. Każdy znak jest w innym kolorze, lecz tylko dla wyróżnienia go.

3. Menu — wykaz podstawowych rozkazów edytora. W celu wybrania funkcji należy wprowadzić literę umieszczoną z lewej strony. Po wybraniu funkcji (nie należy naciskać klawisza (RETURN) tekst podświetlany jest na niebiesko.

4. Obraz zestawu znaków — reprezentacja aktualnych kształtów znaków.

5. Obszar we—wy — służy do wyświetlania komunikatów błędów i innych informacji.

Tryb kursora u góry menu, podświetlany jest na niebiesko, kiedy działamy w siatce edycji. W tym trybie dostępne jest sześć następujących funkcji:

1. Przesuwanie kursora. Do ustawienia kursora używamy klawiszy strzałek. Nie należy naciskać (CONTROL) lub (SHIFT).

2. Kreślenie. Używamy jej do kreślenia i kasowania wewnątrz siatki edycji. Jeśli kursor jest na pustym kwadracie, naciśnięcie (CONTROL) i klawisza strzałki wypełni go. Jeśli kursor jest na wypełnionym kwadracie, użycie (CONTROL) i klawisza strzałki oczyści go. Kursor przesuwany jest w kierunku naciśniętej strzałki.

3. Powrót. Naciśnięcie (H) przesuwa kursor do pozycji początkowej: lewego górnego rogu siatki edycji. Naciśnięcie (CONTROL) (H) przesuwa kursor do prawego dolnego rogu siatki.

4. Przesuwanie siatki. Naciśnięcie (SHIFT) i klawisza strzałki przesuwa znak w siatce edycji o jeden wiersz w kierunku strzałki.

5. Wartość bajtu. Aby zobaczyć wartość bajtu każdego wiersza, należy nacisnąć (N). Dla wyłączenia tych wartości naciskamy (CONTROL) (N).

6. Przełączanie kursora. Naciśnięcie klawisza (SPACEBAR) wyłącza kursor. Naciśnięcie klawisza strzałki — włącza.

Funkcje wprowadź (L) i przechowaj () służą do przemieszczania zestawów znaków z do pamięci zewnętrznej. Po wybraniu ich musimy podać nazwę zbioru lub C:

W celu pobrania znaku do siatki edycji należy nacisnąć (G) i (RETURN). Klawisze strzałek ustawiają kursor na znaku, który chcemy poprawiać.

W celu przeniesienia znaku z siatki edycji i umieszczenia go w zestawie znaków należy nacisnąć klawisz (P). Jest to funkcja odwrotna do pobrania. Można ją wykorzystać do tworzenia wielokrotnych kopii.

Naciśnięcie (C) czyści siatkę edycji. Kasuje tylko wybrany znak. Nie zmienia innych znaków.

Funkcja (I) inicjuje działanie edytora. Odtwarza wszystkie znaki w ich oryginalnych kształtach.

Naciśnięcie (A) umieszcza kolejno znaki zarówno w siatce edycji, jak i obok niej. Przedtem musimy podać znaki, które będą animowane.

Składany znak wygląda podobnie jak duży znak, lecz jest aktualnie składany z kilku standardowych. Naciśnięcie (B) powoduje przejście do budowy złożonego znaku i wymaga wprowadzenia czterech znaków. Pojawią się one w obszarze wyświetlania w trybie GR. 2.

Naciśnięcie () kończy pracę programu. Upewnia się on dodatkowo, czy na pewno chcemy to zrobić. Jakkolwiek odpowiedź inna niż (T) powoduje powrót do programu.

Animacja w trybach graficznych wymaga nieco innego podejścia i być może będzie przedstawiona w ramach cyklu.

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Prosram nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
ON 1040 GRAPHICS 0:POKE 752,1
IX 1050 SETCOLOR 2,0,0:SETCOLOR 1
   ,0,10
VY 1060 FOR R=15 TO 22
LQ 1070 POSITION 2,R:? "_";
JI 1080 NEXT R
GI 1090 POSITION 2,14:? "H      "
WO 1100 POSITION 3,13:? "PPPPPP"
HH 1110 POSITION 1,15:? "H"
KV 1120 FOR R=6 TO 13
MG 1130 POSITION 8,R:? "_"
IY 1140 NEXT R
AL 1150 POSITION 1,9:? "RWRRWR"
RU 1160 POSITION 1,10:? "      "
RH 1170 POSITION 1,11:? "ZAMEK"
QF 1180 POSITION 13,0:? "HJ"
DH 1190 FOR R=1 TO 6
LW 1200 POSITION 13-R,R:? "H"
NF 1210 POSITION 14+R,R:? "J"
IU 1220 NEXT R
JQ 1230 FOR R=6 TO 11
KZ 1240 POSITION 19,R:? "_"
JD 1250 NEXT R
MV 1260 POSITION 20,11
BS 1270 ? "            _"
OU 1280 POSITION 24,10
LM 1290 ? "         "
SZ 1300 FOR R=10 TO 22
MX 1310 POSITION 33,R:? "_";
IW 1320 NEXT R
FK 1370 POSITION 13,5:? "FMMMG"
RZ 1380 POSITION 13,6:? "V   B"
JT 1390 POSITION 13,7:? "GNNNF"
EE 1400 POSITION 0,23
MD 1410 FOR I=1 TO 39:? "_";:NEXT
   I
OM 1420 RESTORE 1480
EB 1430 FOR GW=1 TO 10
ET 1440 READ C,R:POSITION C,R:? "
   ."
SU 1450 NEXT GW
VG 1480 DATA 2,1,3,3,19,0,23,0,29
   ,0,18,2,24,5,26,1,31,2,38,2
```

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Prosram nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
QM 100 REM
QS 130 REM
IK 140 DIM ZNACZKI$(24)
MV 160 TM=PEEK(106)
MD 170 CB=TM-8:CA=CB*256:NCA=CA+2
   4
PI 180 POKE 106,CB-8
SI 190 X=USR(ADR(MM$),224*256,CA,
   1023)
NY 200 X=USR(ADR(MM$),ADR(ZNACZKI
   $),NCA,24)
PK 2000 POKE 756,CB
QN 2010 C=5:R=22
AS 2020 POSITION C,R
MF 2030 ? #6;"#";:GOSUB 2090
AY 2040 POSITION C,R
MX 2050 ? #6;"$";:GOSUB 2090
BE 2060 POSITION C,R
NP 2070 ? #6;"%";:GOSUB 2090
OY 2080 GOTO 2020
GJ 2090 FOR I=1 TO 50
ES 2100 NEXT I
AF 2110 RETURN
```

```
FK 1 REM ****************
FJ 2 REM *              *
YO 3 REM * Prosram nr 3 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
PJ 35 REM UTWORZENIE LINII 110-15
   0
KA 45 REM ZMIEN LINIE 70 JESLI PR
   ACUJESZ ZE STACJA DYSKOW
MG 50 DIM FN$(20),TEMP$(20),AR$(9
   3)
HO 60 DPL=PEEK(10592):POKE 10592,
   255
AJ 70 FN$="C:"
YS 80 GRAPHICS 0:? "     ANTIC'S
   GENERIC BASIC LOADER"
CD 90 ? ,"BY CHARLES JACKSON"
PW 100 POKE 10592,DPL:TRAP 170
JO 110 ? :? :? "Tworzenie ";FN$:?
    "...prosze czekac."
LQ 120 RESTORE :READ LN:LM=LN:DIM
    A$(LN):C=1
BK 130 AR$="":READ AR$
XW 140 FOR X=1 TO LEN(AR$) STEP 3
   :POKE 752,255
OT 150 LM=LM-1:POSITION 10,10:? "
   (Odliczanie...T-";INT(LM/10);"
   )    "
UY 160 A$(C,C)=CHR$(VAL(AR$(X,X+2
   ))):C=C+1:NEXT X:GOTO 130
OT 170 IF PEEK(195)=5 THEN ? :? :
   ? CHR$(125);"ZA DUZO LINII DAT
   A !":? "UTWORZENIE ZBIORU NIEM
   OZLIWE!":END
WU 180 IF C<LN+1 THEN ? :? CHR$(1
   25);"ZA MALO LINII DATA!":? "U
   TWORZENIE ZBIORU NIEMOZLIWE!":
   END
AL 200 OPEN #1,8,0,FN$
PP 210 POKE 766,1:? #1;A$;:POKE 7
   66,0
TY 220 CLOSE #1:GRAPHICS 0:? "KON
   IEC"
FK 1000 DATA 104
IT 1010 DATA 04904904803206807307
   703207707703604005204904105807
   707703606103410410413320710413
   3206104133209
ZF 1020 DATA 10413320810417016025
   513820800210416817720614520813
   619225520824723020723020920222
   4255208233096
SZ 1030 DATA 03415504905304803209
   007806506709007507303606103405
   609225412406807406416005610825
   4124068068068
MG 1040 DATA 17005611625412406816
   4004010034155
```

```
FK 1 REM ****************
FJ 2 REM *              *
ZI 3 REM * Prosram nr 4 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
NA 99 GOTO 110
BH 100 FOR I=1 TO 50:NEXT I:RETUR
   N
QS 130 REM
IH 140 DIM ZNACZKI$(76):DLY=100
QW 150 REM
QY 160 REM
ET 170 CB=PEEK(740)-4:POKE 106,CB
   -12:CA=CB*256:GRAPHICS 0
RC 180 REM
TU 200 CA=CA+(96*8):FOR SEC=0 TO
   1:GOSUB 2110+10*SEC:X=USR(ADR(
   MM$),ADR(ZNACZKI$),CA,75)
```

```
GF 210 CA=CA+76:NEXT SEC
OV 1040 POKE 752,1
HX 2000 REM
XB 2010 X=5:Y=20
AJ 2020 POKE 756,CB+2
SJ 2030 POSITION X,Y:? #6;"ab":PO
   SITION X,Y+1:? #6;"cd":POSITIO
   N X,Y+2:? #6;"ef":GOSUB DLY
QU 2040 POSITION X,Y:? #6;"gh":PO
   SITION X,Y+1:? #6;"ij":POSITIO
   N X,Y+2:? #6;"kl":SCROLL=SCROL
   L+2:GOSUB DLY
PZ 2050 POSITION X,Y:? #6;"mn":PO
   SITION X,Y+1:? #6;"op":POSITIO
   N X,Y+2:? #6;"qr":SCROLL=SCROL
   L+4:GOSUB DLY
PF 2060 GOTO 2030
IS 2070 REM
IY 2080 REM
IY 2090 REM
HZ 2100 REM
```

```
FK 1 REM ***************
FJ 2 REM *             *
AC 3 REM * Prosram nr 5 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
GI 35 REM UTWORZENIE LINII 2110-2
   120
KA 45 REM ZMIEN LINIE 70 JESLI PR
   ACUJESZ ZE STACJA DYSKOW
MG 50 DIM FN$(20),TEMP$(20),AR$(9
   3)
HO 60 DPL=PEEK(10592):POKE 10592,
   255
AJ 70 FN$="C:"
YS 80 GRAPHICS 0:? "     ANTIC'S
   GENERIC BASIC LOADER"
CD 90 ? ,"BY CHARLES JACKSON"
PW 100 POKE 10592,DPL:TRAP 170
JO 110 ? :? :? "Tworzenie ";FN$:?
   "...prosze czekac."
LQ 120 RESTORE :READ LN:LM=LN:DIM
   A$(LN):C=1
BK 130 AR$="":READ AR$
XW 140 FOR X=1 TO LEN(AR$) STEP 3
   :POKE 752,255
DT 150 LM=LM-1:POSITION 10,10:? "
   (Odliczanie...T-";INT(LM/10);"
   )    "
UY 160 A$(C,C)=CHR$(VAL(AR$(X,X+2
   ))):C=C+1:NEXT X:GOTO 130
WE 170 IF PEEK(195)=5 THEN ? :? :
   ? CHR$(125);"ZA DUZO LINII DAT
   A !":? "UTWORZENIE ZBIORUNIEMO
   ZLIWE!":END
WU 180 IF C<LN+1 THEN ? :? CHR$(1
   25);"ZA MALO LINII DATA!":? "U
   TWORZENIE ZBIORU NIEMOZLIWE!":
   END
AL 200 OPEN #1,8,0,FN$
PP 210 POKE 766,1:? #1;A$;:POKE 7
   66,0
TY 220 CLOSE #1:GRAPHICS 0:? "KON
   IEC"
GH 1000 DATA 214
IX 1010 DATA 05004904904803209007
   8065067090075073036040049044O5
   505404106103400000000000000000
   0000000000003
EX 1020 DATA 00501101101100500300
   024023221224413223224000400901
   101901901701603101620020013620
   0200008248001
GA 1030 DATA 00300701402805606006
   012019219222409611206006000701
   102302302301100700822420816823
   2008208224016
BH 1040 DATA 00901701701703405808
   206908408508207803215505004905
   004803209007806506709007507303
   6040049044055
LG 1050 DATA 05404106103401601603
   100320020020023223200824019200
   300301506212011205602419219219
   2192192192240
ZB 1060 DATA 24000701102302302301
   100700822420816823200820822401
   600900901701601601603100320020
   023224012000O
EK 1070 DATA 24019200300700700701
   501501501522422419212812019219
   200083405800820690840850820780O3
   2155
```

```
FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Prosram nr 6 *
VJ 4 REM *    cz. 1     *
FM 5 REM *             *
FP 6 REM ***************
NM 7 REM
IC 2110 REM
SC 2120 GOTO 2640
ZY 2130 REM WCZYTANIE CIAGU ZNAKO
   W
ZB 2140 X=1
DX 2150 POSITION 1,22:? D$;M$;:M$
   =""
IA 2160 GET #1,KEY:? CHR$(KEY);
JS 2170 IF KEY=155 THEN 2210
NY 2180 IF KEY<>126 THEN 2200
IT 2190 IF X>1 THEN X=X-1:GOTO 21
   60
VJ 2200 M$(X)=CHR$(KEY):X=X+1:GOT
   O 2160
GU 2210 POKE 702,64:POSITION 1,22
   :? D$;
AK 2220 RETURN
GY 2230 REM POBRANIE NAZWY ZBIORU
   ZNAKOW
DV 2240 IF M$="" THEN M$="Q"
QE 2250 IF M$="Q" THEN POP :GOTO
   3170
YQ 2260 F$="D:":IF M$="C:" THEN F
   $=M$:GOTO 2350
DZ 2265 IF LEN(M$)<3 THEN 2300
ZC 2270 IF M$(2,2)=":" THEN 2290
GA 2280 IF M$(3,3)<>":" THEN 2300
RB 2290 F$=M$:GOTO 2310
TW 2300 F$(3)=M$
MH 2310 FOR I=1 TO LEN(F$)
HP 2320 IF F$(I,I)="." THEN 2350
FF 2330 NEXT I
DD 2340 F$(LEN(F$)+1)=".FNT"
PQ 2350 TRAP 3020:POKE 752,1:? F$
   ;
AY 2360 RETURN
DY 2370 REM MALUJ MATRYCE
GI 2380 POKE 752,1:POSITION 1,3:?
OQ 2390 X=USR(ADR(PX$),ADR(X$),MA
   )
ZG 2400 FOR I=0 TO 7
DN 2410 POSITION 10,I+5:? "   ";C
   HR$(30);CHR$(30);CHR$(30);
GH 2420 IF N=1 THEN ? PEEK(I+CHA)
FH 2430 NEXT I
AU 2440 RETURN
IU 2450 REM
LT 2460 IF CH<32 THEN CHR=CH+64
IY 2470 IF CH>31 THEN CHR=CH-32
SA 2480 IF CH>95 THEN CHR=CH
TE 2490 CHA=CA+CHR*8
QP 2500 X=USR(ADR(MB$),CHA,ADR(X$
   ))
MR 2510 POKE SA+22+4*40,CHR
FU 2520 X=USR(ADR(MM$),CHA,PA+542
   ,7)
EM 2530 IF BLD=0 THEN 2610
DA 2540 X=USR(ADR(MM$),BA(1),PA+6
   90,7)
OV 2550 X=USR(ADR(MM$),BA(3),PA+6
   98,7)
GU 2560 X=USR(ADR(MM$),BA(2),PA+8
   18,7)
HM 2570 X=USR(ADR(MM$),BA(4),PA+8
   26,7)
BI 2580 RETURN
ZS 2590 X=USR(ADR(FM$),PA+640,0,2
   55)
UG 2600 BLD=0:POKE 53249,138
NO 2610 X=USR(ADR(MM$),CHA,PA+693
   ,7)
AS 2620 RETURN
```

```
FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Prosram nr 6 *
VZ 4 REM *    cz. 2     *
FM 5 REM *             *
FP 6 REM ***************
NM 7 REM
ZN 2630 REM DEKLARACJE
WF 2640 DIM B$(4),BA(4),C$(26),D$
   (38)
PZ 2650 DIM F$(20),M$(40),P(26),X
   $(64)
UM 2660 X$(1)=CHR$(127):X$(64)=X$
   :X$(2)=X$
AV 2670 D$(1)=" ":D$(19)=D$(1):D$
   (2)=D$
XO 2680 FOR K=20 TO 38:D$(K,K)=CH
   R$(30):NEXT K
YK 2710 TM=PEEK(106)
MZ 2720 PB=TM-4:PA=PB*256
DZ 2730 CB=PB-4:CA=CB*256
YZ 2740 POKE 106,CB-4
GA 2750 GRAPHICS 0:REM POKE 559,0
FG 2760 POSITION 1,3:?
JX 2770 DL=PEEK(560)+PEEK(561)*25
   6
HZ 2780 SA=PEEK(DL+4)+PEEK(DL+5)*
   256
AN 2790 MA=SA+1+5*40
KD 2800 FOR I=1 TO LEN(C$)
FU 2810 IF I<18 THEN B=PA+920
QL 2820 P(I)=B+4:B=B+4
FP 2830 NEXT I
ZI 2840 GOSUB 5000
FT 2850 OPEN #1,4,0,"K:"
XT 2860 REM GRAFIKA P/M
ZY 2870 POKE 54279,PB:POKE 53277,
   3
OG 2880 POKE 623,1
AZ 2890 POKE 53256,0:POKE 53257,1
GH 2900 POKE 53258,1:POKE 53259,3
SR 2910 POKE 704,84:POKE 705,38
RP 2920 POKE 706,38:POKE 707,132
SN 2930 X=USR(ADR(FM$),PA+512,0,5
   12)
KF 2940 POKE 53248,150:POKE 53249
   ,138
WL 2950 POKE 53250,146:POKE 53251
   ,170
PV 2960 GOTO 3020
BC 2970 REM INICJOWANIE ZNAKOW
FY 2980 POSITION 1,22:? D$;
MX 2990 ? "INICJACJA (T/N)? ";:GE
   T #1,KEY
EJ 3000 POSITION 1,22:? D$;
UQ 3010 IF KEY<>84 THEN 3160
SI 3020 X=USR(ADR(MM$),224*256,CA
   ,1024)
NQ 3030 FOR I=0 TO 7
RV 3040 POKE CA+I+126*8,255
ZX 3050 POKE CA+I+127*8,64
XR 3060 IF I=0 THEN POKE CA+127*8
   ,0
WW 3070 IF I=7 THEN POKE CA+7+127
   *8,0
FP 3080 NEXT I
FR 3090 C1=15:R1=2:CH=ASC("@")
AB 3100 POKE 756,CB:POKE 82,1
YU 3110 POKE 709,10:POKE 710,0
AJ 3120 POKE 559,46
CQ 3130 REM MALUJ ZNAK
RK 3140 GOSUB 2460:GOSUB 2380:POK
   E 752,0
IA 3150 REM WEJSCIE Z KLAWIATURY
SL 3160 C=1:R=1:CX=1:RX=1
DA 3170 POSITION C,R+4:? CHR$(31)
   ;CHR$(30);
EL 3180 GET #1,KEY:POKE 752,0
KP 3190 REM KASUJ BLAD
IE 3200 IF  NOT E THEN 3250
OS 3210 POKE 752,1
ZA 3220 POSITION 1,22:? D$:E=0
CQ 3230 POSITION C,R+4:? CHR$(31)
   ;CHR$(30);
DM 3240 POKE 752,0
HP 3250 FOR B=1 TO LEN(C$)
FE 3260 IF KEY=ASC(C$(B,B)) THEN
   3300
CR 3270 NEXT B
BB 3280 POSITION 1,22:? CHR$(253)
   ;" BLAD "
```

```
OL 3290 E=1:GOTO 3170
OY 3300 X=USR(ADR(FM$),P(1),0,40)
VQ 3310 X=USR(ADR(FM$),P(B),255,4
 )
YR 3320 ON B GOTO 4580,4590,4600,
 4610
QD 3330 ON B-4 GOTO 4630,4640,465
 0,4660
PA 3340 ON B-8 GOTO 4550,4550,455
 0,4550
CY 3350 ON B-12 GOTO 3430,3440,34
 60,3480
TL 3360 IF B=17 THEN 3500
BL 3370 IF  NOT BLD OR B=26 THEN
 3390
NI 3380 IF B<20 OR B>21 THEN GOSU
 B 2590
HR 3390 ON B-17 GOTO 4240,3560,35
 20,3860
UZ 3400 ON B-21 GOTO 2980,3690,38
 60,3750
DF 3410 IF B=26 THEN 4890

FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Program nr 6 *
WP 4 REM *    cz. 3    *
FM 5 REM *             *
FP 6 REM ***************
NM 7 REM
HF 3420 REM KURSOR POZ. POCZ.
NK 3430 C=1:R=1:CX=C:RX=R:GOTO 31
 70
BX 3440 C=8:R=8:CX=C:RX=R:GOTO 31
 70
IV 3450 REM
XK 3460 N=1:GOSUB 2400:GOTO 3170
JB 3470 REM
XI 3480 N=0:GOSUB 2400:GOTO 3170
HP 3490 REM WYLACZ KURSOR
PB 3500 POKE 752,1:GOTO 3170
IL 3510 REM
GS 3520 X=USR(ADR(FM$),ADR(X$),12
 7,64)
HA 3530 X=USR(ADR(MS$),ADR(X$),CH
 A)
QU 3540 GOTO 3140
EX 3550 REM BLD
DN 3560 M$="4 ZNAKI? ":GOSUB 2140
BS 3570 B$=M$:IF M$="Q" THEN 3170
DH 3580 IF LEN(B$)=4 THEN 3600
IO 3590 ? CHR$(253):GOTO 3560
OV 3600 X=USR(ADR(FM$),PA+690,0,1
 6)
JM 3610 POKE 53249,130:BLD=1
RF 3620 X=USR(ADR(FM$),PA+818,0,1
 6)
VN 3630 FOR I=4 TO 1 STEP -1
PM 3640 CH=ASC(B$(I,I)):GOSUB 246
 0
FP 3650 BA(I)=CHA
FV 3660 NEXT I
RF 3670 GOTO 3140
DR 3680 REM LADUJ ZNAKI
UZ 3690 M$="LADUJ? "
LS 3700 GOSUB 2140:GOSUB 2240
QX 3710 OPEN #3,4,0,F$:POKE 752,1
AI 3720 X=USR(ADR(CC$),3,7,CA,100
 8)
WC 3730 GOTO 3790
CW 3740 REM ZAPISZ ZNAKI
JG 3750 M$="ZAPISZ? "
MK 3760 GOSUB 2140:GOSUB 2240
TT 3770 OPEN #3,8,0,F$:POKE 752,1
EE 3780 X=USR(ADR(CC$),3,11,CA,10
 24)
PA 3790 CLOSE #3
CN 3800 POSITION 1,22:? D$
MQ 3810 POKE 752,1:GOTO 3140

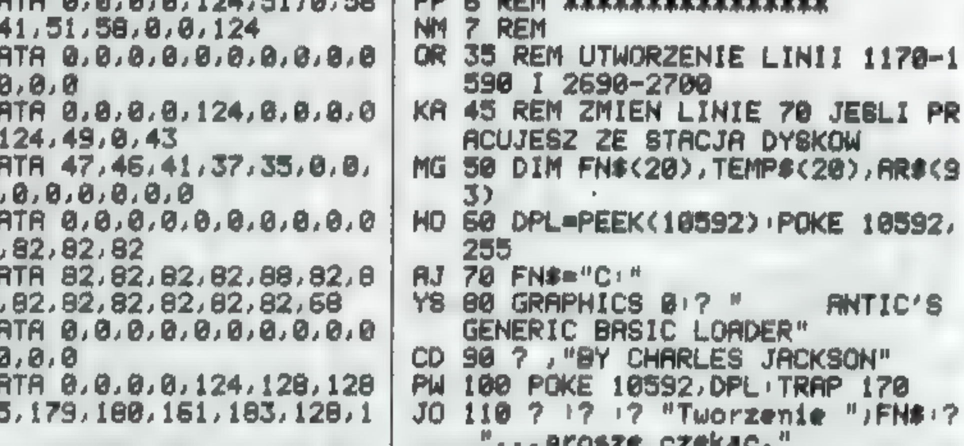
OD 3820 CLOSE #3:POSITION 1,22:?
 D$;
OW 3830 ? CHR$(253);" ZLA NAZWA Z
 BIORU"
FX 3840 E=1:POKE 752,0:GOTO 3170
YA 3850 REM WEZ/WSTAW ZNAK
MR 3860 IF  NOT BLD THEN 3940
PK 3870 M$="WYBIERZ ":M$(9)=B$
PA 3880 M$(13)="? ":GOSUB 2140
ZD 3890 FOR I=1 TO 4
IJ 3900 IF M$=B$(I,I) THEN 3930
FM 3910 NEXT I
LU 3920 ? CHR$(253):GOTO 3870
ZQ 3930 CH=ASC(M$):GOTO 3140
DA 3940 POSITION C1+20,R1+15:? CH
 R$(31);CHR$(30);
DD 3950 POSITION C1+20,R1+15:? CH
 R$(31);CHR$(30);
IA 3960 GET #1,K
TM 3970 CHA=CA+(C1-1+(R1-1)*18)*8
AA 3980 IF K=81 THEN 3170
FR 3990 IF K=42 THEN C1=C1+1
GH 4000 IF K=43 THEN C1=C1-1
EZ 4010 IF K=45 THEN R1=R1-1
CG 4020 IF K=61 THEN R1=R1+1
FR 4030 IF R1=0 THEN R1=7
EK 4040 IF R1=8 THEN R1=1
IF 4050 IF C1=0 THEN C1=18
KO 4060 IF C1=19 THEN C1=1
HC 4070 IF K<>155 AND K<>82 THEN
 3860
DN 4080 CHR=(CHA-CA)/8
JR 4090 IF CHR<64 THEN CH=CHR+32
QQ 4100 IF CHR>63 THEN CH=CHR-64
ML 4110 IF CHR>95 THEN CH=CHR
ZV 4120 IF K<>82 THEN 4170
LI 4130 RA=(224*256)+(C1-1+(R1-1)
 *18)*8
BZ 4140 X=USR(ADR(MM$),RA,CHA,7)
QM 4150 GOSUB 2500:GOSUB 2380
TF 4160 POKE 752,0:GOTO 3940
OX 4170 IF B<>21 THEN 4200
RH 4180 X=USR(ADR(MB$),CHA,ADR(X$
 ))
RC 4190 GOTO 3140
GM 4200 X=USR(ADR(MS$),ADR(X$),CH
 A)
QG 4210 GOTO 3140

FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Program nr 6 *
XF 4 REM *    cz. 4    *
FM 5 REM *             *
FP 6 REM ***************
NM 7 REM
ZL 4230 REM ANIMACJA
NS 4240 D=4:N=0:M$="ZNAKI? ":GOSU
 B 2140
HN 4250 IF LEN(M$)<2 THEN 3160
IR 4260 POKE 752,1:? "WOLNO SZYBK
 O KONIEC"
MK 4270 X=USR(ADR(MB$),CA,ADR(X$)
 )
DV 4280 GOSUB 2380
MX 4290 K=PEEK(764):POKE 764,255
TJ 4300 IF K=255 THEN 4420
ZY 4310 IF K<>46 THEN 4350
MF 4320 D=D+4:F=1:IF D>32 THEN D=
 32
QU 4330 POSITION 1,22:? "WOLNO SZ
 YBKO"
RM 4340 GOTO 4420
CS 4350 IF K<>62 THEN 4390
LW 4360 D=D-4:F=1:IF D<1 THEN D=0
UI 4370 POSITION 1,22:? "WOLNO SZ
 YBKO"
RY 4380 GOTO 4420
CS 4390 IF K<>5 THEN 4420
ES 4400 POSITION 1,22:? D$;
QK 4410 GOTO 3140
SF 4420 FOR I=1 TO LEN(M$)
UT 4430 CH=ASC(M$(I,I)):GOSUB 246
 0
OF 4440 X=USR(ADR(PX$),ADR(X$),MA
 )
MT 4450 FOR WAIT=1 TO D:NEXT WAIT
FS 4460 NEXT I
CL 4470 IF F=0 THEN 4530
JF 4480 POSITION 1,22:F=0
PS 4490 IF D=0 THEN ? "WOLNO
 "
TZ 4500 IF D=32 THEN ? "       SZY
 BKO"
FH 4510 IF D=0 OR D=32 THEN 4530
WP 4520 ? "WOLNO SZYBKO"
UC 4530 GOTO 4290
NM 4540 REM PRZESUNIECIE
JN 4550 POKE 752,1:X=USR(ADR(SF$)
 ,B,CHA)
NN 4560 GOSUB 2500:GOSUB 2380:GOT
 O 3170
FW 4570 REM RUCH KURSORA
JQ 4580 R=R-1:GOTO 4680
JB 4590 R=R+1:GOTO 4680
AS 4600 C=C-1:GOTO 4680
AD 4610 C=C+1:GOTO 4680

IR 4620 REM
RA 4630 R=R-1:W=1:GOTO 4680
QL 4640 R=R+1:W=1:GOTO 4680
JE 4650 C=C-1:W=1:GOTO 4680
ZT 4660 C=C+1:W=1
WE 4670 REM GRANICE RUCHU KURSORA
WY 4680 IF R>8 THEN R=1
YS 4690 IF R<1 THEN R=8
GN 4700 IF C>8 THEN C=1
IH 4710 IF C<1 THEN C=8
WQ 4720 IF  NOT W THEN 4830
JC 4730 REM PISZ  BIT
YL 4740 P=CX+(RX-1)*8:M$=X$(P,P)
CB 4750 IF M$=CHR$(126) THEN M$=C
 HR$(127):GOTO 4770
YV 4760 IF M$=CHR$(127) THEN M$=C
 HR$(126)
JI 4770 X$(P,P)=M$:? CHR$(27);M$;
 CHR$(30);
JL 4780 REM WYSWIETL OBRAZ
QB 4790 POKE 752,1
GY 4800 X=USR(ADR(MS$),ADR(X$),CH
 A)
NA 4810 GOSUB 2510:POKE 752,0
IV 4820 REM
UQ 4830 IF  NOT N THEN 4860
JA 4840 POSITION 10,RX+4
TQ 4850 ? "   ";CHR$(30);CHR$(30)
 ;CHR$(30);PEEK(RX-1+CHA)
XP 4860 CX=C:RX=R:W=0
SX 4870 GOTO 3170
XE 4880 REM KONIEC
TD 4890 IF BLD THEN GOSUB 2590:GO
 TO 3160
FC 4900 POSITION 1,22:? D$;
NE 4910 ? "KONIEC (T/N)? ";:GET #
 1,KEY
WS 4920 IF KEY=84 THEN 4940
GL 4930 POSITION 1,22:? D$:GOTO 3
 160
OZ 4940 POKE 106,TM:GRAPHICS 0
BG 4950 POKE 53248,0:POKE 53249,0
PS 4960 POKE 53250,0:POKE 53251,0
MZ 4970 POKE 82,2
VD 4980 X=USR(ADR("h1JG"))
GZ 5000 FOR I=0 TO 959:READ A:POK
 E SA+I,A:NEXT I
AG 5010 RETURN

FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Program nr 6 *
XV 4 REM *    cz. 5    *
FM 5 REM *             *
FP 6 REM ***************
NM 7 REM
TI 19990 REM EKRAN
IH 20000 DATA 81,82,82,82,82,82,8
 2,82,82,82,82,82,82,82,82
XE 20010 DATA 82,82,82,82,87,82,8
 2,82,82,82,82,82,87,82,82,82
RO 20020 DATA 82,82,82,82,82,82,8
 2,69,124,128,165,164,185,180,1
 75,178
BU 20030 DATA 128,128,128,128,128
 ,186,174,161,171,175,183,128,1
 24,128,175,162
XN 20040 DATA 178,161,186,128,124
 ,128,128,128,173,165,174,181,1
 28,128,128,124
JA 20050 DATA 90,82,82,82,82,82,8
 2,82,82,82,82,82,82,82,82
TR 20060 DATA 82,82,82,82,83,82,8
 2,82,87,82,82,82,83,82,82,82
GW 20070 DATA 82,82,82,82,82,82,8
 2,68,0,0,0,0,0,0,0,0
BA 20080 DATA 0,0,0,0,0,0,0,0,0,0
 ,0,0,124,0,0,0
QF 20090 DATA 124,0,0,0,124,0,0,4
 3,53,50,51,47,50,0,0,124
```

```
MD 20100 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
LJ 20110 DATA 0,0,0,0,124,0,0,0,1
   24,0,0,0,124,33,0,33
VI 20120 DATA 46,41,45,33,35,42,3
   3,124,0,0,0,0,0,0,0,0
AJ 20130 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,124,0,0,0
IQ 20140 DATA 124,0,0,0,124,34,0,
   34,53,36,47,55,33,0,0,124
MX 20150 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
RM 20160 DATA 0,0,0,0,124,0,0,0,1
   24,0,0,0,124,35,0,43
LW 20170 DATA 33,51,53,42,0,0,0,1
   24,0,0,0,0,0,0,0,0
PE 20180 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,65,82,82,82
VX 20190 DATA 88,82,82,82,68,39,0
   ,55,37,58,0,0,0,0,0,124
MG 20200 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
ZJ 20210 DATA 0,0,0,0,124,0,0,0,0
   ,0,0,0,124,41,0,41
YO 20220 DATA 46,41,35,42,53,42,0
   ,124,0,0,0,0,0,0,0,0
AM 20230 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,124,0,0,0
VR 20240 DATA 0,0,0,0,124,44,0,44
   ,33,36,53,42,0,0,0,124
NA 20250 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
UX 20260 DATA 0,0,0,0,124,0,0,0,0
   ,0,0,0,124,48,0,55
OM 20270 DATA 51,52,33,55,0,0,0,1
   24,0,0,0,0,0,0,0,0
BG 20280 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,124,0,0,0
PU 20290 DATA 0,0,0,0,124,51,0,58
   ,33,48,41,51,58,0,0,124
MJ 20300 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
QK 20310 DATA 0,0,0,0,124,0,0,0,0
   ,0,0,0,124,49,0,43
KE 20320 DATA 47,46,41,37,35,0,0,
   124,0,0,0,0,0,0,0,0
OQ 20330 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,65,82,82,82
YN 20340 DATA 82,82,82,82,88,82,8
   2,82,82,82,82,82,82,82,68
ND 20350 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
RU 20360 DATA 0,0,0,0,124,128,128
   ,186,165,179,180,161,183,128,1
   28,186
CW 20370 DATA 174,161,171,175,183
   ,128,128,124,0,0,0,0,0,0,0,0
PK 20380 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,65,82,82,82
TN 20390 DATA 82,82,82,82,82,82,8
   2,82,82,82,82,82,82,82,68
MM 20400 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
LW 20410 DATA 0,0,0,0,124,0,1,2,3
   ,4,5,6,7,8,9,10
OE 20420 DATA 11,12,13,14,15,16,1
   7,124,0,0,0,0,0,0,0,0
DP 20430 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,124,18,19,20
XK 20440 DATA 21,22,23,24,25,26,2
   7,28,29,30,31,32,33,34,35,124
NG 20450 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
TY 20460 DATA 0,0,0,0,124,36,37,3
   8,39,40,41,42,43,44,45,46
EP 20470 DATA 47,48,49,50,51,52,5
   3,124,0,0,0,0,0,0,0,0
UF 20480 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,124,54,55,56
WJ 20490 DATA 57,58,59,60,61,62,6
   3,64,65,66,67,68,69,70,71,124
MP 20500 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0
XF 20510 DATA 0,0,0,0,124,72,73,7
   4,75,76,77,78,79,80,81,82
MK 20520 DATA 83,84,85,86,87,88,8
   9,124,81,82,82,82,82,82,82,82
RO 20530 DATA 82,82,82,82,82,82,8
   2,82,82,82,82,82,68,90,91,92
EX 20540 DATA 93,94,95,96,97,98,9
   9,100,101,102,103,104,105,106,
   107,124
QZ 20550 DATA 124,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0,0
YE 20560 DATA 0,0,0,0,124,108,109
   ,110,111,112,113,114,115,116,1
   17,118
QQ 20570 DATA 119,120,121,122,123
   ,124,125,124,90,82,82,82,82,82
   ,82,82
VX 20580 DATA 82,82,82,82,82,82,8
   2,82,82,82,82,82,88,82,82
RN 20590 DATA 82,82,82,82,82,82,8
   2,82,82,82,82,82,82,82,67
FK 1 REM ***************
FJ 2 REM *             *
BQ 3 REM * Program nr 7 *
FM 5 REM *             *
FP 6 REM ***************
NM 7 REM
OR 35 REM UTWORZENIE LINII 1170-1
   590 I 2690-2700
KA 45 REM ZMIEN LINIE 70 JESLI PR
   ACUJESZ ZE STACJA DYSKOW
MG 50 DIM FN$(20),TEMP$(20),AR$(9
   3)
HO 60 DPL=PEEK(10592):POKE 10592,
   255
AJ 70 FN$="C:"
YS 80 GRAPHICS 0:? "     ANTIC'S
   GENERIC BASIC LOADER"
CD 90 ? ,"BY CHARLES JACKSON"
PW 100 POKE 10592,DPL:TRAP 170
JO 110 ? :? :? "Tworzenie ";FN$:?
   "...prosze czekac."
LQ 120 RESTORE :READ LN:LM=LN:DIM
   A$(LN):C=1
BK 130 AR$="":READ AR$
XW 140 FOR X=1 TO LEN(AR$) STEP 3
   :POKE 752,255
DT 150 LM=LM-1:POSITION 10,10:? "
   (Odliczanie...T-";INT(LM/10);"
   )    "
UY 160 A$(C,C)=CHR$(VAL(AR$(X,X+2
   ))):C=C+1:NEXT X:GOTO 130
ZB 170 IF PEEK(195)=5 THEN ? :? :
   ? CHR$(253);"ZA DUZO LINII DAT
   A !":? "UTWORZENIE ZBIORUNIEMO
   ZLIWE!":END
```



— Gorzej, że brakuje także komputerów w szkołach...

```
ZB 180 IF C<LN+1 THEN ? :? CHR$(2
   53);"ZA MALO LINII DATA!":? "U
   TWORZENIE ZBIORU NIEMOZLIWE!":
   END
AL 200 OPEN #1,8,0,FN$
PP 210 POKE 766,1:? #1;A$;:POKE 7
   66,0
TY 220 CLOSE #1:GRAPHICS 0:? "KON
   IEC"
IK 1000 DATA 1307
PG 1010 DATA 049049055048032080708
   403604004904106103410410417022
   400024002516925514125500613816
   0006162255202
HS 1020 DATA 208253136208248003415
   504904905604003200708403604005
   005204106103420625500620824117
   0202708231104
NX 1030 DATA 170224000240018141125
   500618000616225503415504904905
   704803208708403604005205404106
   1034202208253
BY 1040 DATA 136208248206255006200
   824109603415504905004904803206
   807307703207007703604005305204
   1059082069077
OI 1050 DATA 032070073076077069007
   704607906607415504905005004803
   207007703604004904106103410410
   4133205104133
BV 1060 DATA 204104104133206104107
   022400024001616025516520614520
   403415504905005104803207007703
   6040050052041
OC 1070 DATA 061034136208251145200
   423020520220824010416819200024
   001416520613619200024003415504
   9050052048032
MO 1080 DATA 070077036040052054040
   106103400514520413620825114520
   409603415504905005404803206807
   3077032077077
DT 1090 DATA 036040052057041058080
   206907703207707908607706907704
   607906607415504905005504803207
   7077036040049
TU 1100 DATA 041061034104104133200
   710413320610413320510413320410
   417022400024001816025517720603
   4155049050056
NO 1110 DATA 048032077077036040050
   005204106103414520413619225520
   824723020723020520220823810416
   8177206145204
CT 1120 DATA 136192034155049050050
   704803207707703604005205404106
   103425520824709603415504905104
   9048032068073
AZ 1130 DATA 077032067067036040050
   105504105808206907703206707307
   905708407604607906607415504905
   1050048032067
OX 1140 DATA 067036040049041061030
   410410410417016900002410501620
   220825017010410415706600310415
   7069003104034
HJ 1150 DATA 155049051051048032060
   706703604005005204106103415706
   800310415707300310415707200307
   6086228034155
FO 1160 DATA 049051053048032068070
   307703207708303604005605004105
   808206907703207707908608308408
   2046079066074
UE 1170 DATA 155049051054048032070
   708303604004904106103410410413
   320510413320410413320710413320
   6169000141000
XW 1180 DATA 007169000141001007030
   415504905105504803207708303604
   005005204106103416900014100200
   7169128141003
UN 1190 DATA 007162008172000007230
   800000717720420112603415504905
   105504803207708303604005205404
   1061034208009
ER 1200 DATA 173002007013003007140
   100200707800300702420220822817
   200100723803415504905105704803
   2077083036040
```

```
SH 1210 DATA 054056041061034001007173002007145206173000007201064208198096034155049052049048032068073077032
IB 1220 DATA 077066036040057049041058082069077032077079086066089084046079066074155049052050048032077066036
KR 1230 DATA 040049041061034104104133205104133204104133207104133206160000169127145206200192064208034155049
QZ 1240 DATA 052051048032077066036040050052041061034249169000141000007169000141001007172001007177204141002
AX 1250 DATA 007169128141034155049052052048032077066036040052054041061034003007162008172001007173002007056
QS 1260 DATA 237003007048010141002007172000007034155049052053048032077066036040054056041061034169126145206
CC 1270 DATA 078003007238000007202208225238001007173000007201064208034155049052054048032077066036040057048
CZ 1280 DATA 041061034200096034155049052056048032068073077032083070036040049049056041058082069077032083072
AB 1290 DATA 073070084089046079066074155049052057048032083070036040049041061034104104104170104133205133207
XK 1300 DATA 104133204133206224009240040224010240009224034155049053048048032083070036040050052041061034011
CE 1310 DATA 240073224012240050096160007177204133208230206136177204145206136034155049053049048032083070036
VD 1320 DATA 040052054041061034208249177204145206165208198206145206096160001772041332082302041770341 55049
HE 1330 DATA 053050048032083070036040054056041061034204145206201920072082471652081452060961600001772 04024
AR 1340 DATA 074144003024034155049053051048032083070036040057048041061034105128145204200192008208240096160
ZQ 1350 DATA 000177204024010144003024105001145034155049053052048032083070036040049049050041061034204200192
EL 1360 DATA 008208240096034155049053053054048032058073077032000080036040054050041058082069077032080065073078
GM 1370 DATA 084088046079065074155049053053048032080083604004904105103410410413320510413320410413320710 4
GM 1380 DATA 133206162008160000177204145206200192034155049053056048032080088036040056052041061034008208247
ZN 1390 DATA 202240032165204024105000133204144006165205105000133205165034155049053057048032080088036040052
DG 1400 DATA 054041061034206024105013320614421816520710500013320714421009603415505005405704803207703 6061
IZ 1410 DATA 034104162016169007157066003104157069003104157068003104157073003104157072003076086220034155058
WN 1420 DATA 055048048032067036061034045061043042028029030031095124092094072008078014032065066067071073076
AE 1430 DATA 080083081034155
```

Tomasz MROWIEC
Ludwik PIELA

GRA W TYSIĄCA

Janusz JANIEC

Jest to gra waszych ojców. Może brać w niej udział dwóch graczy tzn. ty i twój komputer. Jak w klasycznej grze, tak i tu wykorzystano talię składającą się z 24 kart (od asa do „9"). Obowiązuje następująca hierarchia wartości: as — 11 punktów, „10" — 10, król — 4, dama — 3, walet — 2 oraz „9" — bez punktów.

Każdy z graczy posiada po 5 kart, które rozdaje generator liczb losowych. W związku z tym, że jest to tysiąc dobierany nie ma początkowej licytacji. Rozpoczynając grę wychodzisz pierwszy, natomiast komputer jest pierwszy w następnym rozdaniu. Na ekranie masz zobrazowany aktualny stan kart „na ręku".

Nie obawiaj się, że komputer w momencie podejmowania decyzji będzie podglądał twoje karty, z pewnością tego nie uczyni. Jeśli wychodzący ma parę, czyli damę i króla w tym samym kolorze, powinien to zameldować, a następnie wyjść w damę. Za meldunek gracz otrzymuje: 100 punktów za kier, 80 dla kar, 60 dla trefli oraz 40 dla pików. Gdy skończy się talia, komputer tasuje karty, podaje aktualny wynik, a następnie proponuje dalszą grę. Wygrywa ten, który zdobędzie jako pierwszy 1000 punktów.

Przyporządkowanie klawiszy poszczególnym kolorom oraz honorom kart przedstawia poniższa tabela.

KOLOR	KLAWISZ
KIER	K
TREFL	T
KARO	C
PIK	P
HONOR	**KLAWISZ**
AS	A
10	1
KROL	K
DAMA	D
WALET	W
* 9 *	9
MAM MELD.	M

Program można podzielić na dwa zasadnicze bloki. Pierwszy dotyczy wykonywania poleceń wydawanych przez ciebie i drugi tworzący z komputera gracza. Poszczególne podprogramy są oddzielone komentarzem. Powinno to ułatwić zrozumienie algorytmu programu oraz dociekliwszym umożliwić dokonanie przeróbek.

```
1000 REM ************************
1002 REM *                      *
1004 REM *    GRA W TYSIACA     *
1006 REM *      * Atari *       *
1007 REM *                      *
1008 REM ************************
1050 REM DEKLARACJA ZMIENNYCH
1052 DIM CZ$(15),D$(1),DD$(1),WW
$(4),OD$(5)
1055 DIM TA(4,6),HO$(5),KO$(4),W
A(6),W$(1),T$(1)
1060 WY=0:SUC=0:SUG=0
1065 RESTORE 1088
1066 FOR B=1 TO 6:READ D:WA(B)=D
:NEXT B
1067 WW$="KCTP"
1068 CZ$="               "
1070 KO$(1,1)=CHR$(0):KO$(2,2)=C
HR$(96):KO$(3,3)=CHR$(16):KO$(4,
4)=CHR$(123)
1081 DATA *AS* ,A
1082 DATA *10* ,1
1083 DATA KROL ,K
1084 DATA DAMA ,D
1085 DATA WALET,W
1086 DATA * 9 *,9
1088 DATA 11,10,4,3,2,0
1100 REM ZEROWANIE CZESCI DANYCH
1105 GRAPHICS 0:RE=0:KT=0:PO=1
1150 REM TWORZENIE TABLICY
1155 FOR A=1 TO 4:FOR B=1 TO 6
1160 TA(A,B)=0:NEXT B:NEXT A
1200 REM ILE KART W TALII
1204 ? CHR$(125)
1205 LI=0
1206 FOR A=1 TO 4:FOR B=1 TO 6
1210 IF TA(A,B)=0 THEN LI=LI+1
1215 NEXT B:NEXT A
1220 IF LI=0 THEN 1354
1250 REM LOSOWANIE KARTY
1255 Y=INT(RND(0)*4)+1:X=INT(RND
(0)*6)+1
1300 REM SPRAWDZANIE CZY BYLA
1301 REM JUZ LOSOWANA KARTA
1305 IF TA(Y,X)<>0 THEN 1255
1307 RE=RE+1:BE=1:GOSUB 2005
1310 IF WY=0 THEN TA(Y,X)=1:GOTO
 1352
1315 TA(Y,X)=2
1350 REM POBRANIE DANYCH HONORU
1351 REM KARTY I WYSWIETLENIE NA
1352 REM EKRANIE.

1354 RZ=0:? CHR$(125)
1355 FOR A=1 TO 4:FOR B=1 TO 6
1360 IF TA(A,B)=2 THEN 1385
1365 RESTORE 1080+B
1370 READ HO$
1372 IF LI=0 AND TA(A,B)=1 THEN
2105
1375 IF RE=10 AND TA(A,B)=1 THEN
 RZ=RZ+1:POSITION 5,4+RZ:? HO$;"
 ";KO$(A,A)
1385 NEXT B:NEXT A
1388 IF LI>0 AND RE<10 AND WY=1
THEN WY=0:GOTO 1205
1390 IF LI>0 AND RE<10 AND WY=0
THEN WY=1:GOTO 1205
1400 REM CZY KONIEC ROZDANIA
1405 IF RE=0 AND LI=0 THEN 1500
```

```
1410 REM KTO WYCHODZI
1415 IF KT=1 THEN 1805
1420 POSITION 20,5:? "WYCHODZ !
"
1425 POSITION 20,6:? "---------
"
1430 POSITION 27,8:? " ":POSITIO
N 20,8:? "Kolor ";:INPUT W$:IF W
$<>"M" THEN 1460
1435 FOR A=1 TO 4
1437 IF TA(A,3)=1 AND TA(A,4)=1
THEN 1442
1439 NEXT A
1441 POSITION 11,16:? "Nie oszuk
uJ !":FOR CZ=1 TO 300:NEXT CZ:PO
SITION 11,16:? CZ$:GOTO 1430
1442 POSITION 11,16:? "Masz meld
unek":POSITION 11,17:? "nie klam
iesz !"
1444 SUG=SUG+120-20*A:GOTO 1430
1460 POSITION 20,9:? "Honor ";:I
NPUT D$
1465 REM ODNALEZIENIE KOLORU W
1466 REM TABLICY
1470 FOR A=1 TO 4:IF WW$(A,A)=W$
 THEN 1600
1480 NEXT A:GOTO 1625
1500 REM SUMOWANIE WYNIKOW TABLI
CY
1505 FOR A=1 TO 4:FOR B=1 TO 6
1510 IF TA(A,B)=3 THEN SUG=SUG+W
A(B):GOTO 1520
1515 SUC=SUC+WA(B)
1520 NEXT B:NEXT A:REM *********
1530 ? CHR$(125):GRAPHICS 1:POSI
TION 4,6:? #6;"SUMA PUNKTOW"
1535 POSITION 4,7:? #6;"--------
----"
1540 POSITION 4,9:? #6;"** TY **
 ";SUG
1545 POSITION 4,11:? #6;"KOMPUTE
R ";SUC:SETCOLOR 0,8,14
1550 IF (SUC<1000) OR (SUG<1000)
 THEN ? :? "      Czy grasz dale
J [T/N]";:INPUT T$
1552 IF T$="T" THEN 1105
1555 ? :? "          * KONIEC *"
1560 ? CHR$(125):END :REM ******
1600 FOR B=1 TO 6
1605 RESTORE 1080+B
1610 READ HO$,DD$
1615 IF DD$=D$ AND TA(A,B)=1 THE
N 1645
1620 NEXT B
1625 FOR CZY=8 TO 11:POSITION 20
,CZY:? CZ$:NEXT CZY
1630 POSITION 11,16:? "Nie oszuk
uJ !":FOR CZ=1 TO 300:NEXT CZ:PO
SITION 11,16:? CZ$
1635 GOTO 1430
1640 REM KOMPUTER SPRAWDZA CZY
1641 REM MA TEN KOLOR I HONOR
1645 Y=A:X=B
1648 REM * CZY MA WYZSZY HONOR *
1650 FOR B=1 TO 6
1655 IF TA(Y,B)=2 AND X>B THEN T
A(Y,B)=4:TA(Y,X)=4:KT=1:GOTO 174
5
1660 NEXT B
1663 REM DAJE NAJNIZSZY HONOR
1665 FOR B=6 TO 1 STEP -1
1670 IF TA(Y,B)=2 AND X<B THEN T
A(Y,B)=3:TA(Y,X)=3:KT=0:GOTO 174
5
1675 NEXT B
1678 REM ZRZUTKA GDY BRAK DANEGO
1679 REM KOLORU
1680 B=6
1685 FOR A=4 TO 1 STEP -1
1690 IF TA(A,B)=2 THEN TA(A,B)=3
:TA(Y,X)=3:KT=0:GOTO 1745
1700 NEXT A
1705 B=B-1
1710 GOTO 1685
1745 RE=RE-2
1750 POSITION 5,12:? "KOMPUTER D
AJE:"
1760 RESTORE 1080+B
1770 READ HO$
1785 POSITION 20,12:? HO$;" ";KO
$(A,A):FOR CZ=1 TO 300:NEXT CZ:G
OSUB 2000
1790 GOTO 1205
1800 REM WYCHODZI KOMPUTER
1805 FOR CZY=5 TO 11:POSITION 20
,CZY:? CZ$:NEXT CZY
1810 POSITION 20,5:? "WYCHODZE !
"
1815 POSITION 20,6:? "----------
"
1820 REM CZY NIE NA MELDUNKU
1825 FOR A=1 TO 4
1830 IF TA(A,3)=2 AND TA(A,4)=2
THEN POSITION 11,16:? "MAM MELDU
NEK !":SUC=SUC+120-20*A:X=4:GOTO
 1843
1835 NEXT A
1837 GOTO 1860
1840 REM WYCHODZI W DAME
1843 HO$="":HO$="DAMA"
1845 POSITION 20,9:? HO$;" ";KO$
(A,A)
1850 GOTO 1912
1855 REM WYCHODZI W NAJSTARSZA K
ARTE
1860 FOR A=1 TO 4:FOR B=1 TO 6
1865 IF TA(A,B)=2 THEN 1875
1870 NEXT B:NEXT A
1875 X=B
1880 FOR B=1 TO 6
1885 RESTORE 1080+B
1890 READ HO$
1895 IF X=B THEN 1905
1900 NEXT B
1905 POSITION 20,8:? HO$;" ";KO$
(A,A):GOSUB 2000
1910 REM GRACZ DOKLADA
1912 Y=A
1915 POSITION 20,10:? "KOLOR ";:
INPUT W$
1920 POSITION 20,11:? "HONOR ";:
INPUT D$
1925 REM ODNALEZIENIE KOLORU W T
ABLICY
1930 FOR A=1 TO 4:IF WW$(A,A)=W$
 THEN 1940
1935 NEXT A:GOTO 1915
1940 FOR B=1 TO 6
1945 RESTORE 1080+B
1950 READ OD$,DD$
1955 IF DD$=D$ AND TA(A,B)=1 THE
N 1980
1960 NEXT B
1965 FOR CZY=10 TO 11:POSITION 2
0,CZY:? CZ$:NEXT CZY
1970 POSITION 20,8:? "Nie oszuku
J !":FOR CZ=1 TO 200:NEXT CZ
1975 POSITION 20,8:? CZ$:GOTO 19
15
1980 REM POROWNANIE KART I ZAKOD
OWANIE WYNIKU
1983 RE=RE-2
1985 IF A=Y AND B<X THEN TA(A,B)
=3:TA(Y,X)=3:KT=0:GOTO 1205
1990 TA(A,B)=4:TA(Y,X)=4:KT=1:GO
TO 1205
2000 REM ROZDAWANIE KART
2005 FOR A2=1 TO BE
2010 FOR B2=2 TO 14
2015 SOUND 0,0,0,B2

2020 NEXT B2
2025 SOUND 0,0,0,0
2035 IF PO=0 THEN 2050
2040 NEXT A2
2045 PO=0
2050 RETURN :REM *********
2100 REM ****************
2105 RZ=RZ+1
2110 POSITION 5,4+RZ:? HO$;" ";K
O$(A,A)
2115 GOTO 1385
```

* SUMA KONTROLNA / ETYKIETA *

TE	1000	FW	1002	FK	1004	LV	1006
GQ	1007	UK	1008	IY	1050	UY	1052
GI	1055	JW	1060	SI	1065	EG	1066
WO	1067	JW	1068	RS	1070	HN	1081
YI	1082	AM	1083	EI	1084	EP	1085
BF	1086	CV	1088	EH	1100	ZZ	1105
YI	1150	DG	1155	JR	1160	UL	1200
GP	1204	UE	1205	CX	1206	HY	1210
NZ	1215	HE	1220	TK	1250	QQ	1255
VN	1300	YN	1301	PI	1305	YP	1307
BU	1310	CH	1315	QF	1350	KQ	1351
TI	1352	MP	1354	DK	1355	LV	1360
NI	1365	VZ	1370	OG	1372	YT	1375
OW	1385	TN	1388	SU	1390	QZ	1400
CN	1405	JH	1410	MZ	1415	CK	1420
VN	1425	XJ	1430	VT	1435	QI	1437
DG	1439	ZE	1441	HR	1442	OO	1444
CF	1460	KB	1465	VW	1466	IG	1470
PX	1480	JH	1500	CZ	1505	OV	1510
UN	1515	ZH	1520	TA	1530	PV	1535
WV	1540	PF	1545	LM	1550	LG	1552
DI	1555	ZA	1560	WM	1600	MW	1605
TV	1610	SA	1615	CI	1620	XD	1625
HR	1630	RM	1635	RH	1640	IZ	1641
LN	1645	JG	1648	XB	1650	MP	1655
CU	1660	JH	1663	UU	1665	EQ	1670
DR	1675	YZ	1678	RQ	1679	WI	1680
TS	1685	XL	1690	BT	1700	IJ	1705
WR	1710	XT	1745	SH	1750	MW	1760
WH	1770	OY	1785	RT	1790	BE	1800
VL	1805	TW	1810	VS	1815	GB	1820
VY	1825	TD	1830	CY	1835	VH	1837
TA	1840	YR	1843	TB	1845	TM	1850
WT	1855	DD	1860	QU	1865	OJ	1870
FX	1875	XO	1880	NY	1885	WP	1890
UB	1895	CI	1900	RD	1905	BB	1910
ET	1912	LX	1915	IS	1920	VD	1925
TC	1930	RX	1935	XE	1940	NO	1945
SK	1950	VG	1955	DA	1960	LI	1965
NR	1970	RX	1975	IG	1980	YB	1983
IT	1985	RD	1990	YX	2000	NJ	2005
PK	2010	EU	2015	YZ	2020	HX	2025
GR	2035	YU	2040	XB	2045	WM	2050
TJ	2100	JH	2105	KO	2110	VQ	2115

Rady i porady programisty (cz. 2)

Kontynuując rozpoczęty cykl o programach, które powinny znaleźć się w Twojej biblioteczce programów-narzędzi, przedstawiam program przeznaczony do porządkowania (sortowania) elementów zawartych w tablicy.

```
10 ' ***** PORRZADKOWANIE CIAGOW *****
20 INPUT"DŁUGOSC CIAGU";n
30 DIM x(n)
40 FOR i=1 TO n
50 INPUT x(i)
60 NEXT i
70 PRINT"WARTOSCI WCZYTANEGO CIAGU"
80 GOSUB 140
90 GOSUB 230
100 PRINT"CIAG UPORZADKOWANY"
110 GOSUB 140
120 STOP
```

Poniższy program jest segmentem sterującym pozwalającym na wprowadzenie liczby n — długości ciągu i wartości do tablicy x, wywołanie podprogramu drukującego.

```
130 '***** PODPROGRAM DRUK *****
140 PRINT
150 i=0
160 WHILE i<>n
170 i=i+1
180 PRINT x(i);
190 WEND
200 PRINT
210 RETURN
```

Podprogram drukujący wywoływany jest dwukrotnie, pierwsze wywołanie drukuje elementy nieuporządkowanej tablicy, drugie zaś wartości tablicy po uporządkowaniu. W segmencie sterującym umieszczone jest wywołanie podprogramu sortującego.

```
220 '***** PODPROGRAM SORTOWANIA ******
230 j=n
240 j=INT(j/2)
250 IF j=0 THEN 380
260 s=n-j
270 FOR r=1 TO s
280 i=r
290 v=i+j
300 IF x(i)<=x(v) THEN 360
310 h=x(i)
320 x(i)=x(v)
330 x(v)=h
340 i=i-j
350 IF i>0 THEN 290
360 NEXT r
370 GOTO 240
380 RETURN
```

Podprogram sortowania umożliwia ustawienie elementów tablicy w porządku rosnącym, jeśli badany warunek w linii 300 ma postać x(i) <= x (v), w porządku malejącym warunek ma postać x(i) > x(v).

Jeżeli w linii 30 zmienimy definicję tablicy ze zmiennej numerycznej na łańcuchową, tj. linia 30 przyjmie postać DIM x$ (n) i w miejsce x wpiszemy x$, będziemy mogli porządkować łańcuchy znaków — np. ułożyć alfabetycznie dowolną listę nazwisk.

Jak ważne jest posiadanie tego podprogramu w twojej bibliotece przekonasz się o tym czytelniku jeszcze nieraz.

Zdzisław ŁUKASZEWICZ

```
/****************************/
/*                          */
/*       S K A R B          */
/*         ATARI            */
/*                          */
/****************************/
 main()$(
 char kierunek, z, num[6], pierw[6];
 char wydr[17];
 int c, a, b, x, y, p, q;
 c=iml(); /* inicjal. biblioteki */
 do$(
  x=0; y=0;
  a=rnd(19)-10;
  b=rnd(19)-10;
  printf("\f");
  do$(
   printf("\nKierunek N, S, O, W ?");
   kierunek=getchar();
   getchar();
        if (kierunek=='N') y--;
  else  if (kierunek=='S') y++;
  else  if (kierunek=='W') x--;
  else  if (kierunek=='O') x++;
  else  $( printf("\nUWAGA !");
           for(c=1; c<=30; c++)
           sound(0, 100, 10, 9);
      sound(0, 0, 0, 0);
    printf("\nTylko : N, S, O lub W");
      $)
   q=b-y; p=a-x; z=p*p+q*q;
  c=ifp(z, num);
  c=sqrt(num, pierw);
  c=fasc(pierw, wydr);
   printf("\nDo skarbu %s", wydr);
   for(c=1; c<=30; c++)
   sound(0, 100, 10, 9);
  sound(0, 0, 0, 0);
 $)
 while(z>0);
 printf("\nSkarb : X=%d Y=%d", a, b);
$) while(czydalej());
$)
```

```
PY 590 REM ***********************
YS 592 REM *                     *
DE 594 REM *      S K A R B      *
TT 595 REM *     * ATARI *       *
ZH 597 REM *                     *
QW 598 REM ***********************
YW 600 DIM K$(2):X=0:Y=0
UJ 602 PRINT CHR$(125)
TK 604 A=INT(RND(0)*19)-10
TV 606 B=INT(RND(0)*19)-10
KZ 608 ? "KIERUNEK N,S,O,W ";:INPUT
   K$
CS 610 IF K$="N" THEN Y=Y+1:GOTO 63
   0
HA 612 IF K$="S" THEN Y=Y-1:GOTO 63
   0
HM 614 IF K$="W" THEN X=X-1:GOTO 63
   0
CG 616 IF K$="O" THEN X=X+1:GOTO 63
   0
TS 618 ? :? "UWAGA !":GOSUB 650
RF 620 GOTO 608
NI 630 P=A-X:Q=B-Y:Z=SQR(P*P+Q*Q)
DO 634 IF Z=0 THEN 640
NC 636 ? "DO SKARBU ";INT(Z*100)/10
   0:GOSUB 650:GOTO 608
BN 640 ? :? "SKARB : X=";A;" Y=";B
UR 642 ? :? "SZUKAMY JESZCZE RAZ ?"
XQ 644 ? :? "            ";CHR$(212);
YC 646 ? "/";CHR$(206);:INPUT K$
GC 648 IF K$=CHR$(84) THEN 602
PE 649 END
CP 650 FOR C=1 TO 15:SOUND 0,100,10
   ,9:NEXT C:SOUND 0,0,0,0
ZS 652 RETURN
```

SCHOWEK

„Schowek" — to procedura pomocnicza, która może się przydać przy pisaniu złożonych programów w standardowym BASIC-u ATARI. Umożliwi ona dostęp do tych obszarów pamięci RAM komputerów ATARI XL i XE, które normalnie są niewidoczne z poziomu BASIC-a, gdyż zasłania je ROM systemu operacyjnego lub sam BASIC. Prawie całą tę przestrzeń można wykorzystać do czasowego przechowywania danych. Miejsca do wykorzystania jest sporo, 22 kB „Schowek" organizuje tę pamięć w umowne sektory po 128 bajtów, numerowane od Ø do 175, dostępne — niezależnie od siebie — do zapisu bądź odczytu. Wymiany danych dokonuje się za pomocą funkcji USR z parametrami.

USR (*adres, numer, bufor, zapis*), gdzie:
adres — adres programu maszynowego,
numer — numer sektora,
bufor — adres bufora danych o pojemności 128 bajtów,
zapis — znacznik kierunku operacji: Ø to odczyt z pamięci, 1 to zapis.

Program maszynowy jest przemieszczalny, można więc umieścić go w zmiennej tekstowej w BASIC-u. Numer sektora winien się zawierać w granicach Ø do 175. Podanie zbyt dużego numeru spowoduje niewykonanie przez program maszynowy jakiejkolwiek operacji, funkcja USR przybiera wtedy wartość równą adresowi wywołania. Gdy numer będzie o wiele za duży (powyżej 255), to może nastąpić zapis lub odczyt z sektora o innym niż podany numerze. Można się o tym dowiedzieć z wartości funkcji USR, gdyż jeśli wykonała ona jakieś operacje na pamięci, to przybiera wartość równą numerowi odpowiedniego sektora. Jako adres bufora danych najwygodniej podać adres zmiennej tekstowej.

„Schowek" warto wykorzystywać tylko wtedy, gdy rzeczywiście brakuje miejsca, np. gdy chcemy mieć w pamięci jednocześnie trzy wykresy w grafice 8, a mieści się tylko jeden, albo gdy w programie typu disasemblera zakładamy jednocześnie duży bufor programu i duży zbiór etykiet itp. „Schowka" możemy także użyć do przekazywania danych między różnymi programami uruchamianymi kolejno, bez wyłączania komputera. W każdym z takich wypadków trzeba poświęcić trochę pracy na szczegółowe rozwiązanie współpracy programu w BASIC-u ze „Schowkiem", stosownie do potrzeb.

Dołączony program w BASIC-u zawiera procedurę maszynową w liniach DATA, oraz ilustruje samą zasadę wykorzystania „Schowka". Drugi program, w składni dostosowanej do Atari Macroassembler'a.

Robert MIZIŃSKI

```
0 REM Program SCHOWEK
1 REM Autor: Robert Mizinski
2 REM
10 REM DEMONSTRACJA DZIALANIA SCHOWKA
20 DIM SCHOWEK$(96)
30 FOR I=1 TO 96:READ A:SCHOWEK$(I)=CHR$(A):NEXT I
40 DATA 104,104,104,201,176,144,5,104,104,104,104,96,162,0,133
41 DATA 212,134,213,74,144,2,162,128,134,216,24,105,160,201,208
42 DATA 144,2,105,7,133,217,104,133,215,104,133,214,104,104,208
43 DATA 16,166,216,165,214,134,214,133,216,166,217,165,215,134,215
44 DATA 133,217,165,20,197,20,240,252,120,173,1,211,73,3,141
45 DATA 1,211,160,127,177,214,145,216,136,16,249,173,1,211,73
46 DATA 3,141,1,211,88,96
50 ? "Demonstracja:":? " tekstowa (1) lub graficzna (2)"
60 ? :? "Wybierz cyfre",:INPUT I
70 IF I=2 THEN 300
80 REM DEMONSTRACJA TEKSTOWA
90 DIM BUFOR$(128)
100 FOR I=1 TO 128:BUFOR$(I,I)="█":NEXT I
105 ? " ZAPIS CZY ODCZYT?":? "(1 - ZAPIS, 0 - ODCZYT)"
110 INPUT ZAPIS:IF  NOT ZAPIS THEN 130
120 ? "WPROWADZ ZNAKI DO BUFORA":INPUT BUFOR$
130 ? "KTORY NUMER SEKTORA ";:INPUT SEKTOR
140 X=USR(ADR(SCHOWEK$),SEKTOR,ADR(BUFOR$),ZAPIS)
150 IF X=ADR(SCHOWEK$) THEN ? "ZLY NUMER":GOTO 100
160 IF ZAPIS THEN ? "ZAPISANO";:GO TO 180
170 ? "ODCZYTANO";
180 ? " SEKTOR ";X
```

```
190 IF  NOT ZAPIS THEN ? "OTO BUFOR:":? :? BUFOR$
200 GOTO 100
290 REM DEMONSTRACJA GRAFICZNA
300 GRAPHICS 8:COLOR 1
310 FOR I=0 TO 159 STEP 2:PLOT 0,I:DRAWTO 319,I:NEXT I
320 ? "Zapis do schowka ekranu 1"
330 ZAPIS=1:SEKTOR=1:GOSUB 500
400 GRAPHICS 8:COLOR 1
410 FOR I=1 TO 319 STEP 2:PLOT I,0:DRAWTO I,159:NEXT I
420 ? "Zapis do schowka ekranu 2"
430 ZAPIS=1:SEKTOR=51:GOSUB 500
440 ? "Odczyt ze schowka ekranu 1"
450 ZAPIS=0:SEKTOR=1:GOSUB 500
460 ? "Odczyt ze schowka ekranu 2"
470 ZAPIS=0:SEKTOR=51:GOSUB 500
480 GOTO 440
490 REM ZAPIS/ODCZYT EKRANU ZE SCHOWKA
500 EKRAN=PEEK(88)+256*PEEK(89)
510 FOR I=0 TO 49
520 X=USR(ADR(SCHOWEK$),SEKTOR,EKRAN+I*120,ZAPIS)
530 SEKTOR=SEKTOR+1
540 NEXT I
550 RETURN
```

```
;Procedura "Schowek"
;Autor: Robert Mizinski
;
;Warunki wejscia:
;Na stosie lezy od gory: ile parametrow,
;numer sektora HI, LO,
;adres bufora danych HI, LO,
;znacznik zapis-odczyt HI, LO.
;
RTCLOK  EQU     $14         ;definicje
PORTB   EQU     $D301       ;etykiet
FR0     EQU     $D4         ;systemowych
ZRODLO  EQU     FR0+2       ;rezerwacja
CEL     EQU     FR0+4       ;strony 0
        ORG     $600
        PLA                 ;ile parametrow
        PLA                 ;HI numeru sektora
        PLA                 ;LO numeru
        CMP     #[$50*2]    ;sprawdz wartosc
        BCC     DOBRZE      ;numeru
ZLE     PLA: PLA: PLA: PLA: RTS
DOBRZE  LDX     #0          ;zwrot numeru
        STA     FR0         ;sektora
        STX     FR0+1       ;do BASIC'a
        LSR     A           ;oblicz LO
        BCC     JESTLO      ;rzeczywistego
        LDX     #$80        ;adresu
JESTLO  STX     CEL         ;schowka
        CLC                 ;oblicz HI adresu
        ADC     #$A0
        CMP     #$D0        ;omin obszar I/O
        BCC     PODD0
        ADC     #7
PODD0   STA     CEL+1
        PLA
        STA     ZRODLO+1
        PLA
        STA     ZRODLO
        PLA                 ;HI znacznika
        PLA                 ;LO znacznika
        BNE     PRZESL      ;idz jesli zapis
        LDX     CEL         ;tu odczyt
        LDA     ZRODLO      ;zamiana
        STX     ZRODLO      ;celu ze zrodlem
        STA     CEL
        LDX     CEL+1
        LDA     ZRODLO+1
        STX     ZRODLO+1
        STA     CEL+1
PRZESL  LDA     RTCLOK      ;czekanie na brak
CZEKAJ  CMP     RTCLOK      ;przerwan
        BEQ     CZEKAJ
        SEI
        LDA     PORTB       ;zamiana ROM(=)RAM
        EOR     #%00000011
        STA     PORTB
        LDY     #$7F        ;licznik
PETLA   LDA     (ZRODLO),Y
        STA     (CEL),Y     ;przeslanie danych
        DEY
        BPL     PETLA
        LDA     PORTB       ;zamiana RAM(=)ROM
        EOR     #%00000011
        STA     PORTB
        CLI
DODOSA  RTS                 ;powrot do BASIC'a
        END     DODOSA      ;lub DOS'a
```

INFORMATYKA

Temat nr 2: REPREZENTACJA INFORMACJI W KOMPUTERZE

1. System binarny, a cyfrowy zapis czarno-białego obrazu.
2. Liczby w systemie binarnym i dziesiętnym.
3. Interpretacja terminów: bit i bajt.
4. Przykłady operacji artymetycznych i logicznych na liczbach zapisanych w systemie binarnym.
5. Interpretacja „kodu znaku" i jego graficznego zapisu. Kod ASCII.
6. Atrybuty pikseli.

Temat nr 3: BUDOWA MIKROKOMPUTERA cz. 1

1. Pamięć RAM. Podstawowe parametry (pojemność, czas cyklu) i przeznaczenie.
2. Pamięć ROM — przeznaczenie.
3. Mikroprocesor, przeznaczenie, parametry.
4. Różnice między procesorami (zegar, liczba bitów).

Propozycja materiałów pomocniczych:

1. Komputery osobiste, D. Madej i inni, Warszawa WKiŁ 1987.
2. Mikrokomputery, A. J. Dirksen, red., Warszawa 1985, WKiŁ.
3. Sami tworzymy symbole graficzne, M. Skonieczny, R. Zugehor, „IKS" nr 6/1988, s. 26.
4. Mikroprocesory 32-bitowe, A. J. Piotrowski, „Komputer" nr 4/87, s. 38

SPRZĘT INFORMATYCZNY cz. 1

(Materiały do podstaw informatyki cz. 2)

Usiłując zrozumieć znaczenie podstawowych parametrów sprzętu informatycznego nie można, niestety, uniknąć tematu: „system dwójkowy (binarny)".

Podczas ostatnich wakacji miałem okazję rozmawiać z Anglikiem. Rozmowa zeszła na temat mieszkań (tu i tam) i ich wielkości. Po krótkiej chwili okazało się, że porównanie wielkości mieszkań stanowi nie lada problem. Przyzwyczajeni jesteśmy bowiem do różnych miar. Dla niego 10 m² było tak abstrakcyjnym pojęciem, jak dla mnie 50 stóp kwadratowych. Na szczęście w rozmowie tej pomogła nam 10-letnia córka mego rozmówcy. Zna bowiem (ze szkoły) relacje obu miar.

Wydaje się, że przykład ten ilustruje kłopoty zrozumienia zwrotu „pojemność pamięci komputera" lub uświadomienie różnic między procesorami 8- a 16-bitowymi, osobie która nie zna pojęcia bit lub bajt.

W poprzednim nrze „IKS-a" omówiona została relacja między rzeczywistym obrazem, a jego cyfrową reprezentacją (zero-jedynkową) w pamięci komputera. W ten sposób zostało wskazane, iż zera i jedynki nie są abstrakcyjnymi pojęciami — odzwierciedlają bowiem czarne i białe punkty obrazu komputerowego lub inaczej mozaikę tworzącą wyświetlany na monitorze obraz. Zatem pamięć maszyny przechowuje informacje w postaci zer i jedynek.

Tezy do zajęć z Podstaw Informatyki

(Materiały do Podstaw Informatyki cz. 2)

Oczywiście w pamięci komputera zapisywane są nie tylko obrazy — przede wszystkim liczby, liczby i jeszcze raz liczby. W jaki sposób? Skoro dysponujemy tylko zerami i jedynkami. Możliwe jest to dzięki systemowi dwójkowemu, który różni się od stosowanego przez nas na co dzień tym, że wykorzystywane są w nim właśnie tylko dwa znaki **1** i **0** — stąd nazwa systemu — „dwójkowy", „binarny".

Zatem w komputerze każdą liczbę zapisujemy nie kombinacją dziesięciu cyfr (od 0 do 9), a tylko dwóch — zero i jeden. Zasada binarnego zapisu liczb jest prosta i wygląda następująco:

System dziesiętny	System binarny (komputerowy)
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
itp.	

Należy zwrócić uwagę, że każdej cyfrze dziesiętnej odpowiada aż osiem znaków w zapisie maszynowym — ogromne marnotrawstwo! Osiem znaków, by zapisać jeden! Rzeczywiście tak jest. Pamięć komputera zbudowana jest z „szufladek", niezmiennej wielkości „kontenerków", każdy z nich to osiem elementów pamiętających zero lub jedynkę. Zapisując zatem w komputerze liczbę 123 wykorzystujemy aż 24 elementy pamiętające — BITY lub inaczej trzy kontenerki — BAJTY. Czyli zapis owej liczby wygląda następująco:

00000001 (pierwszy bajt)
00000010 (drugi bajt)
00000011 (trzeci bajt)

Oczywiście istnieją inne, bardziej ekonomiczne, sposoby zapisywania liczb, jednak ze względu na ogólny charakter niniejszych rozważań proponuję poszukać ich opisu w dowolnej książce mającej w tytule „komputer" lub „mikrokomputer".

Wydawać by się mogło, że binarny zapis liczb jest nadzwyczaj sztuczny. Jak bowiem dodawać takie wartości, mnożyć, dzielić? Okazuje się, że nie jest to wcale trudne. Na przykład przy dodawaniu należy tylko pamiętać, że: 0+0=0, 1+0=1, 0+1=1 i 1+1=... właśnie, przecież nie ma znaku 2, czyli wynik należy zapisać w postaci „10".
1+2 w komputerze wykonuje się:

```
 0000001
+0000010
 0000011 — to jest właśnie 3.
```

```
3+3 —  00000011
      +00000011
       00000110    czyli 6.
```

System binarny pozwala w sposób bardzo prosty wykonywać wiele ciekawych i użytecznych operacji. Na przykład dzielenie lub mnożenie przez 2, 4, 8, itp., polega na „przesuwaniu" określonej liczby w lewo lub w prawo.

4_{10} = 00000100_2 dzieląc przez dwa, przesuwamy w prawo liczbę dwójkową o jedno miejsce — $00000010_2=2_{10}$. Podobnie mnożenie przez dwa polega na przesunięciu, ale w lewo, zatem w wyniku otrzymujemy: $000001000_2=8_{10}$ (oznaczenia, wskaźniki wyróżniają system w jakim zapisana jest liczba).

Wydaje się, że możemy teraz uznać, iż rozumiemy sposób zapisu czarno-białych obrazów oraz liczb w komputerze. Wiadomo jednak, że korzystając z maszyny, w zasadzie nigdy nie widzimy bitów. Na ekranie monitora komputera oglądamy normalne znaki, co prawda brakuje zazwyczaj wśród nich polskich liter (ń, ś itp.), ale dwójka to „2", a litery są podobne do tworzących te słowa. Jak to jest możliwe?

W tym miejscu bardzo ważne jest poprawne zrozumienie pojęcia „kodu znaku" i odpowiadającego mu „obrazka".

Jeśli idzie o „obrazek znaku", wiemy już, że w pamięci komputera tworzy go mozaika zer i jedynek, a na ekranie odpowiednio ułożone białe i czarne „punkty" (piksele). Powszechnie, a konkretnie w standardzie ASCII („aski" obrazków cyfr, małych i dużych liter, znaków interpunkcyjnych jest aż 96 (128—32 znaki sterujące). Każdy z tych obrazków ma swój kod lub inaczej „liczbową nazwę". Na przykład cyfry od 0 do 9 mają kody od 48 do 57, małym literom od „a" do „z" odpowiadają kody od 97 do 122, dwukropek to 58, itp.

Zatem w jednej części komputera zapisane mamy obrazki znaków. W drugiej natomiast kody znaków tworzące wprowadzane przez nas do komputera słowa. Na przykład w maszynie słowo „IKS" zapisane jest w trzech bajtach (umownych kontenerkach, każdy po 8 bitów!). W pierwszym bajcie mamy kod litery I — 73, w drugim — 75 (K) i w trzecim 83 (S). W postaci binarnej wygląda to następująco:

1001001 (I)
1001011 (K)
1010011 (S)

Przykład ten jeszcze raz zwraca uwagę na to, iż w pamięci komputera zapisywane są tylko liczby! Liczby te — to owe kody lub inaczej „nazwy" znaków, które maszyna potrafi wyświetlić. Natomiast obrazy znaków, a właściwie ich szablony, czcionki, komputer ma

A W SZKOLE

zapisane w specjalnej „trwałej" części pamięci.

Nasuwa się w tym momencie pytanie — a gdyby zmienić tylko szablony? Właśnie tak się robi! Zmieniając pamięć „trwałą" ze starymi wzorcami na nową, np.: gdzie zamiast danych o postaci obrazka znaku „&" (kod — 38) odzwierciedlona będzie, odpowiednio ułożonymi jedynkami, postać naszej polskiej litery „ł". Zatem, gdy komputer „zechce" wyświetlić znak o kodzie 38, „pobierze" ze swej pamięci, odpowiadający temu kodowi obrazek, szablon litery „ł". Czyli na ekranie pojawi się nie znak „&", a litera „ł". W podobny sposób można zmienić postać wszystkich szablonów, tak by komputer „posługiwał się" cyrylicą lub znakami chińskimi.

Kończąc tematykę relacji: obraz na ekranie — jego cyfrowy zapis w pamięci komputera, celowym wydaje się przypomnienie, iż często na ekranie monitora ukazują się migające, a nawet kolorowe obrazki. Uzyskuje się to dzięki odpowiedniej elektronice oraz „mniej oszczędnemu" wykorzystaniu pamięci komputera. Upraszczając problem — każdy punkt obrazu, który dotychczas zapisywany był jako zero lub jedynka (w jednym bajcie mieściło się aż 8 punktów obrazu) zapisywany jest w jednym **bajcie**. Oznacza to, iż każdemu punktowi obrazu można przyporządkować liczbę mieszczącą się w jednym bajcie — od 0 do 256, innymi słowy 257 kolorów — od bieli przez szarości, żółty, niebieski, czerwony i inne, do czerni. Dzieje się to niestety kosztem wykorzystanej do tego celu pamięci. Przy takim sposobie zapisu obrazu potrzebujemy jej osiem razy więcej (nic za darmo!). Dlaczego 256? W każdym bajcie można zapisać 256 różnych kombinacji zer i jedynek.

Wydaje się, że można już uznać, że przebrnęliśmy jeden z najmniej ciekawych etapów naszego wstępu do informatyki. Było to jednak niezbędne, choćby po to, by móc odpowiedzieć na pytanie: co możemy zapisać w pamięci wielkości 48 000 bajtów (pamięć ZX SPECTRUM)? Przyjmując, że kartka maszynopisu ma 30 wierszy po 60 znaków, to w pamięci tej wielkości można zarejestrować... ponad 26 stron. Mało!

Sądzę, że usiłując omówić budowę mikrokomputera (bardzo ogólnie) uzasadnione jest rozpoczęcie od pamięci. Wiemy już bowiem do czego służy (przechowywanie informacji w postaci liczb binarnych), potrafimy także szacować jej wielkość (bajtami). Praktycznie stosuje się jednak nieco większe miary pojemności pamięci:

1 KB (kilobajty) tj. około 1000 bajtów (dokładnie (1024)

1MB (megabajty) — 1 000 000 bajtów

1 GB (gigabajt) — 1 000 000 000 bajtów.

Typowy IBM PC/XT wyposażony jest w pamięć 512 KB (tj. 512 000 bajtów).

Należy podkreślić, iż jest szereg różnych pamięci. Umownie dzielimy je na: zewnętrzne i wewnętrzne. Na początek zajmiemy się pamięcią wewnętrzną — o niej to właśnie dotychczas mówiliśmy. Tworzą je małe elementy elektroniczne, tak zwane układy scalone („kości") wielkości niewielkiej gumki do wycierania. W komputerach klasy IBM PC wykorzystuje się najczęściej kostki o pojemności 64 i 256 Kb (kilobity). Obecnie dostępne są także pojemniejsze kostki — nawet do jednego mega! Mówimy teraz o tak zwanych kostkach RAM. Tworzą one pamięć dostępną dla nas — użytkowników komputera. Możemy w niej zapisywać informacje i odczytywać. RAM zazwyczaj przechowuje aktualnie wykonywany program oraz przetwarzane dane (np. kolumny sumowanych liczb).

Poza pojemnością ważna jest szybkość pamięci, to znaczy czas wyszukania i odczytania (lub zapisania) np. jednego z 500 000 znaków (które wcześniej były zapisane w pamięci). Czas ten (czas cyklu) zazwyczaj wynosi mniej niż 400 ns (ns—nano sekund tj. 10^{-9}s.). Oznacza to, że podczas jednej sekundy można użyć tę pamięć (czytając z niej lub zapisując) aż dwa i pół miliona razy (obecnie dostępne są oczywiście także znacznie szybsze pamięci).

Niestety nie wszystkie cechy RAM są takie „sympatyczne". Wystarczy bowiem, że wyłączą prąd i wszystkie zapisane w niej pracowicie informacje bezpowrotnie giną! Przydała by się zatem taka pamięć, która nie gubi in-

...aż człowiek stworzył komputer

Szybki w latach osiemdziesiątych rozwój techniki komputerowej, powstanie mikrokomputerów mających zdolności obliczeniowe dużych komputerów z lat siedemdziesiątych, spowodowały ogromne zainteresowanie możliwościami coraz doskonalszej techniki obliczeniowej. A jakie były początki urządzeń liczących. Kogo i co możemy dzisiaj uważać za antenatów czy protoplastów współczesnych twórców komputerów i coraz doskonalszych maszyn liczących.

Za przodków komputera uznać należy wszystkie urządzenia służące do liczenia. Zanim jednak zaczęto myśleć o urządzeniach musiano nauczyć się liczyć. Zaczynano od liczenia na palcach. Liczby oznaczano węzełkami, muszelkami czy kamyczkami. Przodkiem komputera było z pewnością liczydło wynalezione 5000 lat temu, w Babilonie i rozpowszechnione następnie w Chinach, Japonii, Egipcie, Grecji i Rzymie. Około 3000 roku p.n.e. powstał pierwszy przyrząd do przeprowadzania obliczeń zwany abakusem. Początkowo były to deski posypane piaskiem, na których zaznaczano kreskami zliczane ilości, a następnie deski z wyżłobionymi rowkami, w których przesuwano przedmioty np. kamyczki. Około 200 roku p.n.e. w starożytnej Grecji opracowano „kalkulator analogowy" używany do obliczeń ruchu gwiazd i planet. W sto lat później **Heron z Aleksandrii** opracował licznik drogowy, zwany drogomierzem Herona, który dokonywał samoczynnego pomiaru długości drogi przebytej przez pojazd.

Pod koniec pierwszego wieku n.e. Arab **Alfazari** zastosował cyfry 0,1,2,3,4,5,6,7,8,9 używane do dnia dzisiejszego. Cyfry są pochodzenia indyjskiego, do Europy Zachodniej zostały przeniesione w X—XII wieku (według niektórych publikacji w 1226 roku). W 960 roku **Gerbert z Francji** udoskonalił **abakus**, co pozwoliło na dokonywanie obliczeń pozycyjnych. Około 1200 roku wprowadzono pierwsze liczydła skonstruowane na bazie stosowanego powszechnie abakusa. W 1464 roku niemiecki matematyk i astronom Regiomontanus wprowadził tablice sinusów dla ułatwienia mnożenia dwu liczb wielocyfrowych. W 1494 roku Włoch **L. Pacioli** opisał system księgowości podwójnej, uważany za początek prowadzenia ewidencji o zjawiskach gospodarczych w sposób kompletny i ciągły. W 1614 roku szkocki matematyk **John Napier** wprowadził pojęcie logarytmu. Opracowane tablice logarytmów uprościły czynności mnożenia, dzielenia, potęgowania i pierwiastkowania. W 1617 roku John Napier zastosował sztabki rachunkowe do obliczania wielokrotności liczb całkowitych (metodą dodawania logarytmów).

Sztabki rachunkowe znane są również jako „kostki Napiera". W 1620 roku **E. Gunter** wynalazł suwak logarytmiczny, mający dwie ruchome skale logarytmiczne do wykonywania mnożenia i dzielenia. Należy tu zaznaczyć, że w niektórych publikacjach Gunterowi przypisuje się wynalezienie skali logarytmicznej, natomiast suwaka logarytmicznego — **Oughtredowi** (w 1621 lub 1624). W 1623 roku profesor języków biblijnych i astronomii w Tybindze **Wilhelm Schickard** wynalazł pierwszą maszynę do dodawania i odejmowania (a więc pierwszy arytmometr), która wykonywała przeniesienie liczb dziesiętnych przez sześć miejsc.

Można było również, za pomocą tabliczki mnożenia, wykonywać operacje mnożenia i dzielenia. Niezależnie od Wilhelma Schickarda w 1645 roku francuski matematyk i filozof **Blaise Pascal** przedstawił skonstruowaną w 1642 roku maszynę do sumowania, opartą na zasadzie obrotowych kół zębatych. Liczby były przedstawione za pomocą wielkości kątowych. Arytmometr Pascala miał możliwość automatycznego przeniesienia dziesiętnego, wykonywania odejmowania przez odwrotny obrót tarczy liczącej oraz mnożenia przez wielokrotne dodawanie. Urządzenie to było maszyną jednookresową, tzn. ustawianie cyfr liczby i otrzymanie wyniku realizowane było w tym samym cyklu pracy. W 1666 roku skonstruował maszynę do sumo-

INFORMATYKA W SZKOLE

formacji także wówczas, gdy wyłączymy zasilanie. Rolę tę pełnią pamięci zewnętrzne (o czym będzie mowa innym razem) oraz tak zwana pamięć ROM. Trwałość zapisu w ROM uzyskana jest niestety kosztem możliwości zapisania w niej czegokolwiek. Oznacza to, że informacje zawarte w niej są niezmienne — to co wpisał do ROM producent można czytać tysiące razy, ale nigdy zmienić. Pamięć Tylko Do Czytania. Potrzebę takiej pamięci uzasadniają „szablony znaków" — raz zapisane w fabryce mogą niezmiennie być używane przez lata. Pozostałe cechy pamięci ROM są podobne do RAM.

Wracając do szybkości RAM, po co tak duża? Kto w pełni wykorzysta tę cechę? Jest jeden jej „współpracownik", w zasadzie „pracodawca" — **mikroprocesor** (procesor) — także układ scalony. Praktycznie to w nim wykonywane są wszystkie operacje związane z przetwarzaniem informacji. Układ ten między innymi dodaje, mnoży, dzieli, porównuje. Czynności te (i wiele innych) wykonywane są bardzo szybko, w takt tak zwanego **zegara**. W popularnym procesorze 8066 wymusza on (zegar) wykonanie do 10 milionów mikrooperacji (10 Mega) w czasie jednej sekundy. Jest to jednak już przeszłość, dzisiaj bowiem coraz częściej słychać o procesorach z zegarem powyżej 20 Mega. Co to praktycznie oznacza? Każdy takt zegara to wykonanie przez procesor jednej mikrooperacji typu: 1. Pobierz z RAM pierwszy argument, 2. Pobierz z RAM drugi argument, 3. Dodaj oba argumenty, 4. Wynik zapisz w RAM. To przykładowe dodawanie dwóch liczb składa się z czterech mikrooperacji. W wypadku, gdy zegar ma szybkość 4 Mega — czas wykonania przez procesor operacji dodawania (zgodnie z podanym przykładem) wyniesie 1/1000000 s. Owe mikrooperacje, to nic innego, jak czynności wykonywane przez nas, gdy korzystamy z kalkulatora — wprowadzanie argumentów i zapisywanie wyniku w zeszycie. Różnica polega głównie na szybkości. Aby wprowadzić do kalkulatora liczbę, jedna sekunda może się okazać zbyt krótkim czasem — komputer wykonuje tę czynność co najmniej milion razy szybciej. Możliwe jest to dzięki temu, iż mikroprocesor sam odczytuje argumenty — stąd potrzeba dużej szybkości RAM, aby procesor się nie „nudził" czekając na podanie kolejnych argumentów.

Argumenty, to oczywiście liczby — jak duże? Zależy to od procesora. Taki, który może operować na ośmiu bitach jednocześnie — na przykład dodawać liczby mieszczące się w jednym bajcie każda, nazywamy **procesorem ośmiobitowym.** Zatem procesor taki w jednej chwili może dodawać liczby nie większe od wartości równej 256 — taką bowiem maksymalną liczbę można zapisać w ośmiu bitach. Gdy chcemy sumować większe liczby, robi się to, „na raty". Na przykład najpierw komputer dodaje jedności, potem dziesiątki, itp. Wpływa to znacznie na szybkość działania komputera. Stąd postęp technologii budowy komputerów widoczny jest w budowie szesnastobitowych procesorów (np.: 80286) — mogą operować one, w każdym takcie, liczbami o wielkości do 32 000. Na przykład operację 31245:13567 komputer 16-bitowy z zegarem 10 Mega wykona w czasie: 1 s/10^7 x-liczba mikrooperacji niezbędnych do wykonania operacji dodawania.

Obecnie coraz częściej słychać o procesorach 32-, a nawet 64-bitowych!

Wszystko jest jasne, prawie wszystko — bo skąd on (procesor) wie, i to tak szybko, gdzie znajdują się argumenty (mogą przecież być ich setki tysięcy) i co z nimi zrobić? Dzielić? Dodawać? Co zrobić z wynikiem? Gdzie go zapisać? Przecież pamięć jest bardzo duża. Odpowiedzi na wszystkie pytania (także na inne) bardzo sprawnie i szybko (tak by się procesor nie nudził) udziela RAM. Odpowiedzi te, to instrukcje naszego programu komputerowego — jednego z tematów naszego następnego spotkania.

Włodzimierz GOGOŁEK

---

wania, którą następnie ulepszył **Ch. Mahon** wprowadzając mechanizm przeniesień.

W 1674 roku niemiecki matematyk **Gottfried Wilhelm Leibniz** opracował model arytmometru działającego na zasadzie walców schodkowych stosowanych także we współczesnych konstrukcjach. Konstrukcja modelu przewidywała wykonywanie 4 działań arytmetycznych tj. mnożenia, dzielenia, dodawania i odejmowania. Arytmometr był maszyną dwuokresową, tzn. w jednym cyklu ustawiało się cyfry liczby, natomiast w drugim — za pomocą ruchu korbką ustawione cyfry, liczby były wprowadzone do licznika.

W następnych latach G. W. Leibniz opisał operacje rachunkowe w systemie dwójkowym oraz opracował projekt maszyny liczącej w systemie dwójkowym, by w 1694 roku skonstruować pierwszy działający model arytmometru. Udoskonalony przez **Charlesa Thomasa de Calmara** arytmometr Leibniza był pierwszą, sprzedawaną maszyną liczącą.

W 1784 roku francuski matematyk **L. Carnot** założył pierwsze w świecie biuro obliczeniowe. Na początku XIX wieku **Joseph M. Jacquard** wynalazł automatyczny warsztat tkacki, w którym zastosował zasadę przedstawiania informacji przez rozmieszczenie otworów w karcie papierowej. Za pomocą tych otworów realizowane było sterowanie zmian koloru przędzy przy produkcji wielobarwnej tkaniny. Powszechnie uważa się, że zastosowana przez Jacquarda papierowa karta dziurkowana stała się pierwowzorem kart dziukowanych.

W dalszym wyliczaniu nazwisk wybitnych uczonych i wynalazców warto dłużej zatrzymać się przy nazwiskach **Charles'a Babbage'a** i **Hermana Holleritha**. Angielski uczony Charles Babbage opracował koncepcję, a następnie zbudował „machinę różnicową przeznaczoną do obliczania i drukowania tablic matematycznych" (np. tablic kwadratów liczb, dużych tablic wielomianów, itp.). W 1833 roku zaprojektował „maszynę analityczną" będącą ideą automatycznej maszyny liczącej ze sterowaniem programowym. Maszyna ta nie została jednak wykonana, gdyż potrzebne do jej budowy części mechaniczne, w tym przede wszystkim koła zębate, nie mogły być wykonane dostatecznie precyzyjnie.

Maszyna analityczna Ch. Babbage'a zawierała wszystkie elementy współczesnego komputera. Program oraz dane miały być wprowadzane do pamięci z kart dziurkowanych (według zasady Jacquarda). Mogły być przechowywane w pamięci, a także istniała możliwość modyfikacji obliczeń w zależności od użytkowych wyników.

W 1887 roku amerykański wynalazca Herman Hollerith opracował mechaniczny system rejestrowania danych przez wycinanie dziurek w długim pasku taśmy papierowej oraz zbudował maszynę sortującą do zliczania opracowanych danych. Dziurkowany pasek papieru przesuwany był ręcznie z szybkością 50—75 kart na sekundę nad pojemnikami z rtęcią. Rząd igieł teleskopowych badał kartę i w wypadku stwierdzenia dziurki zamykał obwód elektryczny, powodując w ten sposób zwiększenie stanu odpowiedniego licznika o 1. Wynalazek Holleritha zastosowano w pracach statystycznych, a przede wszystkim do opracowania wyników spisów ludności (1890 rok — USA, 1891 rok — Kanada, 1897 rok — Rosja). Było to pierwsze urządzenie, zastosowane na tak szeroką skalę do przetwarzania danych. Pozwoliło na skrócenie czasu opracowania wyników z 6 lat do 6 tygodni. Po raz pierwszy zastosowano masowo technikę kart dziurkowanych. W 1896 roku Herman Hollerith założył przedsiębiorstwo Tabulating Machine Company, które zajęło się produkcją i sprzedażą maszyn jego konstrukcji. Przedsiębiorstwo to po reorganizacji i z dwoma innymi przedsiębiorstwami przyjęło w 1911 roku nazwę Computing. Tabulating-Recording Company zmienioną w 1924 roku na International Business Machines Corporation (IBM). W tym też roku po raz pierwszy na świecie uruchomiono produkcję maszyn licząco-analitycznych. Firma IBM jest obecnie największym na świecie producentem komputerów wraz z oprogramowaniem i sprzętu komputerowego.

Warto zwrócić uwagę, że w Polsce po raz pierwszy do prac statystycznych zastosowano maszyny licząco-analityczne w roku 1921. Prace nad stworzeniem pierwszego komputera prowadziło wiele zespołów naukowo-badawczych. W 1936 roku **Allan M. Turing** przedstawił teoretyczny model uniwersalnej, abstrakcyjnej maszyny liczącej zwanej później „maszyną Turinga". W tym samym czasie **John von Neumann** zaproponował zastosowanie w maszynach liczących dwójkowego systemu liczenia w miejsce stosowanego dotychczas systemu dziesiętnego, a Ludwik Conffigual opisał sterowaną programem maszynę cyfrową mającą pracować w systemie dwójkowym. W 1939 roku Georg Stibitz skonstruował przekaźnikową, cyfrową maszynę liczącą nazwaną Complex Computer, a rok później pokazał współpracę prostej maszyny przekaźnikowej połączonej linią telefoniczną z końcówką. Wydarzenie to traktowane jest obecnie jako pierwsze zastosowanie zdalnego dostępu. W 1941 roku **Kondrat Zu-**

se opracował pierwszą całkowicie sterowaną programowo maszynę cyfrową nazwaną 23, pracującą na przekaźnikach, uznaną nawet przez niektórych uczonych za pierwszy, pracujący komputer.

W latach 1939—1944 zespół pod kierownictwem Amerykanina **Howarda Aikena** z uniwersytetu Harwarda we współpracy z firmą IBM prowadził prace nad skonstruowaniem maszyny elektromechanicznej (przekaźnikowej) opartej na idei maszyny liczącej Charlesa **Babbage'a**. Zbudowaną maszynę, którą nazwano Automatic Sequevice Controlled Colenlator, sterowano automatycznie za pomocą programu zapisanego na dwudziestoczterościeżkowej taśmie dziurkowanej. Maszyna ta zwana również Mark I uważana jest za pierwszy komputer. Maszyny pracujące na przekaźnikach elektromechanicznych zaliczane są obecnie do zerowej generacji komputerów.

W 1946 roku **J. M. Brainerd, J. P. Eckert, J. W. Mauchly** i **H. H. Goldstine** z Uniwersytetu Pensylwańskiego zakończyli budowę pierwszej na świecie elektronicznej maszyny cyfrowej do celów obliczeniowych nazwanej ENIAC (Elektronic Integrator and Computer), która zapoczątkowała erę współczesnych, elektronicznych komputerów. W pierwszych komputerach elektronicznych używano lamp elektronowych. Maszyny te zajmowały bardzo dużo miejsca, pobierały bardzo dużo prądu i były zawodne np. ENIAC zbudowany z około 1800 lamp elektronowych zajmował przestrzeń 150 m² i pobierał 150 KW energii. Mimo zawodności ENIAC pracował jak na owe czasy bardzo szybko. W czasie pracy wykonywał 10000 operacji w każdej sekundzie. Dla przykładu podamy, że czas dodawania dwóch liczb dziesiętnych, dziesięciopozycyjnych wynosił około 0,2 ms. ENIAC zbudowany na potrzeby amerykańskiej artylerii do wykonywania obliczeń balistycznych został praktycznie zastosowany w 1947 roku po zainstalowaniu na poligonie Aberdeen w Instytucie Balistyki Szefostwa Artylerii Armii USA. Na kolejne wersje lepszych maszyn nie trzeba było długo czekać. Już w rok po zainstalowaniu ENIAC-a John von Neuman przedstawił koncepcję komputera cyfrowego operującego na liczbach binarnych. Według koncepcji von Neumana program powinien być zapamiętywany w pamięci operacyjnej. Rozkazy i dane należało przedstawić w pamięci operacyjnej jednakowo, tzn. w postaci liczb binarnych, a rozkazy powinny być wykonywane automatycznie. Opierając się na tej koncepcji zbudowano na potrzeby armii amerykańskiej komputer EDVAC (Electronic Discrete Varizble Automatic Computer). Pierwszym komputerem z pamiętanych programem był skonstruowany w 1948 roku pod kierunkiem **M. W. Wilkesa** z uniwersytetu Cambridge komputer EDSAC (Electronic Delay Starage Automatic Calculator).

W 1949 roku zbudowano pierwsze komputery uniwersalne umożliwiające zastosowanie w problematyce przetwarzania danych. Były to skonstruowany w USA — UNIVAC, a w Wielkiej Brytanii — LEO.

Maszyny pracujące na lampach elektronowych zaliczane są obecnie do pierwszej generacji komputerów.

Na początku lat pięćdziesiątych pojawiły się pierwsze języki programowania jako interpretery. **Grace Hopper** z firmy Remington Rand opisała pierwszy kompilator. Uwzględnienie w konstrukcji komputerów możliwości adresowania symbolicznego oraz adresowania pośredniego umożliwiło rozwój języków programowania. W 1956 roku zostaje wprowadzony przez firmę IBM wysokopoziomowy język programowania FORTRAN. W roku 1960 wprowadzono języki ALGOL i COBOL.

Postęp w elektronice umożliwiał dalszy rozwój komputerów. W latach sześćdziesiątych lampy elektronowe zastąpiono tranzystorami. Sam tranzystor wynaleźli J. Bardeen, W. H. Brattai i W. Shockly w 1948 roku. Pierwszy tranzystor został wyprodukowany w laboratorium firmy Bell. Pierwszy tranzystorowy komputer zbudowano w 1956 roku w Massachusettes Institute of Technology, a w cztery lata później firma IBM uruchomiła seryjną produkcję komputerów tej klasy.

Na początku lat sześćdziesiątych uzyskano szybkość pracy komputera równą jednemu milionowi operacji na sekundę, a także skonstruowano pierwsze układy scalone (początkowo układ scalony zawierał w jednej obudowie kilkadziesiąt tranzystorów i innych elementów elektronicznych), które zaczęto stosować przy budowie nowych maszyn. Dzięki temu komputery stały się znacznie mniejsze, a jednocześnie zwiększyła się szybkość ich działania.

Maszyny pracujące na tranzystorach zalicza się obecnie do drugiej generacji, a na układach scalonych do trzeciej generacji komputerów.

W 1965 roku w USA rozpoczęto seryjną produkcję komputerów cyfrowych serii IBM 360 zaliczanych do trzeciej generacji oraz pojawiły się pierwsze komputery analogowe zbudowane na układach scalonych.

W roku 1967 również w USA zastosowano pierwsze abonenckie systemy informatyczne, pracujące w systemie podziału czasu, a także rozpoczęto produkcję komputerów hybrydowych na układach scalonych.

Z ciekawostek warto dodać, że w tym samym roku odbył się pierwszy mecz szachowy pomiędzy radzieckimi i amerykańskimi komputerami. Rozegrano cztery partie. Dwie partie wygrał program opracowany w ZSRR, dwie partie zakończyły się remisem.

W latach sześćdziesiątych udoskonalono urządzenia zewnętrzne wspomagające pracę użytkowników sprzętu komputerowego. W 1968 roku rozpoczęto produkcję czytników dokumentów przystosowanych do automatycznego odczytu, a także graficznych urządzeń zewnętrznych tj. grafoskopów i alfaskopów, miniaturyzacja elementów elektronicznych, produkcja układów scalonych o coraz większym upakowaniu podstawowych elementów elektronicznych doprowadziła do skonstruowania pierwszych minikalkulatorów (kalkulatorów) na układach scalonych LSI (large scale integradion — dużej skali integracji), a także pojawiły się pierwsze kalkulatory kieszonkowe. W 1969 roku wydawca nowojorski William Morrow and Co. dokonał pierwszej próby wykorzystania komputera do wykonania składu książki.

Na przełomie lat sześćdziesiątych i siedemdziesiątych dokonał się ogromny skok w dziedzinie mikroelektroniki. W 1969 roku projektanci z amerykańskiej firmy Datapoint zaprojektowali bardzo prosty procesor i zlecili firmom Texas Instruments i Intel wykonanie go w postaci jednego układu scalonego. Z postawionego zlecenia wywiązała się firma Intel, choć okazało się, że pierwszy mikroprocesor działał z szybkością około 10 razy mniejszą od szybkości oczekiwanej przez firmę Datapoint. Z tego powodu zerwano umowę, ale Intel wykonał jednak prototyp zaprojektowanego procesora. Zdecydowano się także na produkcję seryjną procesora. W ten sposób na rynku pojawił się pierwszy mikroprocesor Intel 4004, o długości słowa równej czterem bitom. W 1974 roku pojawiły się mikroprocesory na ulepszonych układach scalonych i większej długości słowa (8 bitów) i słowa adresowego (16 bitów). W tym samym roku odbyły się pierwsze komputerowe mistrzostwa świata w szachach. Rozegrano je w Sztokholmie na kongresie IFIP, (International Federation for Information Processing). Uczestniczyło 13 programów z 8 krajów. Tytuł mistrzowski zdobył program „Kaissa" opracowany w ZSRR. W 1976 roku opracowano pierwszy mikrokomputer zawierający się w jednym układzie scalonym (seria Intel MCS-48).

Szybki postęp w technologii produkcji układów scalonych (zaczęto wytwarzać układy scalone bardzo dużej skali integracji VLSI-very large scale integradion) oraz komputerów sprawił, że zaczęto mówić o rewolucji komputerowej. Komputery zbudowane z układów VLSI zaczęto zaliczać do czwartej generacji.

W początkowym okresie komputery dla pojedynczego użytkownika produkowano w Hewlett-Packard i IBM. Komputery te były zaprojektowane dla naukowców (kosztowały około 10 000 $), a można je było programować w Basicu i używać od razu po włączeniu do sieci.

Pierwszym mikrokomputerem był Mark 8 opisany jako konstrukcja dla amatorów w czasopiśmie Radio-Electronics w lipcu 1974 roku. W styczniu 1975 roku rozpoczęto produkcję komputera Altair, opartego na 8-bitowym procesorze Intel 8080.

W połowie lat siedemdziesiątych Stere Woźniak i Stere Computer — Apple dla masowego odbiorcy (cena początkowa około 900 $) Apple był zbudowany na bazie mikroprocesora typu 6502 firmy MOS Technology, współpracował z klawiaturą oraz wykorzystywał ekran telewizyjny. Wszystkie układy komputera były umieszczone na jednej płytce. Po udanej konstrukcji Apple wyprodukowano znacznie ulepszony komputer osobisty Apple II, który ciągle modyfikowany, produkowany jest do dziś. Kolejnym sukcesem firmy Apple cieszyła się konstrukcja z 1983 roku Macintosh. W ciągu 100 dni wyprodukowano około 70 000 sztuk tego komputera.

Z innych bardzo udanych modeli mikrokomputerów należy wymienić wyprodukowany w 1980 roku przez firmę Clive Sinclaira Z 80 — pierwszy komputer na każdą kieszeń, w cenie poniżej 200 $. W 1981 roku Sinclair wyprodukował ZX 81, a w rok później tak popularny w Polsce ZX Spectrum. Obecnie kierunek rozwoju komputerów osobistych wytyczają Atari serii ST, Commodore Amiga oraz IBM.

Największy na świecie producent sprzętu komputerowego, amerykańska firma IBM (około 70% produkcji światowej), przystąpił do produkcji komputerów osobistych w 1981 roku. Opracowano od razu komputer IBM PC z 16-bitowym procesorem, choć w tamtym okresie powszechnie stosowano mikroprocesory 8-bitowe. Konstrukcję IBM PC łatwo jest modyfikować poprzez wymianę lub dołączanie nowych modułów, co zwiększa możliwości obliczeniowe i graficzne komputera oraz pozwala na dołączenie nowych urządzeń zewnętrznych. Model IBM PC stał się standardem komputera osobistego przeznaczonego do zastosowań profesjonalnych. **oprac. Jacek SZANIAWSKI**

PROGRAM 60

ZX SPECTRUM

Projektant — znaki

Szanowna Redakcjo !

W numerze 3 Waszego pisma zamieszczono program na "C-64" "Projektant-znaki", a ponieważ posiadam podobny program na ZX SPECTRUM - "UDG", więc postanowiłem przerobić go na język polski i wysłać jego wydruk do Was.

Program działa bezbłędnie, o czym może świadczyć fakt, że polskie litery, którymi się właśnie posługuję zostały utworzone na tym programie.

Program ten po przepisaniu na komputer należy zapisać na taśmę instrukcją :

SAVE "UDG" LINE 400.

Zbigniew Zych

```
   1 REM   polska wersja
 ZBIGNIEW ZYCH
        86.11.01.
  10 GO TO 8000
 100 PLOT 63,98  DRAW 129,0
 110 PLOT 128,160  DRAW 0,-129
 INK 0
 120 IF INKEY$="" THEN GO TO 130
 130 IF INKEY$="1" THEN LET a=1
 LET a$="■". LET b$="■"
 135 IF INKEY$="4" THEN GO SUB 1
800
 140 IF INKEY$="0" THEN LET a=0
 LET a$="*". LET b$=
 145 IF INKEY$="3" THEN GO SUB 1
000
 147 IF INKEY$="2" THEN GO SUB 1
324
 150 LET y$=INKEY$
 155 PRINT AT y,x,b$
 160 LET x=x+(y$="8" AND x<23)-(
y$="5" AND x>8)
 170 LET y=y+(y$="6" AND y<17)-(
y$="7" AND y>2)
 180 PRINT AT y,x,a$
 185 IF a=1 AND POINT (x-8,177-y
)=1 OR a=0 AND POINT (x-8,177-y)
=0 THEN GO TO 290
 190 IF x>16 =1 THEN LET b=2
 195 LET c=(b-1)*16+y-1
 200 IF POINT (x*8+1,175-y*8)=1
THEN LET a(c)=a(c)+2↑(7-(x-8*b))
 205 IF POINT (x*8+1,175-y*8)=0
THEN LET a(c)=a(c)-2↑(7-(x-8*b))
 210 IF a=0 AND POINT (x-8,177-y
)=0 THEN GO TO 250
 220 PLOT INK 0, OVER 0+(1 AND P
OINT (x-8,177-y)=1 AND a=0),x-8,
177-y
 230 PLOT INK 0, OVER 0+(1 AND P
OINT (232+x,177-y)=1 AND a=0),23
2+x,177-y
 255 PRINT AT y,3+(b-1)*23,"   "
 260 PRINT AT y,6+(b-1)*23-LEN S
TR$ a(c); INK 0;a(c)
 270 LET b=1
 290 BEEP .02,-10  BEEP .02,0  B
EEP .02,10
 300 GO TO 100
 400 POKE 23609,60  CLS  PRINT
AT 8,17;"▌  @";AT 9,13;"▌▌▀▀▀▌
";AT 10,13;"▌▌▌▌▌▌▙";AT 11,13;"
▘ ▀ ▀";AT 12,18;"▙"  PAUSE 100
: GO TO 8000
1000 REM FUNKCJE
1010 PRINT AT 19,0;"  p - POKE
   s - SAVE   "
1015 PRINT #1;AT 1,0;"

1020 PRINT AT 20,0;"  k - KONIE
C  c - CONTINUE  "
1025 PRINT AT 21,0;"
 "
1030 IF INKEY$="p" THEN GO SUB 1
100
1040 IF INKEY$="s" THEN GO SUB 1
300
1050 IF INKEY$="k" THEN GO SUB 1
500
1060 IF INKEY$="c" THEN GO SUB 1
700: RETURN
1070 BEEP .005,0  GO TO 1010
1100 REM FUNKCJA POKE
 PONIZSZE ZNAKI SA GRAFICZNE !
1110 PRINT AT 19,0;"A  B  C  D
E  F  G  H  I  J  K"
1120 PRINT AT 20,0;"L  M  N  O
P  Q  R  S  T  U  "
1125 FOR n=1 TO 30: NEXT n: LET
m=0: POKE 23658,8: FOR f=1 TO 4
1130 PRINT #1;AT 1,1;f;". Litera
 (A-U) :". LET p$=INKEY$
1133 IF CODE p$=13 THEN FOR n=1
TO 30: NEXT n: GO TO 1185
1140 IF CODE p$<65 OR CODE p$>85
 THEN GO TO 1130
1160 FOR n=0 TO 7: LET m=m+1: PO
KE USR CHR$ (CODE p$+79)+n,a(m):
 NEXT n
1165 REM
 PONIZSZE ZNAKI SA GRAFICZNE !
1170 PRINT AT 19,0;"A  B  C  D
E  F  G  H  I  J  K"
1180 PRINT AT 20,0;"L  M  N  O
P  Q  R  S  T  U  "
1183 NEXT f
1185 POKE 23658,0  PRINT #1;AT 1
,1;"
1190 PRINT AT 21,0; OVER 1;"
Ucis"+CHR$ 8+"'nij dowolny klawi
sz !"; PAUSE 0: RETURN
1300 REM SAVE
 PONIZSZE ZNAKI SA GRAFICZNE !
1310 PRINT AT 19,0;"A  B  C  D
E  F  G  H  I  J  K"
1320 PRINT AT 20,0;"L  M  N  O
P  Q  R  S  T  U  "
1321 FOR n=1 TO 30: NEXT n
1322 GO TO 1330
1324 PAUSE 5: PRINT #1;AT 1,9;"
COPY ? (t/n)": IF INKEY$="" THE
N GO TO 1324
1325 IF INKEY$="t" THEN PRINT AT
 y,x;"■": COPY : PRINT AT y,x;a$
1326 GO SUB 1700: RETURN
1327 FOR n=1 TO 30: NEXT n
1330 PRINT #1;AT 1,0;"        S
AVE ? (t/n)   ": IF INKEY$="" T
HEN GO TO 1330
1335 IF INKEY$="n" THEN RETURN
1336 IF INKEY$<>"" THEN GO TO 13
36
1340 INPUT "Nazwa  ";n$  IF LEN
n$>10 THEN GO TO 1340
1345 IF n$="" THEN LET n$="UDG"
1350 SAVE n$CODE USR "A",168
```

```
1352 FOR n=1 TO 30: NEXT n
1355 PRINT #1;AT 1,8;" VERIFY ?
(t/n)": IF INKEY$="" THEN GO TO
1355
1356 IF INKEY$<>"t" THEN RETURN
1360 PRINT AT 19,0;"
                          ";AT 19,8; F
LASH 1; OVER 1;"Przewin"+CHR$ 8+
"/ tas"+CHR$ 8+"me"+CHR$ 8+", !
";AT 20,8;"     VERIFY !    "
1365 PRINT AT 18,0
1370 VERIFY p$CODE
1380 RETURN
1500 REM KONIEC
1510 FOR n=1 TO 30: NEXT n
1520 PRINT #1;AT 1,8;"KONIEC ?
(t/n)": IF INKEY$="" THEN GO TO
1520
1525 IF INKEY$="t" THEN GO TO 99
99
1530 RETURN
1700 REM
1710 PRINT AT 19,1;" 1-RYSUJ
          0-MAZ"+CHR$ 8; OVER 1
;"-;"
1720 PRINT AT 20,1;" 3-MENU
4-CLS       2-COPY "
1725 PRINT #1;AT 1,0;"

1730 RETURN
1800 REM CLS
1810 FOR n=1 TO 30: NEXT n
1820 PRINT #1;AT 1,0;"
   CLS  ? (t/n)          ": IF INKEY
$="" THEN GO TO 1820
1830 IF INKEY$="t" THEN GO TO 90
00
1840 PRINT #1;AT 1,0;"
                          ": RETURN
8000 REM INSTRUKCJA
8005 BORDER 7: PAPER 7: INK 0
8010 CLS : PRINT AT 0,5;"User De
fined Graphics";AT 1,4;"

8020 PRINT "Program ten umozliwi
a tworzenie 21 wlasnych symboli
graficznych.": PLOT 123,159
8030 PRINT '"Marker ""*"" przesu
wany jest kur- sorami (5-8)."
8035 PRINT '"Wybor litery,pod kt
ora zdefinio-wany zostanie dany
symbol,jest  dowolny.": PLOT 27,
111: PLOT 27,110: PLOT 155,111:
PLOT 155,110: PLOT 175,105: PLOT
 175,106
8040 PRINT '"Mozliwy jest zapis
wszystkich   symboli na tasmie l
ub otrzymanie kopii na drukarce
(wymaga to   zmiany jednej z ins
trukcji !).": PLOT 19,79: PLOT 1
07,71: PLOT 107,70
8050 PRINT '"Zestaw symboli na t
asmie mozna  zaladowac zleceniem
 LOAD""""CODE  i wykorzystac w i
nnym programie.": PLOT 163,39: P
LOT 163,38: PLOT 219,39: PLOT 18
,27: PLOT 20,28: PLOT 21,29: PLO
T 68,31: PLOT 68,30: PLOT 100,23
: PLOT 100,22
8060 PRINT #1;AT 1,3; OVER 1;"Wc
is"+CHR$ 8+" nij dowolny klawisz
 !"

8100 PAUSE 0
8999 REM
9000 BORDER 6: PAPER 7: INK 0: C
LS
9010 PRINT AT 0,5;"User Defined
Graphics"
9015 INK 0: PLOT 38,166: DRAW 17
2,0
9020 PLOT 63,160: DRAW 129,0: DR
AW 0,-129: DRAW -129,0: DRAW 0,1
29
9025 PLOT 63,96: DRAW 129,0: PLO
T 128,160: DRAW 0,-129
9030 FOR n=28 TO 161 STEP 133
9040 FOR m=64 TO 193 STEP 8
9050 PLOT m,n: DRAW 0,2: NEXT m:
 NEXT n
9060 FOR n=60 TO 195 STEP 134
9070 FOR m=32 TO 164 STEP 8
9080 PLOT n,m: DRAW 2,0: NEXT m:
 NEXT n
9090 PRINT AT 20,1;" 3-MENU
4-CLS       2-COPY"
9095 PRINT AT 19,1;" 1-RYSUJ
          0-MAZ"+CHR$ 8; OVER 1
;"-;"
9100 FOR n=5 TO 28 STEP 23: FOR
m=2 TO 17: PRINT AT m,n; INK 0;0
: NEXT m: NEXT n
9110 PRINT PAPER 7;AT 0,0;"  ";A
T 1,0;"  ";AT 0,30;"  ";AT 1,30
;"  "
9120 INK 0: PLOT 0,159: DRAW 16,


9120 INK 0: PLOT 0,159: DRAW 16,
0: DRAW 0,16: DRAW 1,0: DRAW 0,-
17: DRAW -17,0: DRAW 0,-1: DRAW
18,0: DRAW 0,18
9130 PLOT 237,175: DRAW 0,-18: D
RAW 18,0: DRAW 0,1: DRAW -17,0:
DRAW 0,17: DRAW 1,0: DRAW 0,-16:
 DRAW 16,0: INK 0
9500 REM ZMIENNE
9510 LET y=2: LET x=8: DIM a(40)
9520 LET a=0: LET a$="*"
9530 LET b$=" ": LET b=1
9998 GO TO 180
9999 BORDER 0: CLS : PRINT AT 7,
0; FLASH 1;" Podanie instrukcji
CONTINUE  ,  powoduje wyzerowani
e komputera.": STOP : RANDOMIZE
USR 0
```

Autorzy: M&G Soft.

UWAGA !!

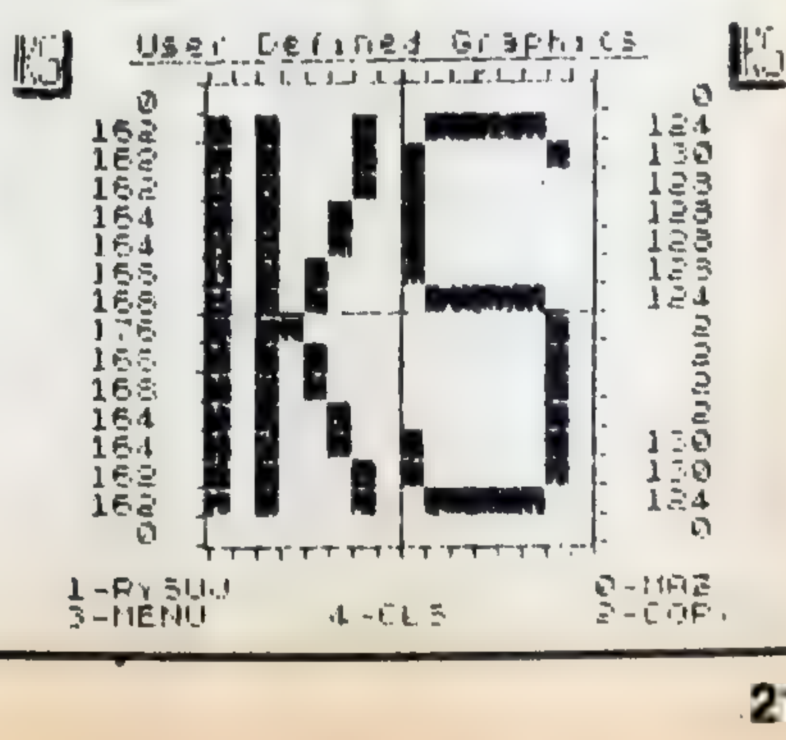
W liniach

1110,1120,1170,1180,1310 i 1320

litery w instrukcjach PRINT sa znakami graficznymi !

Pierwsze uruchomienie : RUN 400, nastepne : RUN .

WYNIK DZIALANIA PROGRAMU



PROGRAM 61

DWA GŁOSY NA SPECTRUM

Powszechnie znane ułomności popularnego ZX-Spectrum, to jego małe możliwości akustyczne. W komputerze tym zainstalowany jest jeden generator akustyczny, który ma tylko dwa zmienne parametry: wysokość tonu i czas trwania dźwięku.

Pozornie niemożliwe jest naśladowanie jakichś instrumentów, czy też granie melodii na dwa głosy. Zamieszczony niżej zestaw trzech programów niemożliwe czyni możliwym. Pierwszy z nich naśladuje dźwięk mandoliny; gra na dwa głosy starą rosyjską melodię „Samotna harmonia". Po wprowadzeniu do pamięci uruchamia się go dyrektywą **RUN**.

Nastepnie, nie wymazując poprzedniego programu, dopiszcie treść drugiego. Nie zrażajcie się, jeśli powtórzy się wiersz o jakimś numerze. Dyrektywa RUN spowoduje zagranie drugiej, równie pięknej melodii „Suliko".

Jeśli, nie wymazując dwu pierwszych programów, wprowadzicie trzeci, to będziecie mogli wybierać na zmianę: pierwszą lub drugą melodię. Aktualnie wybierana melodia wskazywana jest przez mrugający kursor. Zmiany wyboru dokonujemy klawiszem **SPACE**, a wybór potwierdzamy klawiszem **ENTER**. Stop programu następuje po naciśnięciu klawisza **BREAK**.

```
   1 REM Samotna harmonia
   5 DEF FN m()=1-2* INT (1/2)
  10 DATA "aCECazHDEFaHCafezHCDEF
CHCDEFaFDDCHaHCDEFEFFGFGCEaFDDCHaza"
  20 DATA "eaCafeaHCaezafcbezaHCDD
azaHCaCHHazezaHCDDDEDEDCaCHHazeee"
  60 LET t=10: LET tm=23672
  70 LET h=-1: LET c=0: LET d=2:LET e=4:
LET f=5: LET g=7:LET z=8: LET a=9
  80 READ p$: READ d$
 150 FOR l=1 TO 3
 160 LET i=1
 170 FOR j=1 TO 6
 180 LET buf=i+7+ FN m(): FOR i=i
TO buf: GOSUB 1010: NEXT i: LET t=t*
(5- FN m()): GOSUB 1010: LET i=i+1:
LET t=10
 190 NEXT j
 200 NEXT l
1000 CLS: STOP
1010 LET w= VAL p$(i): LET u= VAL d$(i)
1020 IF d$(i) <= "Z" THEN LET w=w+12
1030 IF p$(i) <= "Z" THEN LET u=u+12
1040 POKE tm,0
1050 BEEP 1/60,w: BEEP 1/60,u
1060 POKE tm, PEEK tm+2: IF PEEK tm < t
THEN GOTO 1050
1070 RETURN

   2 REM Suliko
  30 DATA "ggefgafg0HHHHHDHC0CCHCDCHaHagaaggfgg00"
  40 DATA "eecdefde0gggggHgg0aagaHagfgfeffeedeec0"
  50 DATA "111121182111121182111111112251111211 87"
  60 LET t=18: LET tm=23672
  70 LET h=-1: LET c=0: LET d=2: LET e=4: LET f=5:
LET g=7: LET z=8: LET a=9
  90 READ q$: READ b$: READ v$: GOTO 250
 250 FOR k=1 TO 2
 260 LET x=1: GOSUB 1200
 270 LET x=19:GOSUB 1200
 280 NEXT k
1200 FOR i=x TO LEN q$
1210 IF q$(i) = "0" THEN PAUSE t* VAL v$(i):GOTO 1290
1220 LET w= VAL q$(i)
1230 LET u= VAL b$(i)
1240 IF q$(i) <= "Z" THEN LET w=w+12
1250 IF b$(i) <= "Z" THEN LET u=u+12
1260 POKE tm,0
1270 BEEP 1/60,w: BEEP 1/60,u
1280 POKE tm, PEEK tm+2: IF PEEK tm < VAL v$(i) THEN
GOTO 1270
1290 NEXT i
1300 RETURN

  90 READ q$: READ b$: READ v$
  95 DIM s$(2,16): LET s$(1)="Samotna harmonia":
LET s$(2)="Suliko"
 100 PRINT AT 10,1;OVER 1;"Kto"; CHR$ 8;", ra";
CHR$ 8;", melodie";CHR$ 8;", wybiera?"
 105 PRINT AT 12,4;s$(1); AT 13,4;s$(2): LET l=0
 110 PRINT AT 12+l,1; FLASH 1;" "
 120 IF INKEY$ =" " THEN PRINT AT 12+l,1;" ":
LET l=l+1: LET l=FN m(): GOTO 110
 130 IF INKEY$ = CHR$ 13 THEN CLS : PRINT AT 10,
10,s$(l+1): GOTO 150+l*100
 140 GOTO 120
1000 CLS : GOTO 100
```

M P

Programowanie nie jest trudne, czyli MINI-GEM

Siadając do pracy przy komputerze, trudno sobie uzmysłowić, że jeszcze kilka lat temu te elektroniczne cacka nie były tak przyjaźnie „ustosunkowane" do człowieka jak dzisiaj. Starsi informatycy ze wstrętem wspominają czasy, gdy program wprowadzano do komputera za pomocą kart perforowanych, albo nawet taśmy dziurkowanej.

Jeszcze przed pięcioma laty tekst programu wprowadzało się do pamięci maszyny za pomocą prostego edytora, a trzeba było przed przystąpieniem do pracy dokładnie przestudiować instrukcję jego obsługi Kto raz pracował na dzisiejszym komputerze oprogramowanym w systemie GEM, ten nigdy dobrowolnie nie będzie chciał wrócić do wcześniejszych metod.

Co to jest GEM? Jest to skrót od wyrazów angielskich, które można zinterpretować: „Zarządzanie środowiskiem komputerowym za pomocą symboli graficznych". Wykonanie jakiejś funkcji można sprowokować nie poprzez pracowite wystukiwanie dyrektywy na klawiaturze z możliwością popełnienia błędu, lecz dzięki wskazaniu na monitorze komputera odpowiedniego symbolu graficznego. Zamiast pisać „DELETE" (usuń) wystarczy wskazać na ekranie symbol kosza na śmieci. Wyboru symbolu dokonujemy strzałką, która wędruje po całym ekranie dzięki, tzw. myszy. Pomyłka jest prawie wykluczona

Wbrew pozorom nie jest to takie trudne nawet wtedy, gdy nie mamy do dyspozycji „myszy". Nieco mniej komfortowym urządzeniem jest joystick. Poza tym można wykorzystać klawisze ze strzałkami, które znajdują się na klawiaturach wszystkich minikomputerów

Żeby nie być gołosłownym przedstawiamy wam krótki program, dzięki któremu możemy przesuwać w jednym z czterech kierunków dowolny tekst. Program napisany jest w BASIC-u dla komputera ZX-Spectrum. Widać więc, że nawet w tak bardzo ganionym języku i jednocześnie na tak małym i prostym komputerze można zaimplementować opisaną wcześniej metodę sterowania programem

Dla ułatwienia wspomnę, ze zmiany jednej z sześciu opcji dokonujemy klawiszem **SPACE,** a wybór potwierdzamy klawiszem **ENTER.** Dwie „niekierunkowe opcje to

C — czytaj tekst,

R — ruch całego tekstu.

Resztę powiedzą wam symbole. Stop programu nastąpi po wciśnięciu klawisza **BREAK.**

```
10 FOR n = 0 TO 79: READ a:
 POKE USR "a" + n,a: NEXT n
20 GOSUB 9000: LET t$ = u$:
 GOSUB 9000: LET i = 0: GOTO
 1000
30 LET t = i*4+1: PRINT AT 10,2
; t$(1 TO t);FLASH 1;t$(t+1 TO
 t+2); FLASH 0; t$(t+3 TO LEN
 t$) 'TAB 2; u$(1 TO t); FLASH
 1; u$(t+1 TO t+2); FLASH 0;
```

PROGRAM 62

ATARI

Wydruk ekranu

Czy zawsze, gdy chcesz coś wydrukować, potrzebujesz edytora tekstu lub programu graficznego? Z pewnością nie. Do prostych prac przy formacie 40 znaków × 24 wiersze, może posłużyć Ci poniższy program. Trudno oczywiście mówić o tworzeniu złożonej grafiki, gdy pracujemy w trybie graficznym 0. Można jednak, wykorzystując zestaw znaków możliwy do uzyskania na drukarce 1029, tworzyć nieskomplikowaną grafikę, tabele, teksty o maksymalnie 40-znakowym wierszu.

Po uruchomieniu programu, wybieramy rodzaj drukowania. Chodzi o to, że w trybie tekstowym stosuje się odstęp między wierszami. W ten sposób na jeden cal wchodzi sześć wierszy tekstu. W trybie graficznym nie stosuje się odstępu między wierszami, dzięki czemu w jednym calu mieści się dziewięć wierszy. Gdy dokonasz wyboru, program przejdzie w stan wyczekiwania. Będzie gotów do tworzenia obrazu. Przesuwanie kursora po ekranie odbywa się tradycyjnie, np. naciśnięcie klawisza CONTROL i któregoś ze strzałką, przesuwa kursor w kierunku wskazywanym przez strzałkę; naciśnięcie klawisza RETURN powoduje przesunięcie kursora na początek następnego wiersza, itd.

Drukowanie inicjuje się przez naciśnięcie klawisza ESC. W związku z tym, że obraz przechowywany jest tylko w pamięci ekranu, po wydrukowaniu zostaje skasowany. Jest to z pewnością niedogodność, którą rekompensuje fakt, że obrazy czy teksty drukowane tą metodą, będzie można otrzymać znacznie szybciej niż przy zastosowaniu np. złożonych edytorów, których czas wprowadzania jest znaczny.

Fragment zaczynający się od etykiety 600, a kończący się na 670, to zasadnicza część drukująca. Może ona również służyć jako podprogram drukujący dowolne fragmenty wykonującego się programu np. tekst, tabele itp. Sens wykorzystania go do tego celu jest jednoznaczny. Przy dużej różnorodności zmiennych, stosowanie do każdej z nich instrukcji LPRINT staje się często niewygodne, gdyż z reguły chcemy również oglądać je na ekranie. Program niepotrzebnie rozrasta się, jak również staje mniej czytelny. Nie należy przy tym zapomnieć o zadeklarowaniu długości ciągu tekstowego a$.

```
PK 100 REM ************************
KL 105 REM *                      *
GR 110 REM *  TWORZENIE PROSTEJ   *
BH 115 REM *  GRAFIKI  I WYDRUK   *
LZ 120 REM *  NA DRUKARCE 1029.   *
KP 125 REM *                      *
PQ 130 REM ************************
JX 200 REM *                      *
DV 205 REM *  RODZAJ DRUKOWANIA   *
JZ 210 REM *                      *
GU 230 GRAPHICS 0:OPEN #2,8,0,"P:"
ZU 235 ? :? "RODZAJ DRUKOWANIA :"
PK 240 ? "-------------------"
CV 245 ? :? CHR$(177);" TEKSTOWE"
AM 250 ? :? CHR$(178);" GRAFICZNE"
UN 255 ? :INPUT RD
UV 260 IF RD=2 THEN PRINT #2;CHR$(
   27);"9";
KV 345 REM *                      *
UE 350 REM *  TWORZENIE GRAFIKI   *
KX 355 REM *                      *
MA 360 OPEN #1,4,0,"K:":POKE 82,0
TT 365 GRAPHICS 0:DIM A$(40)
TS 370 FOR C=1 TO 20:? CHR$(31);:N
   EXT C:FOR C=1 TO 11:? CHR$(29);
   :NEXT C
KS 375 GET #1,R
QB 380 IF R=27 THEN 600
SH 385 ? CHR$(R);
SB 390 GOTO 375
KA 500 REM *                      *
AG 505 REM * PROGRAM REALIZUJACY *
IN 510 REM * DRUKOWANIE ZAPRO -   *
MK 515 REM * JEKTOWANEJ GRAFIKI.  *
KE 520 REM *                      *
QA 600 FOR A=0 TO 23
XU 610 A$=""
VB 620 FOR B=0 TO 39
LN 630 LOCATE B,A,X
GB 640 D=B+1:A$(D,D)=CHR$(X)
DO 650 NEXT B
YD 660 IF RD=2 THEN PRINT #2;A$:GO
   TO 670
HT 665 LPRINT A$
DI 670 NEXT A
KC 700 REM *                      *
MI 710 GRAPHICS 0:POKE 82,4
DF 720 ? :? "CZY CHCESZ KONTYNUOWA
   C T/N"
AV 730 ? :INPUT TN
OC 740 IF TN=84 THEN RUN
OG 750 END
```

Janusz JANIEC

```
 u$(t+3 TO LEN u$)
50 IF INKEY$ = "" THEN GOTO 50
60 IF INKEY$ = CHR$ 13 THEN
 GOTO 300+i
70 IF INKEY$ = " " THEN LET i
 = i+1: IF i > 6 THEN LET i = 0
80 GOTO 30
300 GOTO 1000
301 GOTO 1010
302 GOTO 1050
303 GOTO 1250
304 GOTO 1200
305 GOTO 1100
306 STOP
999 REM  Czytanie tekstu
1000 CLS: INPUT "Napisz jakis
 tekst" 'LINE p$: GOTO 30
1009 REM Przesuw w lewo
1010 FOR k = 0 TO LEN p$-1
1020 FOR L = 30 TO k STEP -1
1030 PRINT AT 15,l;p$(k+1);" "
1040 NEXT l: NEXT k: GOTO 30
1049 REM Przesuw w prawo
1050 FOR k = LEN p$-1 TO 0
 STEP -1
1060 FOR l = 0 TO 31-LEN p$+k
1070 PRINT AT 15,l;" ";p$(k+1)
1080 NEXT l: NEXT k: GOTO 30
1098 REM Plynny ruch calego
1099 REM tekstu w lewo
1100 LET g$ = " "
1110 FOR k = 1 TO 5
1120 LET g$ = g$+g$
1130 NEXT k
1140 LET o$ = g$+p$+g$+" "
1150 FOR k = 1 TO LEN o$-32
1160 PRINT AT 15,0;o$(k TO
 k+31): PAUSE 5
1170 NEXT k: GOTO 70
1199 REM Przesuw do gory
1200 LET w = LEN p$: IF n > 21
 THEN LET w = 21
1210 FOR k = 1 TO w
1220 FOR l = 20 TO k-1 STEP -1
1230 PRINT AT l,0;p$(k): PRINT
 AT l+1,0;" "
1240 NEXT l: NEXT k: GOTO 30
1249 REM Przesuw do dolu
1250 LET w = LEN p$: IF w > 21
 THEN LET w = 21
1260 FOR k = w TO 1 STEP -1
1270 FOR l = 1 TO k
1280 PRINT AT l,0;p$(k):PRINT
 AT l-1,0;" "
1290 NEXT l: NEXT k: GOTO 30
7999 REM Grafika uzytkowa
8000 DATA 192,96,48,24,12,6,3,1
8010 DATA 1,3,6,12,24,48,96,192
8020 DATA 24,24,24,24,24,24,24
,24
8030 DATA 0,0,0,255,255,0,0,0
8040 DATA 31,31,24,24,24,24,31
,31
8050 DATA 252,254,7,3,3,7,254
,252
8060 DATA 0,7,15,28,24,24,24,24
8070 DATA 24,24,24,24,28,15,7,0
8080 DATA 0,254,255,3,0,0,0,0
8090 DATA 0,0,0,0,3,255,254,0
8799 REM Plansza
8800 DATA 150,152,32
8810 DATA 32,145,147,32
8820 DATA 32,147,144,32
8830 DATA 32,146,146,32
8840 DATA 32,145,144,32
8850 DATA 32,148,149,32
8860 DATA 32,146,145,32
8900 DATA 151,153,32
8910 DATA 32,144,147,32
8920 DATA 32,147,145,32
8930 DATA 32,144,145,32
8940 DATA 32,146,146,32
8950 DATA 32,146,144,32
8960 DATA 32,146,144,32
8999 REM Czytanie planszy
9000 LET u$ = " "
9010 FOR k = 1 TO 27
9020 READ a: LET u$ = u$+CHR$ a
9030 NEXT k: RETURN
```

Mieczysław PŁACHETA

Lilavati (9)

Spłowiałe rękopisy

Danuta KWASIŻUR Mieczysław SKONIECZNY

Z treścią dzisiejszego zadania rozwiązywanego z udziałem mikrokomputera zapoznaje użytkownika ekran informacyjny, który pokazuje się po zainicjowaniu pracy programu. Jego zawartość jest następująca:

```
 Odnaleziono bardzo cenne, stare reko-
pisy matematyczne, ktore jednak wskutek
niewlasciwego przechowywania splowialy
tak daleco, ze tylko zarysy niektorych
cyfr mozna bylo odroznic.

 Cyfry nieczytelne oznaczone sa
 gwiazdkami. Nalezy w ich miejsce
 wstawic pierwotnie tam wpisane
 cyfry.

KLAWISZE STERUJACE:

do sterowania kursorem : ↑,↓,←,→
klawisz 'z' - wstawianie cyfry
klawisz 'ENTER' - koniec rozwiazywania
                   zadania

    NACISNIJ DOWOLNY KLAWISZ
```

Do rozwiązania jest 8 zadań, których kolejność wykazuje coraz większą skalę trudności. Po naciśnięciu klawisza ENTER na ekranie wyświetla się informacja o poprawności rozwiązania. Jeżeli rozwiązanie było błędne to pada pytanie:

„CZY CHCESZ SPRÓBOWAĆ JESZCZE RAZ (T/N)?"

i dalsza akcja programu jest uzależniona od odpowiedzi użytkownika.

Wersja źródłowa segmentu głównego programu zajmującego 6 KB pamięci RAM jest następująca:

```
10 MODE 1:INK 0,24:INK 1,0:INK 2,6:INK 3,26:BORDER 0:PAPER 0:CLS
20 SYMBOL AFTER 126
30 SYMBOL 126,0,0,0,255,0,0,0,0
40 KEY DEF 15,1,126
50 PEN 1:LOCATE 1,3:PRINT "  Odnaleziono bardzo cenne, stare reko- pisy matematy
czne, ktore jednak wskutek niewlasciwego przechowywa
nia splowialy  tak daleco, ze tylko zarysy niektorych  cyfr mozna bylo odroznic.
"
60 WINDOW £1,3,36,10,15:PAPER £1,1:CLS £1:PEN £1,0
70 LOCATE £1,2,2:PRINT £1,"Cyfry nieczytelne oznaczone sa":PRINT £1," gwiazdkami
. Nalezy w ich miejsce":PRINT £1," wstawic pierwotni
e tam wpisane":PRINT £1," cyfry."
80 LOCATE 2,17:PRINT "KLAWISZE STERUJACE:"
90 PRINT:PRINT " do sterowania kursorem : ";CHR$(240);",";CHR$(241);",";CHR$(242
);",";CHR$(243)
100 PRINT " klawisz 'z' - wstawianie cyfry
110 PRINT " klawisz 'ENTER' - koniec rozwiazywania                    zadania"
120 LOCATE 5,24:PRINT "NACISNIJ DOWOLNY KLAWISZ"
130 CLEAR INPUT
140 IF INKEY$="" THEN GOTO 140
150 FOR k=1 TO 8
160 READ wym(k,1),wym(k,2),il(k):NEXT k
170 DATA 5,4,4,5,5,5,5,4,4,5,4,4,5,4,4,6,8,17,6,8,18,10,14,27
180 FOR k=1 TO 8
190 MODE 1:LOCATE 14,5:PRINT "ZADANIE";K
200 WINDOW £1,14,14+wym(k,1)+2,8,8+wym(k,2)+1:PAPER £1,1:CLS £1:PEN £1,0
210 WINDOW £2,1,40,1,3:PAPER £2,1:PEN £2,0:CLS £2
220 GOSUB 380
230 GOSUB 700
240 GOSUB 1020
250 IF BL=0 THEN GOTO 350
260 CLS £2:PRINT £2," TWOJE ROZWIAZANIE ZAWIERA BLEDY !!":INPUT £2," CHCESZ SPRO
BOWAC JESZCZE RAZ (T/N)";T$
270 IF T$="T" OR T$="t" THEN CLS £2:GOTO 230
280 IF T$="N" OR T$="n" THEN GOTO 300
290 GOTO 260
300 IF K=8 THEN GOTO 370
310 INPUT £2," NASTEPNE ZADANIE (T/N)";T$
320 IF T$="T" OR T$="t" THEN CLS £2:GOTO 360
330 IF T$="N" OR T$="n" THEN GOTO 370
340 GOTO 300
350 PRINT £2," TWOJE ROZWIAZANIE JEST POPRAWNE !!":GOTO 300
360 NEXT k
370 END
```

Do programu głównego jest dołączonych 5 podprogramów spełniających następujące funkcje:

1. Podprogram wczytywania danych do aktualnego zadania.

```
380 REM  'podprogram wczytywania danych do aktualnego zadania'
390 IF k>1 THEN ERASE tabl$:ERASE GWIA
400 DIM tabl$(wym(k,2),wym(k,1))
410 DIM GWIA(IL(K),3)
420 FOR i=1 TO wym(k,2)
430 FOR j=1 TO wym(k,1)
440 READ tabl$(i,j):LOCATE £1,1+j,1+i:PRINT £1,tabl$(i,j)
450 NEXT j
460 NEXT i
470 FOR i=1 TO il(k)
480 READ gwia(i,1),gwia(i,2),gwia(i,3)
490 NEXT i
500 RETURN
510 DATA " ","*","*","4","*","+","2","*","*","3"," ","**","**","**","**"," ","6","5"
,"2","9"
520 DATA 1,2,3,1,5,6,2,3,6,2,4,8
530 DATA " "," ","1","0","5"," ","*","*","1","7","+","5","*","*","*","**","**","**"
,"**","**","*","0","8","4","6"
540 DATA 2,2,4,2,3,9,3,4,2,3,5,4,5,1,1
550 DATA " ","5","*","1","7","+","*","4","*","*"," ","**","**","**","**"," ","6","*"
,"1","*"
560 DATA 1,3,3,2,2,1,2,4,9,4,5,5
570 DATA " ","*","*","*","7","-","*","3","5","*"," ","**","**","**","**"," ","6","1"
,"7","7"
580 DATA 1,3,5,1,4,2,2,2,2,2,5,0
590 DATA " ","3","*","5","7","-","*","9","*","*"," ","**","**","**","**"," "," ","4"
,"*","6"
600 DATA 1,3,4,2,2,2,2,5,1,4,4,7
610 DATA " "," "," ","*","*","*"," ",x," ","4","5","7"," ","**","**","**","**","**","
"," ","*","*","*","*"," ","1","7","0","5"," ","*","
*","*","*"," "," ","**","**","**","**","**","**","*","*","*","*","*","*"
620 DATA 1,4,3,1,5,4,1,6,1,4,3,2,4,4,3,4,5,8,4,6,7,6,1,1,6,2,3,6,3,6,6,4,4,8,1,1
,8,2,5,8,3,5,8,4,8,8,5,3,8,6,7
630 DATA " "," ","*","*","*","3"," ",x," ","*","*","*"," ","**","**","**","**","**","
"," ","*","7","0","*"," ","*","2","1","3"," ","*","
*","*","9"," "," ","**","**","**","**","**","**","*","*","*","*","*","*"
640 DATA 1,3,1,1,4,2,1,5,1,2,4,3,2,5,1,2,6,8,4,3,9,4,6,4,5,2,1,6,1,3,6,2,6,6,3,3
,8,1,3,8,2,8,8,3,5,8,4,7,8,5,3,8,6,4
650 DATA " "," ","*","*","*","*"," "," "," "," ","**","**","**","**","**","**"," "," "
," "," ","*","*","*","*","*","*","1","*","*","*","*"
,"*","*"," "," "," "," "," "," ","**","**","**"," "," "," "," "," "," "," "
660 DATA "3","0","2","*"," "," "," "," "," "," ","*","6","9","5"," "," "," "," "
," "," "
670 DATA "**","**","**","**"," "," "," "," "," "," ","*","*","1","1"," "," "," "
," "," "," ","3","*","*","*"," "," "," "," "," "
,"**","**","**","**"," "," "," "," "," "," ","2","*","1","*"," "," "," "," "
," ","*","*","*","*"," "," "," "," "
680 DATA " ","**","**","**","**"," "," "," "," "
690 DATA 1,3,1,1,4,7,1,6,6,3,1,6,3,2,8,3,3,7,3,4,6,3,5,1,3,6,0,3,8,3,3,9,8,3,10,
5,4,1,3,4,2,8,4,3,5,6,4,6,7,1,2,9,2,3,9,3,2,10,3,0,1
0,5,0,12,4,3,12,6,0,13,3,2,13,4,3,13,5,1,13,6,0
```

2. Podprogram sterowania kursorem na ekranie.

```
700 REM  'podprogram sterowania kursorem'
710 i=1:j=1
720 PAPER £1,3:PEN £1,2:LOCATE £1,j+1,i+1:PRINT £1,tabl$(i,j)
730 a$=INKEY$
740 IF a$="" THEN GOTO 730
750 IF INKEY(6)=0 THEN RETURN
760 IF a$="z" OR A$="Z" THEN GOTO 900
770 IF a$=CHR$(240) THEN GOTO 820
780 IF a$=CHR$(241) THEN GOTO 840
790 IF a$=CHR$(242) THEN GOTO 860
800 IF a$=CHR$(243) THEN GOTO 880
810 SOUND 1,458:GOTO 730
820 IF i=1 THEN GOTO 810
830 GOSUB 1200:i=i-1:GOTO 720
840 IF i=wym(k,2) THEN GOTO 810
850 GOSUB 1200:i=i+1:GOTO 720
860 IF j=1 THEN GOTO 810
870 GOSUB 1200:j=j-1:GOTO 720
880 IF j=wym(k,1) THEN GOTO 810
890 GOSUB 1200:j=j+1:GOTO 720
900 FOR V=1 TO IL(K)
910 IF J=GWIA(V,2) AND I=GWIA(V,1) THEN GOTO 940
920 NEXT V
930 GOSUB 1130:GOTO 730
940 LOCATE £2,2,2:PRINT £2,"WPISZ CYFRE"
950 CLEAR INPUT
960 A$=INKEY$
970 IF A$="" THEN GOTO 960
980 IF ASC(A$)>47 AND ASC(A$)<58 THEN GOTO 1000
990 GOSUB 1190:GOTO 940
1000 PAPER £1,0:PEN £1,1:LOCATE £1,J+1,I+1:PRINT £1,A$:TABL$(I,J)=A$
1010 CLS £2:GOTO 720
```

3. Podprogram badania poprawności rozwiązania.

```
1020 REM 'podprogram badania poprawnosci rozwiazania'
1030 bl=0
1040 FOR v=1 TO il(k)
1050 IF J=GWIA(v,2) AND I=GWIA(v,1) AND tabl$(i,j) <>"*" THEN PAPER £1,0:PEN £1,
1:LOCATE £1,j+1,i+1:PRINT £1,tabl$(i,j):GOTO 1080
1060 NEXT v
1070 PAPER £1,1:PEN £1,0:LOCATE £1,j+1,i+1:PRINT £1,tabl$(i,j)
1080 FOR V=1 TO IL(K)
1090 IF RIGHT$(STR$(GWIA(V,3)),1)=TABL$(GWIA(V,1),GWIA(V,2)) GOTO 1110
1100 BL=1
1110 NEXT V
1120 RETURN
```

4. Podprogram z komunikatami o błędach.

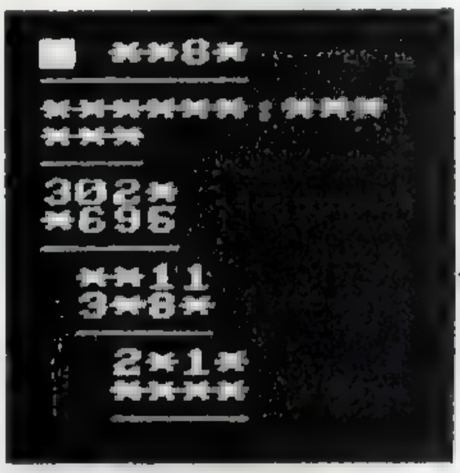
```
1130 REM  'podprogramy z komunikatami o bledach'
1140 CLS £2:LOCATE £2,2,2
1150 IF tabl$(i,j)<"1" OR tabl$(i,j)>"9" THEN GOTO 1170
1160 CLS £2:LOCATE £2,2,2:PRINT £2,"TEJ CYFRY NIE MOZNA ZMIENIAC":GOTO 1180
1170 CLS £2:LOCATE £2,2,2:PRINT £2,"TEJ POZYCJI NIE MOZNA ZMIENIAC"
1180 SOUND 1,458:FOR T=1 TO 1000:NEXT T:CLS £2:RETURN
1190 CLS £2:LOCATE £2,2,2:PRINT £2,"TO NIE JEST CYFRA":SOUND 1,458:FOR T=1 TO 10
00:NEXT T:CLS £2:RETURN
```

5. Podprogram badania pozycji cyfry.

```
1200 REM 'podprogram badania pozycji cyfry'
1210 FOR V=1 TO IL(K)
1220 IF J=GWIA(V,2) AND I=GWIA(V,1) AND tabl$(i,j) <>"*" THEN PAPER £1,0:PEN £1,
1:GOTO 1250
1230 NEXT V
1240 PAPER £1,1:PEN £1,0
1250 LOCATE £1,j+1,i+1:PRINT £1,tabl$(i,j):RETURN
```

Przykładowa postać jednego z zadań jest następująca:

ZADANIE 8



Życzymy Czytelnikom bezbłędnego rozwiązania wszystkich 8 zadań.

Danuta KWASIŻUR,
Mieczysław SKONIECZNY

POLSCRIPT — edytor tekstu z polskimi literami dla C-64

I. Wstęp

POLSCRIPT jest przeróbką programową znanego edytora tekstu EASY SCRIPT, który został opracowany w wersji pierwotnej przez Simona Tranmera i wypuszczony na rynek angielski przez firmę Precision Software Limited w roku 1982. Poniższy opis, bardzo skrótowy (oryginalny opis EASY SCRIPT zajmuje kilkadziesiąt stron), dotyczy zarówno wersji oryginalnej, jak i polskiej modyfikacji, ponieważ różnice między nimi są minimalne.

Autor zetknął się z przeróbkami programu dla drukarek MPS-801, MPS-802, MPS-803, z tym, że na co dzień używa wersji dla drukarki MPS-803, ponieważ pracuje z nią poprawnie posiadana drukarka (OKIDATA-120).

II. Uruchomienie programu

POLSCRIPT jet programem rezydującym i współpracuje równie dobrze z magnetofonem, jak i stacją dysków.

Zajmuje w pamięci komputera 73 bloki (Polscript 803). Po załadowaniu i uruchomieniu edytora komendą RUN ukazuje się menu główne, gdzie jesteśmy proszeni o podanie:
— szerokości tekstu (jest to szerokość tekstu na monitorze, najczęściej wynosi ona 40 znaków),

— rodzaju urządzenia (stacja dysków czy magnetofon),
— rodzaju drukarki (dla drukarek Commodore i zgodnych należy wybrać 0).

Następnie pytani jesteśmy o rodzaj liter (polskie, czy angielskie) i układ klawiatury (polski, czy angielski (QWERTY)).

Przechodzimy do trybu: „EDIT" — pisanie tekstu. Do dyspozycji mamy praktycznie cały ekran monitora. W górnej części ekranu (STATUS LINE) znajduje się informacja o trybie pracy, w jakim aktualnie znajdujemy się oraz o położeniu kursora (wiersz i kolumna). Zmianę kolorów ramki, tła i kursora uzyskuje się przez równoczesne naciśnięcie klawiszy CTRL i odpowiednio 3,2 i 1.

III. Pisanie tekstu

Każdy nowo tworzony dokument (pisany tekst) powinien zostać zatytułowany. W tym celu należy nacisnąć klawisz F3, napisać nb, podać w cudzysłowie nazwę dokumentu (pliku) i nacisnąć RETURN.

Klawisz F3 służy zawsze do zaznaczenia początku linii komend, w której podaje się między innymi, oddzielone od siebie dwukropkiem następujące komendy dotyczące formatu wydruku pisanego tekstu:

lmxx	— lewy margines (xx — nr kolumny),
rmxx	— prawy margines,
plxx	— długość strony,
tlxx	— długość tekstu,
spx(1,2,3)	— odstęp między liniami,
1nxx	— ilość pustych wierszy w tekście,
1pxx	— ilość wierszy na cal,
txx	— ilość liter na cal.

Każda komenda lub linia komend musi być zakończona klawiszem RETURN. Standardowe ustawienie powyższych parametrów wygląda następująco: lm=1, rm=80, pl=66, tl=60, lp=6, pt=10.

Klawisz RETURN służy do rozpoczynania nowego akapitu (paragrafu). Pojedyncze znaki usuwa się klawiszem DEL. Naciśnięcie CTRL i W powoduje przejście do pierwszej litery następnego słowa, a CTRL i „O" do ostatniej litery ostatniego słowa.

IV. Uzyskiwanie polskich liter

Polskie litery uzyskuje się następująco:
ą — średnik
ę — SHIFT + średnik
ś — SHIFT + przecinek
ń — SHIFT + kropka
ć — SHIFT + „/"
ó — „ő"
ź — SHIFT + „α"
ż — znak „+"
Ż — SHIFT + znak „+"
ł — dwukropek
Ł — SHIFT + dwukropek

Ulega również zmianie znaczenie niektórych klawiszy tj:
klawisz „gwiazdka" =„:"
SHIFT + „gwiazdka" = „;"
SHIFT + „—" =„+"
„strzałka pionowa" =„zł"
znak funta =„∮"
SHIFT + „6" =„?"

V. Znaczenie klawiszy funkcyjnych

Klawisz F1 służy do zmiany trybu pracy. Podstawowe kombinacje z klawiszem F1 to:
— F1+0+V — przeglądanie tekstu na monitorze w postaci, w jakiej będzie wydrukowany (do przesuwania tekstu służą klawisze kursora, klawisz Commodore, F5, F7, SPACJA.
— F1+0+P — drukowanie tekstu. Dodanie „C" powoduje wydruk ciągły. Przejście od „oglądania" strony na ekranie do wydruku następnej na drukarce uzyskuje się po naciśnięciu „P".
— F1+L — załadowanie pliku (dokumentu) do pamięci komputera (po podaniu nazwy i naciśnięciu RETURN),
— F1+F — zapisanie pliku (zamiast wpisywania nazwy pliku możemy ją skopiować, o ile plik został uprzednio zatytułowany za pomocą klawisza F2)
— F1+E+A — usunięcie całego kształtu tekstu,
— F1+E+R — usunięcie tekstu od pozycji kursora w dół,
— F1+E+P — usunięcie paragrafu,
— F1+E+S — usunięcia zdania,
— F1+D — usunięcie tekstu podświetlonego kursorem,
— F1+I — przejście do trybu pracy z rozsuwaniem tekstu (INSERT) lub odwrotnie,
— F1+G+E — przejście kursora na koniec tekstu,
— F1+G+xx — przejście kursora do linii xx,
— F1+SPACJA — wyświetlenie zawartości następnego ekranu,
— F1+SHIFT+SPACJA — wyświetlenie zawartości poprzedniego ekranu,
— F1+INST/DEL — usunięcie linii,
— F1+R+oświetlenie kursorem — definiowanie zakresu,
— F1+T+H lub V — tabulacja (pozioma i / lub pionowa),
F7 — przejście do następnej pozycji i tabulatora przy tabulacji poziomej,
F8 — jw., tylko przy tabulacji pionowej,
F1+C — kasowanie tabulacji,
F1+A — duplikacja tekstu (po zaznaczeniu zakresu (RANGE),
F1+S — wyszukiwanie tekstu i zastępowanie,
F5 — załączanie dużych liter,
F4 — przejście do trybu operacji dyskowych. W trybie tym istnieje możliwość między innymi;

a) wyświetlenia katalogu dyskietki na ekran lub jego wydrukowania (odpowiednio F4+„$" i F4+„+"+„$"+O+P,
b) załadowania dowolnego pliku z katalogu do pamięci komputera przez kombinację klawiszy: F4+„+"+„$"+L+F2,

VI. Inne możliwości programu

Omawiany program pozwala również między innymi na:
— formatowanie dyskietki,
— usuwanie, kopiowanie i zmianę nazwy plików,
— justyfikację i centrowanie tekstu,
— drukowanie w nagłówku i w stopce,
— automatyczną numerację stron,
— tworzenie długich dokumentów za pomocą wiązania plików (linked files) i obróbkę takich dokumentów (formatowanie, wydruk),
— tworzenie nowego pliku z zaznaczonych paragrafów,
— wydruk pogrubiony, podkreślony i odwrócony (reverse),
— resetowanie programu bez naruszenia tekstu znajdującego się w pamięci (RUN) STOP +RESTORE) lub z jego skasowaniem (F1+RUN) STOP).

Program umożliwia również definiowanie własnych znaków. Maksymalna wielkość tworzonego pliku wynosi 764 wiersze przy 40 kolumnach.

VII. Podsumowanie

Z przedstawionego powyżej, z konieczności skrótowego opisu edytora POLSCRIPT (EASY SCRIPT) wynika, że w zasadzie spełnia on wszystkie wymagania stawiane „poważnym", profesjonalnym programom tego typu. Niewątpliwą jego wielką zaletą jest „pisanie po polsku". Sam program nie jest może zbyt „przyjazny", bo komend koniecznych do zapamiętania jest dużo, ale z doświadczeń autora wynika, że jest to tylko kwestia ilości godzin, czy też stron napisanych za pomocą programu. Zaletą programu jest również możliwość współpracy z magnetofonem. Jedyną wadą programu odkrytą w ciągu kilku miesięcy jego użytkowania jest zniekształcanie liter po znakach „," i „—" przy stosowaniu automatycznej numeracji stron. Zjawisko to występowało zarówno podczas używania drukarki MPS-803, jak i OKIDATA -120. Nie potrafiąc sobie z tym problemem poradzić autor niniejszego artykułu zastosował „ręczne" numerowanie stron. A może ktoś z czytelników i użytkowników programu POLSCRIPT-803 wie, jak powyższą wadę usunąć?

Tadeusz CISEK

ZAGADKA

Martin CARR

Alma Cari nie była sobą, jej głos znany doskonale wszystkim mieszkańcom Dean Verry, drżał i łamał się, a twarz z nienaturalnym uśmiechem wyglądała niczym maska.

Frank Ctraks, który dopiero teraz włączył swój odbiornik zdążył usłyszeć tylko końcowy fragment komunikatu:
— Dlaczego z Banku Myśli zginął mózg arcyzbrodniarza Citta Seana — trudno w tej chwili odpowiedzieć. Prezydent policji zapowiada, iż sprawą zajmie się osobiście.

Teraz dopiero Frank zrozumiał dlaczego został w trybie pilnym wyrwany ze swego urlopu. Jadąc do centrum starał sobie przypomnieć co wiedział o Citt Seanie. Tak... to musiał być niezwykły facet, jeżeli przez całe dziesięciolecia prowadził swoją prywatną wojnę z cywilizacją ziemską. Miał takie słowo na określenie sylwetki — anarchista niepodporządkowujący. Zwalczająca go policja kontynentalna była bezsilna. Jego dziełem było uprowadzenie najnowocześniejszego na owe czasy pojazdu kosmicznego i wymuszenie za zwrot od Rady Obywatelskiej wysokiego okupu — 10 weryklejsów czyli jego udziału w 10 ekspedycjach pozagalaktycznych. On też porwał prom kosmiczny na Eksporą Gardis i poleciał nim na wycieczkę w przestrzeń Oriona.

Kiedy wreszcie umarł, jego mózg trafił do gabinetów najwybitniejszych uczonych psychiatrów i neurologów. Ich ekspertyza była wyczerpująca, dopatrzyli się wielu odchyłek i nieprawidłowości w funkcjonowaniu. Nazwali go mózgiem arcyzbrodniarza i zalecili umieszczenie w Banku Myśli. Olbrzymi materiał badawczy, jaki zgromadzono, posłużył do opracowania nowej teorii, a zarazem nowej dziedziny nauki zwanej przez opracowujących mianem HMC. Rada Obywatelska wydała wtedy zezwolenie na wprowadzenie korekt u nowo narodzonych, jeżeli stwierdzono jeden z symptomów Citta. Dzięki temu przez całe wieki nie zdarzało się nic, co mogłoby źle wpłynąć na cywilizację ziemską, a życie toczyło się spokojnie.

Kiedy Frank trafił do centrum policji spostrzegł niezwykłe ożywienie. Na ekranach monitorów widać było, jak cymborgi i roboty funkcjonalnie uzbrojone w analizatory poszukiwawcze przeczesywały teren, dom po domu, rezydencja po rezydencji, biurowce i zakłady produkujące. Kiedy wyświetlił swój numer identyfikacyjny usłyszał: proszę iść niebieską drogą i natychmiast wyświetliły się niebieskie światełka. Ruszył. Szedł przez kilka kondygnacji, aż znalazł się przed gabinetem prezydenta policji. Tam na niego czekano, nie zdążył bowiem nawet nacisnąć guzika, kiedy drzwi się rozsunęły i wszedł do gabinetu. Na jego powitanie podniósł się ubrany w mundur policji miejskiej wysoki mężczyzna. Wskazał mu miękki fotel, sam podsunął wózek z napojami i zajął miejsce naprzeciw.

— To, co teraz powiem należy do najściślejszych tajemnic nie znanych ogółowi naszego miasta, przecieki mogą wywołać niepotrzebne napięcia. Dlatego proszę o zachowanie tego dla siebie. Jest pan najlepszym ekspertem w rozwiązywaniu niejasnych spraw. To że zaginął mózg arcyzbrodniarza to jedno, ale wiąże się z tym cała seria innych wydarzeń. Dziesięć dni po jego zaginięciu spłonął Bank Centralnych Danych, dwa dni po tym Centrala Rozdzielnictwa Dóbr. Przyczyn pożarów i sprawców nie ustalono. Poprosiliśmy wtedy o pomoc policję kontynentalną. Gdy awioloty z posiłkami lądowały w naszym mieście został obrabowany Bank Funduszów — zginęło 1000 regredów czyli środki na utrzymanie i zapewnienie wymogów życiowych dla 1000 naszych obywateli na jeden rok.

Nasza hipoteza jest taka: osoba lub grupa osób wykradła mózg i wbudowała go jakiemuś cymborgowi, po to, aby destabilizować sytuację w naszym mieście. Nie mamy tylko odpowiedzi — dlaczego? To, co się jeszcze może zdarzyć trudno przewidzieć. Prosząc pana o pomoc liczymy, że pan potrafi sprawę rozwiązać, ze swej strony udostępnimy wszystko czego będzie panu potrzeba.

Rozmowa była skończona i Frank postanowił wrócić do siebie, musiał głęboko się zastanowić, jak rozwiązać postawione przed nim zadanie.

Ale jeszcze tego dnia dowiedział się, że z oddalonego o dwadzieścia mil od miasta ośrodka terminku kosmicznego zginął ładunek trafianu, najnowszego paliwa do awiolotów B, używanych, jako niszczyciele kosmicznych meteorytów. Następnego dnia rano niezidentyfikowany pojazd z serii awiolotów B zniszczył prom kosmiczny z Lukseru przywożący ładunek platyny, dwa dni później ten sam los spotkał prom z Algera-1. Rada Ziemi podjęła wtedy decyzję wysłania eskadr ochronnych. Nie na wiele się to zdało, gdyż w niewyjaśnionych okolicznościach zaginął kolejny statek kosmiczny, tym razem z ładunkiem almogeru niezbędnego do produkcji walidu, materiału, z którego budowano pokrycia promów kosmicznych.

W czasie, kiedy wystąpiły te tajemnicze wydarzenia Frank pracował z największym napięciem. Za pomocą programów symulacyjnych sprawdzał własną hipotezę. Po kolejnej chyba dwudziestej serii wszystkie wydarzenia zaczęły układać się w jedną spójną całość.

Jak później, już po rozwiązaniu sprawy, mówił na konferencji prasowej nie był pewny, czy poszedł dobrą drogą. Najpierw bowiem udało mu się ustalić listę potomków Citta Seana. Później przebadał je jedną po drugiej i wtedy znalazł rozwiązanie. Okazało się, że tylko dwaj ludzie nie byli przebadani na HMC i to do tego właśnie Harry i Reed Seanowie. Motorem działania był Harry, którego Rada Lekarska z powodu obniżonego progu inteligencji skierowała na najniższy X poziom kierowania. Tam stykając się bezpośrednio z robotami i cymborgami rozpoczął wbudowywanie im układów posłuszeństwa wobec siebie. Mając wielkie aspiracje, a jednocześnie zdając sobie sprawę z tego, że nie będzie mógł być podniesiony na wyższy poziom, postanowił sam doprowadzić do własnego awansu. O co mu chodziło? Chciał zostać dyktatorem Ziemi. Kiedyś w młodości znalazł gdzieś kasetę magnetowidową z zapisem o dyktatorach. To mu się spodobało. A ponieważ jego umysł nie został skorelowany z potrzebami wobec cywilizacji, działał bez skrupułów. Miał swego pomocnika w bracie, który znajdując się w najwyższym pułapie kierowniczym poziomu A-1 dostarczał potrzebnych informacji. Jak wykazały badania był szantażowany i kilkakrotnie naruszono jego nietykalność osobistą. Roboty, które były w dyspozycji Harrego, kilkakrotnie niszczyły ulubione sprzęty Reeda i to wystarczyło.

Kradzież mózgu przodka miała dać legendę i zarazem natchnąć odwagą do działania.

Muszę zapewnić państwa, że obydwaj znajdują się w ośrodku zdrowia psychicznego i właśnie w tej chwili poddawani są zabiegowi dostosowania społecznego. Wyciągnęliśmy też daleko idące wnioski. Nigdy już tak nie będzie, aby ktoś nie był badany na HMC. Rada Obywatelska podjęła decyzję, aby certyfikat HMC był stale noszony na lewej ręce. To byłoby wszystko — zakończył.

GIEŁDA POMYSŁÓW

Szanowna Redakcjo!

Od pierwszych numerów „IKS-a” jestem jego stałym czytelnikiem. Posiadam mikrokomputer Atari 800XL, stację dysków LDW 2000 oraz drukarkę Atari 1025. Od kilku miesięcy układam programy w języku Basic, Pascal, Asembler. Jeden z nich chciałbym opublikować na łamach waszego pisma.

Program ten służy do robienia spektoramu pamięci komputera. Po wpisaniu go do pamięci komputera podajemy adresy, w zakresie których program ma robić spektoram. Program ten pracuje w trybie graficznym 8. Po zapisaniu całego ekranu następuje jego zamazanie i zarysowanie od początku. Program umożliwia wczytanie dowolnego programu z dysku do pamięci komputera i późniejsze wykonanie wykresu.

Myślę, że program ten przyniesie wiele satysfakcji w „badaniu” budowy programów. Ciekawe efekty można uzyskać w posiadaniu różnych programów np.:

Ciekawych efektów w zakresie spektografii komputerowej — życzy autor.

Maciej STANUSCH

```
0 DIM NA$(14)
5 GOTO 125
10 SETCOLOR 2,0,0:? CHR$(125):POSITION 0,5
15 ? "PODAJ OD JAKIEGO ADRESU MAM TESTOWAC    PAMIEC ADR1=";:INPUT A
20 POSITION 0,10:? "PODAJ DO JAKIEGO ADRESU MAM TESTOWAC PAMIEC ADR2=";:INPUT B
25 GRAPHICS 8:SETCOLOR 2,0,0:COLOR 1:POKE 752,1
30 FOR Q=A TO B:X=X+1:IF X=320 THEN X=0:GRAPHICS 8:SETCOLOR 2,0,0:COLOR 1:POKE 7
52,1
35 PLOT X,150:DRAWTO X,150-PEEK(Q)*0.58823
40 IF PEEK(53279)=6 THEN RETURN
45 ? ,Q,PEEK(Q):NEXT Q
50 ? CHR$(125):? "GOTOWE...":? "NACISNIJ START"
55 IF PEEK(53279)<>6 THEN 55
60 RETURN
65 REM WCZYTANIE PROGRAMU Z DYSKU
70 GRAPHICS 0:SETCOLOR 2,0,0:POSITION 0,5:? "PODAJ NAZWE ZBIORU W POSTACI D:NAZW
A":POSITION 5,7:INPUT NA$
75 MEMLO=PEEK(14)+256*PEEK(15)
80 APPMHI=PEEK(741)+256*PEEK(742)
85 TRAP 105:Q=MEMLO
90 OPEN #1,4,128,NA$:POKE 559,0
95 GET #1,X:MEMLO=MEMLO+1:IF MEMLO=APPMHI THEN 105
100 POKE MEMLO,X:GOTO 95
105 GRAPHICS 0:SETCOLOR 2,0,0:CLOSE #1
110 ? "ZALADOWANO ";MEMLO-Q;" BAJTOW "
115 ? :? "OD ADRESU ";Q;" DO ADRESU ";MEMLO:?
120 GOSUB 15
125 REM MENU
130 X=0
135 GRAPHICS 0:SETCOLOR 2,0,0:COLOR 1:? :? :? :? :?
140 ? "1.TESTOWANIE PAMIECI":?
145 ? "2.WCZYTANIE PROGRAMU Z DYSKU":?
150 ? :? :? "PODAJ NR OPCJI I NACISNIJ RETURN":INPUT SKOK
155 ON SKOK GOSUB 10,65
160 GOTO 125
```

Postscriptum do programu „Enigma”

W zamieszczonym programie „ENIGMA” (4/88 „IKS”) nie ma błędu. Opis jego autor przedstawił może zbyt lakonicznie, ale wyraźnie napisał, że składa się z trzech części. Każdą z tych części należy wprowadzić na taśmę oddzielnie.

Pierwsza z nich, to fragment o etykietach 0, 1, 2, część druga rozpoczyna się od etykiety 3, a kończy na 11, zasadniczy fragment programu zaczyna się od etykiety 20. Każdą z części wprowadzamy kolejno na kasetę instrukcją CSAVE. Program pierwszej części ładujemy z taśmy przez wpisanie CLOAD. Gdy zostanie wprowadzony, program zatrzyma się — będzie oczekiwał na wpisanie instrukcji RUN. Gdy to nastąpi rozpocznie się wprowadzanie następnych części, które będą uruchamiały już automatycznie.

Janusz JANIEC

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

FRAME TABLE - tablica w systemach operacyjnych ze stronicowym podziałem pamięci wirtualnej, ustalająca zgodność między stronami wirtualnymi a fizycznymi,
FRAMEWORK - plan, szkielet, struktura,
FRAMING - ramkowanie, obramowanie, także: ustawianie obrazu, synchronizacja obrazu,
FRAMING BIT - bit ramki,
FREE - wolny, swobodny, także: bezpłatny,
FREE-FORM - dowolnego formatu, dowolnej struktury,
FREE FORM CODING - kodowanie niepozycyjne,
FREE FROM ... - nie zawierający ..., pozbawiony ..., wolny od ...,
FREE FROM INTERFERENCE - niezakłócony,
FREE-HAND DRAWING - tryb rysowania swobodnego, tryb rysowania odręcznego,
FREE OF CHARGE - bezpłatny,
FREE OF ERROR - bezbłędny,
FREE OF TURN - bez czekania w kolejce, poza kolejnością,
FREE-RUNNING MODE - tryb swobodnego dostępu,
FREE SPACE - wolna pamięć, wolna przestrzeń pamięci,
FREE VARIABLE - zmienna swobodna,
FREE VECTOR - wektor swobodny,
FREEZING - zamrożenie,
FREQUENCY - częstotliwość,
FREQUENCY DIVISION MULTIPLEXING - zwielokrotnienie częstotliwości,
FREQUENCY MODULATION - modulacja częstotliwościowa,
FREQUENCY MODULATION RECORDING - zapis z modulacją częstotliwościową, zapis FM,
FREQUENCY SHIFT KEYING - kluczowana modulacja częstotliwości,
FRICTION FEED - podawanie papieru przy pomocy wałka,
FRIENDLY ENVIRONMENT - otoczenie sprzyjające, otoczenie dopasowane, dopasowanie,
FRIENDLY SOFTWARE - oprogramowanie "przyjazne", oprogramowanie dopasowane, oprogramowanie sprzyjające,
FROM ABOVE - z góry, od góry,
FRONT-END - interfejs zewnętrzny, także: procesor komunikacyjny, preprocesor,
FRONT-END COMPONENT - podsystem dostępu, także: podsystem wstępnego opracowania danych,
FRONT-END COMPUTER - komputer czołowy,
FRONT-END DATA PROCESSING - przetwarzanie danych czołowe,
FRONT-END INTERFACE - interfejs zewnętrzny,
FRONT-END PROCESSOR - procesor czołowy, specjalizowany procesor wprowadzania i wyprowadzania (danych), także: procesor komunikacyjny,
FS - patrz: FILE SEPARATOR,
FSK - patrz: FREQUENCY SHIFT KEYING,
FULL ADDER - sumator pełny,
FULL DUPLEX - pełny dupleks (jednoczesna komunikacja w obu kierunkach)
FULL-DUPLEX CHANNEL - kanał dupleksowy,
FULL PATHNAME - pełna (kompletna) ścieżka nazwy,
FULL-SCREEN EDITOR - edytor pełnoekranowy,
FULL STOP - znak kropki,
FULL WORD - słowo maszynowe,
FULLWORD - patrz: FULL WORD,
FULLY CONNECTED NETWORK - sieć w pełni połączona,
FULLY INVERTED FILE - plik (zbiór) całkowicie zmieniony (np. przez indeksację według innego indeksu),
FULLY STARTED JOB - zadanie w pełni uruchomione,
FUNCTION - funkcja,
FUNCTIONAL - funkcyjny, także: funkcjonalny, czynnościowy,
FUNCTIONAL CHARACTER - znak sterujący, znak funkcyjny,
FUNCTIONAL DEPENDENCE - zależność funkcyjna,
FUNCTIONAL DESIGN - projekt (schemat) funkcjonalny,
FUNCTIONAL DIAGRAM - patrz: FUNCTIONAL DESIGN,
FUNCTIONALITY - możliwości funkcjonalne,
FUNCTIONAL LANGUAGE - język funkcjonalny, język funkcjonalnego programowania,
FUNCTIONAL QUALITY - jakość użytkowa,
FUNCTIONAL UNIT - zespół funkcjonalny,
FUNCTION BUTTON - klawisz funkcyjny,
FUNCTION CALL - wywołanie funkcji,
FUNCTION GENERATOR - generator funkcji,
FUNCTION KEY - patrz: FUNCTION BUTTON,
FUNCTION KEYBOARD - klawiatura funkcyjna,
FUNCTION MENU - wykaz (spis) funkcji, menu,
FUNCTION MULTIPLIER - układ mnożący dla funkcji,
FUNCTION TABLE - tablica funkcji,
FUNCTOR - funktor, element logiczny,
FUNDAMENTAL - podstawowy,
FUNDAMENTAL FORMULA - wzór podstawowy,
FUNDAMENTAL TYPE - typ podstawowy (danych),
FURNISH - zaopatrywać, wyposażać,
FURNISH INFORMATION - dostarczać informacji,
FUSE BASE - gniazdo bezpiecznikowe,
FUZZY - postrzępiony, rozmyty, także: niedokładny, nieścisły,
FUZZY LOGIC - logika nieścisła, logika niedokładna (w systemach ekspertowych),

G

GAIN - wzrost, także: korzyść, wzmocnienie,
GAIN EXPERIENCE - zdobyć doświadczenie,
GAME - gra, także: grać,
GAME THEORY - teoria gier,
GAME TREE - drzewo gry,
GAMING - teoria gier użytkowych,
GAMIST - specjalista teorii gier,
GAMM-MIX - mieszanka GAMM,
GANG-PUNCH - masowe wprowadzanie do kart dziurkowanych identycznej treści,
GANGPUNCHING - kopiowanie wieloseryjne (kart dziurkowanych),
GAP - przerwa międzyblokowa, przerwa blokowa (np. między dwoma blokami na taśmie lub dysku magnetycznym), także: brak impulsu lub sygnału,
GAP DIGIT - cyfra techniczna,
GAPLESS - bezszczelinowy,
GAPLESS TAPE - taśma magnetyczna z ciągłym zapisem danych,
GAPPED TAPE - taśma magnetyczna z blokami danych,
GARBAGE - dane zbędne, dane pomyłkowe,
GARBAGE COLLECT - czyścić pamięć (usuwać zbędne dane),
GARBAGE COLLECTION - usuwanie danych zbędnych (np. z ciągu znaków),
GARBAGE COLLECTOR - program czyszczący pamięć (usuwający zbędne dane),
GARBAGE IN, GARBAGE OUT - wprowadzisz błędne dane, to uzyskasz błędne wyniki,
GASP - język informacyjnego modelowania,
GAS PANEL - monitor plazmowy (jarzeniowy),
GAS-PLASMA DISPLAY - monitor plazmowy,
GATE - bramka, element logiczny,
GATE CIRCUIT - układ bramkowy,
GATED - bramkowany,
GATEWAY - śluza w sieciach komputerowych (sprzętowe i programowe środki zabezpieczające łączność w sieci),
GATEWAY SERVER - stacja łączności z zewnętrzną siecią komputerową,
GATHER WRITE - pisanie rozproszone, wypis rozproszony (z pamięci),
GB - patrz: GIGABYTE,
GE - sieć komputerowa GE,
GE - patrz: GREATER OR EQUAL,
GEM - patrz: GRAPHICS ENVIRONMENT MANAGER,
GENERAL - ogólny,
GENERAL FLOW - ogólna sieć działań,
GENERALIZATION - uogólnienie,
GENERALIZE - uogólniać,
GENERALIZED DATA BASE - baza danych ogólnego przeznaczenia,
GENERALIZED FORMULA - wzór ogólny,
GENERALIZED ROUTINE - program ogólny, program uniwersalny,
GENERAL LOCAL NAME - ogólna nazwa lokalna,
GENERALLY ACCEPTED - powszechnie przyjęty,
GENERAL PROBLEM SOLVING PROGRAM - ogólny program rozwiązywania problemów,
GENERAL-PURPOSE - ogólnego zastosowania, uniwersalny, wielocelowy,
GENERAL-PURPOSE COMPUTER - komputer uniwersalny, komputer wielozadaniowy.
GENERAL PURPOSE INTERFACE BUS - wspólna magistrala wejścia/wyjścia,
GENERAL-PURPOSE REGISTER - rejestr wielozadaniowy,
GENERAL REGISTER - rejestr ogólnego przeznaczenia,
GENERATE - wytwarzać, wywoływać, generować,

GENERATED ADDRESS - adres generowany,
GENERATE (TO) - generować,
GENERATING PROGRAM - generator programów, program generujący,
GENERATION - generacja (np. komputerów), także: generowanie, wytwarzanie,
GENERATION NUMBER - numer generacji,
GENERATIVE - wytwarzający,
GENERIC - rodzajowy, uniwersalny, ogólny,
GENERIC DESCRIPTION - opis uogólniony,
GENERIC FUNCTION - funkcja ogólna,
GENERIC NAME - nazwa ogólna,
GENERIC OPERATION - operacja ogólna,
GENERIC PACKAGE - pakiet uniwersalny, pakiet z możliwością strojenia,
GENERIC PROCEDURE - procedura ogólna,
GENERIC TYPE - typ uniwersalny, typ ogólny, typ parametryzowany (pojęcie programowania dotyczące klasy typów danych),
GENUINE - autentyczny, prawdziwy,
GEOMETRIC(AL) - geometryczny,
GEOMETRICAL PROGRESSION - postęp geometryczny,
GEOMETRIC MEAN - średnia geometryczna,
GEOMETRY - geometria,
GET - otrzymywać, uzyskiwać, dostać, stać się, wpływać, także: powstać,
GET ACQUAINTED WITH ... - zapoznać się z ...,
GET WORSE - pogarszać się, psuć się,
GIBSON-MIX - mieszanka Gibsona,
GIGABYTE - gigabajt (1048576 bajtów),
GIGACYCLE - gigacykl (1048576 cykli),
GIGO - patrz: GARBAGE IN, GARBAGE OUT,
GIVE INFORMATION - dostarczać informację,
GIVEN QUANTITY - wielkość dana,
GIVE (PROFESSIONAL) ADVICE - konsultować,
GIVE SERVICES - świadczyć usługi,
GKS - patrz: GRAPHICS KERNEL SYSTEM,
GLASS TELETYPE - "szklany dalekopis" (wykorzystanie monitora dla sukcesywnego wprowadzania lub wyprowadzania linii tekstu bez użycia operacji ekranowych i sterowania kursorem),
GLOBAL - globalny, także: ogólnoświatowy,
GLOBAL IDENTIFIER - identyfikator globalny, oznacznik globalny,
GLOBAL OPTIMIZATION - globalna optymalizacja (optymalizacja programu na poziomie instrukcji i procedur), także: znajdywanie minimum lub maksimum funkcji celu,
GLOBAL VARIABLE - zmienna globalna,
GLOSSARY - słownik (wyjaśniający, specjalistyczny),
GLOW - jarzyć się, żarzyć się, świetlić,
GLUABILITY - sklejalność,
GLUE - kleić, sklejać (np. fragmenty tekstu w trakcie edycji),
GLUING - klejenie, sklejanie,
GO - idź, także: stawać się, zapalać się (o lampie),
GOAL DRIVEN - wnioskowanie wsteczne (w systemach ekspertowych),
GOAL FUNCTION - funkcja celu (w zadaniach optymalizacji),
GOAL-INVOKED INTERPRETATION - interpretacja od celu (proceduralna interpretacja zasady "jeśli A to B"),
GOTO - przejście, przechodzenie, przekazanie sterowania,
GOTO STATEMENT - instrukcja "GOTO", instrukcja przejścia,
GOVERNING - regulacja, sterowanie,
GOVERNOR - regulator,
GPS - patrz: GENERAL PROBLEM SOLVING,
GRACEFUL DEGRADATION - "amortyzacja uszkodzeń", także: płynne obniżenie efektywności,
GRADUAL - stopniowy,
GRADUALLY - stopniowo,
GRADUATE - kończyć (np. szkołę, uczelnię),
GRAINED - ziarnisty,
GRAMMAR - gramatyka,
GRAMMATICAL - gramatyczny,
GRAMMATICS - patrz: GRAMMAR,
GRANDFATHER - grupa danych starsza o dwa pokolenia od danych rozpatrywanych,
GRAND TOTAL - suma całkowita, suma ogólna, suma łączna,
GRANULARITY - stopień detalizacji, także: granulacja,
GRAPH - wykres, graf,
GRAPH COLORING - kolorowanie grafu,
GRAPHEME - grafon (elementarna jednostka tekstu np. litera),
GRAPH FOLLOWER - urządzenie wprowadzania konturowych obrazów graficznych,
GRAPHIC - znak drukarski (np. litera, cyfra),
GRAPHICAL - wykreślny, graficzny,
GRAPHICAL DATA PROCESSING - przetwarzanie danych graficznych,
GRAPHICAL DEVICE - patrz: GRAPH PLOTTER,
GRAPHICAL DISPLAY UNIT - grafoskop, ekranopis graficzny,
GRAPHICAL KERNEL SYSTEM - system jądra graficznego,
GRAPHICAL LANGUAGE - język graficzny,
GRAPHICAL METHOD - metoda wykreślna,
GRAPHICAL OUTPUT PRIMITIVE - niepodzielny element obrazu (np. kropka, przecinek, myślnik, itp.),
GRAPHICAL SYMBOLS - symbole graficzne,
GRAPHIC ART - grafika,
GRAPHIC CHARACTER - znak graficzny,
GRAPHIC DATA - dane graficzne,
GRAPHIC DISPLAY - zobrazowanie graficzne,
GRAPHIC FORMAT - postać graficzna,
GRAPHIC FORMULA - wzór strukturalny,
GRAPHIC INPUT DEVICE - graficzne urządzenie wejściowe,
GRAPHIC INTERFACE - interfejs graficzny, środki graficznego współdziałania (oddziaływania),
GRAPHIC LANGUAGE - język graficzny,
GRAPHIC PALLET - paleta graficzna (zgodność pomiędzy kodami kolorów, a kolorami zobrazowanymi na ekranie monitora),
GRAPHIC PRIMITIVE - element obrazu (np. punkt, odcinek, znak alfanumeryczny, itp.),
GRAPHIC PROCESSOR - procesor graficzny,
GRAPHICS - grafika,
GRAPHIC SCHEDULE - harmonogram,
GRAPHICS CURSOR - kursor graficzny,
GRAPHICS DIGITIZER - urządzenie (cyfrowe) wprowadzania obrazów,
GRAPHICS EDITOR - edytor graficzny, edytor zobrazowania,
GRAPHICS ENVIRONMENT MANAGER - zarządzanie środowiskiem grafiki,
GRAPHICS KERNEL SYSTEM - bazowy system graficzny,
GRAPHICS MODE - tryb graficzny,
GRAPHICS PAD - plansza graficzna, rysownica graficzna,
GRAPHICS PROGRAMMING LANGUAGE - język graficzny,
GRAPHICS TERMINAL - terminal graficzny,
GRAPHIC TABLETT - rysownica graficzna (urządzenie do wprowadzania konturów obrazów, składające się z tabliczki oraz specjalnego pióra świetlnego), koordynatograf,
GRAPHIC VIEWPORT - wziernik graficzny,
GRAPH PLOTTER - urządzenie rysujące, pisak cyfrowy, ploter,
GRAPH THEORY - teoria grafów,
GRAVE ACCENT - znak akcentu,
GRAY CODE - kod Gray'a,
GRAY LEVEL - poziom jaskrawości (dla obrazów czarno-białych),
GRAY SCALE - skala jaskrawości,
GRAY-SCALE IMAGE - zobrazowanie półtonowe (z różną skalą jaskrawości),
GRAY SCALE SIGNAL - sygnał skali kontrastów,
GREATER OR EQUAL - większy lub równy,
GREATER-THAN SIGN - znak większości ("większy od"),
GREATER THEN - większy od,
GREAT MAJORITY - znaczna większość,
GROSS ERROR - gruby błąd,
GROSS INDEX - indeks główny, indeks podstawowy,
GROUND - uziemienie,
GROUND COLOUR - kolor tła,
GROUNDWORK - podłoże, baza, podstawa, fundament,
GROUP - grupa, zespół,
GROUP DELAY - opóźnienie grupowe,
GROUPED RECORDS - zapisy zgrupowane,
GROUPING - grupowanie,
GROUPING OF DATA - grupowanie danych,
GROUP SYMBOL - symbol grupy,
GROWING SYSTEM - system rozszerzalny, system dostosowany do rozbudowy,
GT - patrz: GREATER THEN,
GUARANTEE - gwarancja, poręczenie, także: gwarantować, zapewnić,
GUARANTEE CERTIFICATE - karta gwarancyjna,
GUARANTEED ACCURACY ALGORITHM - algorytm o zagwarantowanej dokładności,
GUARD - zabezpieczenie, osłona, dozór, także: osłaniać, zabezpieczać,
GUARD BIT - bit zabezpieczenia,
GUARD DIGIT - pozycje chronione (wyniku),
GUARDED COMMANDS - komendy zabezpieczone,
GUARD PLATE - płyta ochronna,
GUESS - domysł, przypuszczenie, próba,
GUIDANCE - wskazówka, wytyczne,
GUIDE - poradnik, także: prowadzić,
GUIDE BOOK - poradnik (książka), przewodnik (książka), informator,
GUIDE HOLE - dziurka prowadząca, dziurka przesuwu,
GUIDE ROLLERS - rolki prowadzące (np. taśmę magnetyczną),
GUIDE TO OPERATION - wprowadzenie do pracy,
GULP - grupa bajtów przetwarzana jako jedna całość,

LIGA MYŚLĄCYCH

ZADANIE 1

Dwa ciała poruszają się po okręgu w tym samym kierunku i spotykają się co 56 minut. Gdyby ciała te poruszały się z tymi samymi prędkościami w przeciwnych kierunkach, to spotykałyby się co 8 minut. Przy ruchu w przeciwnych kierunkach odległość (mierząc po okręgu) między zbliżającymi się ciałami zmniejszyłaby się z 40 metrów do 26 metrów w ciągu 24 sekund. Ile metrów na minutę przebywa każde ciało oraz jaki jest obwód okręgu, po którym poruszają się oba ciała?

ZADANIE 2

Naczynie o pojemności 8 litrów napełniono powietrzem zawierającym 16% tlenu. Z naczynia tego odprowadzono pewną ilość powietrza i doprowadzono taką samą ilość azotu, po czym ponownie odprowadzono taką samą (jak za pierwszym razem) ilość mieszaniny i ponownie doprowadzono taką samą ilość azotu. W powstałej mieszaninie było 9% tlenu. Wyznaczyć po ile litrów gazu odprowadzono i doprowadzono za każdym razem z naczynia.

ZADANIE 3

Barka o ładunku 600 ton była wyładowana w ciągu 3 dni, przy czym pierwszego i trzeciego dnia wyładowano 2/3 całego ładunku. W drugim dniu wyładowano mniej, niż w pierwszym, a w trzecim mniej niż w drugim, przy czym różnica między procentowym zmniejszeniem wyładunku trzeciego dnia w stosunku do wyładunku drugiego dnia i procentowym zmniejszeniem wyładunku drugiego dnia w stosunku do wyładunku pierwszego dnia była równa 5. Ile ton ładunku wyładowano każdego dnia?

ZADANIE 4

Należy wyznaczyć kąt rozwarcia stożka, opisanego dookoła czterech równych kul ułożonych w ten sposób, że każda z nich jest styczna do trzech pozostałych.

ZADANIE 5

Z całej ilości towaru a % sprzedano z zyskiem p %, zaś b % sprzedano z zyskiem q %. Jaki był zysk sprzedaży pozostałej części towaru, jeżeli ogólny procent zysku wyniósł r %?

Rozwiązania zadań prosimy przesyłać do redakcji do końca grudnia br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

PROGRAM 63

ATARI

Trałowanie

Znane przysłowie wojskowe mówi, że saper myli się tylko raz. Dotyczy ono również i załogi trałowców. Pewną różnicą charakteryzującą ich zadania jest to, że miny na morzu mogą zmieniać swoje położenie. Potrzeba dużo czasu i cierpliwości, by je unieszkodliwić. Do pomocy mamy urządzenia wykrywające. Po każdym ruchu meldują nam, w którym kierunku mamy poruszać się, aby dotrzeć do miny np. miny z lewej. Okrętem sterujemy za pomocą klawiszy: L — lewo, P — prawo, N — naprzód, W — wstecz

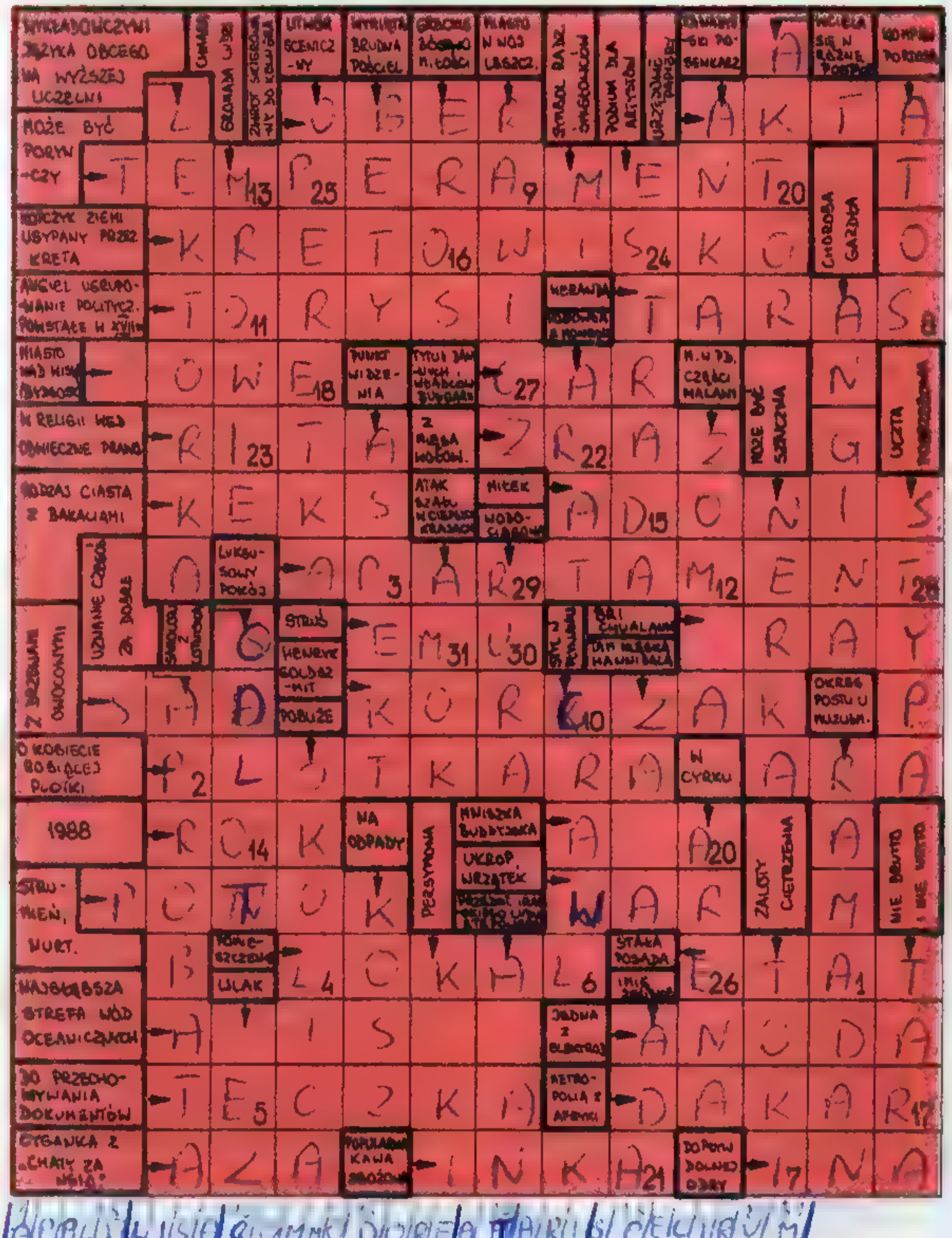
Jeśli na otrzymany meldunek zareagujemy prawidłowo, to zostaniemy powiadomieni, że miny są bliżej. Po kilku ruchach minę należy zdetonować. Gdy to ci się nie uda, uważaj się za jednego z tych pechowców, którzy pomylili się ten jeden raz.

```
PG 500 REM ************************
YA 502 REM *                      *
HY 504 REM *  T R A L O W A N I E *
GS 505 REM *      * A t a r i *   *
YP 507 REM *                      *
QH 509 REM ************************
MK 510 DIM T$(65),TS$(8),A$(1),B$(1
   )
VG 515 DATA Z LEWEJ ,Z PRAWEJ,Z PRZ
   ODU
QX 516 DATA Z TYLU  ,W LEWO  ,W PRA
   WO ,NAPRZOD ,WSTECZ  ,K
CI 520 RESTORE
MF 525 FOR K=1 TO 9:READ TS$
FK 530 T$(LEN(T$)+1)=TS$:NEXT K
BR 535 ? CHR$(125)
FB 537 ? "T R A L O W A N I E"
GA 539 EP=3:GOSUB 800
LL 541 M=5+INT(RND(0)*6):KA=0:N=M
XB 545 FOR K=1 TO N:A$="":B$="":OD=
   INT(RND(0)*3)+3
TA 548 P=0:R=INT(RND(0)*4)+1
DV 552 ? "M I N Y ! ";:RR=R*8:? T$(
   RR-7,RR)
EC 555 GOSUB 800:A=P:IF R>2 THEN 61
   0
TB 560 ? T$(33,40);"/";T$(41,48);"=
   ";
QR 565 ? "L/P";:INPUT A$
MK 570 IF (A$="L" AND R=1) OR (A$="
   P" AND R=2) THEN P=P+1:GOTO 650
PQ 580 GOTO 640
DV 610 ? T$(49,56);"/";T$(57,64);"=
   ";
TX 620 ? "N/W";:INPUT B$
CK 630 IF (B$="N" AND R=3) OR (B$="
   W" AND R=4) THEN P=P+1:GOTO 650
UJ 640 P=P-1
AW 650 IF P>A THEN ? " MINY BLIZEJ
   !"
OJ 660 IF P<A THEN ? " MINY DALEJ !
   "
FN 665 EP=2:GOSUB 800
RC 670 IF P>2 THEN ? "DETONUJESZ ?"
   :EP=3:GOSUB 800:GOTO 730
QQ 680 GOTO 552
NY 690 Q=Q+1:GOSUB 830:? Q;" MINA Z
   NISZCZONA"
XJ 700 NEXT K:WYNIK=INT((N*100)/M)
GR 705 ? "ZNISZCZONO ";WYNIK;" % MI
   N"
DV 710 ? "TRALUJEMY T/N";:INPUT A$
JS 715 IF A$="N" THEN ? CHR$(125):E
   ND
YK 720 Q=0:GOTO 535
LH 730 ? "            T/N ";:INPUT A$
CV 735 IF A$="T" AND P=OD THEN 690
ZN 740 IF P=OD THEN ? "WPADLES NA M
   INE !":GOSUB 830:END
AF 745 IF A$="T" AND P<OD THEN ? "P
   UNKT KARNY !":KA=KA+1
XG 746 IF KA=3 THEN N=N-1:KA=0
QL 750 GOTO 552
RI 800 FOR BE=1 TO EP:FOR G=10 TO 0
    STEP -1
WK 805 SOUND 0,100,10,G:NEXT G:NEXT
    BE
ZG 810 RETURN
GO 830 FOR S=20 TO 240 STEP 3:SOUND
    0,S,4,8:NEXT S:SOUND 0,240,0,14
DF 840 FOR S=0 TO 40:NEXT S
IJ 850 FOR S=15 TO 0 STEP -1:POKE 7
   06,62:SOUND 0,250,4,S:SOUND 2,24
   0,0,S:NEXT S
ZQ 860 RETURN
```

Janusz JANIEC

** W algorytmie programu wykorzystano pomysł J. Millera*

KRZYŻÓWKA NR 8



Litery z kratek ponumerowanych od 1 do 31 utworzą nazwy 5 komputerów, które wystarczy podać jako rozwiązanie zadania, przesyłając pod adresem redakcji na kartach pocztowych, w terminie do końca grudnia, naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

GRAFIKA KOMPUTEROWA

Jedną z najbardziej dynamicznie rozwijających się dziedzin zastosowania maszyn cyfrowych jest grafika komputerowa. Jeszcze do niedawna była ona domeną szybkich komputerów profesjonalnych. Obecnie dzięki zastosowaniu nowoczesnych technologii mogą się nią posługiwać posiadacze współczesnych komputerów osobistych.

Z tą myślą ciekawe i tanie rozwiązanie zaproponowała angielska firma Digithrust Ltd. Oferuje ona zestaw składający się z pakietu programów i pełnowymiarowej karty rozszerzającej do komputerów zgodnych z IBM PC AT, która pozwala połączyć komputer z magnetowidem lub kamerą wideo. Używając tego zestawu można zapamiętać obraz w postaci cyfrowej — na monitorze RGV z rozdzielczością 640 na 400 punktów. Specjalny algorytm pozwala uzyskać efekt płynnego przejścia między sąsiednimi kolorami i uzyskać ten sam obraz na monitorze i karcie EGA. Obraz można zapamiętać na nośniku magnetycznym w postaci akceptowanej przez programy graficzne PC Paint, GEM Paint i Ventura Publisher. Funkcja „format" określa, w której części ekranu ma pojawić się przetworzona grafika. Aby przyspieszyć operacje ustawienia kierunku, ostrości i przysłony kamery przewidziano możliwość przyjęcia przez komputer obrazu czarno-białego. Grafika komputerowa jest narzędziem o niekwestionowanej już dziś przydatności w pracy architektów, inżynierów czy lekarzy. Dlatego coraz więcej firm oferuje dodatkowe karty graficzne, które zainstalowane wewnątrz komputera znacznie rozszerzają jego możliwości i to pod względem rozdzielczości obrazu, jak i szybkości realizowania skomplikowanych operacji ekranowych.

Nowością w tej dziedzinie jest karta VTP amerykańskiej firmy Verticom. Jej przeznaczeniem jest monochromatyczny tryb 1280 na 960 punktów, chociaż może też pracować w trybie CGA. W karcie zastosowano najnowszy procesor graficzny INTEL 82786 — o nie wykorzystanych do końca możliwościach — taktowany z częstotliwością 20 MHz, 512 KB pamięci RAM i 8 KB ROM. VTP współpracuje z szeroko rozpowszechnionymi programami, na przykład Microsoft Windows, Auto CAD ADI, GEM, Ventura Publisher. Zainstalowanie takiej karty porównywalne jest ze wstawieniem drugiego komputera do jednej obudowy.

(RAJ)
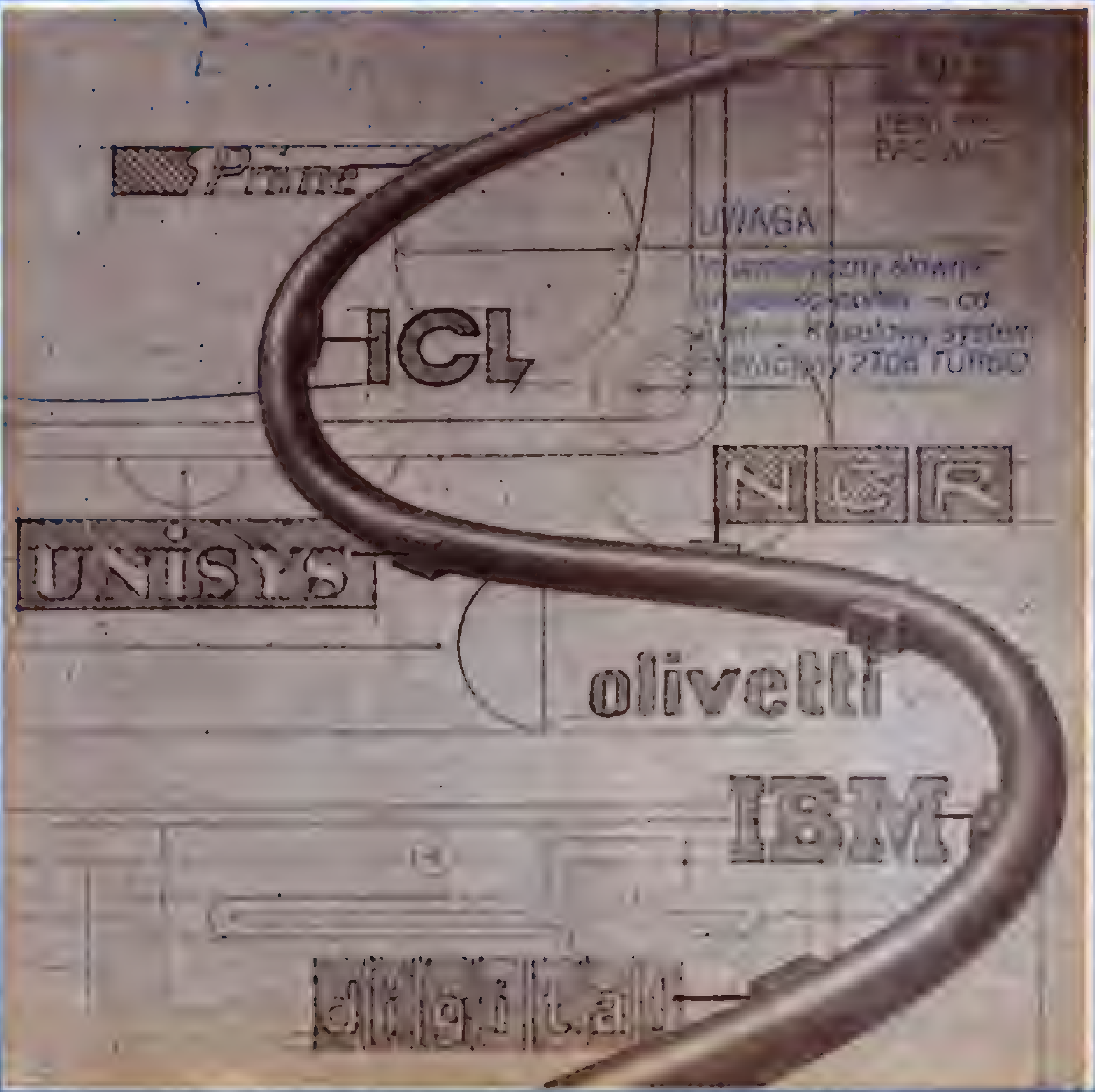
„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Pielà, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1611. Nr ind. 361682. U-11

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 zł

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" NR 11/1988 ISSN 0860-2794



Jeśli ocenialibyśmy poziom informatyki i jej zastosowań na podstawie ilości i rozmachu imprez targowych poświęconych tej dyscyplinie to z pewnością znaleźlibyśmy się w ścisłej czołówce państw europejskich i na czele krajów socjalistycznych.

Niestety gala trwa tylko kilka dni i zaraz po niej pojawia się szara rzeczywistość. *Komputer 88*, *Infosystem 88*, *Targi Poznańskie*, *Softarg* i *Informacja 88* w Katowicach przyciągały do swoich stoisk kolorowymi monitorami. A że forma dla pospolitego "oglądacza" ważniejsza jest niż treść, kiedy na koronie katowickiego Spodka na ustawionch tam kilkudziesięciu monitorach zaczęto wyświetlać filmy, arena opustoszała.

Nie oznacza to oczywiście, aby podobne imprezy były zbędne. Informatykastworzyła zdrowy rynek (jeden z nielicznych) i wszelkiego rodzaju targi, wystawy czy pokazy pozwalają wyłowić nowości, a i w konsekwencji urealnić ceny. A jeśli już o nie chodzi to ceny sprzętu dyktowane są przez czarnorynkowy kurs dolara, zaś cena oprogramowania zależy od podaży, popytu i jego wartości.

Oprogramowanie jest coraz lepsze. Gołym okiem już widać jego profesjonalizm. W Katowicach, pomimo iż zabrakło tam wielu softwarowych potentatów nie brakowało ciekawych propozycji.

Oczywiście *Informacje 88* można (i słusznie) skrytykować. Ale jej największa zaletą jest fakt, że po prostu była. Rodzima informatyka z pewnością na tym nie ucierpi. Wypada mieć nadzieję, że organizatorzy w przyszłym roku znacznie uatrakcyjnią formę, a i zwiedzającym przedstawią zanacznie więcej wzstawców.

I jeszcze kilka słów o katowickim Spodku: najlepsze naszym zdaniem propozycje programowe opiszemy na naszych łamach. Już w następnym numerze przedstawimy oprogramowanie dziennikarzowi najbliższe – czyli polska wersja Deskop Publishing Rolanda Wacławka.

REDAKCJA

W NUMERZE

Kasetowy system operacyjny 2T06 TURBO — str. 4
Losowy start generatora RND — str. 8
Loteryjka — str. 9
Programy narzędziowe na CPC — str. 10
Lilavati (10) — str. 12
Rady i porady programisty (2) — str. 13
Informatyka w szkole — Programowanie — str. 14
Komputerowe piractwo — niebezpieczeństwo XXI wieku — str. 21
Chińczyk — str. 22
Jak płynnie i szybko przesuwać... — str. 24
Drugie prawo dynamiki — str. 26
Żeglowanie — str. 28
Informatyczny słownik — str. 29
Liga Myślących — str. 31
Giełda pomysłów — str. 31

HEWLETT-PACKARD

Drukarka laserowa firmy Hewlett Packard model Laser Jet II jest w chwili obecnej najbardziej popularnym typem drukarki laserowej na świecie, nie tylko ze względu na niską cenę, ale przede wszystkim z powodu bardzo dużych możliwości użytkowych.

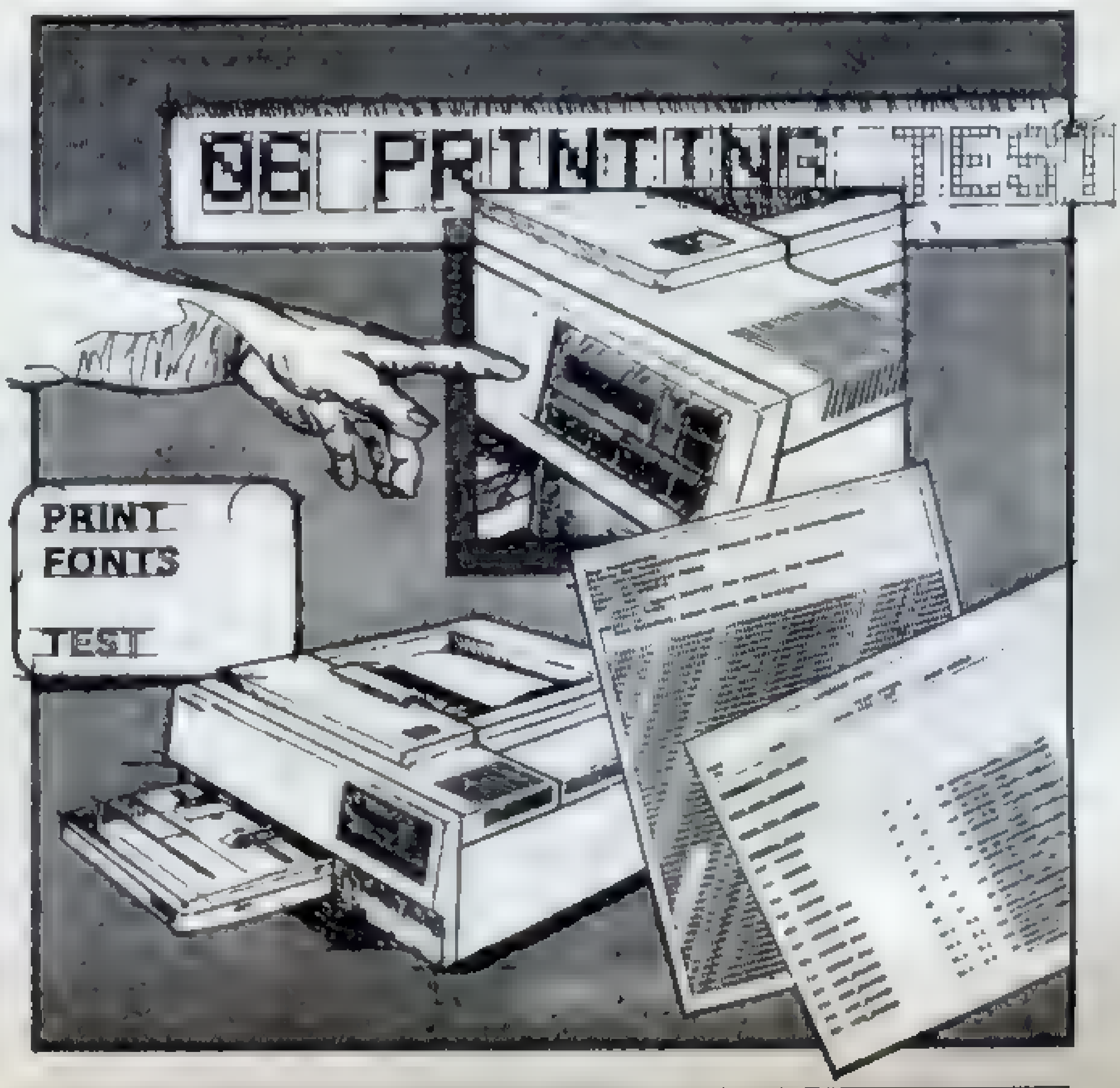
Standardowo jest ona wyposażona w pamięć 512 kB RAM z możliwością rozszerzenia do 1MB, 2MB, lub 4 MB. Jako jedna z nielicznych drukuje na papierze każdego rodzaju, foliach przeźroczystych, foliach i papierach samoprzylepnych typu JAC.

Przy zastosowaniu scanera można wykorzystywać ją jako kopiarkę z możliwością obróbki cyfrowej obrazu i druku. Znajduje szerokie zastosowanie w małej poligrafii do składania druku i zdjęć.

Jedną z nielicznych firm oferujących ten model drukarki laserowej jest znana z wysokiej jakości sprzętu służącego do opracowań graficznych przy zastosowaniu techniki komputerowej BIURO OBSŁUGI IMPORTU i EKSPORTU BOIiE mające siedzibę w Warszawie przy ulicy Chłodnej 35/37, pawilon 1a, telefon 24 76 16, tlx 817073 BOIiE PL.

Ostatnim szlagierem proponowanym przez firmę BOIiE jest stacja graficzna APOLLO w standardowej konfiguracji 4MB lub 8 MB RAM, 167 MB hard dysk i rozdzielczości 1024 x 1024 z 1 MB pamięci RAM na dwudziestocalowym kolorowym monitorze.

Na zorganizowanym w listopadzie br. pokazie sprzętu mikrokomputerowego nowej generacji oprócz drukarek laserowych, stacji graficznych, zaprezentowano różnego rodzaju zestawy komputerowe i kartę do obróbki cyfrowej obrazu. Atrakcyjna forma tej prezentacji, z wykorzystaniem dużych ekranów wizyjnych i profesjonalnych mikserów video, cieszyła się dużym zainteresowaniem.



LaserJet series II Printer

Nauka • Technika • Nauka

Firma Dell zapowiedziała rozpoczęcie, z końcem bieżącego roku, produkcji komputera będącego pierwszym odpowiednikiem IBM PS/2 jaki ukaże się na rynku. Jednakże zanim to nastąpi najlepszym komputerem firmy pozostanie najnowszy model o nazwie System 310, bazujący na mikroprocesorze 80386, z częstotliwością zegara 20 MHz, pracujący w standardzie AT. Jego pamięć statyczna RAM jest ograniczona do 32 kbajtów, jednakże zastosowanie procesora o częstotliwości 20 MHz i układu kontrolującego Intel 82385 powoduje, że jest on jednym z szybszych komputerów tej klasy. Jego cena bez wyposażenia dodatkowego 2699 dolarów USA.

Na amerykańskim rynku komputerowym coraz większą popularność zdobywa firma Elxsi, której siedziba znajduje się w San Jose w Kalifornii. Firma produkuje szeroki asortyment komputerów konkurujących wysoką jakością wykonania i niższą ceną z produktami takich renomowanych firm jak IBM czy Digital Equipment Corporation. W 1987 roku obroty jej osiągnęły wielkość 25,2 mln dolarów USA, co stanowi wzrost o prawie 9 mln w stosunku do roku poprzedniego, natomiast w roku bieżącym przewidywany jest wzrost o prawie 10 mln.

Komputery firmy Elxsi są najczęściej wykorzystywane do symulacji i modelowania w czasie rzeczywistym, a ich duża szybkość działania czyni je bardzo przydatnymi do obsługi dużych baz danych. Podstawę konstrukcji komputerów Elxsi stanowi bus danych pracujący z szybkością 320 Megabajtów na sekundę. Zastosowanie wspólnego busu pozwala na łatwą rozbudowę systemu poprzez wymianę procesora na szybszy i zwiększenie pamięci. W najnowszych komputerach Elxsi zastosowany jest 64-bitowy procesor równoległy 6460 i pamięć o pojemności 2000 megabajtów.

Wspomagający procesor neuronowy ANZA-Plus firmy Hecht-Nielsen Neurocomputers umożliwia komputerom IBM PC-AT lub kompatybilnym pracę jako komputer neuronowy wykonujący obliczenia w czasie rzeczywistym. Tak „przerobione" komputery znajdują zastosowanie szczególnie przy analizowaniu obrazów i sygnałów. Sieci neuronowe mogą być stosowane z dowolnymi kombinacjami nie więcej niż 2,5 miliona elementów obliczeniowych i połączeń między nimi. Maksymalna szybkość działania takiego systemu wynosi 10 milionów połączeń na sekundę. Wszystkie funkcje wiążące się z działaniem sieci neuronowej wykonywane są w pamięci procesora wspomagającego umożliwiając tym samym niezależną obsługę funkcji wejścia/wyjścia komputera.

Firma Hetch-Nielsen Neurocomputers (HNC) z San Diego uzyskała zamówienie od armii Stanów Zjednoczonych na zaprojektowanie sieci komputera neuronowego do analizowania i rozwiązywania problemów współczesnego pola walki. Na opracowanie projektu architektury wojskowego komputera sieci neuronowej firma otrzymała zaliczkę wysokości 50 tys. dolarów USA i pół roku na jego zaprezentowanie. Przewiduje się, że komputery sieci neuronowej znajdą zastosowanie głównie przy analizie danych z urządzeń kontrolnych, czujników oraz przy rozwiązywaniu problemów naukowych współczesnego pola walki, takich jak obróbka sygnałów radarowych, sonarowych czy obrazów. Ponieważ komputery sieci neuronowej tworzą własne algorytmy, ich sposób rozróżniania szczegółów jest podobny do ludzkiego, to mogą być przydatne w przewidywaniu zachowań przeciwnika na podstawie obserwacji z ćwiczeń. Wyraża się nadzieję, że komputery te poprzez obserwację czynności człowieka, będą w stanie nauczyć się sterowania robotami zastępującymi człowieka tam, gdzie jest to szczególnie niebezpieczne lub tam, gdzie wykonywane czynności są stale powtarzane. Przewiduje się także, że przyczynią się one do powstania nowych generacji broni, które będą eliminować zmęczenie ludzkie, samoczynnie naprawiać uszkodzenia powstałe w wyniku działań bojowych oraz uwzględniające zmiany zachodzące na polu bitwy.

Firma GTE Governement Systems Corp. otrzymała zamówienie wartości 7,3 mln dolarów USA od dowództwa amerykańskich sił powietrznych na dostarczenie oprzyrządowania komputerowego wraz z oprogramowaniem dla elektronicznych systemów ćwiczebnych strategicznego lotnictwa bombowego. Za pomocą tych systemów załogi samolotów bombowych wykonują symulacje lotów bojowych, w trakcie których doskonalą się w wyszukiwaniu celów strategicznych i zwalczaniu obrony przeciwlotniczej przeciwnika.

Kasetowy system operacyjny 2TO6 TURBO

Wojciech ZABOŁOTNY

Program K.S.O. 2TO6 powstał jako system usprawniający programowanie i posługiwanie się programami narzędziowymi (SPEEDSCRIPT, SOUNDMACHINE i inne), dzięki zwiększeniu szybkości i wygody obsługi magnetofonu. Program umożliwia korzystanie z takich języków, jak: ASSEMBLER-EDYTOR, MAC-65, BASIC-XL, MICROSOFT-BASIC i innych, eliminując długie oczekiwanie na wgranie się używanego języka i uruchamianego programu. W praktyce jest możliwe wykorzystanie programu także do przyspieszonego wczytywania gier. Uzyskano szybkość transmisji około 3000 bit/sek. W obrębie rekordu, co daje efektywną prędkość transmisji około 2000 bit/sek (oznacza to, że na przykład gra „TOMAHAWK" ładuje się 4 minuty zamiast 20). Przy zastosowaniu dość dobrego magnetofonu (M 8010) i taśm z nośnikiem AGFA lub BASF program działa pewnie. Jeśli wystąpią kłopoty, należy skontrolować równomierność przesuwu taśmy i czystość głowicy. Przy używaniu magnetofonu innego typu niż M 8010 mogą się okazać konieczne następujące zmiany:

— rezystor RS należy dobrać tak, aby uzyskać właściwą wartość sygnału na wejściu magnetofonu przy zapisie,

— można pobierać sygnał z gniazda słuchawkowego magnetofonu, jeśli sygnał na wyjściu DIN jest zbyt słaby,

— można pobierać sygnał z gniazda słuchawkowego magnetofonu, jeśli sygnał na wyjściu DIN jest zbyt słaby.

Aby było możliwe przenoszenie programów między magnetofonami, konieczna jest także staranna regulacja skosu głowicy. Dla pełnego wykorzystania funkcji programu konieczne jest sterowanie przez komputer pracą silnika magnetofonu.

W dalszym ciągu artykułu będzie opisany interfejs dopasowujący sygnały komputera i magnetofonu oraz sterujący silnikiem oraz opis koniecznej przeróbki magnetofonu.

Niestety, nie jest możliwe przerobienie w prosty sposób magnetofonów „firmowych". Większość z nich zawiera kondensator (niezbędny do normalnej pracy) dołączony równolegle do głowicy uniwersalnej, co poprawia zapis i odczyt częstotliwości używanych w zapisie „firmowym", ale uniemożliwia zapis i odczyt impulsów używanych przez system TURBO. Oznacza to, że ci, którzy korzystają z magnetofonu firmowego, będą musieli dołączyć do zestawu zwykły magnetofon, ale jeśli ktoś używa zwykłego magnetofonu z dołączonym interfejsem (a jest to dość powszechne), może dołączyć do niego także interfejs **TURBO**.

Jak otrzymać program TURBO

Do artykułu dołączone są dwa listingi. Są to programy napisane w języku ATARI-BASIC. Aby wygenerować program K.S.O. 2TO6 należy wprowadzić do komputera PROGRAM1. Po uruchomieniu zapisze on na kasecie K.S.O. 2TO6 jako plik 3OOT (przed uruchomieniem dobrze by było zapisać ten program na kasecie instrukcją CSAVE).

Program 2 umożliwia otrzymanie programu kopiującego, pozwalającego przenosić pliki dowolnej długości między dyskietkami a taśmą w systemie TURBO.

Po uruchomieniu na dyskietce zostanie zapisany program kopiujący pod nazwą D:KOPIARKA.TUR.

Wgrywanie programu

1. Jeżeli chcesz pracować w języku ATARI-BASIC, to włącz komputer trzymając wciśnięty klawisz START. Jeżeli BASIC ma być wyłączony — trzymając OPTION i START.
2. W magnetofonie firmowym umieść kasetę z K.S.O. 2T06 i wciśnij klawisz PLAY (START w niektórych magnetofonach).
3. Naciśnij RETURN i program powinien się wczytać do komputera. Jeżeli był wyłączony, to na ekranie pojawi się napis READY, a o tym, że program wgrał się poprawnie, możesz się przekonać wprowadzając instrukcję DOS. Powrót do BASIC-a umożliwia opcja C menu K.S.O. (UWAGA! Jeżeli po wgraniu systemu nie był naciskany klawisz RESET, to przejście do K.S.O. i powrót do BASIC-a spowoduje skasowanie programu w BASIC-u).

UWAGA! Niektóre programy mogą działać błędnie, jeśli po wgraniu K.S.O. był używany klawisz RESET (dotyczy to zwłaszcza programów kopiowych z cartridge'ów, przykładem jest THE LAST STARFIGHTER).

Menu K.S.O.

K.S.O. 2TO6 umożliwia współpracę nie tylko z magnetofonem w trybie TURBO, ale także z magnetofonem firmowym. Dlatego w menu K.S.O. 2TO6 znajdują się opcje związane ze zwykłym trybem transmisji:

D — długie przerwy,

K — krótkie przerwy.

Powyższe funkcje służą do ustalenia rodzaju przerw międzyrekordowych w plikach ładowanych funkcjami R lub L z magnetofonu firmowego.

E — włączenie silnika,

H — zatrzymanie silnika.

Te funkcje dotyczą także magnetofonu firmowego. Ich zadaniem jest ułatwienie odnalezienia początku pliku na taśmie, co przy klasycznym trybie zapisu jest sprawą niezwykle ważną. Ponieważ tryb TURBO jest odporny na ustawienie taśmy daleko przed początkiem zbioru, a ponadto interfejs jest wyposażony w wyłącznik umożliwiający włączanie silnika niezależnie od sterowania przez komputer, nie wprowadziłem podobnych funkcji dla magnetofonu TURBO.

Następne funkcje dotyczą obu rodzajów magnetofonów.

R — załadowanie i uruchomienie programu w kodzie maszynowym,

L — załadowanie programu w kodzie maszynowym.

Wyczerpujące omówienie działania tych funkcji wymaga przedstawienia struktury plików wczytywanych przez system DOS.

Na początku takiego pliku znajdują się dwa bajty FF określające rodzaj pliku. Po nich następują segmenty pliku. Każdy segment na początku zawiera cztery bajty o specjalnym znaczeniu. Dwa pierwsze określają adres komórki, w której znajdzie się pierwszy bajt segmentu. Następne dwa określają adres, pod który zostanie wpisany ostatni bajt segmentu. Ilość bajtów segmentu, które są zapisane po tych czterech bajtach, jest oczywiście równa ADRES KOŃCA—ADRES POCZĄTKU. Niektóre programy generujące pliki w formacie DOS oddzielają segmenty dwoma bajtami $FF (np. KYAN PASCAL).

Specjalne znaczenie mają komórki 736,737,738 i 739. Jeśli wczytany segment zapisze niezerową wartość do komórek 738 739, to K.S.O. uruchamia procedurę zaczynającą się od wpisanego tam adresu. Jeżeli owa procedura zakończy się instrukcją RTA, to będzie kontynuowane ładowanie pliku. Zawartość komórek 736, 737 jest badana po zakończeniu wczytywania pliku. Jeśli jest tam wpisana wartość niezerowa, to nastąpi uruchomienie programu od adresu podanego w tych dwóch komórkach. Jeśli nie, to gdy program był ładowany opcją L, nastąpi powrót do MENU K.S.O., jeśli zaś była użyta funkcja R., nastąpi skok na początek ostatniego wczytanego segmentu pliku. Przeciętnemu użytkownikowi wystarczy informacja, że funkcja R może służyć do ładowania większości programów wczytywanych normalnie przez loader „z wykrzyknikiem" lub BLC. Programy takie można próbować po przekopiowaniu ich (np. programem FCOPY) wczytywać w systemie TURBO.

C — powrót do cartridge'a

Ta funkcja umożliwia powrót do BASIC-a (lub programu umieszczonego na cartridge'u) po wywołaniu K.S.O. instrukcją DOS. Funkcja ta działa oczywiście tylko wtedy gdy włączony jest BASIC lub włożony cartridge z innym programem.

— *zimny start systemu*

Ta funkcja nie wymaga komentarza.

Nazwy plików

Nazwa pliku może zawierać do 10 znaków. Użycie nazwy w instrukcji np. SAVE pokazane jest na poniższym przykładzie

SAVE „D: s0123456789"

Znaki „D:" stanowią identyfikator urządzenia. Użyto identyfikatora takiego samego, jak dla stacji dysków ze względu na zgodność z takimi programami narzędziowymi jak SOUNDMACHINE czy MAGIC PAINTER. Znak „s" jest znakiem sterującym, ważnym tylko w trybie zapisu. Informuje on system, czy ma nastąpić rzeczywiście zapis pliku, czy jego weryfikacja. Umieszczenie na jego miejscu znaku „V" powoduje nie zapisywanie nowego pliku, lecz weryfikację tzn. porównanie tego, co ma być zapisane na taśmie z tym, co się na niej znajduje. W poniższych przykładach znakiem sterującym jest V (dla weryfikacji) albo spacja (dla zapisu). Niektóre programy (np. program graficzny KOALA) nie tolerują jednak spacji w nazwie pliku. Można wówczas dla zapisu użyć znaku sterującego będącego dowolną literą różną od „V"

PRZYKŁAD

Używamy programu SPEEDSCRIPT, chcemy zapisać znajdujący się w pamięci tekst na taśmę pod nazwą „TEKST-12". W tym celu naciskamy CONTROL+S, podajemy nazwę pliku „D: TEKST-12", naciskamy RETURN, włączamy ZAPIS i START w magnetofonie i naciskamy RETURN jeszcze raz. Po zapisaniu programu należałoby sprawdzić poprawność zapisu. Dyrektywa CONTROL+L wprowadza tekst z urządzenia zewnętrznego za kursor, usuwając tekst znajdujący się tam wcześniej. Gdybyśmy chcieli jej użyć do wczytania właśnie zapisanego tekstu, to przy ustawieniu kursora na końcu zapisywanego tekstu mogłoby zabraknąć wolnej pamięci. Natomiast po ustawieniu kursora na początku ewentualny błąd spowodowałby utratę całego tekstu. Na szczęście K.S.O. 2TO6 oferuje TRYB WERYFIKACJI.

Cofamy taśmę do początku pliku, naciskamy CONTROL+S podajemy, nazwę pliku „D: „TEKST-12", naciskamy RETURN, włączamy START w magnetofonie i naciskamy RETURN jeszcze raz. Ewentualne przekłamanie zostanie zgłoszone jako błąd 145, 143, 140 lub 136

Posługiwanie się trybem weryfikacji w języku ATARI-BASIC (i niektórych innych) nastręcza pewne trudności: mianowicie instrukcja SAVE zapisuje także linię wprowadzoną w trybie bezpośrednim, a więc zapisana zostaje także podana nazwa pliku. Można sprawdzić, że zapisanie programu instrukcją SAVE „D: PROGRAM1" i zweryfikowanie go instrukcją SAVE „D: VPROGRAM1" zakończy się błędem 145. Przyczyną tego jest właśnie różnica między nazwami pliku użytymi w pierwszej i drugiej instrukcji. Nie oznacza to jednak, że instrukcja SAVE nie może współpracować z trybem weryfikacji. Podam jeden ze sposobów uniknięcia tego kłopotu: Należy wprowadzić obie instrukcje SAVE w jednej linii, a więc piszemy SAVE „D: PROGRAM1" SAVE „D: VPROGRAM1" i naciskamy RETURN, włączamy ZAPIS i START w magnetofonie i naciskamy ponownie RETURN. Program zostanie zapisany. Następnie cofamy taśmę do początku zapisanego właśnie zbioru i naciskamy RETURN. Teraz weryfikacja nastąpi w sposób właściwy

Te kłopoty występują jednak tylko przy użyciu instrukcji SAVE. Znaki „0123456789" w pierwszym przykładzie oznaczają nazwę pliku i są zapisywane w jego nagłówku na taśmie. Nazwy krótsze są uzupełniane spacjami do 10 znaków, nazwy dłuższe są „obcinane".

Oto odpowiednie przykłady:
SAVE: „D: TEXT" zapisze na taśmie plik o nazwie „TEXT".
SAVE: „D: PROGRAM BARDZO CIEKAWY" zapisze plik o nazwie „PROGRAM BA".

Nazwa może zawierać dowolne znaki poza trzema zastrzeżonymi: *, /, @. Te znaki spełniają ważne funkcje przy odnajdywaniu na taśmie zbiorów przy odczycie lub weryfikacji

ZNAK / oznacza, że dany znak nazwy nie będzie porównywany przy identyfikacji pliku.
Na przykład.
Instrukcja LOAD „D: PROGRAM15" może spowodować wczytanie zarówno programu „PROGRAM15", jak i programu „PROGRAMY5" (w zależności od tego, który będzie znaleziony wcześniej).

Znak @ oznacza, że identyfikację nazwy należy zakończyć na tym znaku.
Na przykład
Instrukcja LOAD: „D: PR@" spowoduje wczytanie pierwszego programu o nazwie zaczynającej się od znaków „PR" (na przykład „PROGRAMY" „PRZYKŁADY" itp.).

Znak * powoduje, że jeżeli znaki występujące przed nim były zgodne, to zostaje wprowadzona pełna nazwa pliku z pytaniem czy jest on właściwy.
Przykład:

Instrukcja LOAD: „D: KW*" po napotkaniu na taśmie pliku o nazwie „KWIECIEŃ" spowoduje wyprowadzenie komunikatu „KWIECIEŃ (T/N)" i czekanie na decyzję użytkownika. Naciśnięcie klawisza „T" oznacza akceptację pliku i przejście do jego czytania (weryfikacji), naciśnięcie innego klawisza powoduje szukanie następnego pliku

Ważne dla „Hackerow"

Kasetowy system operacyjny w swej aktualnej wersji umożliwia także niestandardowe wykorzystanie trybu TURBO — transmisja przebiega wtedy bez podziału na bloki, czyli znacznie szybciej. Poza tym program tak zapisany jest trudniejszy do skopiowania.

Oto informacje niezbędne do wykorzystania tych możliwości: Pod adresem 1792($700) znajduje się instrukcja skoku do procedury transmisji. Procedura ta wymaga jako danych wejściowych adresu początku bloku danych, adresu końca bloku danych, i kodu operacji. Informacje te wprowadzamy w następujący sposób:

— adres początku umieszczamy w komórkach $50, $51,

— w komórkach $52, $53 umieszczamy adres pierwszego bajtu po przesłanym bloku (bajt ten nie będzie już przesłany),

— kod operacji umieszczamy w akumulatorze i w komórce pamięci $704 w następujący sposób: niezerowa zawartość komórki 704 wymusza tryb weryfikacji bez względu na zawartość akumulatora, zero w akumulatorze oznacza zapis, a wartość różna od zera — odczyt

Procedura transmisji zapisuje lub odczytuje wskazany blok danych uzupełniony sumą kontrolną

Informację o wyniku operacji zawiera rejestr Y, podobnie jak przy innych operacjach I/O

Struktura zapisu na taśmie

Do zapisu na taśmie wykorzystuje się system uproszczonej modulacji częstotliwości (podobnie jak w ZX SPECTRUM).

Występują trzy rodzaje impulsów generowanych na taśmie

1 — impuls synchronizacji (stan wysoki przez 800 cykli zegara i stan niski przez 800 cykli zegara),

2 — impuls „1" (stan wysoki przez 400 cykli zegara, i stan niski przez 400 cykli zegara),

3 — impuls „0" (stan wysoki przez 200 cykli zegara i stan niski przez 200 cykli zegara).

Blok tworzony na taśmie przez procedurę transmisji składa się z 4096 impulsów synchronizacji, po których następują impulsy kodujące bajty danych (od najstarszego do najmłodszego bitu), a po nich bajt sumy kontrolnej utworzonej jako suma modulo 256 (nie EXOR!) wszystkich nadanych bajtów

Przy korzystaniu z urządzenia D: na taśmie mogą być zapisane następujące bloki:

1 — blok nagłówka (zawsze na początku pliku),

2 — pełny blok danych (ilość tych bloków zależy od długości pliku),

3 — blok końca pliku (zawsze na końcu pliku).

BLOK NAGŁÓWKA ma długość 12 bajtów. Na początku zawiera bajty 0,255 stanowiące identyfikator nagłówka, a następnie 10 bajtów będących kodami ATASCII znaków nazwy pliku

PEŁNY BLOK DANYCH ma długość 3074 bajtów. Na początku znajdują się bajty 0,12 (0+12*256=3072) informujące o ilości bajtów danych, a potem 3072 bajty danych

BLOK KOŃCA PLIKU ma długość 3074 bajtów. Na początku znajdują się dwa bajty L,H informujące o ilości bajtów danych (L+255*H= ilość bajtów danych). Potem postępują bajty danych, a za nimi bajty uzupełniające blok do długości 3074 bajtów.

Oczywiście każdy z tych bloków poprzedzony jest impulsami synchronizującymi i zakończony bajtem sumy kontrolnej

Przeróbki „sprzętowe"

Podstawowym urządzeniem koniecznym do korzystania z K.S.O. 2TO6 jest interfejs zmieniający analogowy sygnał z magnetofonu w cyfrowy sygnał doprowadzony do komputera

Schemat tego układu i sposób jego realizacji opiszemy w kolejnym numerze IKS-a.

Dostępne oprogramowanie

Niektóre programy mogą pracować w systemie TURBO bezpośrednio po ich przekopiowaniu ze zwykłego magnetofonu. Niektóre wymagają dość daleko idących przeróbek

Mozl we jest przerobienie na standard TURBO nawet tak długiego programu jak TOMAHAWK

Wśród programów użytkowych istnieją specjalnie dostosowane do pracy w systemie TURBO: ASSEMBLER-EDYTOR i MAC85 (uzupełnione dodatkowo ramdyskami) VISICALC. Wiele języków (wśród nich znakomity TURBO-BASIC XL) nie wymaga zadnych przeróbek dla współpracy z systemem TURBO.

Istnieje ponadto program umozliwiający przenoszenie plików dowolnej długości między dyskami i taśmą w systemie TURBO oraz program umozliwiający kopiowanie plików o długości do 55 KB w systemie TURBO

Oto niektóre (jest ich znacznie więcej) programy działające z TURBO K.S.O. 2TO6
BEZ ŻADNYCH PRZERÓBEK
GRY: Alley Cat, Spindizzy, Combat Leader, Submarine Commander, Colossus Chess, Computer War, Pole Position, Moon Patrol, Behind Jaggi Lines, Boulder Dash, Montezumas's Revange. JĘZYKI: Pascal, Turbo Basic XL, ACTION!, Microsoft Basic, Assembler OSS

Jak używać Kyan Pascal w systemie Turbo

System TURBO umozliwia korzystanie z KYAN PASCALA bez zadnych przeróbek (poza oczywiście skopiowaniem z dysku na taśmę w systemie TURBO). Wykorzystuje się tu mozliwość pracy z dwoma magnetofonami. Można umieścić tekst źródłowy na magnetofonie normalnym i kompilować na magnetofon TURBO zleceniem PC/C-OD Xnazwa. Nie kazdy magnetofon firmowy zdązy zatrzymać się i ruszyć w czasie trwania przerwy między rekordami, mogą więc wystąpić klopoty z wczytywaniem tekstu źródłowego

Po wprowadzeniu zlecenia kompilacji kompilator zaząda dostępu do magnetofonu firmowego (ton pojedynczy) i po wciśnięciu klawisza RETURN wczyta tekst źródłowy Jeśli nie zostaną wykryte błędy składniowe, to kompilator zaząda dostępu do magnetofonu TURBO (ton poczwórny) i po wciśnięciu klawisza RETURN zapisze na nim nagłówek pliku z kodem wynikowym, a następnie pojedynczym tekstem zasygnalizuje konieczność powtórnego wczytania pliku z tekstem źródłowym. Nalezy wówczas cofnąć taśmę z tekstem do początku zbioru. Po naciśnięciu RETURN nastąpi powtórne wczytanie tekstu źródłowego z równoczesnym generowaniem kodu wynikowego

Aby uruchomić program nalezy wgrać go dyrektywą >D1 nazwa lub >D1.* Po wczytaniu kodu komputer zaząda wprowadzenia zbioru b: biblioteki, który musi być skopiowany na taśmę pod nazwą D XLIB. Po jego załadowaniu program powinien się uruchomić

Przy pracy z PASCALEM konieczne jest dokładne wprowadzanie nazw plików. Plik AUTORUN. SYS moze być skopiowany pod dowolną nazwą, ale pliki edytora, kompilatora i biblioteki muszą być nazwane D XED D.XPC i D.XLIB (X dowolna litera rózna od V) Edytor i kompilator po wgraniu pliku AUTORUN muszą być ładowane dyrektywami >D1 ED i >D1 PC bez zadnych znaków specjalnych, takze postać zlecenia kompilacji i wczytywania kodu nalezy zachować bez zmian (zamiast słowa „nazwa" nalezy wstawić litery tworzące nazwę programu)

Program do kopiowania z dyskietek

Przedstawiony program po załadowaniu ze stacji dysków (przez odpowiednią instrukcję DOS-a) umozliwia przenoszenie zbiorów dowolnej długości między taśmą w systemie TURBO i dyskiem (w obie strony), mozliwa jest też weryfikacja, to znaczy porównanie zbioru na taśmie ze zbiorem na dysku Przy operowaniu zbiorami na taśmie obowiązują te same reguły co przy K.S.O.; z jedną różnicą: identyfikatorem magnetofonu TURBO jest T: a nie D:

Oto przykłady uzycia programu

Aby skopiować zbiór LIB PASCALA trzeba podać

Plik wejściowy: D LIB

Plik wyjściowy: T:XLIB

Aby następnie sprawdzić poprawność zapisu:

Plik wejściowy: D.LIB

Plik wyjściowy: T.VLIB

lub

Plik wyjsciowy: T.V

itp

Wprowadzenie zamiast nazwy pliku wyjściowego lub wejściowego *1 spowoduje podanie zawartości dysku w stacji pierwszej (*2 — w drugiej itd)

Aby skopiować plik TOMAHAWK z taśmy na dysk pod nazwę TOM GRA nalezy podać

Plik wejściowy: T XTOMAHAWK

Plik wyjściowy: D:TOM.GRA

Uwagi

Procedura TURBO używa procesora do pomiaru czasu impulsów, w związku z tym konieczne było zablokowanie przerwań, czego konsekwencją jest zablokowanie klawisza BREAK. Przerwanie transmisji osiąga się przez równoczesne wciśnięcie klawiszy OPTION, SELECT i START.

TURBO nie toleruje starszych wersji systemu operacyjnego, dlatego też nie będzie pracować poprawnie z TRANSLATOREM, czy z innymi programami symulującymi stare wersje ROM-u. Oczywiście nie moze pracować także na starych modelach ATARI

Ewentualnych użytkowników systemu TURBO chciałbym ostrzec przed mogącymi się pojawić na giełdach wcześniejszymi wersjami (oznaczonymi iT...). Ze względu na inny sposób generowania sumy kontrolnej i niepełną diagnostykę błędów występuje w nich niebezpieczeństwo niewykrycia błędu transmisji (co na przykład przy kopiowaniu będzie prowadzić do wytworzenia błędnej kopii). Przy częstym korzystaniu z systemu jego wgrywanie moze okazać się uciązliwe Została w związku z tym opracowana także wersja umieszczona na cartridge'u, zgodna z wersją wgrywaną z kasety

```
10 REM PROGRAM 1
20 REM Generuje K.S.O. 2T06 Jako
plik BOOT na kasecie.
30 REM Nie mozna zmieniac numerow
 linii DATA !!!
40 GRAPHICS 0:? "▶▶PROSZE CZEKAC"
50 READ LB
60 DIM A$(LB),B$(22)
70 LICZN=0:CHKSUM=0
80 FOR I=1 TO LB
90 IF LICZN=0 THEN READ B$
100 GOSUB 270:A$(I)=CHR$(BYTE)
110 CHKSUM=CHKSUM+BYTE CHKSUM=CHK
SUM-256*(CHKSUM>255)
120 LICZN=LICZN+1
130 IF LICZN<>10 AND I<>LB THEN 1
70
140 GOSUB 270
150 IF CHKSUM<>BYTE THEN A=10*INT
((I-1)/10)+320:? "BLAD DANYCH W L
INII ";A:LIST A:END
160 LICZN=0
170 NEXT I
180 POKE 764,255:GRAPHICS 0
190 ? " Prosze wlozyc do magnetof
onu kasete,"
200 ? " na ktorej ma byc zapisany
 program"
210 ? " w formie pliku BOOT, wcis
nac PLAY"
220 ? " i RECORD w magnetofonie i
 nacisnac"
230 ? " RETURN."
240 OPEN #1,8,128,"C:"
250 ? #1;A$;:CLOSE #1
260 END
270 B=ASC(B$(LICZN*2+1)):GOSUB 30
0:BYTE=Y
280 B=ASC(B$(LICZN*2+2)):GOSUB 30
0:BYTE=Y+16*BYTE
290 RETURN
300 Y=(B-48)*(B<58)+(B-87)*(B>96)
:RETURN
310 DATA 2056
320 DATA 0011f306c208a93c8d0248
330 DATA d318604c8d08000d2135d7
340 DATA 499199361bcad0fd60be50
350 DATA 0407200c07a9008d00d397
360 DATA a9008d1ad0ad1fd0f01154
370 DATA be0507200c07a9df8d0066
380 DATA d3a9088d1ad060a0804c2d
390 DATA 6208a20886300a48a000e9
400 DATA 9002a00220100768c630b2
410 DATA f008a20c200c074c3f071d
420 DATA 60a008ad1fd0f0d7ad0035
430 DATA d330f6a9008d1ad0aae8e0
440 DATA 301ead00d310f8a9088df4
450 DATA 1ad0ec0a07b00b6649ec31
460 DATA 0b07263088d0d460244992
470 DATA 30cfa08c4c5208209b0836
480 DATA a900a01085309848a004c8
490 DATA 20100768a8c630d0f38850
500 DATA d0f0a432a9008532a202ea
510 DATA 200-073648b132481855a5
520 DATA 31853168203b0768a8c82e
530 DATA f005c4004cce07e633c4e5
540 DATA 34d009a535c533d0054ce6
550 DATA e107eaeaea4cb407a53163
560 DATA a201200c07203b07a00142
570 DATA 4c62094c3607209b08a0e4
580 DATA 00a9ff8549a200ad1fd098
590 DATA f0edad00d330f6a9008d51
600 DATA 1ad0e830ecad00d310f8c7
610 DATA a9088d1ad0ec0a0790d450
620 DATA c8d0daa432a90085329890
630 DATA 4820580768a8a530ae03ed
```

```
640 DATA 07d02691321865318531111
650 DATA c8d002e633c434d0e2a513
660 DATA 33c535d0dc205807a0010c
670 DATA a530c531f002a08f4c62a6
680 DATA 08d132f0d3a091aec10821
690 DATA 9aa9408d0ed4a9228d006b
700 DATA d4a9388d02d3a9008d00b8
710 DATA d3a93c8d02d35860863444
720 DATA 8435a2fa8632a20e8633ba
730 DATA ae0307d006aad0034c90a1
740 DATA 074cf307a9008d0ed4858b
750 DATA 318d00d4a9088d1fd0a9f3
760 DATA 388d02d3a9608d00d3a99f
770 DATA 3c8d02d378bae8e88ec18e
780 DATA 086000a90c850ba9bf8528
790 DATA 0aa91a950f3de302a9fca5
800 DATA 850e8de702a9808d280a96
810 DATA a244a90aa01b4c86e4a949
820 DATA 31a00e20250ca21da90ce0
830 DATA 9d420320c8094cb00c18d3
840 DATA 9001386648207e0ca20399
850 DATA a9009de002ca10faa21047
860 DATA a9039d4203a9209d440382
870 DATA a90b9d4503a9049d4a03b2
880 DATA ad290a9d4b0320c803203d
890 DATA 980990b5209809b0fba988
900 DATA 079d4203a90a9d4503a9b2
910 DATA 139d4403a9023d4803a9eb
920 DATA 009d490320c809386e1681
930 DATA 0aad170a9d4403ad180a0c
940 DATA 9d450338ad190aed170a07
950 DATA 9d4203ad1a0aed180a9d8c
960 DATA 4803fe4803d003fe48031e
970 DATA 20c809ade202d005ade305
980 DATA 02f00b20130aa9009de257
990 DATA 029de7024c310e1d6e16ed
1000 DATA 0aa21da3079d4203a90aee
1010 DATA 9d4503a9179d4403a90222
1020 DATA 9d4803a9009d490320c884
1030 DATA 09ad170a2d180ac9fff062
1040 DATA 02186033602056e4983096
1050 DATA 01604aa90c9d420320564c
1060 DATA e46e2c160a3024c90ed059

1070 DATA 20ade002d00cade102d044
1080 DATA 07a548300f4cbf0cade01b
1090 DATA 028d170aade1028d180a0a
1100 DATA 6c170a48a948a00e2025c3
1110 DATA 0ca00068204c0c4cb00c57
1120 DATA 6ce20200bf0c0000500acc
1130 DATA 390bc10b6e0b100c120c8f
1140 DATA 000086438444a001a9006a
1150 DATA 9143a9ffc89143c8b1241f
1160 DATA c99bf0089143c8c00dd0b4
1170 DATA f360a9209143c8c00dd009
1180 DATA f960a9008d0307854585f1
1190 DATA 46a52a290cc904f035c9f6
1200 DATA 08f00160a002b124c956e5
1210 DATA f025a90420fcfd301ba2ad
1220 DATA f9a00e20290aa900a206f8
1230 DATA a00f20810820140c983058
1240 DATA 05a9808545604c840b38c3
1250 DATA 6646186647a90320fcfdf9
1260 DATA 30f0a901a206a00f2081bb
1270 DATA 09c080f0e39830f0a22d5d
1280 DATA a00b20290aa200bd2e0bf3
1290 DATA c92ff00dc92af016c940ea
1300 DATA f00addfa0ed0d3e8e00c40
1310 DATA d0e720140ca00184456001
1320 DATA a2028a48bdfa0e20b0f2fe
1330 DATA 68aae8e00cd0f1a2008ad1
1340 DATA 48bd1a0b20b0f268aae8b7
1350 DATA e006d0f1a9ff8dfc0220b1
1360 DATA fdf2295f09404820b0f27b
1370 DATA a99b20b0f268c954f0bcb2
1380 DATA 4ca10a2028542f4e29442f
1390 DATA a908a0c42086e4a24ca965
1400 DATA 08a0d02086e4a24ea90b0b
1410 DATA a0552086e4a446100a20ae
1420 DATA c20b98103ec088d026a03f
1430 DATA 01a5451020a900a891437f
1440 DATA e643d002e644a91ac54470
1450 DATA d0efa9fcc543d0e9a9003e
1460 DATA a2fca01a20810860a44689
1470 DATA 10188d270a20c20bc0001c
1480 DATA 300dcd270af008a091a929
1490 DATA 00854585466a44530033a
1500 DATA a08760a0009143e643d02e
1510 DATA 02e644eefa0ed003eefb0c
1520 DATA 0eadfb0ec90cf003a00139
1530 DATA 60a900a2fca01a20810843
1540 DATA 981004a900854520140ca2
1550 DATA 60a44588f003a08360ad96
1560 DATA fa0e0dfb0ed022a54710a2
1570 DATA 03a08860a901a2fca01a2f
1580 DATA 208108201c0c93309ca92d
1590 DATA 0ccdfb0ef005386647d0b9
1600 DATA d6a000b14348e643d00266
1610 DATA e64438adfa0ee9018dfaee
1620 DATA 0eadfb0ee9008dfb0e6899
1630 DATA a00160a9008dfa0e8dfb60
1640 DATA 0ea90e8544a9fc854360bb
1650 DATA a20084cc85cb1869029d1d
1660 DATA 44039869009d4503a000ea

1670 DATA b1cb9d4803c8b1cb9d4978
1680 DATA 03a90b9d42034c5be4851c
1690 DATA d484d520aad920e6d8a973
1700 DATA 0085cdf00520b0f2e6cd2f
1710 DATA a4cdb9800510f4498020cb
1720 DATA b0f24c770ca92020b0f2c7
1730 DATA e6cda906c5cdd0f360a987
1740 DATA 81a00e20250ca200a90557
1750 DATA 9d4203a90b9d4503a9209b
1760 DATA 9d4403a90e9d4803a900c7
1770 DATA 9d49032056e41004c08967
1780 DATA d016bc480388f01060a9e5
1790 DATA 5ea00e20250ca9ff8dfc73
1800 DATA 0220fdf2a2ff9aa90185ee
1810 DATA 09a900852ba90c852a20d4
1820 DATA 8eefa917a00d20250cadbc
1830 DATA 280a100aa928a00e2025cc
1840 DATA 0c4cee0ca920a00e2025da
1850 DATA 0ca951a00d20250ca9ff86
1860 DATA 8dfc0220fdf2295fa00850
1870 DATA d99f0ef0058810f830b43f
1880 DATA 980aa8b9a90e48b9a80eb0
1890 DATA 48603800cbe1f3e5f4eff7
1900 DATA f7f9a0f3f9f3f4e5eda0cc
1910 DATA eff0e5f2e1e3f9eaeef910
1920 DATA a0d7dae1e2a0b2d4b0b6b0
1930 DATA 9b9b2020202020202205016
1940 DATA 727a657277793a20202063
1950 DATA cd009b9b7f7fa0cda0c536
1960 DATA a0cea0d5a09b9bcc2d4c34
1970 DATA 61646f77516e696520700c
1980 DATA 6c696b7520d22d4c6164f1
1990 DATA 6f77616e6965206920739
2000 DATA 746172749bd32d5374610e
2010 DATA 72742070726f6772616d0c
2020 DATA 7520201b9d2d5a696d6e44
2030 DATA 79207374617249bcb2d9e
2040 DATA 4b726f746b696520707279
2050 DATA 7a6572777920c42d446c7b
2060 DATA 756769652070727a657278
2070 DATA 77799bc52d5572756368fc
2080 DATA 6f6d69656e696520202042
2090 DATA 20c82d5a6174727a796d58
2100 DATA 616e69659b20207369 6c18
2110 DATA 6e696b612020202020207b
2120 DATA 202020202073696c6e693a
2130 DATA 6b619bc32d506f77726fa3
2140 DATA 7420646f20636172747 24b
2150 DATA 69646765276196060004451
2160 DATA 4c5547494507004b524fba
2170 DATA 544b494515009b7ffd5a6d
2180 DATA 6c7920665f726d6174201b
2190 DATA 706c696b7520211 4009b30
2200 DATA 7ffd426c616420747 26186
2210 DATA 6e736d69736a693a2021fe
2220 DATA 009b2020202020a0cec7
2230 DATA e1e3e9f3eee9eaa0e4ef3b
2240 DATA f7efeceef9a0ebece1f7a3
2250 DATA e9f3faa01c009b20202030
2260 DATA 20202020[illegible]f64516abe

2270 DATA 206e617a776520706c6968
2280 DATA 6b759b3f4c52534b4445e7
2290 DATA 481d43f908fc08fe09b954
2300 DATA 0ec10ec90ed10ef10edec4
2310 DATA 0ea9808d280a4cbf0ca97a
2320 DATA 008d280a4cbf0ca9348dba
2330 DATA 02d34cbf0ca93c8d02d3ed
2340 DATA a9038d0fd24cbf0cadf8c3
2350 DATA 03d0f8a9fd8d01d320efa4
2360 DATA 0e6cfabf6cfebfa9018d37
2370 DATA 44026cfcff00e4
```

```
10 REM PROGRAM 2
20 REM Zapisuje na dyskietce w pl
iku D:KOPIARKA.TUR
30 REM program przenoszacy pliki
miedzy systemem TURBO i dyskami
40 GRAPHICS 0:? "▶▶PROSZE CZEKAC"
50 READ LB
60 DIM A$(LB),B$(22)
70 LICZN=0:CHKSUM=0
80 FOR I=1 TO LB
90 IF LICZN=0 THEN READ B$
100 GOSUB 270:A$(I)=CHR$(BYTE)
110 CHKSUM=CHKSUM+BYTE:CHKSUM=CHK
SUM-256*(CHKSUM>255)
120 LICZN=LICZN+1
130 IF LICZN<>10 AND I<>LB THEN 1
70
140 GOSUB 270
150 IF CHKSUM<>BYTE THEN A=10*INT
((I-1)/10)+320:? "BLAD DANYCH W L
INII ";A:LIST A:END
160 LICZN=0
170 NEXT I
180 POKE 764,255:GRAPHICS 0
190 ? " Prosze wlozyc do stacji d
yskow dys- "
200 ? " kietke na ktorej ma byc z
apisany "
210 ? " program kopiujacy i nacis
nac"
220 ? " RETURN."
230 POKE 764,255:GET #16,A
240 OPEN #1,8,128,"D:KOPIARKA.TUR
"
250 ? #1;A$;:CLOSE #1
260 END
270 B=ASC(B$(LICZN*2+1)):GOSUB 30
0:BYTE=Y
280 B=ASC(B$(LICZN*2+2)):GOSUB 30
0:BYTE=Y+16*BYTE
290 RETURN
300 Y=(B-48)*(B<58)+(B-87)*(B>96)
:RETURN
310 DATA 1865
320 DATA fff00209e224c8d2100d8
330 DATA 0d2135493199361bcad099
340 DATA fd60be0420200c20a900cd
350 DATA 8d00d3a9008d1ad0ad1f19
360 DATA d0f011be0520200c20a9c2
370 DATA df8d00d3a9088d1ad06089
380 DATA a0804c6221a20886300ae2
390 DATA 48a0009002a0022010204e
400 DATA 66c630f008a20c200c209e
410 DATA 4c3f2060a008ad1fd0f0dd
420 DATA d7ad00d330f6a9008d1aaa
430 DATA d0aae8301ead00d310f8e2
440 DATA a9088d1ad0ec0a20b00bdb
450 DATA 6688ec0b20263088d0d462
460 DATA 60248830cfa03c4c622168

470 DATA 30cfa08c4c6208209b0836
480 DATA a900a01085309848a004c8
490 DATA 20100768a8c630d0f38850
500 DATA d0f0a432a9008532a202ea
510 DATA 200c079848b13248186 5a5
520 DATA 31853168203b0768a8c82e
530 DATA f005c4004cce07e633c4e5
540 DATA 34d009a535c533d0064ce6
550 DATA e107eaeaea4cb407a53169
560 DATA a201200c07203b07a00142
570 DATA 4c62084c3607209b08a0e4
580 DATA 00a9ff8549a200ad1fd098
590 DATA f0edad00d330f6a9008d51
600 DATA 1ad0e830ecad00d310f8c7
610 DATA a9088d1ad0ec0a0790d450
620 DATA c8d0daa432a90085329890
630 DATA 4820580768a8a530ae03ed
```


```
640 DATA 07d02691321865318531111
650 DATA c8d002e633c434d0e2a513
660 DATA 33c535d0dc205807a0010c
670 DATA a530c531f002a08f4c62a6
680 DATA 08d132f0d3a091aec10821
690 DATA 9aa9408d0ed4a9228d006b
700 DATA d4a9388d02d3a9008d00b8
710 DATA d3a93c8d02d35860863444
720 DATA 8435a2fa8632a20e8633ba
730 DATA ae0307d006aad0034c90a1
740 DATA 074cf307a9008d0ed4858b
750 DATA 318d00d4a9088d1fd0a9f3
760 DATA 388d02d3a9608d00d3a99f
```

```
1710 DATA 207a61776172746f7363e6
1720 DATA 206479736b75206e9b50af
1730 DATA 6c696b2077656a73636994
1740 DATA 6f7779203a11009b506cb5
1750 DATA 696b2077796a7363696fb1
1760 DATA 7779203a18009b4b6f70d8
1770 DATA 696f77616e6965207a61bf
1780 DATA 6b6f6e637a6f6e65209be1
1790 DATA 06019b7d20202020202020c0
1800 DATA 4b6f706961726b61204456
1810 DATA 59534b3c3e545552424f53
1820 DATA 544150459b9b2070726f24
1830 DATA 6772616d206f7072616300
1840 DATA 6f77616e792070727a650f
1850 DATA 7a9b202020202020202024
1860 DATA 202020202020576f6a34
1870 DATA 63696563686120 5a6162ce
1880 DATA 6f6c6f746e65676f9b9b6b
1890 DATA 20202020207765772736a36
1900 DATA 6120322e36202050494167
1910 DATA 53544f57202031393838ce
1920 DATA 9b9b555741474120212 1db
1930 DATA 21204162792075 6e696b0f
1940 DATA 6e6163206b6f6c697a6af4
1950 DATA 69207a65207374612d9b8c
1960 DATA 636a61206479736b6f777b
1970 DATA 206964656e747966696b62
1980 DATA 61746f72656d206d61673f
1990 DATA 6e65746f666f2d9b6e7575
2000 DATA 20545552424f206a657383
2010 DATA 7420543a2061206e696582
2020 DATA 20443a206a616b2077202d
2030 DATA 7379732d9b74656d696568
2040 DATA 204b2e532e4f2e205455c6
2050 DATA 52424f9b48a220a90b9da1
2060 DATA 4203a9009d48039d490360
2070 DATA 692056e460a210a9079d81
2080 DATA 4203a9009d48039d490340
2090 DATA 2056e460a001b1868d4dac
2100 DATA 27204025a903a2109d4295
2110 DATA 03a94c9d4403a9279d4523
2120 DATA 03a9069d4a03a9009d4b50
2130 DATA 032056e4301aa907a21059
2140 DATA 9d4203a9009d48039d49b2
2150 DATA 032056e4300620b0f24c53
2160 DATA 2827c088d0034cbe244c37
2170 DATA 2b25444d3a2a2e2a00e0b4
2180 DATA 02e102ae246b
```

```
1840 DATA 0c4cee0ca920a00e2025da
1850 DATA 0ca951a00d20250ca9ff86
1860 DATA 8dfc0220fdf2295fa00850
1870 DATA d99f0ef0058810f830b43f
1880 DATA 980aa8b9a90e48b9a80eb0
1890 DATA 48603800cbe1f3e5f4eff7
1900 DATA f7f9a0f3f9f3f4e5eda0cc
1910 DATA eff0e5f2e1e3f9eaeef910
1920 DATA a0d7dae1e2a0b2d4b0b6b0
1930 DATA 9b9b20202020202020205016
1940 DATA 727a657277793a20202063
1950 DATA cd009b9b7f7fa0cda0c536
1960 DATA a0cea0d5a09b9bcc2d4c34
1970 DATA 61646f77616e696520700c
1980 DATA 6c696b7520d22d4c6164f1
1990 DATA 6f77616e696520692073 90
2000 DATA 746172749bd32d5374610e
2010 DATA 72742070726f6772616d0c
2020 DATA 7520201b9d2d5a696d6e44
2030 DATA 792073746172749bcb2d9e
2040 DATA 4b726f746b696520707279
2050 DATA 7a6572777920c42d446c7b
2060 DATA 756769652070727a657278
2070 DATA 77799bc52d5572756368fc
2080 DATA 6f6d69656e696520202042
2090 DATA 20c82d5a6174727a796d58
2100 DATA 616e69659b2020 73696c18
2110 DATA 6e696b612020202020207b
2120 DATA 20202020207369 6c6e693a

2130 DATA 6b619bc32d506f77726fa3
2140 DATA 742064 6f2063617274724b
2150 DATA 69646765276 19b06004451
2160 DATA 4c5547494507004b524fba
2170 DATA 544b494515009b7ffd5a6d
2180 DATA 6c7920666f726d6174201b
2190 DATA 706c696b75202114009b30
2200 DATA 7ffd426c61642074726186
2210 DATA 6e736d69736a693a2021fe
2220 DATA 009b20202020202020a0cec7
2230 DATA e1e3e9f3eee9eaa0e4ef3b
2240 DATA f7efeceef9a0ebece1f7a3
2250 DATA e9f3faa01c009b20202030
2260 DATA 202020202050 6f64615abe
1670 DATA 009bfdfdfd2020202020a4
1680 DATA 20426c61642074 72616e0c
1690 DATA 736d69736a693a202d0022
1700 DATA 9b2a6e20706f64616a65e8
```

```
770 DATA 3c8d02d378bae8e88ec18e
780 DATA 086000a90c850ba9bf8528
790 DATA 0aa91a850f8de802a9fca5
800 DATA 850e8de702a9808d280a96
810 DATA a244a90aa01b4c86e4a949
820 DATA 31a00e20250ca210a90ce0
830 DATA 9d420320c8094cb00c18d3
840 DATA 900138664820 7e0ca20399
850 DATA a9009de002ca10faa21047
860 DATA a9039d4203a9209d440382
870 DATA a90b9d4503a9049d4a03b2
880 DATA ad280a9d4b0320c809208d
890 DATA 980990b5209809b0fba988
900 DATA 079d4203a90a9d4503a9b2
910 DATA 199d4403a9029d4803a9eb
920 DATA 009d490320c809386e1681
930 DATA 0aad170a9d4403ad180a0c
940 DATA 9d450338ad190aed170a07
950 DATA 9d4803ad1a0aed180a9d6c
960 DATA 4903fe4803d003fe49031e
970 DATA 20c809ade202d005ade305
980 DATA 02f00b20130aa9008de257
990 DATA 028de3024c3109186e16ed
1000 DATA 0aa210a9079d4203a90aee
1010 DATA 9d4503a9179d4403a90222
1020 DATA 9d4803a9009d490320c884
1030 DATA 09ad170a2d180ac9fff062
1040 DATA 0218603860205 6e4983096
1050 DATA 016048a90c9d420320564c
1060 DATA e4682c160a3024c988d059

1670 DATA b1cb9d4803c8b1cb9d4978
1680 DATA 03a90b9d42034c56e4851c
1690 DATA d484d520aad920e6d8a973
1700 DATA 0085cdf00520b0f2e6cd2f
1710 DATA a4cdb9800510f4498020cb
1720 DATA b0f24c770ca92020b0f2c7
1730 DATA e6cda906c5cdd0f360a987
1740 DATA 81a00e20250ca200a90557
1750 DATA 9d4203a90b9d4503a9209b
1760 DATA 9d4403a90e9d4803a900c7
1770 DATA 9d49032056e41004c08967
1780 DATA d016bc480388f01060a9e5
1790 DATA 5ea00e20250ca9ff8dfc73
1800 DATA 0220fdf2a2ff9aa90185ee
1810 DATA 09a900852ba90c852a20d4
1820 DATA 8eefa917a00d20250cadbc
1830 DATA 280a100aa928a00e2025cc
```

Losowy start generatora RND

Generator **RND** w ZX Spectrum posiada pewną niedogodność: po włączeniu komputera do sieci wartość wyjściowa generatora jest zerowana, czyli cały cykl generacji jest zawsze jedną i tą samą sekwencją:

```
 10 REM Krzysztof Pozniak ©1988
    losowy start generatora
    liczb pseudolosowych
 20 RANDOMIZE 1: REM ustawienie
                 stalego startu
                 generatora RND
 30 RANDOMIZE 256*PEEK 23672+PE
EK 23673: REM losowe ustawienie
      genetatora RND
 40 PRINT RND
 50 GO TO 20
```

Basic przewiduje możliwość ustawienia generatora za pomocą rozkazu **RANDOMIZE**, ale trzeba ją ustalić samodzielnie. Jeśli zależy nam na całkowicie losowym (i niezależnym od użytkownika) ciągu liczb losowych, to w standardzie **Basic'a** tego nie zrobimy.

Musimy skorzystać z komórek pamięci zliczających ramki (licznika ramek). Licznik zwiększa swoją wartość o 1 co 1/50 sekundy. Z dużą dokładnością możemy założyć, że ustalenie startu generatora **RND** na podstawie dwóch młodszych bajtów licznika zapewni całkowicie niepowtarzalny ciąg liczb.

```
pierwsze liczby generatora RND:

1 liczba = .0011291504
2 liczba = .08581543
3 liczba = 0.43719482
4 liczba = 0.79025269
5 liczba = 0.2691803
6 liczba = 0.18934631
7 liczba = 0.20186904
8 liczba = 0.14257813
9 liczba = 0.69433594
10liczba = .075531006
11liczba = 0.6658783
12liczba = 0.94125366
13liczba = 0.59408569
14liczba = 0.55688477
15liczba = 0.76686096
```

Krzysztof POŹNIAK

PROGRAM 64

ATARI

Loteryjka

W związku z rozpoczęciem publikowania na łamach „IKS-a" artykułów dotyczących języka „C", chciałbym przedstawić program napisany w Basic-u oraz jego implementację w „C". Początkujący zwolennicy języka „C" mogą w ten sposób obejrzeć i porównać efekty pracy obu języków. Aby nie rozbudowywać programu napisanego w „C", wykorzystano w nim funkcję czydalej (J) z drugiego numeru zeszytów programów „IKS-a". Zbiór wskazówek tego programu dla linkera składa się z następujących nazw:

```
PG 500 REM ************************
YA 502 REM *                      *
VQ 504 REM *      LOTERYJKA       *
TE 506 REM *     *  ATARI  *      *
YS 508 REM *                      *
PI 510 REM ************************
JA 511 POKE 752,0
LL 512 PRINT CHR$(125):J=0:I=0
MS 514 DIM J$(255),I$(15):J$=""
TJ 515 DIM T$(1):OPEN #1,4,0,"K:"
AL 516 PRINT "ILU GRACZY ";:INPUT N
UD 517 PRINT
GY 518 FOR K=1 TO N:? "GRACZ ";K
VX 520 ? "IMIE ";:INPUT I$
TN 521 PRINT
BQ 522 J$(LEN(J$)+1)=I$:I=LEN(I$)+1
PP 523 FOR DL=I+(K*10) TO 9+(K*10)
ND 524 J$(LEN(J$)+1)=CHR$(32)
KU 525 NEXT DL
OR 526 I=LEN(I$)+1
IU 527 J$(LEN(J$)+1)=CHR$(44):J=J+I
HR 528 NEXT K
RY 536 FOR K=1 TO N:SOUND 0,99,10,5
DO 537 KS=(K-1)*10
MD 538 FOR KK=1+KS TO 9+KS
XU 539 PRINT J$(KK,KK);
MQ 540 NEXT KK
FN 542 SU=0
PL 543 FOR G=1 TO 5:SOUND 0,0,0,0
FF 544 W=1+INT(RND(0)*8):SU=SU+W
DV 546 PRINT CHR$(32);W;:NEXT G
CV 547 PRINT "  ";SU
JM 548 IF K=J THEN 550
HY 549 NEXT K
EE 550 ? :? "GRAMY DALEJ ?"
JW 551 POKE 752,1
KL 552 ? :? "          ";CHR$(212);"/
   ";CHR$(206);:GET #1,T
UE 554 IF T<>78 THEN ? :? :GOTO 536
OW 556 END
```

LOTE c
AIO
PRINTF
GRAPHICS
DBC.OBJ
CZYDALEJ

Pierwszy wiersz, to oczywiście nazwa samego programu, którą można zmienić. Pozostałe związane są z biblioteką. Całość zostanie skomplikowana pod warunkiem uczynienia tego samego z funkcją czydalej (J).

Zadanie programu jest proste. Po wprowadzeniu ilości graczy, a następnie ich imion, komputer losuje pięć liczb w zakresie od 1 do 7 oraz oblicza ich sumę. Wygrywa ten, kto zdobędzie więcej punktów. Można do tej zabawy podejść od innej strony. Mechanizm jej przypomina trochę grę w kości. Będzie wygrywał ten, kto uzyska więcej par, trójek itd. o większej liczbie oczek.

J. J.

```
/****************************/
/*                          */
/*        LOTERYJKA         */
/*          ATARI           */
/*                          */
/****************************/
main()$(
 int c, ilu, dl, i, j, k, g, su, t;
 char w, tab[100], imie[10];
  printf("\fIlu graczy (max 9)?");
  ilu=getchar();
  c=getchar();
for(dl=0; dl<=ilu-49; dl++)$(
  printf("\nGracz %d ", dl+1);
  printf("\nImie  ");
  gets(imie);
  t=strlen(imie);
 for(i=0; i<t; i++)$(
 j=i+dl*10;
 tab[j]=imie[i];
$)
 for(i=t; i<=9; i++)$(
 j=i+dl*10;
 tab[j]=32; $)
$)
do$(
  printf("\n");
  for(k=0; k<=ilu-49; k++)
$(sound(0, 99, 10, 5);
 printf("\n");
 for(i=10*k; i<=10*k+9; i++)
 printf("%c", tab[i]);
 sound(0, 0, 0, 0);
  su=0;
 for(g=1; g<=5; g++)$(
  w=1+rnd(8);
  su=su+w;
 printf(" %d", w);
$)printf("   %d", su);
$)$)
 while(czydalej());
$)
```

Programy narzędziowe na CPC — AMSTRAD 6128

W niniejszym opracowaniu przedstawiamy niezbędne w pracy z komputerem CPC—AMSTRAD 6128 programy narzędziowe: DYSKMAPA, SEKTORED, KATALOG, DYSKCOPY i binarny program PROGBIN wykorzystywany przez każdy z wymienionych powyżej programów.

Program PROGBIN jest niezbędny przy operacjach wykonywanych na zbiorach. Warto wprowadzić i zachować na dyskietce jeszcze jeden program: JOB (wydruk 6), który ułatwi nam ściąganie prezentowanych programów do pamięci z jednoczesnym ich uruchomieniem.

Najpierw proponujemy wprowadzić i uruchomić komendą RUN program PROGBIN (wydruk 1). Jeśli wprowadzając program nie popełniłeś błędu pojawią się komunikaty:

OK — WSZYSTKIE DANE PRAWIDŁOWE

WŁÓŻ DYSKIETKĘ DO ZAPISU PROGRAMU BINARNEGO — WCIŚNIJ DOWOLNY KLAWISZ.

Jeśli postąpiłeś zgodnie z komunikatem to możesz uważać, że masz poza sobą wprowadzenie (i zachowanie na dyskietce) najbardziej żmudnego z wymienionych programów.

Programy narzędziowe DYSKMAPA, SEKTORED, KATALOG, DYSKCOPY.

Opublikujemy w kolejnych numerach „IKS-a"

```
10 RESTORE 200:A=&6000
20 FOR K=1 TO 256:REM 256 - liczba linii DATA zawierajacych po 20 bajtow kodu
30 SUMK=0
40 FOR I=1 TO 20
50 READ BAJT
60 SUMK=SUMK+BAJT
70 POKE A+I-1,BAJT
80 NEXT I
90 READ SUMA
100 IF SUMA<>SUMK THEN PRINT "BLEDNE DANE W linii ";K:STOP
110 A=A+20:NEXT K
120 CLS:LOCATE 1,1:PRINT "  OK  -  WSZYSTKIE dane PRAWIDLOWE"
130 LOCATE 1,12
140 PRINT "WLOZ DYSKIETKE DO ZAPISU PROGRAMU BINARNEGO - WCISNIJ DOWOLNY KLAWISZ"
150 IF INKEY$="" THEN 150
160 SAVE "progbin.bin",b,&6000,5120
200 DATA 195,31,16,195,159,16,195,93,16,195,177,16,0,193,198,194,199,195,200,196,2679
210 DATA 201,197,193,198,194,199,195,200,196,201,197,17,255,1,205,72,16,30,0,221,2980
220 DATA 86,0,223,90,16,33,13,16,229,223,81,16,223,48,11,58,81,190,119,35,1793
230 DATA 125,254,31,32,239,175,50,12,16,17,0,10,123,223,84,16,122,223,87,16,1855
240 DATA 201,100,197,7,114,202,7,3,198,7,99,199,7,221,86,0,221,94,2,221,2194
250 DATA 126,4,245,213,71,17,22,0,221,229,225,25,209,221,33,196,16,126,221,114,2534
260 DATA 0,221,119,2,221,35,221,35,221,35,221,35,43,43,16,237,241,254,9,40,2249
270 DATA 4,62,1,24,2,62,193,213,223,236,16,209,33,196,16,223,239,16,201,221,2390
280 DATA 86,0,221,78,2,221,110,4,221,102,5,30,0,223,242,16,201,221,86,0,2069
290 DATA 221,94,2,221,78,4,221,110,6,221,102,7,223,245,16,201,4,0,193,2,2171
300 DATA 4,0,198,2,4,0,194,2,4,0,199,2,4,0,195,2,4,0,200,2,1016
310 DATA 4,0,196,2,4,0,193,2,4,0,198,2,0,0,0,0,129,197,7,82,1020
320 DATA 198,7,102,198,7,78,198,7,0,0,0,0,0,0,0,0,229,229,229,229,1711
330 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
340 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
350 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
360 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
370 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
380 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
390 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
400 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
410 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
420 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
430 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
440 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
450 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
460 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
470 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
480 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
490 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
500 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
510 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
520 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
530 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
540 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
550 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
560 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
570 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
580 DATA 229,229,229,229,229,229,229,229,[illegible],116,0,221,[illegible],1,6,16,197,229,6,16,2957
590 DATA 126,205,61,19,62,72,205,90,197,35,[illegible],244,[illegible],205,90,187,16,251,[illegible],2265
600 DATA 6,16,126,254,32,40,2,62,46,205,90,157,35,[illegible],6,243,62,[illegible],205,90,187,1925
610 DATA 62,10,205,90,187,193,16,254,201,245,15,15,15,15,205,90,19,241,[illegible],2253
620 DATA 198,144,39,206,64,39,195,50,187,229,229,229,229,229,229,229,229,229,229,229,3681
630 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
640 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
650 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
660 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
670 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
680 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
690 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
700 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
710 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
720 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
730 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
740 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
750 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
760 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
770 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
780 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
790 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
800 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
810 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
820 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
830 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
840 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
850 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
860 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
870 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
880 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
890 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
900 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
910 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
920 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
930 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
940 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
950 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
960 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
970 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
980 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
990 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1000 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1010 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1020 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1030 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1040 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1050 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1060 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1070 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1080 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1090 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1100 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1110 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1120 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1130 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,231,4582
1140 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1150 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1160 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1170 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1180 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1190 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1200 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1210 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1220 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1230 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1240 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1250 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1260 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1270 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1280 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1290 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1300 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1310 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
```

```
1320 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1330 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1340 DATA 229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,229,4580
1350 DATA 229,229,229,229,237,140,243,59,32,206,196,53,115,173,29,21,27,156,251,26,2990
1360 DATA 232,149,163,36,118,179,89,36,225,224,194,61,188,190,58,18,209,46,252,190,2677
1370 DATA 115,118,164,66,17,71,8,215,17,236,159,230,36,71,212,4,30,180,204,96,2249
1380 DATA 246,11,109,178,22,97,203,198,16,115,85,73,47,204,159,157,148,216,103,21,2408
1390 DATA 126,68,194,95,149,22,226,191,21,82,176,176,178,118,128,179,172,247,99,199,2776
1400 DATA 208,72,175,149,167,229,196,149,64,43,158,247,116,88,46,142,133,83,134,32,2631
1410 DATA 193,102,59,188,24,252,209,52,47,172,211,230,74,195,100,82,3,35,214,124,2566
1420 DATA 177,157,161,148,121,218,6,145,210,85,82,216,11,188,22,39,97,113,58,76,2330

1430 DATA 107,180,41,79,164,88,230,116,252,159,248,113,68,99,93,26,17,243,216,65,2604
1440 DATA 235,145,86,64,91,105,254,85,162,137,188,177,68,73,169,128,110,233,139,165,2814
1450 DATA 112,243,140,40,249,217,80,7,85,167,190,75,69,175,170,210,162,40,183,153,2767
1460 DATA 173,98,3,48,82,173,122,35,47,102,182,68,210,238,29,53,242,14,204,82,2205
1470 DATA 62,156,167,169,31,137,140,37,214,215,8,68,119,135,152,78,182,130,200,198,2598
1480 DATA 74,174,166,37,243,200,240,28,147,105,109,46,200,99,168,143,216,13,234,198,2840
1490 DATA 111,116,38,67,126,188,210,235,198,236,122,210,233,172,29,41,108,115,252,173,2980

1500 DATA 100,180,214,34,75,25,79,126,45,64,21,59,161,102,118,249,90,224,13,79,2058
1510 DATA 57,220,192,160,65,80,1,150,0,204,212,19,206,46,107,34,120,174,52,196,2295
1520 DATA 18,170,32,94,59,15,19,61,13,139,165,182,148,135,64,178,235,105,255,223,2310
1530 DATA 79,161,86,12,163,197,178,220,5,128,58,133,182,135,96,134,238,138,122,103,2568
1540 DATA 59,140,195,18,25,27,37,70,130,61,166,190,10,106,21,206,167,20,55,78,1781
1550 DATA 49,53,201,164,167,49,57,29,217,232,97,91,156,188,156,168,204,241,88,227,2834
1560 DATA 119,207,253,93,59,15,122,41,168,145,43,247,5,133,157,34,158,200,4,24,2227

1570 DATA 22,100,203,88,82,3,214,71,201,45,224,106,95,198,55,241,63,135,231,112,2501
1580 DATA 88,193,84,57,25,17,174,163,140,110,47,104,139,111,198,197,161,236,244,240,2728
1590 DATA 79,186,105,183,2,113,243,107,122,100,129,176,154,161,217,6,126,25,97,246,2577
1600 DATA 4,173,165,31,190,137,247,11,215,173,247,234,69,67,48,24,211,212,60,59,2577
1610 DATA 71,179,169,251,165,174,246,103,42,100,224,146,236,232,209,11,209,91,254,41,3153
1620 DATA 213,175,124,111,208,9,227,184,253,234,40,2,234,226,126,100,200,78,234,7,2985
1630 DATA 219,65,198,79,31,119,46,89,98,254,70,246,109,233,11,61,226,76,165,78,2473
1640 DATA 44,38,15,183,206,189,148,143,12,117,152,246,164,100,51,220,89,105,165,243,2630

1650 DATA 36,85,141,175,9,178,47,44,22,141,41,169,39,45,146,206,148,173,109,11,1965
1660 DATA 70,230,113,172,226,107,142,2,143,215,6,180,112,35,221,31,219,178,183,104,2699
1670 DATA 53,248,131,107,48,139,159,175,15,73,168,2,210,133,226,55,159,251,118,16,2486
1680 DATA 150,61,206,114,47,117,119,37,187,180,120,120,13,11,237,50,102,131,208,75,2285
1690 DATA 129,27,146,147,252,254,141,251,55,21,181,152,112,123,18,146,19,3,161,242,2580
1700 DATA 143,122,189,216,220,232,76,30,81,57,219,199,148,222,122,17,149,186,243,18,2889
1710 DATA 199,215,238,201,9,3,216,185,253,60,162,165,169,15,119,239,61,7,19,158,2693
1720 DATA 22,175,254,67,248,211,2,104,254,207,168,227,114,62,115,139,19,176,8,83,2655

1730 DATA 73,68,49,84,142,196,169,53,186,251,44,250,40,27,51,151,215,179,222,113,2563
1740 DATA 116,189,243,214,140,27,205,253,220,96,172,103,196,126,52,148,237,146,147,187,3217
1750 DATA 6,255,13,175,57,65,116,232,39,18,218,67,69,9,93,203,14,208,223,30,2130
1760 DATA 55,49,217,105,167,179,48,180,170,58,109,17,90,32,216,44,79,30,100,101,2046
1770 DATA 230,178,185,216,1,195,254,109,233,223,208,136,73,224,236,150,30,22,237,137,3276
1780 DATA 10,36,27,10,205,147,53,101,204,22,180,54,42,118,123,210,106,20,63,246,1997
1790 DATA 38,4,132,160,53,104,101,161,244,107,129,213,123,108,209,254,185,221,48,47,2661
1800 DATA 225,254,68,133,205,230,174,133,26,21,16,160,65,252,26,88,30,104,5,222,2447

1810 DATA 119,233,159,16,121,31,202,202,180,81,250,129,198,101,85,28,98,243,95,91,2682
1820 DATA 32,153,161,77,14,68,95,39,86,12,114,147,151,183,139,176,204,202,32,24,2109
1830 DATA 7,10,182,233,184,164,16,9,197,4,129,39,202,194,45,139,41,216,146,165,2322
1840 DATA 74,187,192,76,89,101,1,23,127,29,245,24,83,159,19,94,194,231,24,107,2079
1850 DATA 91,29,106,39,13,54,109,81,240,73,146,9,130,250,202,41,96,16,119,168,2012
1860 DATA 127,113,252,86,214,111,131,61,17,77,165,236,240,72,206,46,30,74,39,73,2370
1870 DATA 80,132,189,86,183,212,43,243,209,10,33,47,200,131,213,148,92,152,12,4,2424
1880 DATA 159,114,132,52,206,205,108,185,169,54,44,169,2,98,56,205,226,203,92,211,2690
1890 DATA 121,12,148,107,74,96,204,174,232,224,212,163,23,27,70,101,154,227,102,20,2491

1900 DATA 45,200,31,160,104,28,88,75,92,189,211,218,236,174,127,88,48,205,141,142,2602
1910 DATA 105,51,209,80,83,125,119,100,111,131,146,62,189,130,14,212,26,149,220,59,2321
1920 DATA 224,3,123,216,191,38,186,30,50,238,200,226,242,215,143,15,245,159,159,18,2921
1930 DATA 158,122,157,136,3,134,187,136,174,48,161,25,241,33,19,70,209,146,240,223,2624
1940 DATA 12,14,68,128,161,157,234,129,114,157,110,68,196,60,199,26,46,0,210,148,2237
1950 DATA 78,122,50,186,100,13,212,198,34,114,88,179,232,245,134,82,192,57,156,151,2625
1960 DATA 163,122,30,190,225,20,118,156,153,209,42,250,139,143,127,17,205,236,148,11,2704

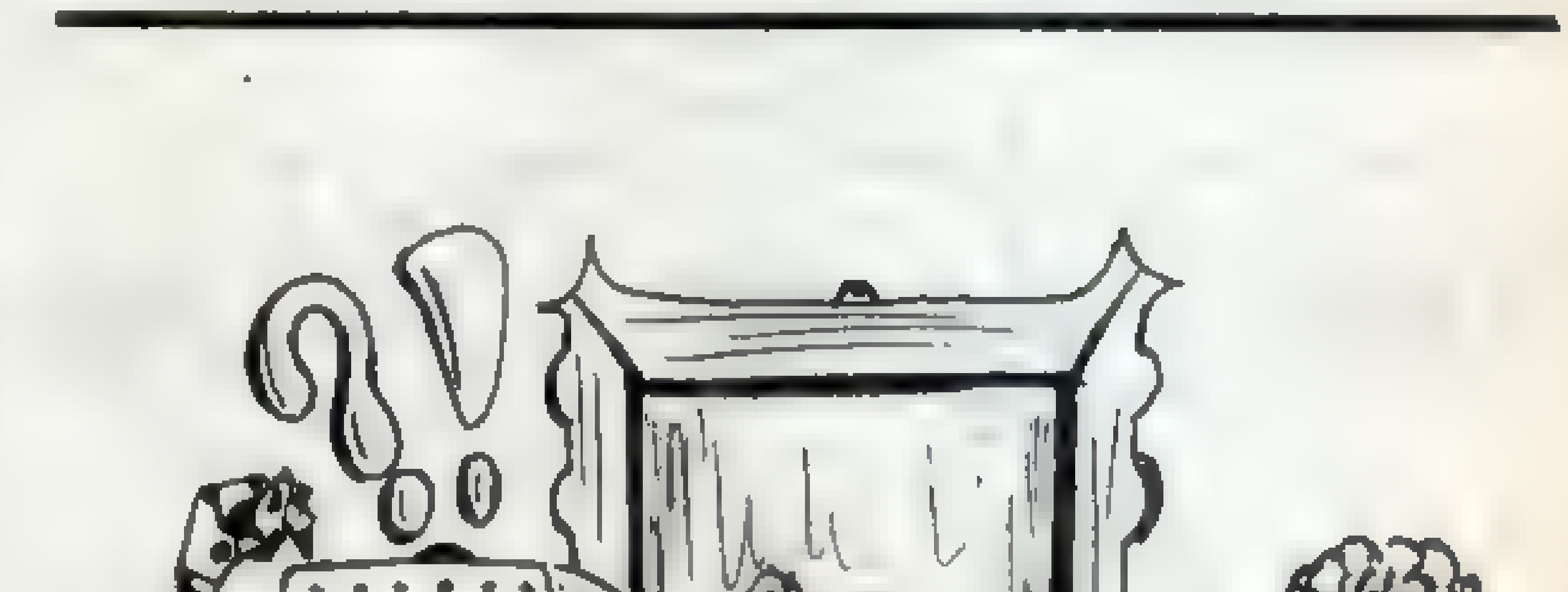
1970 DATA 33,143,63,165,0,75,29,36,135,150,60,85,253,63,248,13,252,244,55,237,2379
1980 DATA 49,208,129,54,129,6,88,248,22,4,55,58,164,88,27,95,31,43,27,235,1755
1990 DATA 92,84,49,1,194,44,144,93,155,47,120,39,220,92,242,104,6,85,218,202,2221
2000 DATA 221,24,64,237,194,143,98,170,161,191,88,146,74,228,191,233,179,176,143,12,2843
2010 DATA 77,61,236,5,199,201,54,29,30,4,11,59,146,21,178,247,67,198,97,110,2030
2020 DATA 249,217,220,68,110,197,130,102,201,141,234,206,19,122,99,187,184,203,117,86,3102

2030 DATA 173,151,245,246,76,200,105,196,144,130,22,153,32,116,44,240,209,13,44,83,2622
2040 DATA 15,150,246,153,189,178,20,244,74,28,181,223,232,187,168,162,242,181,138,83,3055
2050 DATA 188,196,77,114,104,92,210,184,119,187,248,248,83,193,92,251,6,47,47,131,2727
2060 DATA 218,51,80,24,179,115,50,184,45,[illegible],30,226,88,4,216,166,171,95,175,107,2244
2070 DATA 177,45,199,109,248,4,197,44,[illegible],171,70,219,146,46,53,152,128,117,144,164,2428
2080 DATA [illegible],198,155,242,97,42,2290

2090 DATA [illegible],174,219,84,110,3059
2100 DATA [illegible],184,155,146,253,227,58,2542
2110 DATA [illegible],179,54,143,79,2710
2120 DATA 4,200,56,22,20,17,119,154,144,75,106,225,[illegible],85,127,19,2,253,112,51,1920
2130 DATA [illegible],245,178,81,128,2659

2140 DATA 219,55,5,251,17,42,29,59,230,178,216,109,49,127,90,18,20,140,30,148,2032
2150 DATA 129,128,119,64,250,106,150,135,42,21,92,154,95,201,247,242,120,80,238,6,2619
2160 DATA 5,69,224,179,44,242,91,88,242,80,164,166,82,127,91,81,124,63,240,210,2612
2170 DATA 15,116,136,195,239,131,230,39,41,92,193,169,228,86,169,191,135,127,172,199,2904
2180 DATA 197,79,46,226,113,127,219,229,168,157,67,224,132,95,130,186,125,34,236,214,3004
2190 DATA 27,87,27,115,112,2,172,155,23,220,253,48,15,157,253,179,73,252,33,216,2419
2200 DATA 187,149,47,227,73,148,8,24,91,247,211,54,116,7,222,64,196,248,182,157,2668
2210 DATA 189,26,105,108,41,105,117,99,198,193,225,10,133,38,211,21,188,232,155,98,2490
2220 DATA 202,103,189,70,252,192,176,143,253,134,94,221,91,140,166,60,83,228,141,52,2990
2230 DATA 51,239,212,207,227,243,134,19,192,113,208,17,232,108,177,132,237,153,25,238,3164
2240 DATA 235,117,230,123,156,241,190,252,120,97,179,105,239,80,51,52,0,0,0,0,2467
2250 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2260 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2270 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2280 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2290 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2300 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2310 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2320 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2330 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2340 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2350 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2360 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2380 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2390 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2400 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2410 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2420 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2430 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2440 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2450 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2460 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2470 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2480 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2490 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2500 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2510 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2520 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2530 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2540 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2550 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2580 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2590 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2600 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2610 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2620 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2630 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2640 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2650 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2660 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2670 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2680 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2690 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2700 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2710 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2720 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2730 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2740 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2750 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Jacek Szaniawski



Nie martw się kochanie! Za życia wyglądałeś jeszcze gorzej.

Lilavati (10)

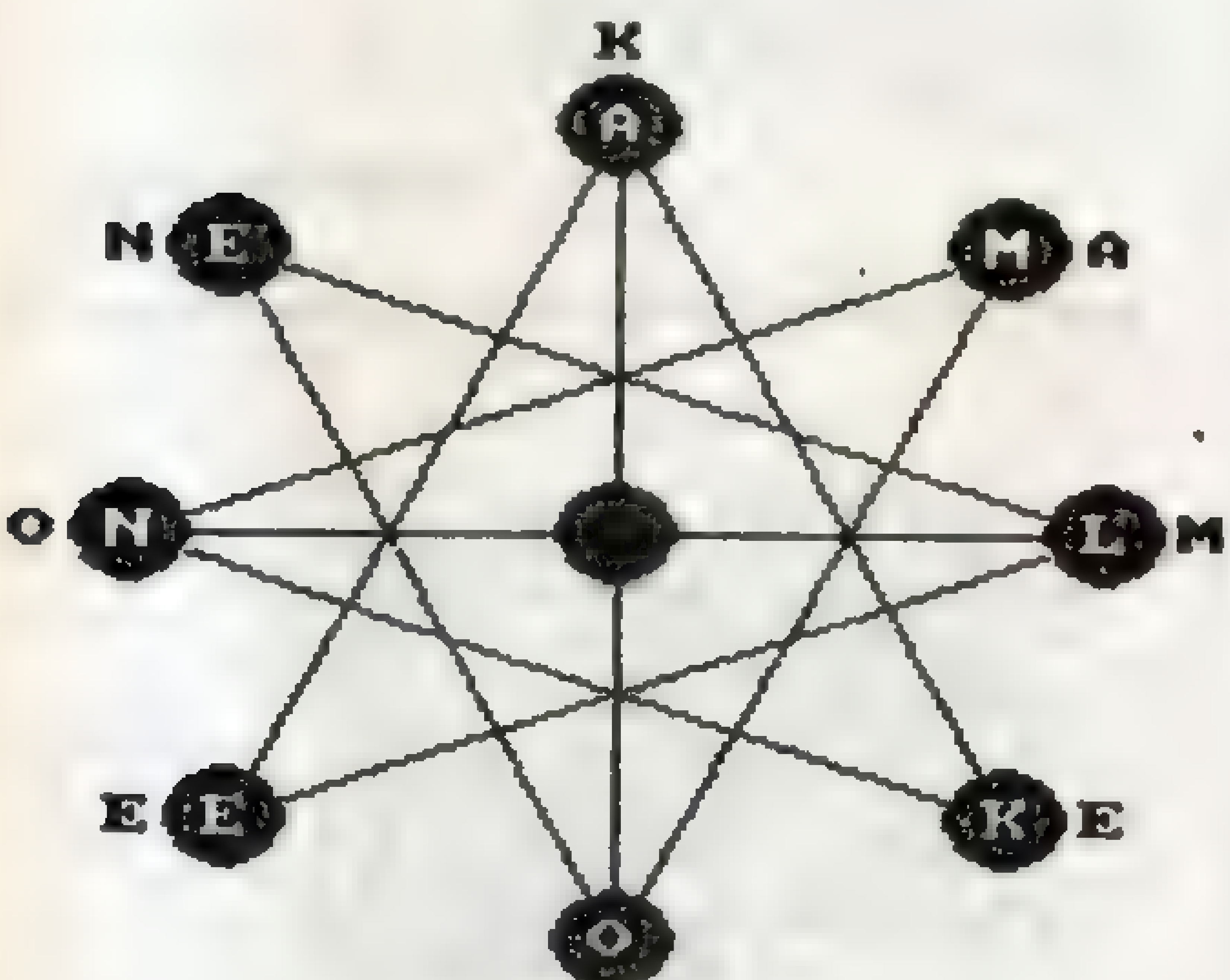
KAMELEON

Danuta KWASIŻUR, Mieczysław SKONIECZNY

W kolejnym, niejako jubileuszowym odcinku cyklu poświęconego rozwiązywaniu anegdot matematycznych przy użyciu mikrokomputera, prezentujemy prostą grę planszową przeznaczoną dla jednego gracza o nazwie „Kameleon". Zasady tej gry są bardzo proste, a prezentuje je program w początkowej fazie realizacji na ekranie monitora. Kopia tego obrazu ma następującą postać

```
Na osmiu wierzcholkach gwiazdy osmio-
ramiennej i w jej srodku rozmieszczone
sa czarne krazki. Ponad krazkami stoja
litery, ktore w kierunku obiegu wskazow-
ki zegara skladaja poczawszy od wierz-
cholka gornego slowo KAMELEON.Oprocz
tego jest 8 luznych ruchomych krazkow,
na ktorych wypisane sa te same litery;
K,A,M,E,L,E,O,N. Krazki ruchome ustawio-
ne sa w przypadkowej kolejnosci na osmiu
zewnetrznych krazkach gwiazdy. Krazek
srodkowy pozostaje nie zajety. Gra pole-
ga na takim kolejnym przesuwaniu krazkow
ruchomych, aby wypisane na nich litery
znalazly sie obok takich samych liter
zapisanych zewnatrz gwiazdy, a wiec
utworzyly wyraz KAMELEON.
STEROWANIE RUCHEM KRAZKOW:
Kolejne naciskanie SPACJI powoduje migo-
tanie liter na kolejnych krazkach.
Nacisniecie klawisza ENTER spowoduje
przesuniecie krazka z migoczaca litera
na pusty krazek /jezeli takie przesunie-
cie jest mozliwe/
NACISNIJ DOWOLNY KLAWISZ
```

Po zapoznaniu się z tekstem informacyjnym rozpoczyna się właściwy etap gry. Widok planszy i początkowego rozstawienia krążków jest widoczny na poniższym rysunku:



Wersja źródłowa programu opracowanego w języku BASIC jest następująca

```
10 '
20 '            Ekran informacyjny
30 '
40 MODE 1:INK 0,14:INK 1,0:INK 2,6,26:INK 3,26:BORDER 14:PAPER 0:CLS
50 PEN 1:PRINT "Na osmiu wierzcholkach gwiazdy osmio-   ramiennej i w jej srodku
 rozmieszczone  sa czarne krazki. Ponad krazkami sto
ja  litery, ktore w kierunku obiegu wskazow-ki zegara skladaja poczawszy od wier
z-  cholka gornego slowo KAMELEON.Oprocz"
60 PRINT "tego jest 8 luznych ruchomych krazkow,  na ktorych wypisane sa te same
 litery:  K,A,M,E,L,E,O,N. Krazki ruchome ustawio-ne
 sa w przypadkowej kolejnosci na osmiuzewnetrznych krazkach gwiazdy. Krazek - sr
odkowy pozostaje nie zajety. Gra pole-";
70 PRINT "ga na takim kolejnym przesuwaniu krazkowruchomych, aby wypisane na nic
h litery  znalazly sie obok takich samych liter   za
pisanych zewnatrz gwiazdy, a wiec     utworzyly wyraz KAMELEON.
80 PEN 3:PRINT "STEROWANIE RUCHEM KRAZKOW:"
90 PEN 1:PRINT "Kolejne naciskanie SPACJI powoduje migo-tanie liter na kolejnych
 krazkach.      Nacisniecie klawisza ENTER spowoduje
   przesuniecie krazka z migoczaca litera  na pusty krazek /jezeli takie przesu
nie-cie jest mozliwe/."
100 PEN 3:PRINT "NACISNIJ DOWOLNY KLAWISZ"
110 CLEAR INPUT
120 a$=INKEY$
130 IF a$="" THEN GOTO 120
140 '
150 '          Segment sterujacy
160 '
170 GOSUB 920
180 GOSUB 640
190 PAPER 1:PEN 0
200 FOR k=2 TO 9
210 LOCATE wsp(k,1)+1,wsp(k,2)+1:PRINT LIT$(tabl(k))
220 NEXT k
230 PAPER 1:l=1
240 K=1
250 IF TABL(K)=1 THEN k=k+1:GOTO 250
260 PEN 2:LOCATE wsp(k,1)+1,wsp(k,2)+1:PRINT lit$(tabl(k))
270 CLEAR INPUT
280 a$=INKEY$
290 IF a$="" THEN GOTO 280
300 IF INKEY(6)=0 THEN GOTO 360
310 IF a$=" " THEN GOTO 330
320 SOUND 1,450:GOTO 280
330 PEN 0:LOCATE wsp(k,1)+1,wsp(k,2)+1:PRINT lit$(tabl(k))
340 IF k=9 THEN GOTO 240
350 k=k+1:GOTO 250
360 FOR v=1 TO 4
370 IF pow(k,v)=l THEN GOTO 400
380 NEXT v
390 SOUND 1,500:LOCATE £1,2,2:PRINT £1,"PRZESUNIECIE":PRINT £1," NIEMOZLIWE":FOR
 T=1 TO 1000:NEXT T:CLS £1:GOTO 250
400 PEN 0:LOCATE wsp(L,1)+1,wsp(L,2)+1:PRINT lit$(tabl(k))
410 LOCATE wsp(K,1)+1,wsp(K,2)+1:PRINT " ":TABL(L)=TABL(K):TABL(K)=1:L=K
420 GOSUB 770
430 ON BL+1 GOTO 180,240
```

W segmencie głównym programu znajdują się wywołania podprogramów, których rola i wersje źródłowe są następujące:

1. Definiowanie symboli własnych (kształt krążka) oraz sparametryzowane wpisywanie symbolu na ekran (wg wartości parametrów x,y będących współrzędnymi znakowymi).

```
440 '
450 '          Podprogram definicji symboli wlasnych
460 '
470 SYMBOL AFTER 125
480 SYMBOL 125,0,0,0,0,3,7,15,15
490 SYMBOL 126,0,0,60,255,255,255,255,255
500 SYMBOL 127,31,31,63,63,63,63,31,31
510 SYMBOL 128,248,248,252,252,252,252,248,248
520 SYMBOL 129,15,15,7,3
530 SYMBOL 130,255,255,255,255,255,60
540 SYMBOL 131,240,240,224,192
550 SYMBOL 132,0,0,0,0,192,224,240,240
560 RETURN
570 LOCATE x,y:PRINT CHR$(125);CHR$(126);CHR$(132)
580 LOCATE x,y+1:PRINT CHR$(127);CHR$(143);CHR$(128)
590 LOCATE x,y+2:PRINT CHR$(129);CHR$(130);CHR$(131)
600 RETURN
```

2. Losowe generowanie początkowego ustawienia liter na planszy

```
610 '
620 '        Podprogram losowego ustawiania liter na planszy
630 '
640 los=2+INT(RND*8)
650 tabl(1)=1:tabl(2)=los
660 FOR k=3 TO 9
670 los=2+INT(RND*8)
680 FOR i=2 TO k-1
690 IF tabl(i)=los THEN GOTO 670
700 NEXT i
710 tabl(k)=los
720 NEXT k
730 RETURN
```

3. Badanie poprawnego rozwiązania zadania — zakończenie gry i ewentualne powtórzenie gry.

```
740 '
750 '        Podprogram badania poprawnosci zadania
760 '
770 BL=0
780 FOR J=1 TO 9
790 IF TABL(J)=J THEN GOTO 810
800 BL=1
810 NEXT J
820 IF BL=1 THEN RETURN
830 LOCATE £1,2,2:PRINT £1,"BRAWO!":PRINT £1," UDALO SIE!"
840 FOR J=0 TO 26:BORDER J:FOR T=1 TO 100:NEXT T:NEXT J
850 CLS £1:PRINT £1:PRINT £1,"SPROBUJESZ":PRINT £1,"JESZCZE RAZ? ":INPUT £1,"(T/
N) ",T$
860 IF T$="T" OR T$="t" THEN LOCATE 13,13:PRINT " ":CLS £1:RETURN
870 IF T$="n" OR T$="N" THEN STOP
880 GOTO 850
```

4. Ryśowanie planszy gry

```
890 '
900 '        Podprogram rysowania planszy
910 '
920 MODE 1
930 DIM wsp(9,2):DIM pow(9,4)
940 GOSUB 470
950 FOR i=1 TO 9:FOR j=1 TO 4:READ pow(i,j):NEXT j:NEXT i
960 DATA 2,4,6,8,1,5,7,0,6,8,0,0,1,7,9,0,8,2,0,0,1,3,9,0,2,4,0,0,1,3,5,0,6,4,0,0
970 FOR i=1 TO 9:READ wsp(i,1),wsp(i,2):NEXT i
980 DATA 12,12,12,2,20,5,22,12,20,19,12,22,4,19,2,12,4,5
990 DIM LIT$(9):FOR i=1 TO 9:READ LIT$(i):NEXT I
1000 DATA " ","K","A","M","E","L","E","O","N"
1010 PEN 1:GRAPHICS PEN 1
1020 FOR I=1 TO 9:X=WSP(I,1):Y=WSP(I,2):GOSUB 570:NEXT I
1030 FOR I=2 TO 4 STEP 2
1040 FOR K=5+(I-2) TO 7+(I-2)
1050 PLOT WSP(I,1)*16+9,(24-WSP(I,2))*16+7
1060 DRAW WSP(K,1)*16+9,(24-WSP(K,2))*16+7
1070 NEXT K
1080 NEXT I
1090 FOR K=3 TO 5
1100 PLOT WSP(8,1)*16+9,(24-WSP(8,2))*16+7
1110 DRAW WSP(K,1)*16+9,(24-WSP(K,2))*16+7
1120 NEXT K
1130 WINDOW £1,27,40,11,15:PAPER £1,1:CLS £1:PEN £1,3
1140 FOR K=3 TO 9 STEP 6
1150 PLOT WSP(6,1)*16+9,(24-WSP(6,2))*16+7
1160 DRAW WSP(K,1)*16+9,(24-WSP(K,2))*16+7
1170 NEXT K
1180 PEN 3:LOCATE 13,1:PRINT"K":LOCATE 23,6:PRINT"A":LOCATE 25,13:PRINT"M":LOCAT
E 23,20:PRINT"E":LOCATE 13,25:PRINT"L":LOCATE 3,20:P
RINT"E":LOCATE 1,13:PRINT"O":LOCATE 3,6:PRINT"N"
1190 RETURN
```

Rady i porady programisty (2)

Z. ŁUKASZEWICZ

Pisząc program rzadko zwracam uwagę na jego elegancję i rozwiązania konstrukcyjne, dążąc najprościej do osiągnięcia zamierzonego celu. Pamiętać jednak należy również o tym, że dobry program, to nie tylko taki program, który się „nie zacina", ale również jest czytelnie napisany. Przez czytelność i elegancję programu rozumiemy ograniczenie do minimum wierszy zawierających kolejno po sobie występujące instrukcje, instrukcje "go to", czyli unikanie zbędnych skoków. O tym i o wielu innych problemach wpływających na przejrzystość programów wielokrotnie pisano na łamach „IKS-a".

Jednym ze sposobów uniknięcia tych błędów jest posiadanie biblioteki krótkich, sprawdzonych podprogramów najczęściej używanych w pracy programisty-amatora. Takim przykładem jest podprogram umożliwiający sterowanie w programie w zależności od wybranego klawisza.

```
10 '*****STEROWANIE PRZY UŻYCIU KLAWIATURY*****
20 RESTORE 80
30 FOR r%=1 TO n
40 READ s%
50 IF INKEY (s%)=0 THEN 90
60 NEXT
70 GOTO 20
80 DATA K1, K2, ..., Kn
90 ON r% GOSUB E1, E2, ..., En
```

W podprogramie rzeczywiste wartości zastąpiono parametrami, i tak:

n — liczba klawiszy użytych do sterowania;

K1, ..., Kn — numery klawiszy użytych do sterowania (numery umieszczone są na obudowie stacji dysków);

E1, ..., En — numery linii, w której zaczynają się podprogramy, które należy wykonać po wciśnięciu klawisza o numerze Kn.

Aby ułatwić zrozumienie przedstawiono poniżej przykład, w którym do sterowania użyto czterech klawiszy o numerach 64, 65, 57, 56 czyli „1", „2", „3", i „4".

Zastosowanie tego podprogramu w twoich programach na pewno poprawi ich przejrzystość i czytelność.

Spróbuj, sprawdź!

```
10 MODE 2
20 WINDOW£1,30,50,12,12
30 '**** PRZYKŁAD ***
40 CLS£1
50 RESTORE 110
60 FOR r%=1 TO 4
70 READ s%
80 IF INKEY (s%)=0 THEN 120
90 NEXT
100 GOTO 50
110 DATA 64,65,57,56
120 ON r% GOSUB 150,160,170,180
130 CALL &BB06
140 GOTO 40
150 CLEAR INPUT: PRINT£1," WYBRANO KLAWISZ 1":RETURN
160 CLEAR INPUT: PRINT£1," WYBRANO KLAWISZ 2":RETURN
170 CLEAR INPUT: PRINT£1," WYBRANO KLAWISZ 3":RETURN
180 CLEAR INPUT: PRINT£1," WYBRANO KLAWISZ 4":RETURN
```

INFORMATYKA W SZKOLE

Lektura poprzednich dwóch „odcinków" omawiających tematy naszej propozycji podstaw informatyki upoważnia już nas zapewne do przyjrzenia się niektórym problemom programowania komputerów. Komputerów? Tak. Komputerów, nie ma bowiem większego znaczenia, jak wielki jest to sprzęt, mikro, mini, stołowy czy też super lub hiper, ogólne zasady programowania są niezmienne.

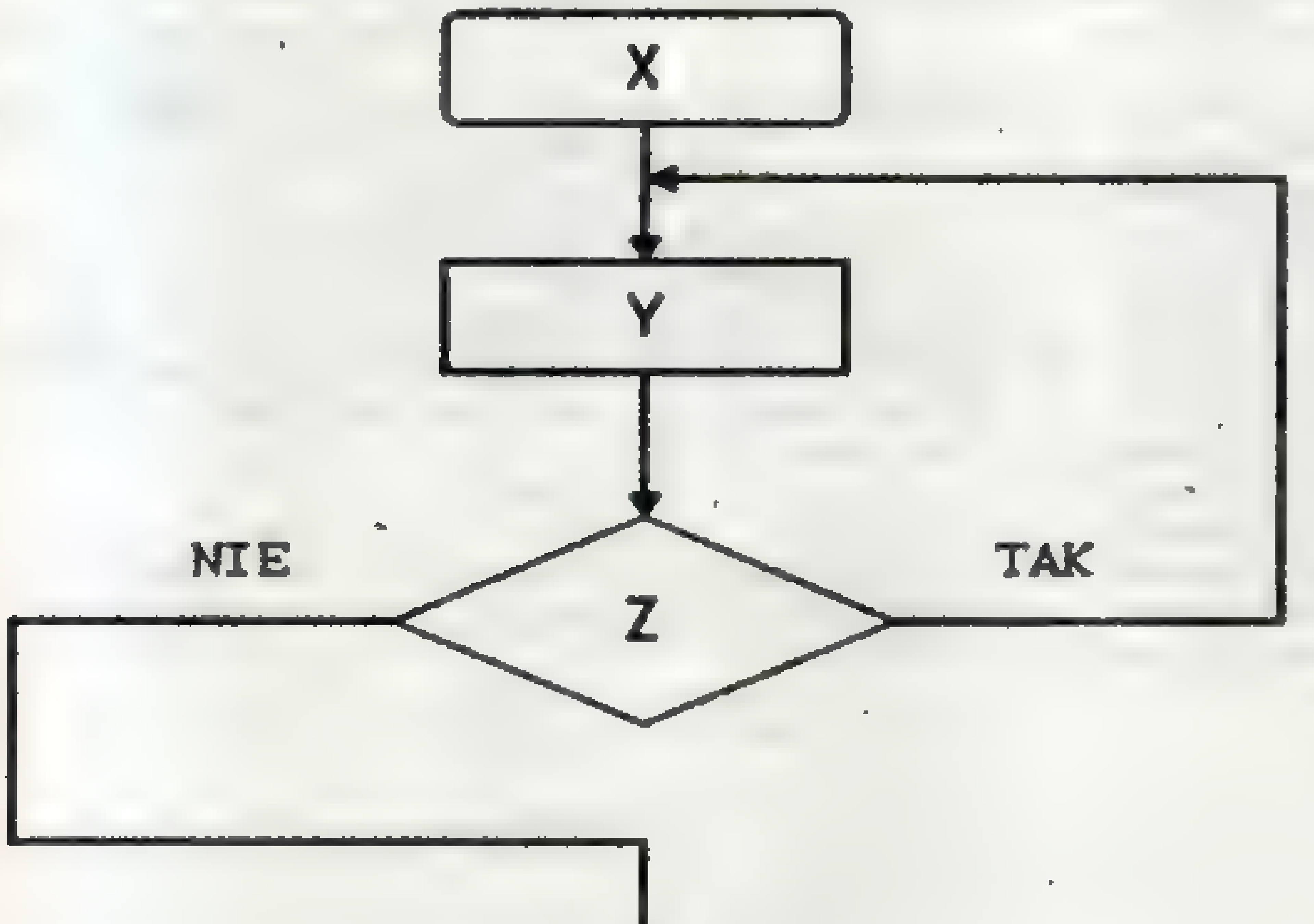
Programowanie

(Materiały do Podstaw Informatyki cz. 3)

Aby choć w części zasady te zrozumieć, koniecznie trzeba wiedzieć co to jest mikroprocesor, pamięć i oczywiście bajt! Wróćmy do naszego przykładu dodawania dwóch liczb. Załóżmy, że są one zapisane w pamięci (oczywiście RAM) komputera. Pierwsza liczba np.: 123 w komórce nr 100, druga natomiast np.: 056 w komórce nr 101. Pozostałe komórki pamięci są „puste" (w wypadku Spectrum, pozostaną wolne następujące komórki: od 0 do 99 i od 102 do 48 000). Usiłując dodać, za pomocą komputera, owe liczby (123+056) maszyna powinna wykonać określone czynności:

1. *Pobrać (odczytać) zawartość setnej komórki pamięci i przekazać (tę zawartość = 123) do procesora.*
2. *Pobrać drugi argument z komórki sto jeden (056) i przekazać do procesora.*
3. *Wysterować procesor tak by dodał wcześniej wprowadzone argumenty (123+056=179).*
4. *Przekazać z procesora wynik do komórki np.: 102.*

W efekcie wykonania podanych operacji wynik dodawania (179) będzie umieszczony w komórce nr 102. Operacje te (czynności), w podanym przykładzie, tworzą pewien „przepis" — przykład prostego algorytmu procesu dodania dwóch liczb przez komputer. Innymi słowy jest to opis kolejnych czynności, których wykonanie umożliwia nam osiągnięcie zamierzonego celu. Zatem algorytm ułatwia poprawne, jednoznaczne i czytelne sprecyzowanie czynności, tworzących pracę powierzoną maszynie. Czyli przed przystąpieniem do pisania programu celowe jest zbudowanie algorytmu. Zadanie to jest nieco ułatwione przez stosowanie umownych symboli graficznych.



x — treść opisująca czynność wykonywaną na początku lub na końcu algorytmu
y — czynności związane z przetwarzaniem informacji
z — sprawdzenie warunku i odpowiednie skierowanie do wykonywania kolejnych czynności

Rys. 1 Niektóre symbole graficzne stosowane w algorytmach

Nareszcie możemy przystąpić do próby napisania programu. Polega to na tłumaczeniu treści zawartych w algorytmie na taką postać, która jest „zrozumiana" przez maszynę. Podobnie jak tłumaczenie, osobie posługującej się tylko językiem angielskim, poleceń wypowiadanych w języku polskim. W praktyce nie jest to takie łatwe, procesor bowiem wykonuje tylko bardzo proste operacje — podobne do wykorzystanych w podanym wcześniej przykładzie. Mikroprocesor jest jak małe dziecko — należy mówić do niego, wyraźnie (bez błędów), bardzo prostym językiem — najlepiej pojedynczymi słowami: idź, dodaj, skocz, przesuń. Oczywiście owe polecenia mają swoje kody — „liczbowe nazwy" — na przykład operację dodawania wykona procesor na „polecenie" (rozkaz) 10000010. Zatem, aby dodać dwie liczby musimy „tylko" przekazać do procesora z dziesięć odpowiednich kodów operacji i „już" mamy wynik. Trzydzieści, dwadzieścia lat temu był to zresztą jedyny sposób programowania komputerów. Dzisiaj komputery są bardziej „życzliwe" („dorosły"), możemy bowiem opisywać nasz algorytm (pisać program) niemal w języku naturalnym, na przykład: PRINT A+B. Ten prosty zapis spowoduje dodanie przez komputer zawartości komórki A do komórki B i wyświetlenie wyniku na ekranie. Jeśli wcześniej do A wpisaliśmy 123 (LET A=123), a do B wprowadzono liczbę 056 (LET B=56), na ekranie ukaże się wynik: 179.

Tak wielkie zbliżenie języka programowania do tego jakim posługują się ludzie, możliwe stało się dzięki specjalnym **programom tłumaczącym**. Należy bowiem pamiętać, że mikroprocesory nadal „rozumieją" kody tylko bardzo prostych operacji. Zatem wspomniany „tłumacz" zamienia naszą jedną instrukcję np.: PRINT A+B na kilkadziesiąt kodów „rozumianych" przez procesor.

Innymi słowy mamy dwa rodzaje, a poprawniej — dwa poziomy języków. Język składający się z kodów (liczb) wymuszających proste operacje — jest to tak zwany **język wewnętrzny** (maszynowy). Bardzo **szybki**, elastyczny, ale kłopotliwy w użyciu. Mówi się o nim, że jest to język niskiego poziomu. Inny dla każdego typu procesora. Zważywszy jednak na niezmiennie **dużą użyteczność** języka wewnętrznego, a jednocześnie na kłopoty wynikające z konieczności stosowania kodów operacji w postaci liczb — powstał **assembler.** W języku tym posługujemy się literowy-

mi, bardziej czytelnymi niż liczby, kodami operacji (także adresów). Na przykład zamiast kodu 10000100, piszemy ADD (ang. dodaj), zamiast kodu 010000100, piszemy MOV (ang. MOVE — przesuń). W przykładach tych, mając na uwadze ich czytelność, pominięto kwęstię adresacji, zainteresowanym polecam np.: „Mikrokomputery", A. J. Dirksen, WKiŁ, Warszawa 1985. Kończąc tę krótką prezentację assemblera należy pamiętać, że czas pisania programu w tym języku jest średnio osiem razy dłuższy od programowania w **językach wysokiego poziomu.** Jednak należy mieć na uwadze to, że zysk ten powstaje kosztem znacznego zmniejszenia szybkości działania programów.

Różne zastosowania komputerów wymusiły tworzenie odpowiednich języków wysokiego poziomu. Na przykład do oprogramowania obliczeń matematycznych stosuje się język FORTRAN (także w superkomputerach), do przetwarzania tekstów LISP, do budowy oprogramowania systemowego — język C, itp.

Warto w tym miejscu trochę więcej miejsca poświęcić wspomnianym „tłumaczom" języków wysokopoziomowych na język wewnętrzny komputera. Wykorzystuje się do tego celu, **kompilatory** lub **interpretery.** Kompilator jest to (w znacznym uproszczeniu) program komputerowy, który zamienia instrukcje języka wysokopoziomowego na język wewnętrzny. Wprowadzamy zatem do komputera nasz program napisany np. w Basicu, tam (w pamięci komputera) czeka na niego kompilator. Następnie rozpoczyna się proces kompilacji po jej ukończeniu, w pamięci komputera pozostaje nasz program, ale już zupełnie w innej postaci — jako program napisany w języku wewnętrznym (zrozumiałym dla procesora). W takiej właśnie postaci jest on dalej wykonywany.

Natomiast **interpretery,** to także programy tłumaczące, ale „na żywo". Oznacza to, że **za każdym razem,** gdy uruchamiamy program napisany np.: w Basicu, jest on (w trakcie realizacji) tłumaczony (przez interpreter) instrukcja po instrukcji na język wewnętrzny i dopiero wykonywany przez komputer. Zatem wykonanie takiego programu jest znacznie dłuższe (wolniejsze), bowiem do okresu wykonania każdej instrukcji programu zawsze dodaje się czas jej tłumaczenia.

W większości domowych komputerów, właśnie w pamięci ROM, najczęściej zapisane mamy interpretery Basica, (były także, nieudane ze względu na brak zainteresowania próby z Pascalem, Logo i innymi programami.

Zważywszy na informacyjny zakres naszej nauki podstaw informatyki należy zwrócić uwagę na ryzyko, jakie niosła by obecnie sobą próba zbyt dokładnego poznania jakiegokolwiek języka programowania. Dlatego też, zważywszy na konieczność równoległego uczestniczenia PT Czytelników w zajęciach praktycznych, proponuję analizę funkcji **tylko kilku** instrukcji języka BASIC: PRINT, INPUT, DIM, GO TO, FOR, NEXT. Wcześniej jednak koniecznie powinniśmy poznać klawiaturę naszego komputera (klawisze funkcyjne, sposób pisania komend systemowych i instrukcji, itp.). Niestety pamiętać także trzeba, iż istnieją różne „dialekty" języka Basic. Oznacza to, że nie wszystkie nasze przykłady będą bezpośrednio dotyczyły różnych komputerów, dotyczą one bowiem, nadal najpopularniejszego u nas mikrokomputera Spectrum. W szczególności dotyczy to instrukcji DIM, która jest odmiennie interpretowana w Spectrum i innych komputerach np.: w Amstradzie. Proponuję w tej sytuacji przestudiować dokładniej niektóre fragmenty dokumentacji dostępnego komputera.

Wiemy zatem, że komputer dysponuje pamięcią, mikroprocesorem, szeregiem innych ciekawych elementów (mowa o tym będzie innym razem) — w sumie — zdolnością wykonywania (bardzo szybko) instrukcji w różnych językach programowania.

Przystępując do analizy niektórych zagadnień związanych z programowaniem komputerów należy przyjąć pewne założenie dotyczące interpretacji słowa „komórka". Wcześniej bowiem, dla ułatwienia, termin ten był jednoznaczny z bajtem. Jednak w językach wysokopoziomowych komórka staje się już wyróżnionym obszarem pamięci, składającym się z wielu bajtów. Istotnym ułatwieniem jest sposób identyfikacji owych komórek — mają one swoje (nadawane przez nas!) nazwy. Na przykład: IKS, nowa, P13, itp. Chcąc zatem cokolwiek zapisać do komórki (lub odczytać), posługujemy się jej nazwą, a nie numerem. Znakomicie ułatwia to pisanie programów. Szczególnie ten sposób **adresowania** miejsca w pamięci (jedna nazwa dla wielu bajtów) przydatny jest gdy chcemy posługiwać się wieloma danymi tego samego typu. Na przykład rejestrując w pamięci skróty informatyczne (zob. „IKS" nr 5/1988) budujemy tak zwaną **tablicę** o wielkości umożliwiającej zapamiętanie wielu skrótów. Np.: A$(a,b). Znak $ oznacza, że w tablicy o nazwie A zapisywane będą teksty, a — ile będzie skrótów, b — ile przewidujemy znaków dla każdego skrótu. Ten sposób gospodarki pamięcią okaże się bardzo przydatny. Aby można było korzystać z takiej tablicy należy wcześniej wykonać instrukcję DIM. Jej działanie można porównać z pracą mającą na celu dzielenie naszej działki na grządki: tu cebula, tam marchew, itp. Działka — to cała pamięć, grządki to jej części przydzielone różnym informacjom. DIM „pilnuje" porządku, np. by nie zrobić zbyt wielu grządek, bo po prostu działka okaże się zbyt mała. Stosując zatem DIM, należy podać ile danych chcemy wprowadzić — np. 200 skrótów i jak wielki jest każdy skrót (ile zajmuje znaków), np. dziesięć. Zapisujemy to następująco: DIM A$(200, 10).

Oznacza to, że w komputerze zostanie zarezerwowany obszar — nasza tablica A o wielkości 2000 bajtów z przeznaczeniem na dwieście skrótów, każdy po 10 znaków. Lub inaczej — w RAM zostanie zarezerwowane 200 komórek, każda po 10 znaków. Czyli komórki:

A$ (1, 10) — komórka nr 1
A$ (2, 10) — komórka nr 2
A$ (3, 10) — komórka nr 3
.
.
.
A$ (199, 10)
A$ (200, 10) — komórka nr 200.

Wróćmy teraz do naszego głównego tematu — programowanie. Na dobry początek wprowadzamy do pamięci komputera instrukcję:

INPUT „ALA", a

Po jej uruchomieniu (klawisz RETURN lub ENTER), na ekranie ukaże się słowo: ALA?, ze znakiem zapytania. Jest to efekt „bezmyślnego" odtworzenia przez maszynę tekstu w cudzysłowie (ALA) i komputer czeka na to byśmy napisali jakąś liczbę. Po napisaniu tej liczby, np. 18, zostanie ona zapamiętana w pamięci RAM. W którym miejscu (przecież pamięć jest bardzo duża)? W komórce o nazwie a. Jest to możliwe do realizacji w tak prosty sposób dzięki interpreterowi Basica. Szybko nadaje on, wymyślone przez nas nazwy kolejnym komórkom pamięci RAM. W naszym wypadku jedna z komórek będzie miała nazwę „a".

Gdy napiszemy: INPUT „ALA", nic nazwa naszej komórki brzmi „nic". Z doświadczenia wiadomo, iż zrozumienie relacji nazwy komórki i jej zawartości jest dosyć kłopotliwe. Dlatego omówienie tego tematu zostało powtórzone. Warto zresztą wrócić do tego tematu, jeśli nie pamiętamy w tej chwili zawartości komórki „a"?

Wobec powyższego, instrukcja INPUT może wyświetlić dowolny tekst „w uszach" (w cudzysłowie) oraz, co najważniejsze, wprowadzić do pamięci pod określony adres dowolną liczbę. Czy tylko liczbę? Mogą być także litery, ale wówczas, jak już było wcześniej — mówione — nazwę komórki musi kończyć znak $.

Na przykład: INPUT „Podaj nazwę miasta", m$

Po uruchomieniu tej instrukcji na ekranie ukaże się:

Podaj nazwę miasta?

w odpowiedzi piszemy:

Ciechocinek

zatem w komórce m$ będzie zapisana nazwa naszego pięknego uzdrowiska.

Podsumowując naszą wiedzę o instrukcji INPUT, należy podkreślić, iż komputer wykonując ją wypisuje dowolny tekst „w uszach" (najlepiej, gdy dotyczy on treści oczekiwanej przez komputer informacji) i czeka na przekazanie mu (przez nas) danych.

W ten sposób, niestety dosyć złożony, możemy stwierdzić, iż wiemy jak wprowadzić do komputera dowolną liczbę lub tekst. Mało tego — wiemy także, gdzie wprowadzone przez nas informacje są zapisane, np. w komórce: a, nic, m.

Wróćmy do przykładów. Każda instrukcja budowanego przez nas programu musi mieć swój numer (żeby interpreter Basica się nie pogubił, w jakiej kolejności ma wykonywać wprowadzane przez nas instrukcje). W praktyce instrukcje numeruje się od 10 z odstępem też 10. Jest to uzasadnione potrzebami, które szybko podczas pracy z komputerem okażą się istotne.

Gdy wprowadzimy do komputera instrukcje:

10 INPUT „Podaj swoje nazwisko", n$

20 INPUT „Podaj swoje imię", i$
po ich uruchomieniu i udzieleniu odpowiedzi (Abacki Tadeusz) w RAM zapisane zostanie nazwisko i imię. I co dalej? Po co to wszystko?

W rezultacie w komórce o adresie ACK — zapisana zostanie interpretacja tego skrótu, pod adresem ACU, ADA i innych podobnie.
Zatem pisząc:

Trzeba jednak bardzo szybko zaznaczyć, że program ten jest fatalny — INPUT powtórzone 249 razy? Stwierdzenie to sugeruje, że istnieją inne sposoby konstrukcji programów.

Programowanie

(Materiały do Podstaw Informatyki cz. 3)

Przejdźmy zatem do wykorzystania tak pracowicie wprowadzonych informacji. Służy temu, w sposób bardzo czytelny, instrukcja PRINT. Działa ona akurat odwrotnie od „INPUTa". Oznacza to, że gdy wykonamy instrukcję:

PRINT „Twoje nazwisko brzmi:", n
na ekranie ukaże się tekst:

Twoje nazwisko brzmi: Abacki

Czyli PRINT wypisuje „bezmyślnie" tekst „w uszach" oraz to co wcześniej wprowadziliśmy do komórki n.

Realizacja instrukcji: PRINT „Mieszkam w:", i będzie następująca:

Mieszkam w: Tadeusz

Bez sensu! Właśnie przykład ten wskazuje, że tekst w uszach musi mieć sens związany z treścią wypisywanej komórki. Zatem poprawnie powinno być:

PRINT „Mieszkam w:", m
Na ekranie ukaże się:

Mieszkam w: Ciechocinek

Prawie dobrze.

Uporaliśmy się zatem z tekstem wyświetlanym na ekranie (skąd się on bierze? I po co?) oraz danymi, które wprowadzamy, by móc je w dowolnej chwili odczytać. Można powiedzieć, że jest to bardzo odległe od praktyki. Czyżby?

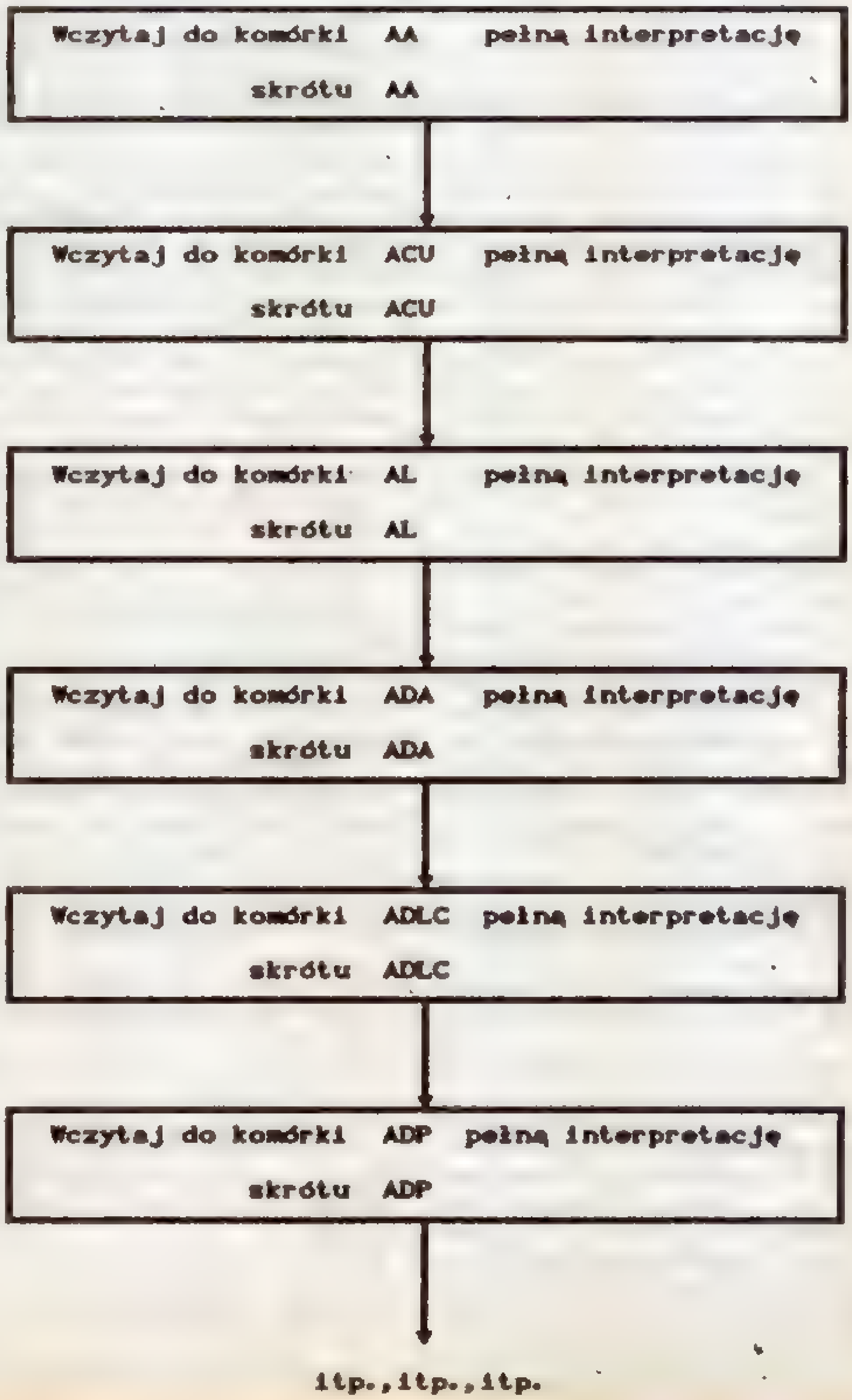
Wyobraźmy sobie, że „Inputami" wprowadziliśmy do pamięci komputera oryginalną interpretację wszystkich skrótów informatycznych, np.:

10 INPUT ACK$ (pomijamy tekst w uszach)
20 INPUT ACU$
30 INPUT ADA$
40 INPUT ADLC$
.
.
.
249 INPUT WPR$

Po uruchomieniu naszego programu, korzystając z „IKS-a" nr 5 wpisujemy kolejno:
Affirmative Acknowledgment (AA)
Automatic Calling Unit (ACU)
All Points Addressable (ADA)
itd.

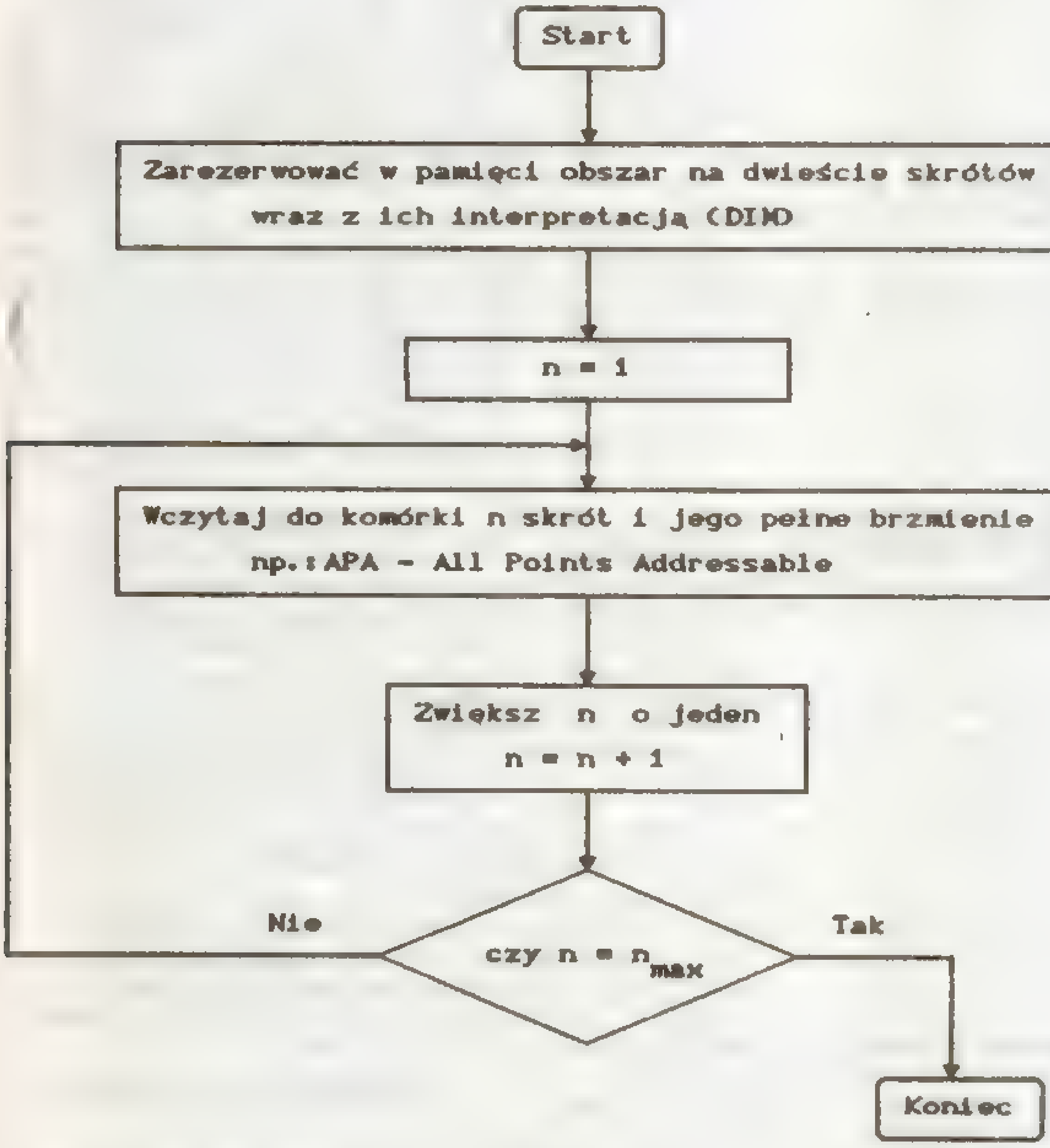
PRINT ACU
na ekranie ukaże się nam pełna interpretacja tego skrótu.

Wróćmy do algorytmu. Algorytm naszego dotychczasowego programu wygląda następująco:



Rys. 2

Wygląda to okropnie. Na szczęście można znacznie prościej, czytelniej:



Rys. 3

A jak to będzie wyglądało w postaci pro gramu?

```
10 DIM S$(200,100)
20 FOR n = 1 to 200
30 INPUT "Podaj kolejny skrot minimum 4 znaki" i jego interpretacj
   S$(n,1 to 200)
40 NEXT n
50 STOP
```

Prawda, że prościej. W tym miejscu należy przedstawić kilka słów wyjaśnienia o znaczeniu kolejnych linii naszego nowego programu. Linia 10 DIM S$ (200, 100) to wspomniana wcześniej rezerwacja miejsca w pamięci dla wprowadzanych przez nas informacji — 200 skrótów (każdy może zajmować do 100 znaków). Kolejne linie naszego programu tworzą tak zwaną pętlę. Jej wykonanie polega na wielokrotnej realizacji czynności zapisanych między instrukcjami FOR i NEXT dla wypadków gdy n=1, n=2, n=3 aż do n=200 — czyli 200 razy. Komputer wykonując instrukcję zawartą w linii nr 20 ustala wartość n, początkowo wynosi ona jeden, i „idzie" dalej do linii 30. Jej wykonanie powoduje wyświetlenie na ekranie monitora tekstu

Podaj kolejny skrót (minimum 4 znaki) i jego interpretację: w odpowiedzi piszemy np.:

ACU — Automatic Calling Unit,

tekst ten zostanie zapamiętany w komórce S$ (1, 100). Należy zwrócić uwagę, że skrót ten musi mieć co najmniej cztery znaki. Jeśli jest krótszy, należy uzupełnić go od [illegible] tak zwane „spacje"). Potrzebę tę uzasadnia konstrukcja dalszego ciągu naszego programu. Instrukcja NEXT n z linii nr 40 dodaje jedynkę do n i „na warsztat" dostaje się komórka S$ (2, 100), ponadto sprawdzany jest warunek czy n nie jest większe od 200, jeśli tak, to koniec programu, jeśli nie — program automatycznie wraca (pętla!) do linii 20 i proces wprowadzania danych powtarza się, ale już dotyczy to komórki S$ (2, 100).

Proszę zwrócić uwagę, że pięcioma instrukcjami zastąpiliśmy 249! Efekt pracy obu programów jest ten sam (niemal). Wskazuje to na rangę problemu poprawnego pisania programu. Programy wykonujące tę samą pracę mogą być napisane na dziesiątki sposobów, lepiej, gorzej. Wskazuje to na rangę problemu poprawnego pisania programu — kwalifikacji programisty.

Analizując rozwiązywanie naszego problemu — dokładnie widać, że dane są sprawnie wprowadzane do komputera. Umknęła nam jednak kwestia zasadnicza — po co to wprowadzanie? Aby móc szybko wyszukiwać wprowadzone w ten sposób informacje. W związku z tym nasz pięcioliniowy program powinniśmy uzupełnić o fragment (modół, część), którego wykonanie przez komputer powoduje, iż maszyna zapyta nas — użytkowników, jaki skrót chcemy rozszyfrować. Modół ten powinien także oczywiście wyszukiwać w pamięci owo rozszyfrowanie — pełną

INFORMATYKA W SZKOLE

interpretację podanego przez nas skrótu. Zatem nasz program, wraz z uzupełnieniem będzie wyglądał następująco:

```
10 DIM S$(200,100)
20 FOR n = 1 to 200
30 INPUT "Podaj kolejny skrot minimum 4 znaki
" i jego interpretacje:"; S$(n,1 to 200)
40 NEXT n
50 STOP
110 INPUT "Podaj skrot, ktorego interpretacje
poszukujesz"; a$ 120 FOR n = 1 to 200
130 IF a$ = S$(n,1 to 4) THEN GO TO 150
140 NEXT n
150 PRINT S$(n,1 to 100)
160 GO TO 110
```

Gdy uruchomimy naszą nową część programu (należy napisać GO TO 110) instrukcja z linii 110 wypisze na ekranie tekst:

Podaj skrót, którego interpretację poszukujesz: i czeka na nasze pytanie. Podajemy je pisząc skrót, np.: ACU. Zostaje on wpisany do komórki a$. Znając działanie instrukcji z linii 120 i 140, przyjrzyjmy się nowej:

IF a$ = S$(n, 1 to 4) THEN GO TO 150

IF to po prostu pytanie „czy? (jeśli?)" zawartość komórki a$ (a tam jest nasze pytanie: skrót ACU) jest identyczna z zawartością pierwszych czterech znaków komórki S$ (n, 1 to 4) — zapis ten oznacza, że przedmiotem naszego zainteresowania jest komórka nr n, a konkretnie znaki od 1 do 4 (to w jęz. ang. oznacza „do"). Zamieszanie to z czterema znakami wynika z tego, iż wpisywane przez nas pierwsze cztery znaki każdej pełnej informacji o skrótach, to cztery pierwsze znaki skrótu. Jeśli odpowiedź jest twierdząca (znaki pytania są identyczne z początkowymi znakami interpretacji) to oznacza, że w tej właśnie komórce S$(n, 100) mieści się pełna interpretacja skrótu o poszukiwanej interpretacji. Wobec tego program „każe iść" komputerowi do linii nr 150 (GO TO 150). W linii nr 150 jest instrukcja wyświetlająca oczekiwaną przez nas informację — pełną informację o skrócie ACU:

ACU — Automatic Calling Unit

Po wyświetleniu tego tekstu wykonuje się linia nr 160 — czyli skok (idź do) do linii nr 110. Zabawa zaczyna się od nowa — możemy zapytać o interpretację kolejnego skrótu.

Jeśli wynik porównania pierwszych czterech liter zawartych w komórce a$ i S$ (n, 100) jest negatywny (litery są różne), wykonuje się znana nam już instrukcja w linii 140 — czyli dodanie jedynki do n (n=n+1) i powrót na początek do linii 120, aż do chwili, gdy n będzie większe od 200 — wówczas, to już praca domowa.

Niewątpliwie omawiany przykład nie jest prosty podczas pierwszego czytania. Sądzę jednak, że wkrótce będziemy mogli przejść do kolejnego tematu. Ale to już w następnym numerze „IKS-a".

Włodzimierz GOGOŁEK

...aż człowiek stworzył komputer (2)

Kontynuując temat podjęty w poprzednim numerze „IKS-a" przedstawimy osiągnięcia polskiej myśli technicznej, polskich twórców komputerów, przedstawimy wreszcie rys historyczny rozwoju informatyki w Polsce.

W 1921 roku po raz pierwszy w Polsce zastosowano maszyny licząco-analityczne do prac statystycznych.

W 1950 roku Zbigniew Pawlak zbudował pierwszą w Polsce cyfrową maszynę liczącą na przekaźnikach, którą zastosowano do celów dydaktycznych. W cztery lata później zbudowano pierwszy eksperymentalny, elektroniczny analizator równań różniczkowych specjalnych.

W 1958 roku zespół konstruktorów Zakładu Aparatów Matematycznych PAN zbudował pierwszy w Polsce komputer cyfrowy pierwszej generacji XYZ. Komputer XYZ był zbudowany na przerzutnikach dynamicznych lampowo-tranzystorowych i zawierał pamięć operacyjną rtęciową o pojemności 1024 słów 18-bitowych. Był modelem laboratoryjnym i wykonywał 800 dodawań na sekundę. W tym samym roku z inicjatywy Zakładu Aparatów Matematycznych Instytutu Matematycznego PAN założono pierwszy w Polsce usługowy ośrodek obliczeniowy na bazie komputera XYZ pod nazwą Biuro Obliczeń i Programów.

W 1959 roku w związku z produkcją sprzętu komputerowego utworzono Wrocławskie Zakłady Elektroniczne, a rok później w Zakładzie Doświadczalnym Instytutu Maszyn Matematycznych PAN zbudowano komputer ZAM 2. Otwierał on serię komputerów wyprodukowanych w kraju i zaliczanych do pierwszej generacji. Komputer ZAM 2 miał pamięć operacyjną o pojemności 1024 słów 18-bitowych, urządzenie wejściowe w postaci czytnika tasiemki dziurkowanej, a niektóre egzemplarze były także wyposażone w czytnik kart dziurkowanych. Urządzenie wyjściowe tworzyły dziurkowane tasiemki oraz pamięć zewnętrzna, zawierająca do 4 jednostek pamięci bębnowej, każda o pojemności 16 Ksłów. Wyniki obliczeń były odtwarzane z tasiemki dziurkowanej na dalekopisie. ZAM 2 pracował z szybkością 1000 dodawań na sekundę. Pracą sterował prosty system operacyjny, a w skład oprogramowania wchodziły dwa języki programowania SAS (skrót od pierwszych liter wyrazu pełnej nazwy: SYSTEM ADRESÓW SYMBOLICZNYCH) i SAKO (SYSTEM

AUTOMATYCZNEGO KODOWANIA), które umożliwiały obliczenia naukowo-techniczne, a nawet proste przetwarzanie danych. W tym samym mniej więcej okresie we Wrocławskich Zakładach Elektronicznych ELWRO zbudowano pierwszy model komputera ODRA 1001.

W 1961 roku opracowano translator dla pierwszego języka programowania SAKO, stworzonego w Instytucie Maszyn Matematycznych w Warszawie. Rok później w zakładach ELWRO we Wrocławiu skonstruowano komputer ODRA 1003, którego seryjna produkcja ruszyła od 1964 roku. Komputer ODRA 1003 był pierwszym małym komputerem (produkowanym do 1965 roku), przeznaczonym do obliczeń naukowo-technicznych i sterowania procesami technologicznymi. Był zbudowany na zasadach techniki tranzystorowej. Przy operacji dodawania pracował z szybkością około 1000 operacji na sekundę. W skład zestawu komputera wchodziły: arytmometr, układ sterowania, pamięć bębnowa, urządzenia wejścia-wyjścia (czytniki i dziurkarka tasiemki papierowej oraz dalekopis).

W 1963 roku na podstawie projektu Katedry Budowy Maszyn Politechniki Warszawskiej, Wrocławskie ELWRO rozpoczęło pierwszą, seryjną produkcję komputerów uniwersalnych typu UMC-1 (UNIWERSALNA MASZYNA CYFROWA). Komputer UMC-1 wykonywał 100 operacji na sekundę i był wyposażony w pamięć bębnową o pojemności 4096 słów 36-bitowych. Urządzenia zewnętrzne to czytnik tasiemki dziurkowanej i dalekopis. W Wojskowej Akademii Technicznej im. J. Dąbrowskiego skonstruowano cyfrowy analizator różnicowy JAGA 63.

W 1964 roku w Instytucie Maszyn Matematycznych PAN opracowano komputer ZAM 21 do obliczeń naukowo-technicznych. Jego produkcję rozpoczęto w 1966 roku. W skład zestawu komputera ZAM 21 wchodziła jednostka centralna z pamięcią operacyjną o pojemności 8 Ksłów 24-bitowych i cyklu pamięci 10 μs. Na urządzenia zewnętrzne składały się: monitor dalekopisowy, czytnik kart dziurkowanych, czytnik tasiemki dziurkowej, dziurkarka tasiemki papierowej, pamięć bębnowa, a później także dziurkarka wierszowa i pamięć na taśmie magnetycznej.

W 1965 roku Wrocławskie Zakłady Elektroniczne ELWRO skonstruowały i rozpoczęły produkcję komputerów ODRA 1013, będących udoskonaloną wersją komputera ODRA 1003. Komputer ODRA 1013 był kolejnym małym komputerem produkowanym po zaniechaniu w 1965 roku produkcji komputera ODRA 1003 i w odróżnieniu od niego miał pamięć operacyjną ferrytową o pojemności 256 słów 40-bitowych i zwiększoną wydajność obliczeń. Kolejnymi osiągnięciami ELWRO były : komputer ODRA 1103, nazwany elektronicznym kalkulatorem dziesiętnym, przeznaczonym do współpracy z maszynami licząco-analitycznymi i komputer ODRA 1204, którego produkcję rozpoczęto w 1968 roku. Komputer ODRA 1103, dzięki możliwości zastosowania operacji mnożenia i dzielenia wykorzystywano w stacjach maszyn licząco-analitycznych do przyśpieszania i usprawnienia obliczeń. Był wyposażony w pamięć operacyjną ferrytową oraz w pamięć bębnową. Szybkość pracy przy operacji dodawania lub odejmowania (liczby 8-cyfrowej) wynosiła 0,18 ms, mnożenia — 2 ms, dzielenia 20 ms. Komputer ODRA 1204 był komputerem drugiej generacji, przeznaczonym głównie do wykonywania obliczeń naukowo-technicznych oraz sterowania procesami technologicznymi w czasie rzeczywistym. W skład zestawu wchodziły: jednostka centralna z pamięcią operacyjną o pojemności od 4 do 64 Ksłów 24-bitowych i cyklu dostępu do pamięci równym 6 ms oraz z możliwością dołączenia do kanałów obsługujących urządzenia zewnętrzne do 8 urządzeń wejścia-wyjścia. Urządzenia zewnętrzne: pamięć bębnowa w postaci do 4 jednostek po 64 Ksłów 24-bitowych każda i czasie dostępu około 10 ms, czytnik tasiemki dziurkowanej, monitor w postaci elektrycznej maszyny do pisania, dziurkarka tasiemki papierowej, drukarka wierszowa. Szybkość działania przy operacji dodawania lub odejmowania liczb stałoprzecinkowych wynosiła 16 ms, a mnożenia — 80 ms.

W 1966 roku opracowano prototyp komputera ZAM 41 — pierwszego komputera do przetwarzania danych, wyprodukowanego w kraju w ramach rodziny komputerów ZAM.

ZAM 41 składał się z jednostki centralnej, zawierającej pamięć operacyjną o pojemności do 20 Ksłów, pamięci zewnętrznej taśmowej i bębnowej, urządzeń wejściowych w postaci czytników tasiemki dziurkowanej i kart dziurkowanych oraz urządzeń wyjściowych w postaci dziurkarki tasiemki papierowej i drukarki wierszowej. Szybkość działania wynosiła około 40 000 dodawań na sekundę, a organizacja komputerowa umożliwiała pracę wieloprogramową. Oprogramowanie pozwalało użytkownikom na korzystanie z następujących języków programowania: COBOL, ALGOL, SAKO, SAS, CEMMA i ZAM-GPSS. Łącznie wyprodukowano i zainstalowano od 1968 roku 16 egzemplarzy komputera ZAM 41.

W 1968 roku rozpoczęto prace koncepcyjne nad stworzeniem rodziny komputerów wspólnej produkcji państw socjalistycznych, nazywanej początkowo rodziną komputerów RIAD, a później rodziną komputerów Jednolitego Systemu. W tym samym roku utworzono Zakład Obsługi Technicznej Maszyn Matematycznych MERA-ELWRO-SERWIS, świadczący usługi w zakresie dostaw sprzętu, serwisu technicznego i oprogramowania oraz szkolenia użytkowników.

W 1969 roku w Zakładach Mechaniki Precyzyjnej w Błoniu pod Warszawą rozpoczęto produkcję drukarek wierszowych (montaż na podstawie licencji) oraz czytników tasiemki dziurkowanej CT-1001 i dziurkarki tasiemki papierowej D-102.

Na początku lat siedemdziesiątych we Wrocławskich Zakładach Elektronicznych ELWRO rozpoczęto produkcję komputerów do przetwarzania danych: ODRA 1304 (1970 rok), ODRA 1325 (lata 1971—1972), ODRA 1305 (1972 rok), pracujących do chwili obecnej w wielu ośrodkach w kraju. Komputer ODRA 1304 jest wieloprogramowym komputerem drugiej generacji produkowanym w ramach rodziny komputerów ODRA z pamięcią operacyjną o pojemności 32 Ksłów 24-bitowych i cyklu pamięci 6 μs. Ma możliwość dołączenia 9 kanałów dla urządzeń zewnętrznych m.in. kanału multiplekserowego. W skład zestawu komputera mogą wchodzić między innymi: czytnik kart dziurkowanych, czytnik tasiemki dziurkowanej, dziurkarka tasiemki papierowej, monitor operatorski, drukarka wierszowa, jednostki pamięci taśmowej. Szybkość operacji dodawania dla liczb stałoprzecinkowych wynosi 26 μs, mnożenia — 96 μs, skoku — 9 μs, a dla liczb zmiennoprzecinkowych szybkość operacji dodawania wynosi 160 μs, mnożenia — 450 μs. Komputer ODRA 1304 może być wykorzystywany zarówno do przetwarzania danych, jak i do obliczeń naukowo-technicznych. Do oprogramowania (zgodnego zresztą z oprogramowaniem komputerów firmy ICL serii 1900) oprócz programów usługowych i standardowych, zarówno dla potrzeb przetwarzania danych, jak i obliczeń naukowo-technicznych, wchodzą następujące języki programowania: PLAN, ALGOL, COBOL, FORTRAN. Komputer ODRA 1325 jest komputerem trzeciej generacji, skonstruowanym na obwodach scalonych TTL i produkowanym w ramach rodziny komputerów ODRA. Pamięć operacyjna wynosi od 8 do 128 Ksłów 24-bitowych, a cykl dostępu równy jest 1 μs. Do jednostki centralnej można podłączyć do 16 kanałów dla urządzeń zewnętrznych, w tym jednego kanału multiplekserowego. W skład zestawu mogą wchodzić wszystkie urządzenia komputera wieloprocesorowego, wieloprogramowego i wielodostępnego. W wypadku liczb stałoprzecinkowych szybkość wykonywania operacji dodawania wynosi 2,6 μs, mnożenia — 12 μs, skoku — 7 μs, a dla liczb zmiennoprzecinkowych szybkość operacji dodawania wynosi 9 μs, a mnożenia — 16 μs. Komputer ODRA 1325 jest przeznaczony do sterowania procesami technologicznymi, do obliczeń naukowo-technicznych i przetwarzania danych. Komputer ODRA 1305 jest także komputerem trzeciej generacji zbudowanym na obwodach scalonych TTL i produkowanym w ramach rodziny komputerów ODRA. Pamięć operacyjna może dochodzić do pojemności 256 Ksłów 24-bitowych, a cykl pamięci wynosi 1 μs. ODRA 1305 jest komputerem wieloprogramowym (może pracować do 16 programów), wielodostępnym (do 60 użytkowników) oraz dwuprocesorowym. Szybkość pracy dla operacji dodawania dwóch liczb stałoprzecinkowych wynosi 1,6 μs, mnożenia — 9 μs, a operacji skoku — 1 μs. Komputer ODRA 1305 jest w pełni zgodny funkcjonalnie i programowo z systemami komputerów serii ICL 1900, stąd można zarówno korzystać z oprogramowania tych komputerów, jak i komputerów ODRA 1304 i ODRA 1325. Jest przeznaczony głównie do przetwarzania danych, obliczeń naukowo-technicznych oraz do sterowania szybkimi procesami zachodzącymi w czasie rzeczywistym.

W 1970 roku w Wojskowej Akademii Technicznej zbudowano pierwszy komputer hybrydowy WAT 1001 oraz grafoskop. W skład elementów analogowych komputera hybrydowego WAT 1001 wchodzą między innymi: wzmacniacze całkujące i sumujące, pamięci analogowe, generatory funkcji nieliniowych, układy mnożące, generatory zmiennych losowych, układ samodzielnego sterowania cyfrowego itp. Komputer WAT 1001 może pracować w trybie pracy ciągłej z ręcznym i automatycznym sterowaniem oraz w trybie pracy powtarzalnej. Wprowadzanie programu odbywa się za pomocą wymiennych tablic programowania oraz klawiatury. Wyniki obliczeń można wy-

...aż człowiek stworzył komputer (2)

prowadzić na ekran monitora, drukarkę rejestratory lub woltomierz. Do układu sterowania można podłączyć komputer cyfrowy, który może sterować zarówno blokami operacyjnymi, jak i programem liczenia. Komputer WAT 1001 jest przeznaczony do syntezy i analizy układów dynamicznych opisywanych równaniami różniczkowymi zwyczajnymi lub cząstkowymi oraz do obliczania układów równości algebraicznych.

W 1970 roku w Zakładach Wytwórczych Przyrządów Pomiarowych ERA w Warszawie opracowano model i prototyp minikomputera K-202. Był to pierwszy w Polsce komputer na układach scalonych. Głównym projektantem i konstruktorem był Jacek Karpiński.

W 1972 roku w Instytucie Maszyn Matematycznych PAN w Warszawie opracowano minikomputer MOMIK 8b. MOMIK 8b jest zbudowany na obwodach scalonych TTL i zawiera pamięć operacyjną o pojemności 2, 4, 8—16 Kslów 8-bitowych i czas cyklu równy 1,8 μs. Pracuje z szybkością od 150 tysięcy do 0,5 miliona operacji na sekundę. Wyposażony jest w kanał multipleksowy i kanał bezpośredniego dostępu. Do kanałów można maksymalnie podłączyć 32 urządzenia zewnętrzne. MOMIK 8b jest wykorzystywany jako jednostka centralna w minikomputerach rodziny minikomputerów MERA 300. W tym samym roku we Wrocławskich Zakładach Elektronicznych ELWRO zmontowano pierwszy w Polsce egzemplarz komputera Jednolitego Systemu o symbolu R-30, produkowany następnie od 1975 roku seryjnie już przez Centrum Komputerowych Systemów Automatyki i Pomiarów MERA ELWRO (dawne Wrocławskie Zakłady Elektroniczne ELWRO) jako komputer IS-1032. Komputer IS-1032 jest przeznaczony do przetwarzania danych oraz rozwiązywania obszernych i skomplikowanych zadań naukowo-technicznych. Pamięć ma pojemność od 128 do 1024 Kbajtów. IS-1032 jest komputerem wieloprogramowym i wielodostępnym. Ma jeden kanał multiplekserowy, umożliwiający podłączenie do 256 urządzeń wejściowo-wyjściowych, 3 kanały selektorowe, umożliwiające podłączenie do kanału 8 jednostek sterujących maksymalnie do 256 urządzeń wejściowo-wyjściowych oraz procesowo. Czas dostępu do pamięci operacyjnej wynosi 0,5 μs, a cykl podstawowy komputera wynosi 300 μs.

W 1973 roku rozpoczęto w Zakładach Wytwórczych Przyrządów Pomiarowych ERA produkcję minikomputera systemu MERA 300 i komputera biurowego MERA 302.

W 1976 roku na 48 Międzynarodowych Targach Poznańskich Centrum Komputerowych Systemów Automatyki i Pomiarów MERA-ELWRO wystawiło przewoźny Ośrodek Obliczeniowy, w którego wyposażeniu znajdował się komputer ODRA 1325, umieszczony w typowym kontenerze.

W 1981 roku powstało i ukonstytuowało się Polskie Towarzystwo Informatyczne.



Dłużej zatrzymaliśmy się na produkcji lat siedemdziesiątych — komputerów serii ODRA, gdyż były one i są nadal udaną konstrukcją sprzętową z bardzo bogatym oprogramowaniem.

A co przyniosły lata osiemdziesiąte — niestety przede wszystkim znaczne pogłębienie się luki technologicznej między krajami przodującymi w dziedzinie techniki komputerowej a Polską. W Polsce brak jest układów dużej skali integracji, mikroprocesorów, pamięci półprzewodnikowych, a w szczególności pamięci dyskowych. Aktualnie w Polsce produkuje się mikroprocesor MCY 7880 oparty na przestarzałym 8-bitowym mikroprocesorze Intel 8080, pamięci zewnętrzne na dyskach elastycznych, choć należy zauważyć, że produkcja ta nie zaspokaja wewnętrznego popytu i nie jest najwyższej jakości. W związku z tymi trudnościami produkcja mikrokomputerów nie odpowiada potrzebom. W ostatnich latach powstało wiele prywatnych zakładów (część z nich powstała w kooperacji z firmami zachodnimi), które zaczęły montować mikrokomputery z elementów produkowanych w innych krajach (najczęściej krajów Dalekiego Wschodu), bądź zajęły się tylko pośrednictwem w sprzedaży zestawów mikrokomputerowych. Z mikrokomputerów produkowanych obecnie czy w najbliższej przyszłości należy wymienić: MERITUM I, MERITUM II, COMPAN 8, MK 4501/2, KRAK-86, cała seria mikrokomputerów ELWRO, czy w końcu MAZOVIA 1016. Aktualnie nie wszystkie z wymienionych mikrokomputerów są produkowane, inne okazały się po prostu niewypałem. Zatrzymajmy się przy mikrokomputerach ELWRO 800 Junior i MAZOVIA 1016.

Model mikrokomputera osobistego ELWRO 800 Junior opracowano, z myślą o zastosowaniu go w polskich szkołach, w Instytucie Automatyki Politechniki Poznańskiej w Poznaniu oraz w Instytucie Komputerowych Systemów Automatyki i Pomiarów we Wrocławiu. ELWRO 800 Junior zawiera: 8-bitowy mikroprocesor Z80 A lub jego odpowiednik U880 produkowany w NRD, stronicowaną pamięć operacyjną, kolorową grafikę (obraz może być 16-kolorowy i może migotać), prosty generator dźwięku, pakiet lokalnej sieci komputerowej, sterownik pamięci na dyskietkach, łącze magnetofonowe oraz dodatkowe układy wejścia-wyjścia. Format zapisu informacji na taśmie i adresy urządzeń zewnętrznych są identyczne z mikrokomputerem ZX Spectrum, co zapewnia ELWRO 800 Junior dostęp do obszernej biblioteki oprogramowania. ELWRO 800 Junior jest wyposażony w dyskowy system operacyjny CP08, w pełni zgodny z systemem CP/M wersja 2.2. Można więc używać wielu programów narzędziowych i użytkowych. Do ELWRO 800 Junior przez wbudowane łącze równoległe o standardzie Centronics i szeregowe RS-232 C można podłączyć drukarkę znakową lub graficzną, manipulator drążkowy (joystick), pióro świetlne, myszkę i wiele innych urządzeń. Do końca 1986 roku planowano wyprodukować 500 sztuk tego komputera, do końca 1987 roku 5000 sztuk, a produkcja docelowa (tylko kiedy to nastąpi) miała wynosić 30 000 sztuk rocznie.

Drugim mikrokomputerem osobistym, na który warto zwrócić uwagę, jest długo oczekiwany, a montowany przez spółkę z ograniczoną odpowiedzialnością „Mikrokomputery" w pełni zgodny z IBM PC/XT mikrokomputer MAZOVIA 1016. MAZOVIA 1016 ma 16-bitowy mikroprocesor INTEL 8086 lub jego odpowiednik K1810WM86 produkowany w ZSRR, pamięć operacyjną o pojemności 256 Kbajtów (może być rozszerzona do 640 Kbajtów), pamięć stałą ROM o wielkości 48 Kbajtów. Podstawowym dyskowym systemem operacyjnym jest spolszczony MS DOS. Twórcy MAZOVII 1016 przystosowują do potrzeb polskiego użytkownika również inne systemy operacyjne jak: CP/M86 czy XENIX. Poza tym do sprzedaży przygotowano spolszczone oprogramowanie typu: bazy danych, programy kalkulacyjne, edytory tekstów itp. MAZOVIA 1016 ma tworzyć podstawowe wyposażenie wyższych uczelni oraz instytutów naukowo-badawczych. Do końca 1986 roku planowano wyprodukować 500 sztuk tych komputerów, a w 1987 roku około 5000 sztuk. Do chwili obecnej produkcji nie uruchomiono.

Należy zauważyć, że zarówno konstrukcja ELWRO 800 Junior, jak i MAZOVII 1016 są już przestarzałe w porównaniu ze sprzętem tej klasy produkowanym w przodujących w dziedzinie techniki komputerowej krajach świata.

J. S.

Komputerowe piractwo — niebezpieczeństwo XXI wieku

Jerzy RAJCH

Okazuje się, iż komputer — spełniający tak wiele pożytecznych czynności — może być też narzędziem przestępstwa. I to nie przestępstwa drobnego czy pospolitego, ale takiego, o którym przed paroma laty nikomu się nawet nie śniło. To wielki problem dalszego rozwoju komputeryzacji, który może zaważyć na obliczu wielu istotnych spraw świata w XXI wieku.

Przed paroma miesiącami niezwykle zaszokowany poczuł się premier Belgii, Wilfried Martens. Za pośrednictwem prasy dowiedział się bowiem, że w pamięci „Bistela", czyli rządowego systemu informacji komputerowej — od Belgian Information System by Telephon — zupełnie swobodnie porusza się komputerowy pirat.

Prasa nazwała pirata tajemniczym „X". Wyposażony w normalny komputer osobisty łatwo złamał wszystkie systemy zabezpieczające. Kluczem do tego były inicjały premiera: W. M. Dzięki temu mógł śledzić na ekranie wymianę korespondencji między członkami rządu, miał dostęp do najświeższych wiadomości agencyjnych, notowań giełdowych, prognoz gospodarczych i informacji dotyczących sytuacji ekonomicznej kraju. Ów „X" wprowadził też do systemu pewne poprawki blokujące tym dostęp do informacji niektórym ministrom. Na marginesie, oni sami tego wcale nie zauważyli! Być może sprawa nadal pozostałaby nie odkryta, gdyby pirat się nie znudził i swych możliwości nie zademonstrował dziennikarzom.

Premier Martens wprawdzie zapewnił, że komputer w pamięci nie posiadał żadnych tajnych informacji, ale opinia publiczna wcale nie była przekonana, czy bystry posiadacz komputera nie wszedł w posiadanie jakiejś tajemnicy państwowej. Rządowi specjaliści skwapliwie zapowiedzieli skonstruowanie takiego systemu zabezpieczającego, który uniemożliwi podobną sytuację. Widać zapomnieli, że analogiczny przypadek wydarzył się trzy lata wcześniej i wówczas też zapewniano o nowym niezawodnym zabezpieczeniu...

Dużo poważniejsze następstwa miało błahe z pozoru wydarzenie z lipca 1986 roku. Wówczas do komputera-matki firmy Philips za pomocą tajnego kodu „włamał się" ktoś nieupoważniony. Firma, by rozwikłać zagadkę, kto i po co to uczynił, postanowiła przeprowadzić śledztwo. Okazało się, że piraci nie zmienili programu ani części jego danych, ale zaatakowali cały system i do ich urządzeń wprowadzili wiele tajnych programów, które nie tylko modyfikowały programy Philipsa, ale mogły też je niszczyć.

W takiej sytuacji sprawę przejęła brygada finansowa paryskiej policji. Szybko odkryła piętnastocyfrowy kod piratów. Jego pięć pierwszych cyfr wskazywało, że pochodzą oni z RFN. Z kolei policja zachodnioniemiecka stwierdziła, iż podany kod identyfikacyjny należy do... uniwersytetu w Heidelbergu. Zaś władze uczelni nic nie wiedziały o takim wykorzystaniu ich komputera.

Ale i piraci popełnili błąd. Porozumiewając się za pomocą komputera między sobą użyli skrótu CERN. A to oznacza Centrum Badań i Poszukiwań Nuklearnych z siedzibą w Genewie. Prowadzący śledztwo poszli tym śladem i udali się do Szwajcarii. Tam czekała na nich niespodzianka. Fizycy pracujący w centrum wyjawili, że kłopoty z piratami mają już od ponad roku. A ich komputery Vax są połączone z 300 laboratoriami na całym świecie.

Stało się jasne, że piraci uzyskali dostęp do ścisłych tajemnic zachodniego świata nauk fizycznych. Obawy te potwierdził Interpol. Wreszcie przestali być tajemnicą. Tym „komputerowym postrachem" okazali się być członkowie oficjalnego klubu CCC — Chaos Computer Club, z siedzibą w Hamburgu i skupiającego prawie stu ludzi. W swym programie deklarują — ich status złożony jest i zarejestrowany w sądzie — że w informatycznym świecie panuje olbrzymi chaos w zakresie bezpieczeństwa. By tę tezę udowodnić, rozpracowywali najbardziej zawiłe programy.

Tu zaczęły się kłopoty i dyskusje między zachodnioniemieckim a francuskim wymiarem sprawiedliwości. W RFN nie istniały jeszcze przepisy prawne karzące za tego typu przestępstwa informatyczne. We Francji też nie, ale policja zakwalifikowała te czyny jako... włamanie i zniszczenie. Nie mogąc się dogadać Francuzi kontynuowali śledztwo na terenie swego kraju. W krótkim czasie wyłapali prawie dwustu piratów grasujących we francuskiej sieci komputerowej.

Tymczasem członkowie Chaos Computer Club rozpoczęli ofensywę i w ciągu pół roku „włamali się" do 135 komputerów ważnych instytucji naukowych — i nie tylko — świata. Między innymi rozszyfrowali w NASA program wahadłowca kosmicznego i dane związane z „gwiezdnymi wojnami".

Ale niemieckie prawodawstwo poszło z duchem czasu. Podczas rewizji w siedzibie klubu znaleziono kilka wręcz wybitnych pirackich programów oraz kilkusetstronicowy spis z kodami i hasłami wejściowymi do olbrzymiej ilości komputerów. Członkowie klubu bronili się, że dopięli tego , co zamierzali. Zwrócili uwagę na luki, które mogą zagrażać całej ludzkości. Bo skoro można zawładnąć w ten sposób tajnymi informacjami czy badaniami lub wpłynąć na wystrzelenie czy lot rakiety jądrowej — to jak tę sytuację inaczej nazwać?

Śledztwo trwa. Nie ma bowiem pewności, jak wielu podobnych, a bardzo zdolnych piratów, będzie chciało pobawić się w cudzym oprogramowaniu. Pewnych systemów zabezpieczających, jak widać, nie ma. To problem do jak najszybszego rozwikłania i przez prawników, i informatyków.

C H I Ń C Z Y K

Krzysztof POŹNIAK

Gra znana jest chyba wszystkim, ale dla porządku przypomnę jej zasady.

W grze bierze udział od 2 do 4 graczy. Zadaniem każdego z nich jest jak najszybsze dotarcie do własnego pola końcowego. O ruchu decyduje rzut kostką. Po rozpoczęciu rozgrywki piony graczy znajdują się w polach startowych. Wyrzucenie kostką 1 lub 6 oczek uprawnia jeden z pionów do wejścia na planszę. Piony poruszają się o tyle pól do przodu, ile oczek wyrzucono kostką. Nie wolno wymijać innych pionów (nawet własnych). Jeżeli ruch zostanie zakończony na polu, na którym stoi pion przeciwnika, to pion przeciwnika uważany jest za zbity i wraca na swoje pole startowe. Po wyrzuceniu 6 oczek i wykonaniu ruchu na planszy (a nie wyjścia z pola startowego) gracz nabywa prawo do powtórnego rzutu.

Po uruchomieniu programu na ekranie ukazuje się plansza atrybutów:

```
[illegible]
opracowal K.Pozniak    ©1988

  > GRACZ 1: ● AUTO
    GRACZ 2: + AUTO
    GRACZ 3: X AUTO
    GRACZ 4: □ AUTO
         OPCJE

<6>,<7> - wybor gracza
<k> - kolor piona
<z> - znak piona
<s> - typ sterowania
<t> - kolor tla planszy
<a> - kolor atramentu

<p> - PLAY
```

Gracz może wybrać znak, kolor i typ sterowania każdym pionem.

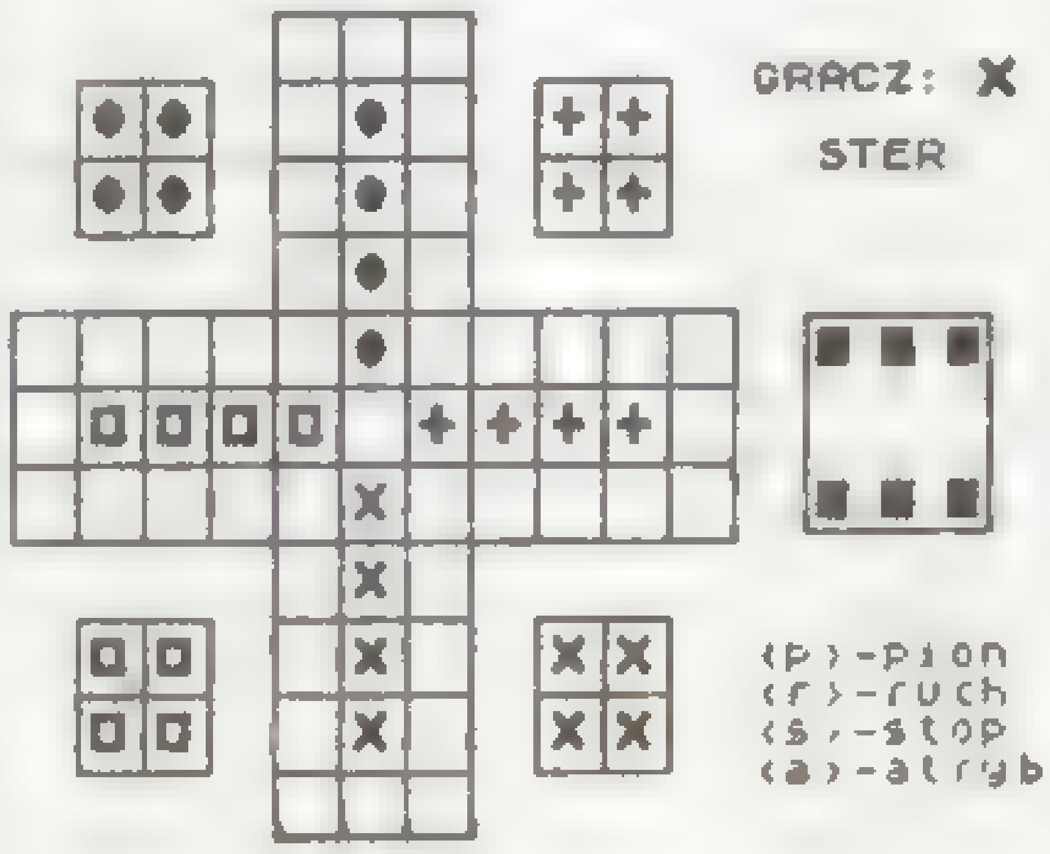
Typy sterowania:

AUTO — gracz automatyczny sterowany przez komputer,

STER — gracz sterowany przez użytkownika z klawiatury,

STOP — gracz wyłączony z rozgrywki.

Uruchomienie gry odbywa się po naciśnięciu klawisza **<p>**. Na ekranie ukazuje się plansza gry z początkowym ustawieniem pionów:



Gracze „rzucają" kostką po kolei. Jeśli gracz ma tryb **AUTO**, to posunięcie odbywa się automatycznie, jeśli tryb **STER**, to nad graczem kontrolę przejmuje użytkownik:

1) nad kostką pokazuje się napis: LOSOWANIE. Wciśnięcie dowolnego klawisza zatrzymuje ten proces i wyświetla wylosowaną liczbę oczek;

2) użytkownik w tym momencie może posłużyć się klawiszem: **<p>** — wybór piona (wybrany pion jest wyświetlany w trybie FLASH), **<r>** — wykonanie ruchu wybranym pionem.
UWAGA: klawiszem **<p>** mamy dostęp jedynie do pionów mogących wykonać ruch.

Jeśli żaden pion nie może się ruszyć, to na dole ekranu ukazuje się napis: RUCH NIEMOŻLIWY i gracz traci kolejkę.

Funkcje dodatkowe:

klawisz **<a>** — zmiana atrybutów pionów i planszy. Po zmianie gra jest kontynuowana.

klawisz **<s>** — ostateczne przerwanie gry.

Nie wyjaśnię dokładnie, jak porusza się pion w trybie **AUTO**. Zapewniam jednak, że nie jest to ruch całkowicie losowy (pomijając oczywiście rzut kostką). Ruch każdego piona określany jest pewnym priorytetem. Wybiera się ruch o najwyższym priorytecie. Tylko wtedy, gdy najwyższy priorytet posiada kilka pionów, wybór jednego z nich jest losowy. Zdradzę jedynie, że źródło decyzji ukryte jest w liniach 1200—1260.

```
   1 REM ****************************
   2 REM
   3 REM                       ©1988
   4 REM     Krzysztof Pozniak
   5 REM
   6 REM      C H I N C Z Y K
   7 REM     gra dla 1 - 4 osob
   8 REM
   9 REM ****************************
  10 REM [illegible]
  12 RESTORE 9900: DIM i(4,8): D
IM k$(7,7). DIM p(2,40): DIM b(4
,2,4): DIM s(4,2,4)
  14 LET ink=7. LET paper=0
  15 INK ink: PAPER paper: BORDE
R paper: CLS
  16 LET k=USR "a": FOR k=k TO k
+31: READ a: POKE k,a: NEXT k
  20 FOR m=1 TO 40: FOR n=1 TO 2
: READ a: LET p(n,m)=a: NEXT n:
NEXT m
  21 FOR m=1 TO 4: FOR n=1 TO 4:
 FOR k=1 TO 2: READ a: LET b(m,k
,n)=a: NEXT k: NEXT n: NEXT m
  22 FOR m=1 TO 4  FOR n=1 TO 4.
 FOR k=1 TO 2: READ a: LET s(m,k
,n)=a: NEXT k: NEXT n: NEXT m
  23 FOR k=1 TO 7: READ a$: LET
k$(k)=a$  NEXT k
  24 FOR k=1 TO 4: LET i(k,3)=7
 LET i(k,4)=CODE "●"+k-1: LET i(
k,6)=10*k-9  NEXT k
  25 LET k=USR "r": FOR k=k TO k
+7: FOR l=0 TO 24 STEP 8: READ a
: POKE k+l,a: NEXT l: NEXT k
  40 REM [illegible]
  41 FOR k=1 TO 4: LET i(k,5)=0
 NEXT k
  42 LET stop=0: LET auto=4
  45 GO SUB 8000
 100 REM [illegible]
 110 LET koniec=0: DIM a(4,4)  D
IM d(40): DIM k(4.4): FOR k=1 TO
 4: LET i(k,1)=4: LET i(k,2)=-4:
 NEXT k
 112 GO SUB 9600
 115 LET x=1
 120 IF i(x,5)<2 THEN PRINT AT 2
,23;"GRACZ: "; INK i(x,3);CHR$ i
(x,4);AT 4,25; INK ink,("AUTO" A
ND NOT i(x,5));("STER" AND i(x,5
))
 125 LET w=0: GO SUB 9000: LET r
uch=0
 130 IF i(x,5)=0 THEN GO SUB 100
0
 140 IF i(x,5)=1 THEN GO SUB 200
0
 150 IF x$="s" THEN GO TO 40
 160 IF x$="a" THEN LET pamx=x.
GO SUB 8000: GO SUB 9800: LET x=
pamx: GO TO 120
 170 IF ruch THEN GO SUB 2200. I
F w=6 THEN GO TO 125
 180 IF koniec=3-stop THEN GO SU
B 200: GO TO 40
 190 LET x=x+1: IF x>4 THEN GO T
O 115
 195 GO TO 120
 200 REM [illegible]
 210 CLS : PRINT AT 2,8, INVERSE
 1,"C H I N C Z Y K";AT 5,8; FLA
SH 1,"gra  zakonczona"
 220 FOR k=1 TO 4  LET q=i(k,5):
 IF q<=2 THEN GO TO 240
 230 PRINT AT 3+q-1,5,"miejsce "
;q-2;" - gracz : "; INK i(k,3);C
HR$ i(k,4)
 240 NEXT k: PRINT #0;TAB 4; INV
ERSE 1;"NACISNIJ DOWOLNY KLAWISZ
": PAUSE 1: PAUSE 0  RETURN
1000 REM [illegible]
1005 LET x$=INKEY$: IF x$="s" OR
 x$="a" THEN RETURN
1010 LET w=INT (6*RND)+1: GO SUB
 9000
1020 GO SUB 8300: IF NOT ruch TH
EN GO TO 2400
1030 GO SUB 1100
1040 LET k=INT (4*RND)+1: GO TO
2150
1100 REM [illegible]
1110 LET s=1
1120 FOR q=1 TO 4: IF i(q,8)>0 T
HEN GO SUB 1200
1130 NEXT q
1140 FOR q=1 TO 4. IF i(q,8)<s T
HEN LET i(q,8)=0
1150 NEXT q: RETURN
1200 REM [illegible]
1205 LET i(q,8)=1: LET e=i(q,7)
AND (i(q,7)<>i(x,6))
1210 IF e>0 THEN IF d(e)=0 THEN
LET e1=a(x,q): IF (x=1 AND e1>=2
1) OR (x=2 AND NOT (e1>=11 AND e
1<=30)) OR (x=3 AND e1<=20) OR (
x=4 AND e1>=11 AND e1<=30) THEN
LET i(q,8)=2: GO TO 1250
1212 IF e>0 THEN IF d(e)<>0 THEN
  IF i(d(e),2)<i(x,2) THEN IF e<
>1 AND e<>11 AND e<>21 AND e<>31
 THEN LET i(q,8)=3: GO TO 1250
1214 IF e>0 THEN IF d(e)<>0 THEN
 IF i(d(e),2)>=i(x,2) THEN LET i
(q,8)=4+(i(d(e),2)>=i(x,2)+1): G
O TO 1250
1216 IF e<0 THEN IF a(x,q)>0 THE
```

```
N LET i(q,8)=4: GO TO 1250
1218 IF e=0 THEN IF d(i(x,6))=0 AND w=1 THEN LET i(q,8)=3: GO TO 1250
1220 IF e=0 THEN IF d(i(x,6))>0 AND w=1 THEN IF i(d(i(x,6)),2)<i(x,2) THEN LET i(q,8)=4: GO TO 1250
1222 IF e=0 THEN IF d(i(x,6))>0 AND w=1 THEN IF i(d(i(x,6)),2)>=i(x,2) THEN LET i(q,8)=4: GO TO 1250
1224 IF e=0 THEN IF d(i(x,6))>0 AND w=6 THEN IF i(d(i(x,6)),2)>=i(x,2) THEN LET i(q,8)=3: GO TO 1250
1226 IF e>0 AND w=6 THEN LET i(q,8)=2: GO TO 1250
1228 IF e<0 THEN IF a(x,q)<0 THEN LET i(q,8)=3: GO TO 1250
1250 IF i(q,8)>s THEN LET s=i(q,8)
1260 RETURN
2000 REM [illegible]
2010 GO SUB 8200
2020 IF x$="s" OR x$="a" THEN RETURN
2030 GO SUB 9000
2035 GO SUB 8300: IF NOT ruch THEN GO TO 2400
2038 LET k=4: GO SUB 2120
2040 LET x$=INKEY$: IF x$="" THEN GO TO 2040
2045 BEEP .05,20
2050 IF x$="s" OR x$="a" THEN RETURN

2052 IF x$="p" THEN GO SUB 2100

2054 IF x$="r" THEN RETURN
2090 GO TO 2040
2100 REM [illegible]
2110 GO SUB 8400
2120 GO SUB 2150
2130 GO TO 8410
2150 REM [illegible]
2152 LET k=k+1-4*(k=4)
2154 IF i(k,8)=0 THEN GO TO 2152
2156 RETURN
2200 REM [illegible]
2210 IF a(x,k)=0 THEN LET e=d(i(k,7)): GO TO 2350
2220 IF a(x,k)>0 THEN LET d(a(x,k))=0
2230 IF a(x,k)<0 THEN LET k(x,-a(x,k))=0
2240 FOR l=1 TO w: GO SUB 8420: GO SUB 8370: GO SUB 8430: NEXT l
2260 IF a(x,k)>0 THEN LET e=d(a(x,k)): LET d(a(x,k))=x: IF e<>0 THEN GO SUB 2300
2270 IF a(x,k)<0 THEN LET k(x,-a(x,k))=1: IF a(x,k)=i(x,2) THEN LET i(x,2)=i(x,2)+1. IF NOT i(x,2) THEN LET koniec=koniec+1 LET i(x,5)=koniec+2
2290 RETURN
2300 REM [illegible]
2310 LET i(e,1)=i(e,1)+1: LET pamx=x: LET pamk=k
2320 FOR s=1 TO 4: IF a(e,s)=a(x,k) THEN LET x=e: LET k=s: LET a(x,k)=0: GO SUB 8430 LET x=pamx: LET k=pamk: RETURN
2330 NEXT s: STOP
2350 REM [illegible]
2352 LET w=1: GO SUB 8420: LET a(x,k)=i(x,6): LET i(x,1)=i(x,1)-1: LET d(a(x,k))=x: GO SUB 8430: IF e<>0 THEN GO SUB 2300
2354 RETURN
2400 REM [illegible]

2410 PRINT #0,AT 1,8, INK ink; PAPER paper; FLASH 1;"RUCH NIEMOZLIWY": BEEP .03,20: PAUSE 50: PRINT #0;AT 1,8;"                ": RETURN
8000 REM [illegible]
8010 PAPER paper: INK ink: BORDER paper: CLS : PRINT TAB 8; INVERSE 1;"C H I N C Z Y K"
8015 PRINT AT 1,16;",";AT 2,1;"opracowal K.Pozniak   ©1988"
8020 FOR x=1 TO 4: GO SUB 8100.' NEXT x: LET x=1
8030 PRINT AT 12,12;"OPCJE:"
8032 PRINT AT 14,4;"<6>,<7> - wybor gracza";TAB 4;"<k> - kolor piona";TAB 4;"<z> - znak piona";TAB 4;"<s> - typ sterowania";TAB 4;"<t> - kolor tla planszy";TAB 4;"<a> - kolor atramentu";TAB 4;"<p> - PLAY"
8035 PRINT AT 2*x+2.8;">"
8040 LET x$=INKEY$: IF x$="" THEN GO TO 8040
8045 BEEP .05,20
8050 IF x$="6" THEN PRINT AT 2*x+2,6;" ": LET x=x+1-4*(x=4): GO TO 8035
8051 IF x$="7" THEN PRINT AT 2*x+2,6;" ": LET x=x-1+4*(x=1): GO TO 8035
8052 IF x$="k" THEN LET i(x,3)=i(x,3)+1-8*(i(x,3)=7): GO SUB 8100: GO TO 8040
8053 IF x$="z" THEN LET i(x,4)=i(x,4)+1-4*(CHR$ i(x,4)="■"): GO SUB 8100: GO TO 8040
8054 IF x$="s" THEN LET q=i(x,5): LET stop=stop-(q=2)+(q=1): LET auto=auto-(q=0)+(q=2) LET i(x,5)=q+1*(q<=2)-3*(q=2): GO SUB 8100: GO TO 8040
8055 IF x$="t" THEN LET paper=paper+1-8*(paper=7): GO TO 8010

8056 IF x$="a" THEN LET ink=ink+1-8*(ink=7): GO TO 8010
8057 IF x$="p" AND stop>=3 THEN BEEP .1,20: BEEP .1,30 PRINT #0;TAB 2, FLASH 1,"TA ROZGRYWKA NIE MA SENSU !": PAUSE 100: INPUT ""
8058 IF x$="p" AND stop<3 THEN RETURN
8060 BEEP .1,30: GO TO 8040
8100 REM [illegible]
8110 PRINT AT 2*x+2,8;"GRACZ ";x;": "; INK i(x,3);CHR$ i(x,4); INK ink;(" AUTO" AND (i(x,5)=0)),(" STER" AND (i(x,5)=1));(" STOP" AND (i(x,5)>=2))
8120 RETURN
8200 REM [illegible]
8210 PRINT AT 7,23; FLASH 1;"LOSOWANIE"
8220 LET w=RND: LET x$=INKEY$: IF x$="" THEN GO TO 8220


8230 LET w=1+INT (6*w): PRINT AT 7,23;"         ": RETURN
8300 REM [illegible]
8310 FOR k=1 TO 4
8320 IF a(x,k)<i(x,2) THEN GO TO 8328
8322 IF a(x,k)=0 AND (w=6 OR w=1) AND d(i(x,6))<>x THEN LET i(k,7)=i(x,6): GO TO 8330
8324 IF a(x,k)<>0 THEN GO SUB 8350: IF i(k,7)<>a(x,k) THEN GO TO 8330
8328 LET i(k,7)=a(x,k): LET i(k,8)=0: GO TO 8335
8330 LET i(k,8)=1: LET ruch=1
8335 NEXT k: RETURN
8350 REM [illegible]
8352 LET pam=a(x,k): FOR q=1 TO w: GO SUB 8360: IF a(x,k)<>pam THEN NEXT q: LET i(k,7)=a(x,k): LET a(x,k)=pam: RETURN
8354 LET i(k,7)=pam: LET a(x,k)=pam: RETURN
8360 REM [illegible]
8362 GO SUB 8370
8364 IF a(x,k)<0 AND a(x,k)>=-4 THEN IF NOT k(x,-a(x,k)) THEN RETURN
8365 IF a(x,k)>0 THEN IF d(a(x,k))=0 OR (q=w AND d(a(x,k))<>x) THEN RETURN
8366 LET a(x,k)=pam: RETURN
8370 REM [illegible]
8372 IF a(x,k)=40 AND x<>1 THEN LET a(x,k)=1: RETURN
8373 IF a(x,k)+1=i(x,6)+40*(x=1) THEN LET a(x,k)=-1: RETURN
8374 IF a(x,k)>0 THEN LET a(x,k)=a(x,k)+1: RETURN
8375 IF a(x,k)<0 THEN LET a(x,k)=a(x,k)-1: RETURN
8400 REM [illegible]
8402 LET q=0: LET v=1: GO TO 8450


8410 REM [illegible]
8412 LET q=1: LET v=1: GO TO 8450
8420 REM [illegible]
8422 LET q=0: LET v=0 GO TO 8450
8430 REM [illegible]
8432 LET q=0: LET v=1
8450 REM procedura wykonawcza
8452 IF a(x,k)<0 THEN PRINT AT b(x,1,-a(x,k)),b(x,2,-a(x,k)); INVERSE NOT v; INK i(x,3); FLASH q;CHR$ i(x,4)
8454 IF a(x,k)=0 THEN PRINT AT s(x,1,k),s(x,2,k); INK i(x,3); FLASH q;(" " AND NOT v);(CHR$ i(x,4) AND v)
8456 IF a(x,k)>0 THEN PRINT AT p(1,a(x,k)),p(2,a(x,k)); INK i(x,3); FLASH q;(" " AND NOT v);(CHR$ i(x,4) AND v)
8458 RETURN

9000 REM [illegible]
9020 PRINT AT 9,25;k$(w+1,1);AT 9,27;k$(w+1,2);AT 9,29;k$(w+1,3);AT 11,27;k$(w+1,4);AT 13,25;k$(w+1,5);AT 13,27;k$(w+1,6);AT 13,29;k$(w+1,7)
9030 RETURN
9800 REM [illegible]
9808 CLS
9810 FOR k=0 TO 3: PLOT 68,171-16*k: DRAW 48,0: NEXT k
9811 FOR k=0 TO 3: PLOT 4,107-16*k: DRAW 176,0: NEXT k
9812 FOR k=0 TO 2: PLOT 68,43-16*k: DRAW 48,0: NEXT k
9813 PRINT #0;AT 0,0; INK ink; PAPER paper;"└┴┴┘"
9814 FOR k=0 TO 3: PLOT 4+k*16,59: DRAW 0,48: NEXT k
9815 FOR k=0 TO 3: PLOT 68+16*k,0: DRAW 0,171: NEXT k
9816 FOR k=0 TO 3: PLOT 132+k*16,59: DRAW 0,48: NEXT k
9820 FOR k=0 TO 2: PLOT 132+k*16,123: DRAW 0,32: NEXT k: FOR k=0 TO 2: PLOT 132,123+k*16: DRAW 32,0: NEXT k
9821 FOR k=0 TO 2: PLOT 132+k*16,11: DRAW 0,32: NEXT k: FOR k=0 TO 2: PLOT 132,11+k*16: DRAW 32,0: NEXT k
9822 FOR k=0 TO 2: PLOT 20+k*16,11: DRAW 0,32: NEXT k: FOR k=0 TO 2: PLOT 20,11+k*16: DRAW 32,0: NEXT k
9823 FOR k=0 TO 2: PLOT 20+k*16,123: DRAW 0,32: NEXT k: FOR k=0 TO 2: PLOT 20,123+k*16: DRAW 32,0: NEXT k
9830 PLOT 197,108: DRAW 45,0: DRAW 0,-45: DRAW -45,0. DRAW 0,45
9832 FOR x=1 TO 4: GO SUB 9850: NEXT x: GO SUB 9840
9833 IF auto=4 THEN PRINT AT 0,23; INVERSE 1;"D E M O"
9834 PRINT AT 17,23;"<p>-pion";AT 18,23;"<r>-ruch";AT 19,23;"<s>-stop";AT 20,23,"<a>-atryb"
9835 RETURN
9840 REM [illegible]
9841 FOR x=1 TO 4. FOR k=1 TO 4: GO SUB 8430: NEXT k: NEXT x
9843 RETURN
9850 REM [illegible]
9851 FOR k=1 TO 4. PRINT AT b(x,1,k),b(x,2,k); PAPER i(x,3); INK paper,CHR$ i(x,4) NEXT k
9852 RETURN
9900 REM [illegible]
9901 DATA BIN 00011000
9902 DATA BIN 01111110
9903 DATA BIN 01111110
9904 DATA BIN 11111111
9905 DATA BIN 11111111
9906 DATA BIN 01111110
9907 DATA BIN 01111110


9908 DATA BIN 00011000
9910 REM [illegible]
9911 DATA BIN 00011000
9912 DATA BIN 00011000
9913 DATA BIN 00011000
9914 DATA BIN 11111111
9915 DATA BIN 11111111
9916 DATA BIN 00011000
9917 DATA BIN 00011000
9918 DATA BIN 00011000
9920 REM [illegible]
9921 DATA BIN 11000011
9922 DATA BIN 11100111
9923 DATA BIN 01111110
9924 DATA BIN 00111100
9925 DATA BIN 00111100
9926 DATA BIN 01111110
9927 DATA BIN 11100111
9928 DATA BIN 11000011
9930 REM [illegible]
9931 DATA BIN 11111111
9932 DATA BIN 11111111

9933 DATA BIN 11000011
9934 DATA BIN 11000011
9935 DATA BIN 11000011
9936 DATA BIN 11000011
9937 DATA BIN 11111111
9938 DATA BIN 11111111
9950 REM [illegible]
9951 DATA 1,11,1,13,3,13,5,13,7,13,9,13,9,15,9,17,9,19,9,21,11,21,13,21,13,19,13,17,13,15,13,13,15,13,17,13,19,13,21,13,21,11
9952 DATA 21,9,19,9,17,9,15,9,13,9,13,7,13,5,13,3,13,1,11,1,9,1,9,3,9,5,9,7,9,9,7,9,5,9,3,9,1,9
9955 REM [illegible]
9956 DATA 3,11,5,11,7,11,9,11
9957 DATA 11,19,11,17,11,15,11,13
9958 DATA 19,11,17,11,15,11,13,11
9959 DATA 11,3,11,5,11,7,11,9
9960 REM [illegible]


9961 DATA 3,3,3,5,5,3,5,5
9962 DATA 3,17,3,19,5,17,5,19
9963 DATA 17,17,17,19,19,17,19,19
9964 DATA 17,3,17,5,19,3,19,5
9970 REM [illegible]
9971 DATA "[illegible]"
9972 DATA "[illegible]"
9973 DATA "[illegible]"
9974 DATA "[illegible]"
9975 DATA "[illegible]"
9976 DATA "[illegible]"
9977 DATA "[illegible]"
9980 REM [illegible]
9981 DATA BIN 00000000,BIN 00001000,BIN 00001000,BIN 00001000
9982 DATA BIN 00000000,BIN 00001000,BIN 00001000,BIN 00001000
9983 DATA BIN 00000000,BIN 00001000,BIN 00001000,BIN 00001000
9984 DATA BIN 00000000,BIN 00001000.BIN 00001000.BIN 00001000

9985 DATA BIN 11111111,BIN 00001111,BIN 11111111,BIN 11111000
9986 DATA BIN 00000000,BIN 00000000,BIN 00000000,BIN 00000000
9987 DATA BIN 00000000,BIN 00000000,BIN 00000000,BIN 00000000
9988 DATA BIN 00000000,BIN 00000000,BIN 00000000,BIN 00000000
9990 REM [illegible]
9991 CLEAR : SAVE "chinczyk" LINE 1: VERIFY "chinczyk"
```

Jak płynnie i szybko przesuwać...

Danuta KWASIŻUR, Mieczysław SKONIECZNY

BASIC mikrokomputera AMSTRAD pozwala na „płynny" ruch dowolnych obiektów na ekranie dzięki odpowiedniemu zastosowaniu instrukcji TAG, MOVE oraz właściwie zorganizowanej pętli FOR. To narzędzie satysfakcjonuje użytkownika przy małych wymiarach obiektu (powierzchnia kilku znaków). Im większy obiekt tym satysfakcja coraz mniejsza. „Płynność" okazuje się ruchem skokowym (każdy wiersz jest przesuwany osobno), co daje się zauważyć na ekranie, a i szybkość takiego ruchu pozostawia wiele do życzenia.

Mankamenty te można usunąć stosując odpowiednio sparametryzowane podprogramy w języku wewnętrznym. Chcemy zaprezentować i omówić 4 podprogramy, które przesuwają dowolnie zadeklarowany obiekt w 4 kierunkach:

— do góry (ruch pionowy)
— na dół (ruch pionowy)
— w lewo (ruch poziomy)
— w prawo (ruch poziomy)

Ilustracją zastosowania ww. podprogramów jest program demonstracyjny, który steruje ruchem obiektu dość dużego (6x9 znaków) po ekranie mikrokomputera w 4 kierunkach. Oczywiście wymiar obiektu jest zmienny, co nie ogranicza stosowania podprogramów. Dla czytelników, którzy pokuszą się o wprowadzenie programu demonstracyjnego z klawiatury do pamięci mikrokomputera i zainicjują jego działanie, kształt ruchomego obiektu będzie niespodzianką (patrz instrukcje 50—150)

W wierszu 40 określono wymiary ruchomego obiektu graficznego

X1 — szerokość obiektu (w znakach)
Y1 — wysokość obiektu (w znakach)

W naszym wypadku X1=9, Y1=6, a więc obiekt ma powierzchnię 54 znaków. Deklarowany obszar, który zostanie przesunięty, jest traktowany jak prostokąt. Poszczególne fragmenty programu realizują następujące czynności:

210—270 — wywołanie podprogramu na ruch poziomy w prawo. Zmienne x,y w wierszu 240 określają współrzędne znakowe prawego, górnego rogu obiektu.

280—360 — wywołanie podprogramu na ruch pionowy w górę. Zmienne x,y w wierszu 310 określają współrzędne znakowe lewego, górnego rogu obiektu.

410—480 — wywołanie podprogramu na ruch poziomy w lewo. Zmienne x,y w wierszu 370 określają współrzędne znakowe lewego, górnego rogu obiektu.

500—580 — wywołanie podprogramu na ruch pionowy w dół. Zmienne x,y w wierszu 390 określają współrzędne znakowe lewego, dolnego rogu obiektu.

Współrzędne, którego rogu należy podać dla odpowiedniego ruchu, pokazuje poniższy rysunek (strzałkami oznaczono kierunek ruchu).



Zmienna L we wszystkich wypadkach określa, o ile znaków należy przesunąć obiekt

Pomocniczą rolę odgrywa podprogram:

```
10 MODE 1:INK 0.0
20 SPEED INK 20,20:INK 1.26.6:INK 2.14.26
30 PAPER 0:BORDER 0:CLS:PEN 1
40 y1=6:x1=9
50 LOCATE 7,9:PRINT CHR$(213);CHR$(143);CHR$(143):CHR$(215)
60 LOCATE 8.10:PRINT CHR$(213):CHR$(143);CHR$(143):CHR$(215)
70 LOCATE 9,11:PRINT CHR$(213):CHR$(143):CHR$(143):CHR$(215)
80 LOCATE 9.12:PRINT CHR$(214):CHR$(143):CHR$(143):CHR$(212)
90 LOCATE 8,13:PRINT CHR$(214);CHR$(143);CHR$(143);CHR$(212)
100 LOCATE 7,14:PRINT CHR$(214):CHR$(143);CHR$(143);CHR$(212)
110 GRAPHICS PEN 1
120 PLOT 192,222:DRAWR 46,46:PLOT 192,222:DRAWR 46,-46
130 PEN 2:LOCATE 16.12:PRINT"INFORMATYKA"
140 LOCATE 16,13:PRINT"KOMPUTERY"
150 LOCATE 16,14:PRINT"SYSTEMY"
160 FOR t=1 TO 500:NEXT
170 x=7:y=9:l=6:GOSUB 1020
180 GOSUB 410
190 x=1:v=14:l=11:GOSUB 1020
200 GOSUB 500
210 GOSUB 730
220 POKE &7F78.y1:POKE &7F81,2*x1
230 POKE &7F8D,2*x1-1:POKE &7F9A,79+2*x1
240 x=9:y=20:l=31:GOSUB 1020
250 POKE &7F7B,w1:FOR Z=w2+1 TO 2*l
260 FRAME:POKE &7F7A,Z:CALL &7F76
270 FOR t=1 TO 5:NEXT:NEXT
280 GOSUB 590
290 POKE &7F01,y1*8:POKE &7F5A,2*x1
300 POKE &7F60,2*x1
310 x=32:y=20:l=19:GOSUB 1020
320 POKE &7F74,w2:POKE &7F75,w1
330 FOR B=1 TO l
340 FOR C=1 TO 8
350 POKE &7F02,c:CALL &7F00
360 FOR t=1 TO 2:NEXT:NEXT:FRAME:NEXT
370 x=32:y=1:l=31:GOSUB 1020
380 GOSUB 410
390 X=1:Y=6:L=19:GOSUB 1020:GOSUB 500
400 GOTO 210
410 GOSUB 950
420 POKE &7F78,y1:POKE &7F81,2*x1
430 POKE &7F8D,257-2*x1:POKE &7F9A,81-2*x1
440 POKE &7F7B,w1
450 FOR Z=w2 TO W2-2*l+1 STEP -1
460 POKE &7F7A,Z:CALL &7F76
470 FRAME:FOR t=1 TO 5:NEXT:NEXT
480 RETURN
490 X=1:Y=6:L=19:GOSUB 1020
500 GOSUB 800
510 POKE &7F01,y1*8:POKE &7F5A.2*x1
520 POKE &7F60,2*x1
530 POKE &7F74,W2:POKE &7F75,W1+56
540 FOR B=1 TO L
550 FOR C=1 TO 8
560 POKE &7F02,c:CALL &7F00
570 FOR t=1 TO 2:NEXT:NEXT:FRAME:NEXT
580 RETURN
```

który przelicza współrzędne znakowe x,y na adresy tych znaków. Oczywiście w mikrokomputerze AMSTRAD CPC 6128 adres początkowy ekranu wynosi 49152.

Podprogramy na ruch pionowy

Podprogram na ruch w górę ma następującą postać:

```
590 RESTORE 610:FOR i=&7F00 TO &7F73
600 READ v:POKE i,v:NEXT
610 DATA 1,0,1,42,116,127,16,93
620 DATA 229,124,198,56,103,17,80,0,237,82,84
630 DATA 93,225,197,229,213,205,95,127,209,98,107
640 DATA 34,116,127,225,193,13,40,46,6,7,84,93,124
650 DATA 198,8,103,197,229,213
660 DATA 205,95,127,209,225,193,13,40,26,16,236
670 DATA 84,93,124,214,56,103,213,17,80,0,237,90
680 DATA 209,197,229,213,205,95,127,209,225,193
690 DATA 24,207,54,0,84,93,19,1,0,0,237,176,201
700 DATA 1,0,0,237,176,201,124,214,8,87
710 DATA 93,229,98,107,34,116,127,225,24,187,201
720 RETURN
```

Podprogram na ruch w dół ma następującą postać:

```
730 RESTORE 750:FOR i=&7F76 TO &7FA1
740 READ v:POKE i,v:NEXT
750 DATA 1,8,0,33,39,192,84
760 DATA 93,19,197,1,0,0,237,184,193,35,54,0
770 DATA 13,40,6,17,0,8,25,24,234,5,40,12,124
780 DATA 214,56,103,17,0,0,25,14,8,24,219,201
790 RETURN
```

Jedno wywołanie podprogramu daje efekt na ekranie przesunięcia obiektu o jedną linię (1/8 znaku) odpowiednio w górę lub w dół. Adres początkowy (ustalony po wyjściu z podprogramu 1020) obiektu jest ładowany do adresów & 7F74 i & 7F75 za pomocą instrukcji POKE (wiersze 320 i 530 programu demonstracyjnego). Jak widać w wierszach 800 i 590 oba podprogramy są wymiennie ładowane w ten sam obszar pamięci określony adresami & 7F00 — & 7F73.

Podprogramy na ruch poziomy

Podprogram na ruch w prawo ma postać:

```
800 RESTORE 820:FOR i=&7F00 TO &7F73
810 READ v:POKE i,v:NEXT
820 DATA 1,0,1,42,116,127,16,93,229
830 DATA 124,214,56,103,17,80,0,237,90,84
840 DATA 93,225,197,229,213,205,95,127,209
850 DATA 98,107,34,116,127,225,193,13,40,46
860 DATA 6,7,84,93,124,214,8,103,197,229,213
870 DATA 205,95,127,209,225,193,13,40,26,16,236
880 DATA 84,93,124,198,56,103,213,17,80,0,237,82
890 DATA 209,197,229,213,205,95,127
900 DATA 209,225,193,24,207
910 DATA 54,0,84,93,19,1,0,0,237,176,201
920 DATA 1,0,0,237,176,201,124,198,8,87
930 DATA 93,229,98,107,34,116,127,225,24,187,201
940 RETURN
```

Podprogram na ruch w lewo ma postać:

```
950 RESTORE 970:FOR i=&7F76 TO &7FA1
960 READ v:POKE i,v:NEXT
970 DATA 1,8,0,33,36,192,84,93
980 DATA 27,197,1,0,0,237,176,193,43,54,0
990 DATA 13,40,6,17,0,7,25,24,234,5,40,12,124
1000 DATA 214,56,103,17,0,0,25,14,8,24,219,201
1010 RETURN
```

Jedno wywołanie podprogramu daje efekt na ekranie przesunięcia obiektu o pół znaku odpowiednio w lewo lub w prawo. Adres początkowy jest analogicznie ustalony w podprogramie 1020 i ładowany za pomocą instrukcji POKE pod odpowiednie adresy. Oba podprogramy są wymiennie ładowane w ten sam obszar pamięci (& 7F76 — &7FA1)

Wszystkie opisane wyżej podprogramy przesuwania wykorzystują następujące rejestry:

bc — przechowuje liczbę, określającą wielkość obiektu;
hl — przechowuje adresy obiektu;
de — roboczy do obliczania adresów.

Wywołanie odpowiedniego podprogramu w języku wewnętrznym w programie użytkowym ma postać:

CALL adres startowy

gdzie adres startowy jest początkiem obszaru rozmieszczania podprogramu w pamięci (odpowiednio & 7F00 — ruch pionowy i & 7F76 — ruch poziomy).

Odpowiednie efekty na ekranie otrzymuje się przez umiejętne zorganizowanie pętli FOR wykorzystującej parametry w1 i w2 (otrzymane z podprogramem 1020) oraz L.

```
1020 A=(Y-1)*80+(x-1)*2
1030 W1=INT(a/256)
1040 W2=a-W1*256
1050 w1=192+w1
1060 RETURN
```

Efekt spowolnienia ruchu może użytkownik uzyskać przez odpowiednie zastosowanie pętli FOR w postaci:
FOR i=1 to T: NEXT
gdzie T jest uzależniony od wymogów ruchu.

Na zakończenie życzymy zainteresowanym czytelnikom cierpliwości we wprowadzaniu programów oraz ciekawych efektów przy formowaniu własnych, odpowiednio zdefiniowanych obiektów.

Danuta KWASIŻUR
Mieczysław SKONIECZNY

ZX-81

Rakieta w cel

W rejon bronionego przez ciebie odcinka pola walki nadleciał nieprzyjacielski helikopter. Ustaw więc odpowiednio kąt wyrzutni, np. 40°, 70° itp. i odpal rakietę klawiszem „ENTER". Z tej walki wyjdzie cało tylko jeden, gdyż w wypadku, kiedy nie trafisz w helikopter, zrzuci on na Twoją pozycję bombę.

J. J.

```
  10 PRINT AT 20,27;"▟"
  15 INPUT B
  20 LET C=27*COS(PI*B/180)*COS(
PI*B/180)
  25 FOR A=1 TO 19
  30 GOSUB 100
  35 LET N=27-INT(SQR(A/19)*C)
  37 A=1 THEN GOTO 50
  40 PRINT AT 21-A,27-INT(SQR((A
-1)/19)*C);" "
  45 PRINT AT 19,27;" "
  50 PRINT AT 20-A,N;"."
  55 IF A=19 AND A=N THEN GOTO 1
20
  60 IF A=19 THEN GOTO 70
  65 NEXT A
  70 FOR A=20 TO 27
  75 GOSUB 100
 76 IF A=27 THEN GOTO 130
 80 NEXT A
 100 PRINT AT 0,A-2;"   "; AT 1
,A-3;"    "
 105 PRINT AT 0,A-1;" - "; AT 1
,A-2;"--▣"
 110 PRINT AT 0,A-1;"-▀-"; AT 1
,A-2;"*-▣"
 115 RETURN
 120 PRINT AT 0,18;"   "
 125 FOR A=1 TO 20
 127 PRINT AT A-1,17;"   "; AT
A,17;"***"
 128 IF A=20 THEN GOTO 140
 129 NEXT A
 130 FOR A=3 TO 20
 132 PRINT AT A-1,27;" "; AT A,
27;"*"
 134 IF A=20 THEN STOP
 135 NEXT A
 140 PRINT AT 5,10;"RAKIETA TRA
FILA"
```

PROGRAM 64

ZX-81

POLE MINOWE

Twoim zadaniem jest przejście przez pole minowe do punktu oznaczonego na ekranie literą „X". Po każdym ruchu gwiazdki wskutek drgań, następuje eksplozja dziesięciu min oznaczonych literą „M". Jeżeli któraś z min wybuchnie w miejscu znajdowania się gwiazdki, następuje zakończenie gry, gdy nie, możesz zabawę kontynuować. Ruch gwiazdce nadaje się za pomocą klawiszy: 5, 6, 7, 8. Gdy przestanie zadowalać cię stopień trudności, możesz zmienić ilość wybuchających min. Dokonuje się tego przez zwiększenie lub zmniejszenie cyfry 10 w wierszu o etykiecie 25.

J. J.

```
  1 REM ************************
  2 REM *     POLE MINOWE      *
  3 REM *        ZX-81         *
  4 REM ************************
  5 LET A=7
  10 LET B=7
  15 PRINT AT 7, 14;"X"
  20 PRINT AT A, B;"*"
  25 FOR I=1 TO 10
  30 LET X=INT(RND*14+1)
  35 LET Y=INT(RND*14+1)
  40 IF (X=7 AND Y=14) OR X=7 A
ND Y=0) THEN GOTO 30
  45 PRINT AT X, Y;"M"
  50 IF X=A AND Y=B THEN GOTO 1
10
  55 NEXT I
  60 INPUT A$
  65 PRINT AT A, B;" "
  70 IF A$="8" THEN LET B=B+1
  75 IF A$="5" THEN LET B=B-1
  80 IF A$="6" THEN LET A=A+1
  85 IF A$="7" THEN LET A=A-1
  90 IF A=7 AND B=14 THEN GOTO
100
  95 GOTO 20
  100 PRINT "WYGRALES"
  105 STOP
  110 PRINT "JUZ PO TOBIE !"
```

ATARI

Drugie prawo dynamiki

J. GOLLA

```
0 REM ***  DRUGIE PRAWO DYNAMIKI
 ***
1 DIM A$(200):A$="---------------
-------"
2 A$(21,100)="....DRUGA ZASADA..
.........DYNAMIKI......       -------
-------'

6 GRAPHICS 2:POKE 752,1:POKE 710
,0 POSITION 0,3
7 FOR T=1 TO 76:SOUND 0,RND(0)*5
0,12,15:? #6;A$(T,T);:SOUND 0,0,
0,0
8 FOR C=1 TO 1500 STEP T:NEXT C:
NEXT T
9 ? "        NACISNIJ DOWOLNY KLAW
ISZ"
10 OPEN #1,4,0,"K:":GET #1,S
20 GRAPHICS 1+16:? #6,"     KLAWI
SZ NR 1     CZESC TEORETYCZNA":?
 #6," "
30 ? #6;"    KLAWISZ NR 2      CZ
ESC SYMULACYJNA":TRAP 20
40 GET #1,S:S=S-48:ON S GOTO 100
,212
50 GOTO 40
100 REM ___________________
105 REM CZESC TEORETYCZNA
110 REM ___________________
115 GRAPHICS 0:POKE 710,0:POKE 8
2,1:POKE 752,1:POKE 709,0
120 ? CHR$(125):? CHR$(127);"  D
RUGA ZASADA DYNAMIKI";CHR$(29)
125 ? "Jezeli na cialo dziala st
ala sila,to"
130 ? "porusza sie ono wzgledem
inercjalnego  ukladu",
135 ? " odniesienia ruchem Jedno
stajnie zmiennym z przyspie";
140 ? "szeniem wprost pro- porcj
onalnym do sily,";
145 ? "a odwrotnie pro-  porcjon
alnym do masy ciala."
150 ? "Kierunek i zwrot wektora
przy";
155 ? "spieszeniasa zgodne z kie
runkiem i";:? " zwrotem wekto-";
160 ? "ra sily.":? :? :? :? CHR$
(127);"NACISNIJ DOWOLNY KLAWISZ"
165 FOR T=0 TO 10 STEP 0.3:POKE
709,T:NEXT T:GET #1,S
170 ? CHR$(125):? CHR$(127);" DR
UGA ZASADA DYNAMIKI"
175 ? CHR$(127);"SILA=MASA*PRZYS
PIESZENIE"
176 ? CHR$(127);CHR$(127);"_    _

"
180 ? CHR$(127);CHR$(127);"F=m*a
";CHR$(29)
185 ? CHR$(127);"PRZYSPIESZENIE=
SILA/MASA";CHR$(29)
186 ? CHR$(127);CHR$(127);"_ _"
190 ? CHR$(127);CHR$(127);"a=F/m
";CHR$(29)
195 ? "Przebyta DROGA jest wpros
t proporcjo-  nalna do";
200 ? " przyspieszenia w stalych
 prze-dzialach czasowych."
205 ? :? :? :? :?
210 ? CHR$(127);"NACISNIJ DOWOLN
Y KLAWISZ":GET #1,S
212 GRAPHICS 2
215 ? #6;"przyspieszenie jest
 proporcjonalne          do sily"
:REM Tekst napisz w INVERSE VIDE
O
220 ? #6;"PRZYSPIESZENIE JEST OD
```

```
WROTNIE PROPORCJO-    NALNE DO M
ASY"
225 POKE 710,0:POKE 752,1
230 ? "   DOWOLNY KLAWISZ:WYBOR
SYMULACJI."
235 ? CHR$(127);"RETURN TO START
 SYMULACJI."
240 GET #1,S:IF S=155 THEN ON A
GOTO 1000,2000
245 IF L=0 THEN POSITION 2,7:? #
6;"masa jest stala":REM Tekst na
pisz w INVERSE VIDEO:A=2:GOSUB 2
70
250 L=-1-L
255 IF L=0 THEN POSITION 2,7:? #
6;"SILA JEST STALA":A=1:GOSUB 26
5
260 GOTO 240
265 POKE 708,15:POKE 711,3:RETUR
N
270 POKE 711,15:POKE 708,3:RETUR
N
1000 REM ________________
1010 REM SILA JEST STALA
1020 REM ________________
1040 GRAPHICS 0:POKE 710,0:POSIT
ION 12,2:? "SILA JEST STALA";CHR
$(253)
1050 POSITION 14,4:? "F1=F2=F3=1
N":IF X=1 THEN X=0:GOTO 1400
1060 POSITION 2,20:? "MASA WOZKO
W MOZE PRZYJMOWAC"
1070 ? "WARTOSC 1,2,3,4 lub 5 kg
"
1080 POSITION 2,15:? "PODAJ MASE
 NR 1 ?";CHR$(253);:GET #1,M1
1090 M1=M1-48:IF M1>5 OR M1<1 TH
EN ? :? " ZLA WARTOSC":GOTO 1080
1100 POSITION 10,8 ? "M1=";M1,"k
g"
1110 POSITION 2,16:? "PODAJ MASE
 NR 2 ?",CHR$(253);:GET #1,M2

1120 M2=M2-48:IF M2>5 OR M2<1 TH
EN ? :? " ZLA WARTOSC":GOTO 1110
1130 POSITION 17,8:? "M2=";M2;"k
g"
1140 POSITION 2,17:? "PODAJ MASE
 NR 3 ?";CHR$(253);:GET #1,M3
1150 M3=M3-48:IF M3>5 OR M3<1 TH
EN ? :? " ZLA WARTOSC":GOTO 1140
1160 POSITION 25,8:? "M3=";M3;"k
g":? CHR$(29);CHR$(29)
1170 ? "CZY MASY SA PODANE PRAWI
DLOWO(T/N) ?";CHR$(253);:GET #1,
S
1180 IF S=84 THEN GOTO 1200
1190 GOTO 1040
1200 H=0:IF M1<=M2 AND M1<=M3 TH
EN M=M1:GOTO 1220
1210 M=M3:IF M2<=M3 THEN M=M2
1220 GRAPHICS 7:SETCOLOR 4,4,4
1230 COLOR 1:SETCOLOR 0,1,0:PLOT
 1,10
1240 DRAWTO 159,10:PLOT 1,40:DRA
WTO 159,40:PLOT 1,70:DRAWTO 159,
70
1250 FOR T=18 TO 159 STEP 14.1:P
LOT T,11:PLOT T,41:PLOT T,71:NEX
T T
1260 PP=PEEK(106)-32:POKE 106,PP
-32:POKE 54279,PP
1270 ? CHR$(127);"PROSZE POCZEKA
C 10 SEKUND!":POKE 559,62
1280 PL=PP*256+1024:FOR I=PL TO
PL+1024:POKE I,0:NEXT I:POKE 532
77,3
1290 POKE 704,8:POKE 705,15:POKE
 706,1
1300 POKE 53256,1:POKE 53257,1:P
OKE 53258,1
1310 POKE 53248,50:POKE 53249,50
:POKE 53250,50
1320 RESTORE 1510:P=PL+51:FOR I=
P TO P-6*M1-5 STEP -1
1321 READ B:POKE I,B:NEXT I
1330 RESTORE 1510:P=PL+367:FOR I
=P TO P-6*M2-5 STEP -1
1331 READ B:POKE I,B:NEXT I
1340 RESTORE 1510:P=PL+683:FOR I
=P TO P-6*M3-5 STEP -1
1341 READ B:POKE I,B:NEXT I:GOSU
B 6000
1350 ? CHR$(28);"KAZDY KLAWISZ T
O START SYMULACJI  "
1360 ? "SPACJA TO WYKONANIE OBLI
CZEN CWICZ.":GET #1,S
1370 IF S=32 THEN X=1:POKE 106,P
P+16:POKE 53277,0:GOTO 1040
1380 H=0:GOSUB 7020:GOTO 1540
1390 REM ________________
1400 REM BLOK OBLICZENIOWY
1405 REM ________________
1410 POKE 53248,0:POKE 53249,0:P
OKE 53250,0
1420 POSITION 8,6:? "m1=";M1;"kg
","m2=";M2;"kg","m3=";M3;"kg":?
:? :?
1450 GOSUB 9000
1455 POSITION 7,14:? "a1=";INT((
1/M1)*100)/100;
1456 ? " a2=";INT((1/M2)*100)/10
0;
1457 ? " a3=";INT((1/M3)*100)/10
0;" [m/s^2]"
1460 POSITION 7,16:? "S1=";(M*10
0)/M1:POSITION 14,16:? " S2=";(M
*100)/M2
1465 POSITION 22,16:? " S3=";(M*
100)/M3
1470 POSITION 30,16:? "        ";
? :?
1480 ? CHR$(127);"KLAWISZ N TO N
OWA SYMULACJA"
1490 ? "     INNY KLAWISZ TO POWR
OT DO MENU"
1495 GET #1,S:IF S=78 THEN GOTO
1040
1500 GOTO 20
1510 DATA 66,66,231,165,231,0,25
5,255,255,255,255,0,255,255,255,
255
1520 DATA 255,0,255,255,255,255,
255,0,255,255,255,255,255,0
1530 DATA 255,255,255,255,255,0,
0,0
1540 ? CHR$(28);CHR$(28);"KAZDY
KLAWISZ ZATRZYMUJE SYMULACJE"
1560 D=H^2
1570 POKE 53248,50+D*(1/M1):POKE
 53249,50+D*(1/M2)
1575 POKE 53250,50+D*(1/M3)
1580 IF (1/M)*D+40)177 THEN 1350
1590 IF PEEK(764)<>255 THEN FOR
T=1 TO 100:NEXT T:POKE 764,255:G
OTO 1350
1600 H=H+0.44:GOTO 1560
1900 REM ________________
1950 REM MASA JEST STALA
1960 REM ________________
2000 GRAPHICS 0:POKE 710,0:POSIT
ION 12,2:? "MASA JEST STALA";CHR
$(253)
2010 POSITION 12,4:? "m1=m2=m3=1
[kg]":IF X=1 THEN X=0:GOTO 3400
2015 POSITION 2,20 ? "SILA DZIAL
AJACA NA WOZEK MOZE"
2016 ? "PRZYJMOWAC WARTOSC 0,1,2
,3,4 lub 5[N]"
2020 POSITION 2,15:? "PODAJ SILE
 NR 1 ?";CHR$(253);:GET #1,F1

2022 F1=F1-48:IF F1>5 OR F1<0 TH
EN ? :? " ZLA WARTOSC":GOTO 2020
2025 POSITION 5,8:? "F1=";F1;"[N
]"
2030 POSITION 2,16:? "PODAJ SILE
 NR 2 ?";CHR$(253);:GET #1,F2
2032 F2=F2-48:IF F2>5 OR F2<0 TH
EN ? :? " ZLA WARTOSC":GOTO 2030
2035 POSITION 15,8:? "F2=";F2;"[
N]"
2040 POSITION 2,17:? "PODAJ SILE
 NR 3 ?";CHR$(253);:GET #1,F3
2044 F3=F3-48:IF F3>5 OR F3<0 TH
EN ? :? " ZLA WARTOSC":GOTO 2040
2050 POSITION 25,8:? "F3=";F3;"[
N]":?:?"
2060 ? "CZY SILY SA PODANE PRAWI
DLOWO(T/N) ?";CHR$(253);:GET #1,
S
2070 IF S=84 THEN GOTO 2100
2075 GOTO 2000
2100 H=0:IF F1>=F2 AND F1>=F3 TH
EN F=F1:GOTO 3000
2300 F=F3:IF F2>=F3 THEN F=F2
3000 GRAPHICS 7:SETCOLOR 4,4,4
3004 COLOR 1:SETCOLOR 0,1,0:PLOT
 1,10
3005 DRAWTO 159,10:PLOT 1,40:DRA
WTO 159,40:PLOT 1,70:DRAWTO 159,
70
3007 FOR T=18 TO 159 STEP 14.1:P
LOT T,11:PLOT T,41:PLOT T,71:NEX
T T

3010 PP=PEEK(106)-32:POKE 106,PP
-32:POKE 54279,PP
3015 ? CHR$(127);"PROSZE POCZEKA
C 10 SEKUND!":POKE 559,62
3020 PL=PP*256+1024:FOR I=PL TO
PL+1024:POKE I,0:NEXT I:POKE 532
77,3
3021 POKE 704,15:POKE 705,15:POK
E 706,15
3022 POKE 53256,1:POKE 53257,1:P
OKE 53258,1
3023 POKE 53248,50:POKE 53249,50
:POKE 53250,50
3030 RESTORE 4000:P=PL+27:FOR I=
P TO P+34:READ B:POKE I,B:NEXT I
3040 RESTORE 4000:P=PL+343:FOR I
=P TO P+34:READ B:POKE I,B:NEXT
I
3050 RESTORE 4000:P=PL+659 FOR I
=P TO P+34:READ B:POKE I,B:NEXT
I
3060 GOSUB 7050
3070 ? CHR$(28);"KAZDY KLAWISZ T
O START SYMULACJI  "

3080 ? "SPACJA TO WYKONANIE OBLI
CZEN CWICZ.":GET #1,S
3090 IF S=32 THEN X=1:POKE 106,P
P+16:POKE 53277,0:GOTO 2000
3100 H=0:GOSUB 7020:GOTO 4560
3399 REM ________________
3400 REM BLOK OBLICZENIOWY
3401 REM ________________
3402 POKE 53248,0:POKE 53249,0:P
OKE 53250,0
3410 POSITION 8,6:? "F1=";F1;"[N
]","F2=";F2;"[N]","F3=";F3;"[N]"
;:? :?
3417 GOSUB 9000
3420 POSITION 8,16:? "S1=";(F1*1
00)/F:POSITION 16,16:? " S2=";(F
2*100)/F
3421 POSITION 24,16:? " S3=";INT
(((F3*100)/F)*100)/100
3422 POSITION 9,14:? "a1=";F1:PO
SITION 16,14:? " a2=";F2
3423 POSITION 24,14:? " a3=";F3;
" [m/s^2]":? :?:? :?
3450 ? CHR$(127);"KLAWISZ N TO N
OWA SYMULACJA"
3460 ? "     INNY KLAWISZ TO POWR
OT DO MENU":GET #1,S:IF S=78 THE
N 2000
3470 GOTO 20
4000 DATA 255,255,255,255,255,25
5,255,255,255,255,255,255,255,25

5,255
4010 DATA 255,255,255,255,255,23
1,165,231,66,66,0,0,0,0,0,0,0,0,
0,0,0,0
4560 ? CHR$(28);CHR$(28);"KAZDY
KLAWISZ ZATRZYMUJE SYMULACJE"
4565 D=H^2
4600 POKE 53248,50+D*F1:POKE 532
49,50+D*F2:POKE 53250,50+D*F3
4610 IF F*D+40)178 THEN 3070
4615 IF PEEK(764)<>255 THEN FOR
```

```
T=1 TO 100 NEXT T POKE 764,255 G
OTO 3070
4620 H=H+0.13 GOTO 4565
6000 COLOR 2 RESTORE 6010 FOR T=
1 TO 15:READ A,B,C,D:PLOT A,B
6001 DRAWTO C,D NEXT T RETURN
6010 DATA 18,6,70,6,68,5,69,5,68
,4,68,4,68,7,69,7,68,8,68,8
7000 DATA 18,36,70,36,68,35,69,3
5,68,34,68,34,68,37,69,37,68,38,
68,38
7001 DATA 18,66,70,66,68,65,69,6
5,68,64,68,64,68,67,69,67,68,68,
68,68
7020 COLOR 0:A=4:GOSUB 7030:A=34
:GOSUB 7030 A=64 GOSUB 7030 RETU
RN
7030 FOR T=A TO A+4 PLOT 0,T:DRA
WTO 90,T:NEXT T

7040 ? CHR$(28);"SILY DZIALAJA D
O ZATRZYMANIA WOZKOW":RETURN
7050 COLOR 2:RESTORE 8000:PLOT 1
8,6:DRAWTO 18+14*F1,6:A=14*F1:B=
4
7051 GOSUB 7100
7055 RESTORE 8000:PLOT 18,36:DRA
WTO 18+14*F2,36:A=14*F2:B=34
7056 GOSUB 7100
7060 RESTORE 8000:PLOT 18,66:DRA
WTO 18+14*F3,66:A=14*F3:B=64
7061 GOSUB 7100:RETURN
7100 FOR T=1 TO 5:READ C:PLOT 18
+A,B:DRAWTO C+A,B:B=B+1:NEXT T
7101 RETURN
8000 DATA 18,19,20,19,18
9000 ? "DROGI WOZKOW SA WYRAZONE
 W LICZBACH"
9001 ? "NIEMIANOWANYCH.DROGA NAJ
SZYBSZEGO"
9002 ? "WOZKA ZOSTALA PRZYJETA J
AKO 100.":? :RETURN
```

```
         * SUMA KONTROLNA / ETYKIETA *
------------------------------------------------
IF     0| AX     1| GX     2| ID     6
ZF     7| ZB     8| XD     9| FH    10
MY    20| KY    30| PO    40| SN    50
WH   100| CA   105| QJ   110| AW   115
BR   120| BY   125| BY   130| HE   135
NK   140| NR   145| TN   150| AN   155
CF   160| OF   165| LA   170| JW   175
MG   176| RR   180| PA   185| FI   186
WO   190| AZ   195| EV   200| OK   205
FT   210| TR   212| DN   215| ER   220
VJ   225| ZD   230| KF   235| ZM   240
FV   245| NE   250| JD   255| NV   260
RX   265| SE   270| CA  1000| ZO  1010
CG  1020| RC  1040| TO  1050| RY  1060
PR  1070| ZF  1080| PC  1090| KI  1100
CY  1110| CF  1120| RA  1130| HT  1140
MG  1150| XE  1160| LW  1170| YH  1180
RR  1190| LF  1200| VX  1210| RB  1220
CS  1230| GE  1240| RJ  1250| UW  1260
IW  1270| SY  1280| EI  1290| KG  1300
QV  1310| EE  1320| DR  1321| YP  1330
DU  1331| BC  1340| LN  1341| RX  1350
FR  1360| OS  1370| WO  1380| UH  1390
UM  1400| UC  1405| WC  1410| TC  1420
AY  1450| OY  1455| HW  1456| JZ  1457
TY  1460| TI  1465| PE  1470| HT  1480
ZJ  1490| PN  1495| PN  1500| DJ  1510
MK  1520| EL  1530| TJ  1540| BU  1560
JP  1570| ZQ  1575| MY  1580| YD  1590
HP  1600| PJ  1900| YG  1950| QB  1960
EU  2000| XI  2010| PN  2015| UC  2016
PZ  2020| LL  2022| AT  2025| UO  2030
QK  2032| WH  2035| ZD  2040| VR  2044
GK  2050| OZ  2060| YC  2070| OP  2075
UO  2100| WF  2300| QT  3000| CX  3004
GK  3005| RU  3007| UF  3010| IW  3015
SE  3020| JE  3021| KQ  3022| RG  3023
FD  3030| TX  3040| DT  3050| CP  3060
RZ  3070| FT  3080| IA  3090| AS  3100
VT  3399| UO  3400| TO  3401| WJ  3402
TZ  3410| BQ  3417| VY  3420| JA  3421
VB  3422| DQ  3423| HM  3450| ZR  3460
GI  3470| OB  4000| BG  4010| TS  4560
CR  4565| XK  4600| DI  4610| AW  4615
MU  4620| CF  6000| FZ  6001| SI  6010
GL  7000| BX  7001| IB  7020| YL  7030
MY  7040| YI  7050| AP  7051| RG  7055
BJ  7056| JQ  7060| JC  7061| WX  7100
PF  7101| PU  8000| CP  9000| RM  9001
LC  9002|         |         |
------------------------------------------------
```


* * *

ŻEGLOWANIE

Nowa pasja, jaka ogarnęła Long Terry zaczęła się pięknego majowego dnia na lekcji historii cywilizacji ziemskiej. Wiemy, że nie należą one do najchętniej wysłuchiwanych przez uczniów szkoły elementarnego przygotowania do życia. Być może stary profesor Duck Long, mówiący monotonnym głosem i niezbyt często ilustrujący tezy wykładu eksponatami nie umiał zapalić do nich swoich uczniów. Tym razem jednak zamiast chorego profesora po włączeniu monitorów uczniowie VI oddziału zobaczyli wysokiego eleganckiego mężczyznę o sylwetce sportowca. Ten mając przed sobą pulpit, na którym włączyły się numery identyfikacyjne słuchaczy, uśmiechnął się szelmowsko i rozpoczął wykład. Mówił o sposobach pokonywania przestrzeni wodnych na różnego rodzaju pływadłach bez silników. Większość uczniów nie dawała temu wiary, jak to bowiem może być możliwe, żeby gdzieś w odległych latach od ich XXX w. ludzie płynąc z szybkością przysłowiowego żółwia mogli pokonywać olbrzymie przestrzenie oceanów i mórz.

Kiedy po zajęciach dzielili się swymi uwagami Tom Caurt zaproponował: słuchajcie możemy przecież spróbować zbudować coś takiego do pływania i sprawdzić. Praktycznie pomysł został przyjęty bez sprzeciwu.

No ...i zaczęło się. Najpierw za pomocą komputerów zgromadzili wiadomości o żeglowaniu i żaglowcach, później rozpoczęli opracowywanie dokumentacji. Nie należało to do najszybszych i bezkolizyjnych zadań. Raz po raz musieli pokonywać bariery. Bo oto nie można było zdobyć drewna na budowę łodzi. Wszystkie drzewa w myśl konwencji o ochronie przyrody były objęte prawem. Jack wymyślił użycie serganu masy plastycznej z ubiegłego wieku, imitującej drewno, ale okazało się, że nigdzie jej już nie produkują. I kiedy sytuacja wydawała się bez wyjścia Tom Frank, który był z rodzicami na wycieczce, przywiózł wiadomość, że czterdzieści mil od miasta stoją opuszczone domy wykonane z tego materiału. Konieczne było uzyskanie zgody na rozebranie i transport. Koniec końców poszło łatwiej niż przypuszczali. Materiał zwieźli i trzeba było przystąpić do prac. Okazało się, że żaden z robotów nie potrafi wykonywać czynności związanych z obrabianiem serganu. Dobrze, że ojciec Jacka jest programistą, napisał i zaprogramował dwa roboty i praca ruszyła. Zawsze po lekcjach cały oddział VI był w swojej stoczni (takie starożytne słowo). Chłopcy przywiązali się do miejsca robót. Gdzieś na początku czerwca wszystko było gotowe. Długa na dwanaście metrów łódź z trzema masztami gotowa była do przetransportowania na wodę. Na wszelki wypadek zainstalowano na niej wyposażenie ratownicze, takie jakie mają pływoloty, ster automatyczny, wspomagane komputerem identyfikatory. I wtedy okazało się, że nie ma żagli i olinowań. Po długich poszukiwaniach zamiast figurującego w zapisach płótna żeglarskiego użyto stersonu, materiału używanego do ochrony przed deszczem.

To było wielkie święto, kiedy łódź została spuszczona na wodę. Okazało się, że jest bardzo przyjemnie, kiedy kołysze się na falach. Przez najbliższe dni wszyscy uczyli się komend i czynności związanych z obsługą pływadła. Instrukcje te zdobył gdzieś niezawodny jak zwykle Jack. I wreście nadszedł dzień pierwszego rejsu. Padły komendy, naciągnięto żagle i nic, żadnego ruchu. Ich pływadło nie chciało się ruszyć. W końcu jednak wypłynęli budząc olbrzymią sensację. Nauczyli się powoli manewrować swoim żaglowcem, wzmacniając lub osłabiając nadmuch i szło. Najgorzej było przy wykonywaniu zwrotów. Z początku dopływali do jakiejś wyspy i tam opuszczali żagle, później zaczepiali liny i obracali statek, podnosili żagle i płynęli z powrotem. Fajne to były przejażdżki. Rodzice z początku z pobłażaniem przyglądali się ich wyczynom, później z coraz większym zainteresowaniem. W tym czasie chłopcy nauczyli się manewrowania swoim statkiem. Ponieważ kończył się rok szkolny postanowili odbyć długi rejs. Gromadzili zapasy, zbierali wiadomości o warunkach żeglowania. Podczas jednego z rejsów, dopłynęli do domu wykorzystując wiatry. I to co niejasne stało się jasnym. Kiedy swą decyzję wypłynięcia w długi rejs zgłosili Radzie Obywatelskiej natychmiast znaleźli się dziennikarze, którzy chcieli razem z nimi płynąć. Na szczęście podczas jednej z przejażdżek większość zachorowała na starą, jak świat chorobę morską i mieli ich z głowy.

Rejs ich był transmitowany na bieżąco. Prawie w każdym dzienniku pokazywano ich jak sobie radzą, a radzili sobie dobrze. Kiedy powrócili do swego miasta czekali na nich rodzice i znajomi. Ale nie to było najważniejsze, nabrali tężyzny fizycznej, sprawności, jakiej nie miał nikt z ich kolegów. Rada Lekarska wydała wtedy oświadczenie zmieniające dotychczasowy zakaz wykonywania prac fizycznych, dopuszczając je, gdyż rozwijają możliwości człowieka.

Od ich powrotu nie było prawie dnia, aby ich statek nie był zwiedzany przez wycieczki szkolne, inżynierów, techników. Dało się zauważyć również i to, że uruchomiono produkcję sergentu, a także budowę podobnych pływadeł. Te wkrótce zdominowały całe wybrzeże, co z kolei doprowadziło do konieczności wydania określonych przepisów porządkowych.

Tylko chłopcy z VII już wtedy oddziału myśleli o nowym pomyśle. Miał nim być statek podwodny z XX w. Kiedy to piszę, przystąpili właśnie do jego budowy.

Martin CARR

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

H

HACKER - 1. programista, potrafiący pisać programy bez wstępnego zapoznawania się ze szczegółowym opisem, a także potrafiący nanosić poprawki do działających programów, nie posiadając dokumentacji, 2. użytkownik systemu komputerowego (sieci komputerowej), zajmujący się poszukiwaniem sposobów dostępu do zabezpieczonych danych (włamywacz do zabezpieczonych programów, systemów lub baz danych),
HALF - połowa, połówka,
HALF-ADDER - półsumator, sumator jednocyfrowy,
HALFADJUST - zaokrąglać,
HALF-BYTE - półbajt,
HALF CARRY - przeniesienie pomocnicze,
HALF DUPLEX - półdupleks, dupleks częściowy (niejednoczesna komunikacja w obu kierunkach),
HALF DUPLEX CHANNEL - kanał półdupleksowy,
HALF-DUPLEX CIRCUIT - patrz: HALF DUPLEX CHANNEL,
HALF-DUPLEX DATA PATH - półdupleksowy tor przesyłania danych,
HALF-DUPLEX OPERATION - system pracy półdupleksowej,
HALF-SUBTRACTOR - układ odejmujący jednocyfrowy,
HALF TITLE - tytuł wstępny, przedtytuł,
HALF-TITLE PAGE - strona przedtytułowa,
HALF-TONE - 1. półton, 2. półtonowy,
HALF-TONING - wytwarzanie półtonów,
HALF-WORD - półsłowo,
HALT - zatrzymanie (np. pracującego komputera),
HALT INSTRUCTION - rozkaz stopu, rozkaz zatrzymania,
HALVE - dzielić na połowę,
HALVING - dzielący na pół,
HAMMER - młotek (np. drukarki),
HAMMERLOCK - zastawka blokująca młotek drukarki,
HAMMER SPRING TENSION - naciąg sprężyny młotka (drukarki),
HAMMING CODE - kod Hamminga,
HAMMING DISTANCE - odstęp Hamminga, dystans Hamminga,
HANDBOOK - podręcznik, poradnik (książka),
HANDBOOK FOR PROGRAMMED LEARNING - podręcznik programowany,
HANDFUL - garść (czegoś), partia (np. kart),
HAND-HELD - test podręczny,
HAND-HELD COMPUTER - komputer podręczny,
HANDHELD MICROCOMPUTER - mikrokomputer podręczny,
HANDLE - wykonywać, mieć do czynienia, manipulować,
HANDLER - 1. podprogram współdziałania z urządzeniami zewnętrznymi, manipulator, 2. program obsługi sytuacji wyjątkowej,
HANDLING - manipulowanie, obsługiwanie, przenoszenie, operowanie,
HANDSHAKE - dwuszynowa metoda kontroli przesyłania informacji,
HANDSHAKING - 1. przesyłanie z potwierdzeniem, 2. uzgadnianie transmisji, 3. procedura synchronizująca (transmisję),
HANDSHAKING SIGNALS - sygnały uzgodnienia,
HANDWRITTEN - napisany ręcznie,
HANDY - poręczny, podręczny,
HANGING INDENT - występ, przesunięcie w lewo (np. tekstu),
HANGUP - zawieszenie (stan systemu komputerowego, w którym nie wyprowadza on już komunikatów i nie reaguje na zewnętrzne przerwania),
HANG-UP - przerwanie (wykonywania programu),
HAPHAZARD - 1. przypadek, 2. przypadkowy, dorywczy,
HAPPEN - zdarzać się,
HARD - 1. twardy, sztywny, 2. stały, niezmienny,
HARD ADDER - przyrostowy sumator sztywny,
HARD COPY - trwała postać zapisu informacji, wydruk na drukarce ekranu monitora,
HARD-COPY TERMINAL - końcówka drukująca, terminal drukujący,
HARD DATA - dane w postaci liczb i wykresów, (w odróżnieniu od opisów jakościowych),
HARD DISC - dysk twardy,
HARD DISK - patrz: HARD DISC,
HARD ERROR - błąd stały,
HARD PAGE BREAK - "twarde" zakończenie strony (w systemach edycji tekstów, przejście do nowej strony, stosowane przy rozpoczynaniu np. nowego rozdziału),
HARD-SECTORED DISK - dysk ze stałym rozmieszczeniem sektorów (dysk magnetyczny o stałym podziale na sektory, rozmieszczenia których nie można zmienić programowo),
HARD SECTORING - odpowiedni zapis na ścieżce kolejnych sektorów poprzez umieszczenie obok otworu indeksowego dyskietki kolejnych otworów, określających położenie początkowe sektorów,
HARD SPACE - "twarda" spacja (spacja wprowadzona do tekstu przez użytkownika, w odróżnieniu od spacji wprowadzanej automatycznie podczas justowania),
HARD TO DISTINGUISH - trudny do odróżnienia,
HARD TO REACH - trudno dostępny,
HARDWARE - sprzęt komputerowy, hardware,
HARDWARE BLOCK PRINT ROUTINES - procedury drukowania zawartości pakietu blokami fizycznymi,
HARDWARE CHECK - kontrola realizowana tylko za pomocą sprzętu,
HARDWARE COMPATIBILITY - wymienność sprzętowa,
HARDWARE-COMPATIBLE - wymienny sprzętowo,
HARDWARE CONTROL - sterowanie układowe,
HARDWARE DIVISION - dzielenie maszynowe,
HARDWARE ENVIRONMENT - środki sprzętowe (sprzęt komputerowy wykorzystywany w trakcie pracy programu),
HARDWARE ERROR - błąd sprzętowy (błąd spowodowany złą pracą sprzętu),
HARDWARE-IMPLEMENTED PROCESS SCHEDULING - sprzętowe sterowanie przebiegiem procesu,
HARDWARE INTERRUPT - przerwanie sprzętowe (przerwanie od urządzenia zewnętrznego np. po błędzie wykonywanego rozkazu, itp.),
HARDWARE MODEL - model fizyczny,
HARDWARE MULTIPLICATION - mnożenie sprzętowe (wykonanie operacji mnożenia rozkazem procesora a nie przez podprogram),
HARDWARE PROGRAM COUNTER - wbudowany licznik rozkazów,
HARDWARE-SOFTWARE TRADE-OFF - podział zadań między sprzęt a oprogramowanie,
HARDWARE SPRITE - sprzętowe środki tworzenia dynamicznego zobrazowania graficznego,
HARDWARE STACK - stos sprzętowy,
HARDWARE SUPPORT - wsparcie sprzętowe, pomoc sprzętowa, realizacja sprzętowa,
HARDWIRED - 1. sprzętowo zaszyty, 2. realizowany środkami sprzętowymi, 3. z wbudowanym konstrukcyjnie układem sterowania lub programowania,
HARDY - mocny, odporny, wytrzymały,
HARMFUL - szkodliwy,
HARMLESS - niegroźny, nieszkodliwy,
HARMONIC - harmoniczny,
HARMONIC DISTORION - zniekształcenie harmoniczne,
HARTLEY - jednostka miary informacji równa informacji reprezentowanej przez jedną cyfrę dziesiętną,
HASH - informacja pozbawiona praktycznego znaczenia, znajdująca się w pamięci,
HASH ADDRESSING - adresowanie z haszowaniem (patrz: HASHING),
HASH DATA - dane zbędne, dane pomyłkowe,
HASHING - haszowanie (sposób organizacji struktur danych, zapewniający efektywne wyszukiwanie i aktualizację danych, tzw. tablice HASH),
HASHING ALGORITHM - algorytm haszowania (patrz: HASHING),
HASHING FUNCTION - funkcja rozmieszczania, funkcja haszowania,
HASH TABLE - tablica przewodnik,
HASH TOTAL - suma kontrolna,
HASP - patrz: HOUSTON AUTOMATIC SPOOLING PROGRAM,
HASTEN COMPLETION - przyspieszyć zakończenie (np. prac programowych),
HAVE A BEARNING ON ... - wywierać wpływ na ...,
HAVE AN EFFECT ON ... - wpływać na ...,
HAVE A RESTRAINING EFFECT - działać hamująco,
HAVE AT ONE'S DISPOSAL - rozporządzać (czymś),
HAZINESS - nieostrość (np. obrazu na ekranie monitora),
HDAM - patrz: HIERARCHICAL DIRECT ACCESS METHOD,
HDLC - patrz: HIGH LEVEL DATA LINK CONTROL,
HDLC STATION - stacja protokołu HDLC,
HDR - patrz: HIGH DENSITY RECORDING,
HD - patrz: HIGH DENSITY,
HD TV - telewizja wysokiej jakości,
HEAD - głowica, także: nagłówek,
HEAD COMPUTER - komputer główny,
HEADER - nagłówek, etykieta początkowa,
HEADER CARD - karta inicjująca, karta wskazująca,
HEADER ENTRY - pozycja tytułowa,
HEADER LABEL - etykieta początku zbioru danych, zapis początku,
HEADER LABEL CHECK - sprawdzanie etykiety nagłówkowej (np. na taśmie magnetycznej),
HEADER RECORD - zapis (rekord) nagłówkowy,
HEAD GAP - szczelina głowicy magnetycznej, także: przerwa między głowicą magnetyczną a nośnikiem danych,

HEADING - patrz: HEADLINE,
HEADLINE - nagłówek,
HEAD MOVEMENT - przesuw głowicy,
HEAD POSITIONING TIME - czas szukania, czas ustawiania głowic,
HEAD SPACING - rozstaw głowic magnetycznych,
HEAD STACK - zespół głowic magnetycznych dla zapisu wielościeżkowego,
HEAD-TO-TAPE GAP - przerwa między głowicą magnetyczną a taśmą,
HEAP - 1. stos (dynamiczny obszar pamięci), 2. wielka ilość, 3. gromadzić,
HEAP MANAGER - program zarządzający stertą (dynamicznym obszrem pamięci programu), program dynamicznego rozdziału obszaru pamięci,
HEAVY - 1. intensywny, 2. ciężki, 3. silny,
HEAVY LOSS - znaczna strata, poważna strata,
HEIGHT - wysokość,
HEIGHT-BALANCED TREE - drzewo o zrównoważonej wysokości,
HEIGHT OF TREE - wielkość drzewa,
HELP - tekst objaśniający, objaśnienie, opis, help,
HELP LIBRARY - biblioteka tekstów objaśniających, biblioteka opisów (objaśnień),
HELP LINE - wiersz (na ekranie monitora) z tekstem objaśniającym,
HERCULES GRAPHIC CARD - karta grafiki Hercules,
HESITATION - opóźnienie, przerwa w przetwarzaniu,
HETEROGENEOUS - niejednorodny, różnorodny,
HETEROGENEOUS COMPUTER NETWORK - niejednorodna sieć komputerowa,
HEURISTIC - heurystyka,
HEURISTIC PROGRAM - program heurystyczny,
HEURISTIC ROUTING - heurystyczny wybór trasy,
HEWLETT-PACKARD COMPANY - firma amerykańska, produkująca między innymi mini- i mikrokomputery, przyrządy pomiarowe oraz tworząca oprogramowanie komputerowe,
HEX - patrz: HEXADECIMAL,
HEXADECIMAL - szesnastkowy,
HEXADECIMAL DIGIT - cyfra szesnastkowa (tzn. jedna z wielkości: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F),
HEXADECIMAL FORMAT - format szesnastkowy (przedstawienie danych w systemie szesnastkowym),
HEXADECIMAL NOTATION - szesnastkowy system liczenia,
HEXADECIMAL NUMBER - liczba szestnastkowa,
HEXADECIMAL NUMBER SYSTEM - system szesnastkowy, system heksadecymalny,
HEX FILE - plik (zbiór) typu szesnastkowego,
HGC - patrz: HERCULES GRAPHIC CARD,
HIBERNATING PROCESS - proces zatrzymujący, proces zawieszający,
HIBERNATING TASK - zadanie zatrzymujące, zadanie zawieszające,
HIBERNATION - stan oczekiwania, stan wyczekiwania,
HIC - patrz: HYBRID INTEGRATED CIRCUIT,
HIDAM - patrz: HIERARCHICAL INDEXED DIRECT ACCESS METHOD,
HIDDEN - 1. ukryty (np. plik), 2. ukryć, schować,
HIDDEN LINE - linia ukryta (np. linie niewidoczne w dwuwymiarowym rzucie bryły),
HIDDEN-LINE REMOVAL - usunięcie linii niewidocznych (w grafice komputerowej sposób odzwierciedlenia trójwymiarowego obiektu),
HIDDEN SURFACE - niewidoczna powierzchnia,
HIDDEN-SURFACE REMOVAL - usunięcie niewidocznych powierzchni (w grafice komputerowej sposób odzwierciedlenia trójwymiarowego obiektu),
HIDE - ukryć, schować,
HIERARCHICAL ACCESS METHOD - hierarchiczna metoda dostępu,
HIERARCHICAL ADDRESSING - hierarchiczna adresacja (sposób przedstawienia obiektu w sieci komputerowej przy pomocy złożonego identyfikatora, odzwierciedlającego strukturę sieci i drogę dostępu),
HIERARCHICAL COMPUTER NETWORK - hierarchiczna sieć komputerowa,
HIERARCHICAL DATA BASE - hierarchiczna baza danych,
HIERARCHICAL DIRECT ACCESS METHOD - hierarchiczna metoda bezpośredniego dostępu,
HIERARCHICAL INDEXED DIRECT ACCESS METHOD - hierarchiczna metoda bezpośredniego dostępu z wykorzystaniem indeksacji,
HIERARCHICAL INDEXED SEQUENTIAL ACCESS METHOD - hierarchiczna indeksowo-sekwencyjna metoda dostępu,
HIERARCHICAL NETWORK - sieć hierarchiczna,
HIERARCHICAL SEQUENTIAL ACCESS METHOD - hierarchiczna sekwencyjna metoda dostępu,
HIERARCHICAL STORAGE - hierarchiczna pamięć komputerowa,
HIERARCHY - kolejność ważności, hierarchia,
HIERARCHY OF SETS - hierarchia zbiorów,
HIGH - 1. duży, 2. większy (przy porównywaniu dwóch wielkości),
HIGH BIT - górny bit (najbardziej znaczący bit),
HIGH BOUND - ograniczenie górne (pliku), kres górny,
HIGH DENSITY - wysoka gęstość (sposób zapisu informacji na dysku),
HIGH DENSITY RECORDING - zapis o dużej gęstości,
HIGH-DENSITY STORE - pamięć o dużej gęstości zapisu,
HIGH-DUTY - o dużej wydajności, wysokosprawny,
HIGHER-LEVEL LANGUAGE - język wyższego poziomu,
HIGHER ORDER EQUATION - równanie wyższego rzędu,
HIGH LEVEL - wyższy poziom,
HIGH-LEVEL COMPILER - kompilator wyższego rzędu,
HIGH LEVEL DATA LINK CONTROL - protokół HDLC,
HIGH-LEVEL GOAL - kres wysokiego poziomu,
HIGH-LEVEL LANGUAGE - język wysokiego poziomu,
HIGH-LEVEL PROTOCOL - protokół wysokiego poziomu,
HIGHLIGHT - uwypuklać, uwydatniać, podświetlać, rozjaśniać, błyszczenie (migotanie lub podświetlanie elementu obrazu),
HIGHLIGHTING - wydzielenie, uwydatnienie, wyodrębnienie (w grafice komputerowej wyodrębnienie części tekstu lub fragmentu obrazu na ekranie monitora poprzez podświetlenie, migotanie, itp.),
HIGH-ORDER DIGIT - cyfra o najwyższej wartości (np. w liczbie),
HIGH-ORDER POSITION - najstarsza pozycja (pierwsza z lewej strony pozycja bajtu, słowa, czy wiersza),
HIGH-PERFOMANCE - szybko działający,
HIGH-QUALITY - wysokiej jakości,
HIGH-RESOLUTION MODE - tryb graficzny wysokiej rozdzielczości,
HIGH-SPEED CARRY - przenoszenie szybkie,
HIGH-SPEED MEMORY - pamięć szybka, pamięć o dostępie szybkim,
HIGH-SPEED MEMORY BLOCK - blok pamięci ultraszybkiej,
HIGH-SPEED PAPER FEED - szybkie zasilanie drukarki papierem,
HIGH-SPEED PRINTER - drukarka szybka,
HIGH-SPEED STORAGE - patrz: HIGH-SPEED MEMORY,
HIGH-SPEED STORE - patrz: HIGH-SPEED MEMORY,
HIGHWAY - szyna, magistrala,
HIGH-YIELD - wysoka wydajność,
HIRED LINE - łącze dzierżawione,
HISAM - patrz: HIERARCHICAL INDEXED SEQUENTIAL ACCESS METHOD,
HIT - pozytywny wynik porównania dwóch wartości,
HITACHI - firma japońska, produkująca sprzęt komputerowy kompatybilny z wyrobami firmy IBM,
HIT-ON-THE-FLY PRINTER - drukarka wierszowa drukująca w ruchu,
HIT RATE - współczynnik zgodności, współczynnik zbieżności,
HLL - patrz: HIGH-LEVEL LANGUAGE,
HLS MODEL - model "barwa-jaskrawość-nasycenie" (w grafice komputerowej charakterystyka koloru za pomocą trzech parametrów: barwy, jaskrawości i nasycenia),
HOLD - przechowywać (np. w pamięci), zatrzymywać, znajdować się, mieścić (w sobie), także: uważać że, utrzymywać,
HOLD-DOWN - przytrzymywać (np. naciśnięty klawisz),
HOLDER - oprawka, uchwyt,
HOLDING - 1. zabezpieczanie, przechowywanie (np. danych), 2. stan posiadania (np. taśm magnetycznych w bibliotece),
HOLD TIME - czas utrzymywania informacji (np. na wejściach bloków sekwencyjnych),
HOLE - dziurka (np. w karcie),
HOLE COUNT - liczenie dziurek (np. na karcie),
HOLE DENSITY - gęstość dziurek,
HOLLERITH CARD - 80-kolumnowa karta dziurkowana z nadrukiem w kodzie Hollerith'a,
HOLLERITH CODE - kod Hollerith'a (kod wykorzystywany do przedstawiania informacji tekstowej na karcie dziurkowanej),
HOLLERITH CONSTANT - wielkość tekstowa (w języku Fortran),
HOLOGRAPHIC MEMORY - pamięć holograficzna,
HOLOGRAPHIC STORAGE - pamięć holograficzna,
HOLOGRAPHIC STORE - patrz: HOLOGRAPHIC STORAGE,
HOLOGRAPHY - holografia,
HOME - początek, pozycja początkowa, (w grafice komputerowej, określenie prawego, górnego rogu ekranu monitora),
HOME ADDRESS - adres własny (np. pole ścieżki dysku, zawierające adres tej ścieżki),
HOME BLOCK - blok początkowy,
HOME BUCKET - porcja macierzysta,
HOME COMPUTER - komputer domowy,
HOME LOCATION - komórka podstawowego obszaru, obszar podstawowy,
HOME POSITION - pozycja główna, pozycja podstawowa,
HOME RECORD - zapis (rekord) główny (pierwszy zapis w pliku), zapis (rekord) odniesienia (pierwszy w łańcuchu zapisów)

LIGA MYŚLĄCYCH

ZADANIE 1

Pięciu studentów, powtarzających trzeci rok studiów wybiera losowo — każdy niezależnie od pozostałych, jedną z trzech równoległych grup. Jakie jest prawdopodobieństwo, że
1) wszyscy studenci znajdą się w pierwszej grupie,
2) wszyscy studenci znajdą się w tej samej grupie,
3) w ustalonej grupie znajdzie się dokładnie trzech studentów?

ZADANIE 2

Ile średnio powinno przypadać rodzynków na bułeczkę aby z prawdopodobieństwem nie mniejszym niż 0,99 twierdzić, że w bułeczce znajduje się choćby jeden rodzynek?

ZADANIE 3

Dwaj piechurzy wyszli w jednakowym czasie naprzeciw siebie i spotkali się po trzech godzinach i dwudziestu minutach. W jakim czasie każdy z nich przejdzie odległość, jaka dzieli miejsca, z których wyszli, jeżeli pierwszy przyszedł na miejsce z którego wyszedł drugi, o 5 godzin później?

ZADANIE 4

Po przejściu w ciągu godziny 3,5 km, pasażer idący do pociągu, zorientował się, że idąc dalej z tą samą prędkością spóźni się na pociąg o 1 godzinę. Dlatego pozostałą drogę przeszedł z prędkością 5 km/h, przybywając na stację, na 30 minut przed odejściem pociągu. Wyznaczyć drogę, jaką przebył pasażer.

ZADANIE 5

Dwóch turystów idzie naprzeciw siebie — jeden z miejscowości A, a drugi z miejscowości B. Pierwszy turysta wyszedł z miejscowości A o 6 godzin później i przy spotkaniu okazało się że przeszedł o 12 km mniej niż drugi. Kontynuując dalej podróż z tą samą prędkością, pierwszy turysta przyszedł do miejscowości B po 8 godzinach, a drugi do A po 9 godzinach. Wyznaczyć odległość AB oraz prędkość obu turystów.

Rozwiązania zadań prosimy przesyłać do redakcji do końca stycznia br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzcę „Ligi" czekają dodatkowe nagrody.

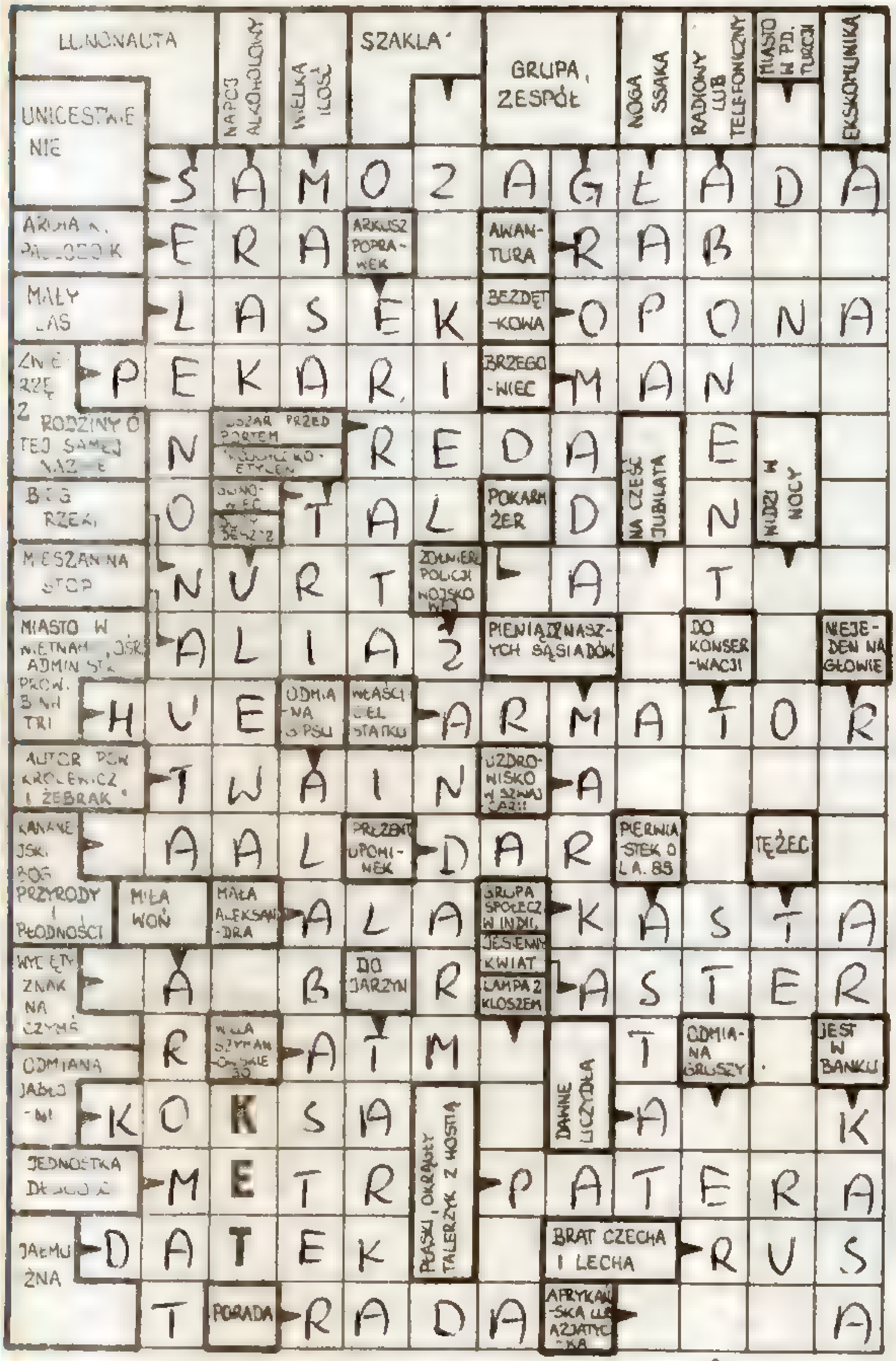
GIEŁDA POMYSŁÓW

ZX-81

Szukanie dziury

```
1 REM **********************
2 REM *  SZUKANIE DZIURY   *
3 REM *       ZX-81        *
5 REM **********************
10 FOR A=1 TO 32
15 PRINT CHR$ 7;
20 NEXT A
25 IF A=32 THEN GOTO 30
30 LET S=0
35 LET J=INT RND*16
40 GOSUB 145
45 PRINT "████████████████
██████████"
50 PRINT AT 9,7; "ODLEGLOSC?"
55 INPUT A
60 PRINT AT 9,7; "          "
65 PRINT AT 17,J;""
70 J=J+INT(A/3+RND)
75 GOSUB 145
80 LET S=S+1
85 PRINT AT 5,12; "ILOSC PODE
JSC: ";S
90 IF J<27 THEN GOTO 50
95 IF J>27 THEN GOTO 120
100 PRINT AT 17,27; "" ;AT 18,
27;"0"
105 LET U=J-S*2
110 PRINT "ILOSC PUNKTOW:"; U
;"- ZA TRAFIENIE"
115 GOTO 125
120 PRINT "BEZNADZIEJNIE"
125 FOR G=1 TO 100
130 NEXT G
135 CLS
140 RUN
145 PRINT AT 17,J; "O"
150 RETURN
```

KRZYŻÓWKA NR 10



W rozwiązaniu krzyżówki wystarczy podać, ile razy w diagramie występują litery: I, K, S. Rozwiązanie zadania należy przesłać pod adresem redakcji na kartach pocztowych, w terminie do końca stycznia, naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.



Nauka • Technika

Nowoczesna technika w strukturze kolejowej, to przede wszystkim wysokiej klasy łączność sygnalizacyjna — czyli zabezpieczenie ruchu coraz szybszych pociągów i szeroko podjęta informacja, realizowana dzięki wykorzystaniu komputerów. Instytuty badawcze kolejnictwa, zespoły naukowców z każdym rokiem oferują coraz to nowsze, bardziej korzystne ekonomicznie i technicznie rozwiązania z użyciem m.in. mikroprocesorów i światłowodów.

Zastosowanie techniki komputerowej do kontroli biegu pociągów i łączności między dyspozytorniami i maszynistami zapewnia kolei efektywniejszą działalność. Zdalne pomiary z lokomotywy umożliwiają wychwycenie każdej najmniejszej niedokładności w poszczególnych urządzeniach hamulcowych w składzie badanego pociągu. Zastosowanie elektronicznego systemu wykrywania wadliwego działania urządzeń hamulcowych wpływa na osiąganie przez pociągi lepszych efektów eksploatacyjnych. Przekazywanie informacji do komputera w lokomotywie zapewnia nie tylko efektywną kontrolę biegu pociągu, ale umożliwia zdalne śledzenie trasy, sygnałów podawanych na semaforach, zamykanych rogatek na przejazdach itp. Biegiem pociągu kieruje maszynista, nadzoruje go komputer, który jednocześnie podaje podobne dane do dyspozytora, posiadającego łączność radiową z maszynistą.

Kierunki podawanej drogi z komputera w lokomotywie do dyspozytora nie tylko umożliwiają zlokalizowanie pociągu na trasie, ale także automatyczne nastawianie drogi przebiegu oraz uzyskanie pełnej kontroli nad pociągami poruszającymi się w danym kierunku. Umożliwia to wprowadzenie na trasę w bezpiecznych odstępach dodatkowych pociągów towarowych.

Handlowcy w Stanach Zjednoczonych oferują dzieciom komplet do demonstracji zjawiska nadprzewodnictwa. To, co nie tak dawno mogła obserwować tylko mała grupa naukowców zatrudnionych w kilku najlepszych ośrodkach badawczych świata, obecnie bawi i uczy młodzież szkolną.

Prototyp zabawki skonstruowali specjaliści z amerykańskiego Uniwersytetu Wisconsin. W skład zestawu wchodzi naczynie z przezroczystego tworzywa połączone ze zbiornikiem na ciekły azot oraz mała płytka o średnicy 25 mm. Krążek ten wykonany został ze związków itru, baru i miedzi z tlenem. Po obniżeniu temperatury w plastykowym naczyniu, płytka zaczyna unosić się w powietrzu, zaś mierniki badające własności próbki rejestrują zmiany w przepływie prądu efektonicznego oraz zachodzące przeobrażenia pola magnetycznego.

Producent zabawki dołącza do niej także rękawice i okulary ochronne, wszak eksperymentator ma do czynienia z ciekłym azotem. Model przedstawiający nadprzewodnictwo można nabyć za 25 dolarów.

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1965. Nr ind. 361682. U-3

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 zł

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" Nr 12/1988 ISSN 0860—2794

NASZYM CZYTELNIKOM I SYMPATYKOM ŻYCZYMY SZCZĘŚLIWEGO NOWEGO ROKU

REDAKCJA

KONKURS — 8
SF — 24
SŁOWNIK — 29

Proponujemy Państwu nową rubrykę w naszym piśmie dzięki niej będziecie mogli Państwo zorientować się w cenach sprzętu komputerowego i peryferyjnego, obowiązujących w różnych firmach na świecie

Dzisiejsze hasło to " HARD DYSKI " !!!

Ceny według firmy "DSCOUNT MICRO PRODUCTSINC " 34360 Glendale Ave ,Livonia, MI48150

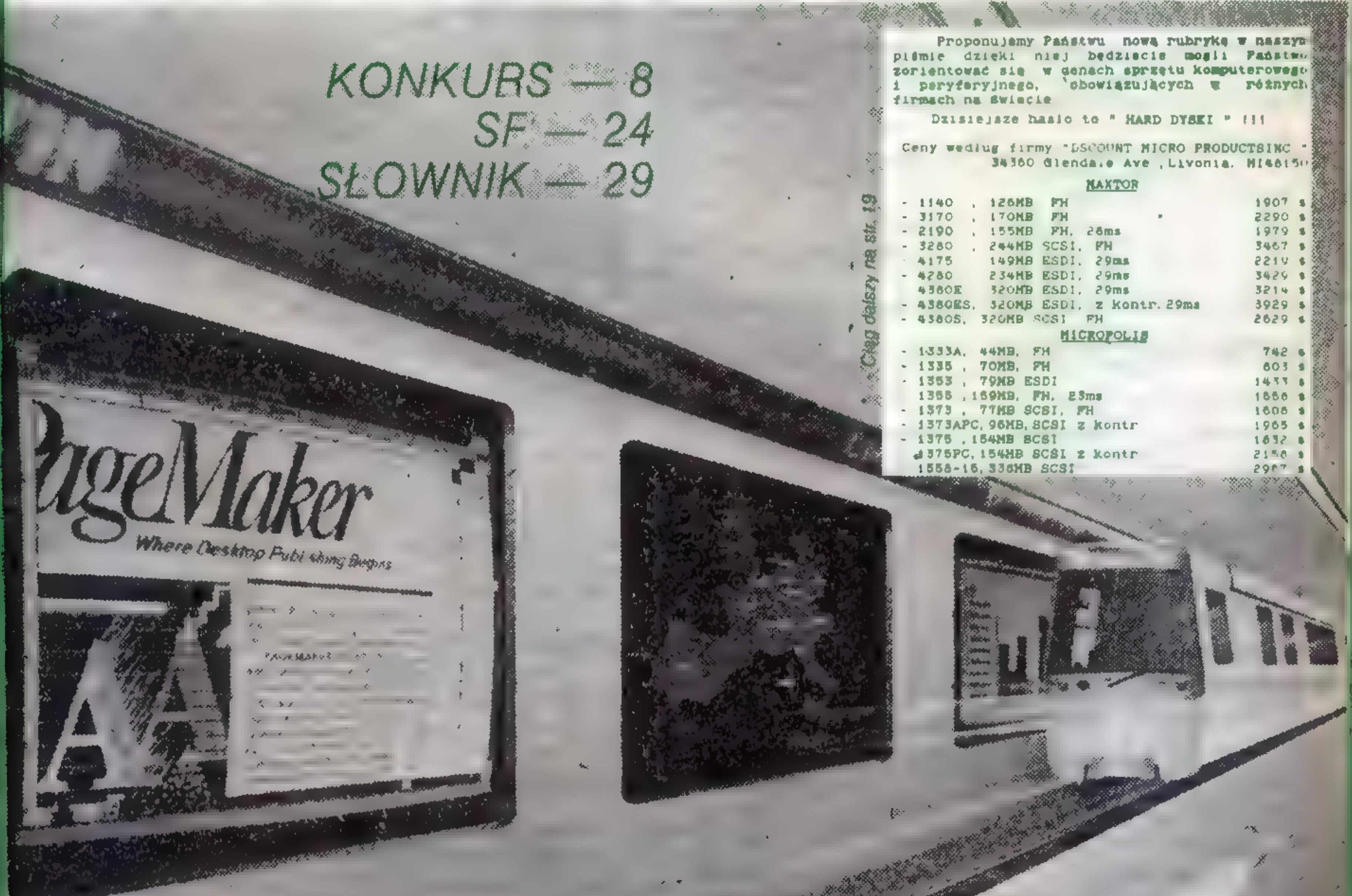
MAXTOR

Model	Cena
- 1140 , 126MB FH	1907 $
- 3170 , 170MB FH	2290 $
- 2190 , 155MB FH, 28ms	1979 $
- 3280 , 244MB SCSI, FH	3467 $
4175 149MB ESDI, 29ms	2219 $
- 4280 234MB ESDI, 29ms	3429 $
4380E 320MB ESDI, 29ms	3214 $
- 4380ES, 320MB ESDI, z kontr. 29ms	3929 $
- 4380S, 320MB SCSI FH	2629 $

MICROPOLIS

Model	Cena
- 1333A, 44MB, FH	742 $
- 1335 , 70MB, FH	803 $
- 1353 , 79MB ESDI	1433 $
1355 , 159MB, FH, 23ms	1858 $
- 1373 , 77MB SCSI, FH	1608 $
- 1373APC, 96MB, SCSI z kontr	1965 $
- 1375 , 154MB SCSI	1832 $
1375PC, 154MB SCSI z kontr	2150 $
1558-15, 336MB SCSI	2967 $

Ciąg dalszy na str. 19



Tym numerem, trzydziestym już od chwili ukazania się pisma, kończymy rok 1988. Był to kolejny rok rozwoju informatyki i coraz powszechniejszych jej zastosowań. Wprowadzanie tej technologii wymuszane jest koniecznością nie tylko postępu technicznego, ale i organizacyjnego. W czasach, w których informacja jest towarem, tym cenniejszym im świeższym, komputery nabierają wartości. Oczywiście mówiąc o komputerze pod uwagę brać trzeba także oprogramowanie, które z kupy żelaztwa czyni sprzęt użyteczny. W konsekwencji już wkrótce powstanie sytuacja, w której prestiż człowieka zależeć będzie nie od tego jakie posiada informacje, ale jak z nich potrafi skorzystać. Z jednej strony informatyka znacznie usprawni procesy decyzyjne w przedsiębiorstwie dostarczając aktualny obraz kondycji ekonomicznej firmy, ale z drugiej pozbawi dogodnego tłumaczenia w razie wpadki typu: *ja jerszcze tego nie wiedziałem.*

Ale to wszystko być może w przyszłym roku, który już na wstępie zachęca przedsiębiorczych do twórczego działania, a zdolnym i pracowitym daje szansę, choć bez gwarancji sukcesu, którego naszym Czytelnikom i Sympatykom życzymy.

Wszystkiego najlepszego w Nowym Roku.

REDAKCJA

W NUMERZE:

Nowe instwukcje ZX-81 — str. 3
Współpraca ATARI 800XL z drukarkami D-100x, DZM-180 KSR i monitorem MERA — str. 6
„Ucieczka Orła" — str. 8
Przesuwanka — str. 10
Lilavati (11) — str. 11
zanim powstanie mapa — str. 12
Informatyka w szkole — str. 15
Twarde dyski — ceny — str. 19
Atari symuluje zjawiska falowe — str. 20
Zmiana liczb podanych cyfrowo na postać słowną — str. 22
Telegrafista — str. 23
SF — Geny — str. 25
Giełda pomysłów — str. 26
Programy narzędziowe na CPC6128 (2) — str. 28
Informatyczny słownik — str. 29
Liga myślących — str. 31
Krzyżówka — str. 32
Komputerowi włamywacze — str. 32

PROGRAM 65

ATARI

Zamek cyfrowy

Tytuł ten jest trochę zwodniczy. Prezentowany program nie służy do programowania komputera jako sterownika zamka elektronicznego. Komputer w zależności od stopnia trudności, który wynosi od 2 do 15 cyfr, kodowanych losowo liczbami od 1 do 9. Twoje zadanie, to odgadnięcie szyfru. Możesz pomylić się dwa razy, trzecia pomyłka powoduje wszczęcie alarmu, a tym samym zakończenie gry.

J. J.

```
PS 750 REM ************************
YM 752 REM *                      *
RG 754 REM *    ZAMEK CYFROWY     *
TQ 756 REM *     * ATARI *        *
ZB 757 REM *                      *
QQ 758 REM ************************
RU 759 DIM S$(1),X$(15)
BK 760 ? CHR$(125)
WT 762 ? :? "ILE CYFR ";:INPUT N
LW 764 IF N<2 OR N>15 THEN GOSUB 81
   0:GOTO 760
EZ 766 FOR K=1 TO N
OJ 768 S$=CHR$(INT(RND(0)*10))
BV 770 X$(LEN(X$)+1)=S$:NEXT K
HY 784 Q=0
FD 786 FOR K=1 TO N
FZ 787 IF Q=3 THEN ? :? "ALARM !":F
   OR C=1 TO 15:GOSUB 810:NEXT C:EN
   D
FE 788 ? K;" CYFRA=";:INPUT Z
GO 790 X=ASC(X$(K,K))
VI 794 GOSUB 810:Q=Q+1
RE 798 IF Z=X THEN ? "ODGADLES ";K;
   " CYFRE":Q=0:NEXT K
WD 800 IF Z<X THEN ? "WYZSZA":GOTO
   787
DF 802 IF Z>X THEN ? "NIZSZA":GOTO
   787
JV 804 ? :? "SZYFR=";:FOR K=1 TO N:
   ? ASC(X$(K,K));:NEXT K:END
JJ 810 FOR D=15 TO 1 STEP -1:SOUND
   0,100,10,D:NEXT D:SOUND 0,0,0,0
ZM 812 RETURN
```

Nowe instrukcje ZX-81

Ireneusz PARAFIAŃCZUK

Najtańszym mikrokomputerem na naszym rynku jest ZX 81. Jest to już klasyczny model o parametrach i możliwościach znacznie gorszych od produkowanych obecnie. Jednak niska cena długo jeszcze zapewni mu powodzenie. Ireneusz Parafiańczuk przedstawia sposoby wprowadzenia 5 nowych instrukcji w celu zwiększenia możliwości ZX 81. Są to instrukcje READ DATA, natychmiastowa inwersja ekranu, a także podawanie liczby wolnych bajtów pamięci.

W celu wprowadzenia nowych instrukcji należy wpisać program z wydruku 1 (linię 1 REM, w której musi być nie mniej niż 120 gwiazdek lub innych znaków, najlepiej jest wprowadzić w trybie FAST. (Następnie należy go uruchomić, wtedy ukaże się liczba 16 514 i znak zachęty do wprowadzenia zmiennej łańcuchowej. Liczba 16 514 jest adresem komórki pamięci, od której zaczyna się program.

WYDRUK 1

```
  1 REM ************************
*********************************
*********************************
*********************************
*****************************
 10 LET A=VAL "16514"
 15 LET B=0
 20 SCROLL
 30 PRINT A;
 40 INPUT A$
 50 IF LEN A$ THEN GOTO VAL "10
0"
 60 LET A=A-SGN PI
 70 IF A<VAL "16514" THEN GOTO
VAL "10"
 80 LET B=B-PEEK A
 90 GOTO 20
100 POKE A,VAL A$
110 LET B=B+PEEK A
120 PRINT TAB 10;A$;TAB 20;B
130 LET A=A+SGN PI
140 IF A<16634 THEN GOTO 20
150 STOP
```

Należy wprowadzać kolejno liczby ze środkowej kolumny wykazują, obserwując cały czas numer komórki pamięci oraz sumę kontrolną w trzeciej kolumnie. Jeżeli popełnimy błąd po wprowadzeniu liczby, należy jeszcze raz nacisnąć NEW LIWE, a program powróci do poprzedniego adresu (naciskając ciągle NEW LINE można powrócić do adresu początkowego). Po wprowadzeniu ostatniej liczby pojawi się komunikat o kodzie 9/150, co oznacza, że program został zatrzymany na polecenie STOP w linii nr 150.

W ten sposób wszystkie nowe instrukcje znajdują się w linii 1 REM. Wpiszmy polecenie LIST, a zobaczymy tylko linię 1 REM, w której są dziwne znaki. Jest to właśnie nasz kod maszynowy. Aby wylistować pozostałe linie, należy wpisać LIST w 10. Pozostaje nam tylko usunąć (skasować) wszystkie linie oprócz 1 wpisując numer linii i klawisz ENTER. Następnie nagrać kod maszynowy z linii 1 na kasetę (w celu późniejszego wykorzystywania) przez wprowadzenie linii 9998 SAVE „KOD" i napisanie w trybie bezpośrednim GOTO 9998. Po zapisaniu programu na kasetę (najlepiej zapisać w dwóch miejscach) możemy do niego dopisywać nowe linie wykorzystując nowe instrukcje znajdujące się w linii 1 REM. Po wpisaniu nowych instrukcji linię 9998 należy skasować (ponieważ nowy program może mieć inną nazwę) lub wpisać nową nazwę programu i dopisać linię 9999 GOTO 10, a następnie nagrać go na kasetę poleceniem GOTO 9998. W ten sposób nagrany program po wczytaniu go z kasety do pamięci za każdym razem zostanie uruchomiony. A oto wykaz nowo otrzymanych komend, sposób ich wywołania i opis.

RAND USR 16608 zawiera wszystko, co jest na ekranie na jego inwencję. Może służyć do natychmiastowego zaczernienia ekranu.

PRINT USR 16593 — podaje liczbę wolnych bajtów pamięci.

500 REM 10.3 SIN A, są to dwie przykładowe linie

520 REM ABE, ALFA, z instrukcjami PSEUDO-DATA.

Zawartością ich mogą być liczby lub łańcuchy, rozdzielone przecinkami. UWAGA! W całym programie (z wyjątkiem linii 1 nie może być żadnych innych linii z instrukcją REM zawierającą coś innego niż dane PSEUDO-DATA, czyli nie może w nim być żadnych komentarzy, nawet nazwy programu.

RAND USR 16514 Wykonuje komendę RESTORE. Musi być użyte, zanim jakakolwiek DATA będzie odczytywana. Może być również wykonywane dla odczytania danych więcej niż jeden raz.

```
10 LET A$ = ""
20 RAND USR 16556
30 LET A VAL A$
```

Sekwencja ta zastępuje instrukcję READ. Czyta kolejne dane z instrukcji REM i podstawia ją pod zadeklarowaną wcześniej pustą zmienną łańcuchową (nie musi to być konieczne A$, ale także B$ czy Z$. W wypadku odczytywania łańcucha znakowych należy pominąć linię 30, w wypadku liczb jest ona konieczna.

Chcąc wykorzystać nowe instrukcje razem z innymi programami w kodzie maszynowym należy bardzo uważać, gdyż może się okazać, iż ten drugi program również znajduje się w linii 1 (spotkałem się z tym już kilkakrotnie). W tym wypadku niezbędne jest dostosowanie naszego programu (gdyż nie znamy możliwości ani uwarunkowań tamtego programu) w miejsce pamięci niedostępne dla programów (żaden program tam nie „wejdzie"), a także dla instrukcji NEW czy CLEAR. Jest to tzw. RAMTOP, miejsce pamięci określane przez zmienne systemowe tzn. ustawiane po włączeniu komputera do sieci, ale które można zmienić, tzn. przesuwać. Zmienna RAMTOP (adres pamięci) jest zapisany w dwóch komórkach o adresie 16388 i 16389 — jest to drugi sposób zwiększenia liczby instrukcji. Najwygodniejsze dla ZX 81 z pamięcią 16 K bajtów i 64 K bajtów jest zapisanie naszego programu w ten sposób:
to ustawienie nowego adresu RAMTOP-32618

```
10 POKE 16388,136
20 POKE 16389,127
30 CLS
40 POKE 32648,33
50 POKE 32649,250
60 POKE 32650,64
```

i tak dalej aż do komórki o numerze 32767 opisując kolejne liczby z wykazu.

Zmieniając niektóre wartości: dla komórki o numerze: 32649 zmienić na 130

"	32684	"	139
"	32685	"	127
"	32700	"	165
"	32701	"	127

Na zakończenie napiszemy linie:

```
9997 NEW
9998 SAVE „KOD"
9999 GOTO 10
```

Dla wersji 1 K bajtów niezbędne są następujące zmiany w programie zasadniczym: ustawienie RAMTOP dla wersji z 1K bajtem pamięci

```
10 POKE 16388,136
20 POKE 16389,67
30 CLS
40 POKE 17288,33
50 POKE 17289,250
```

i dalej numery komórek aż do 17407, wpisując liczby z wykazu, lecz zmieniając niektóre wartości. I tak dla:

komórki o numerze 17289 zmienić na 130
" 17324 " 139
" 17325 " 67
" 17340 " 165
" 17341 " 67

Na koniec programu dla wersji 2 x 91 1K należy wpisać te same linie co dla wersji 16K i 64K pamięci tzn.

```
9997 NEW
9998 SAVE „KOD"
9999 GOTO 10
```

Program nasz przeniesiony za RAMTOP będzie miał inne adresy do wywoływania poszczególnych instrukcji. Zmienione adresy zamieszczone są w tabeli:

instrukcja	wersja pamięci	
	1K	16K i 64K
PSEUDO-RESTORE (RAND USR...)	17 288	32 648
PSEUDO-READ (RAND USR...)	17 330	32 690
INWERSJA EKRANU (RAND USR...)	17 382	32 742
SPRAWDZANIE PAMIĘCI (PRINT USR...)	17 367	32 727

16514	33	33
16515	250	283
16516	64	347
16517	62	409
16518	234	643
16519	1	644
16520	255	899
16521	255	1154
16522	237	1391
16523	177	1568
16524	1	1569
16525	5	1574
16526	0	1574
16527	167	1741
16528	237	1978
16529	66	2044
16530	62	2106
16531	118	2224
16532	190	2414
16533	40	2454
16534	5	2459
16535	9	2468
16536	34	2502
16537	123	2625
16538	64	2689
16539	201	2890
16540	9	2899
16541	24	2923
16542	230	3153
16543	42	3195
16544	123	3318
16545	64	3382
16546	62	3444
16547	118	3562
16548	190	3752
16549	204	3956
16550	133	4089
16551	64	4153
16552	126	4279
16553	35	4314
16554	24	4338
16555	236	4574
16556	42	4616
16557	20	4636
16558	64	4700
16559	43	4743
16560	229	4972
16561	43	5015
16562	43	5058
16563	227	5285
16564	229	5514
16565	205	5719
16566	159	5878
16567	64	5942
16568	225	6167
16569	6	6173
16570	26	6199
16571	184	6383
16572	40	6423
16573	13	6436
16574	119	6555
16575	35	6590
16576	227	6817
16577	52	6869
16578	32	6901
16579	3	6904
16580	35	6939
16581	52	6991
16582	43	7034
16583	227	7261
16584	167	7428
16585	48	7476
16586	233	7709
16587	227	7936
16588	225	8161
16589	0	8161
16590	195	8356
16591	157	8513
16592	20	8533
16593	42	8575
16594	28	8603
16595	64	8667
16596	68	8735
16597	77	8812
16598	33	8845
16599	0	8845
16600	0	8845
16601	57	8902
16602	191	9093
16603	237	9330
16604	66	9396
16605	68	9464
16606	77	9541
16607	201	9742
16608	42	9784
16609	12	9796
16610	64	9860
16611	68	9928
16612	77	10005
16613	22	10027
16614	59	10086
16615	203	10289
16616	34	10323
16617	42	10365
16618	16	10381
16619	64	10445
16620	10	10455
16621	186	10641
16622	40	10681
16623	3	10684
16624	198	10882
16625	128	11010
16626	2	11012
16627	3	11015
16628	191	11206
16629	237	11443
16630	66	11509
16631	32	11541
16632	240	11781
16633	201	11982



Po wpisaniu całego programu (modyfikowanego zależnie od używanej wersji pamięci) przenosimy go na kasetę instrukcją GOTO 9998 (w trybie natychmiastowym). Program najpierw zostanie nagrany na kasetę, po czym sam się uruchomi i wykona. Po wykonaniu programu jako niepotrzebny skasuje się, zostawiając za RAMTOP kod maszynowy. I teraz za każdym razem, gdy chcemy załadować nowe instrukcje, piszemy po prostu LOAD „KOD", włączamy magnetofon, naciskamy NEW LINE, a program załaduje się do pamięci, sam się uruchomi, wykona i skasuje. Dla sprawdzenia, czy kod maszynowy jest w RAMTOP — napiszemy PRINT PEEK 32648. Powinna pokazać się cyfra 33. Można również dodać linię na 35 PRINT „LADUJE SIE KOD", aby naocznie się przekonać, że program się wykonuje (gdyż ekran po załadowaniu będzie cały czas czysty). Na koniec dwa programy wykorzystujące nowe komendy:

„KSIĄŻKA TELEFONICZNA" (WYDRUK 2), w której imiona i numery telefonów wpisuje się w linii 130 (można oczywiście dopisać nowe linie, ze swoimi numerami telefonów. Kursor znacznika linii ustawić na linię 130 i wcisnąć SHIFT + EDIT.

Linia 150 musi kończyć program.

```
WYDRUK 2

  20 PRINT "PODAJ IMIE"
  30 INPUT A$
  40 RAND USR 32648
  50 LET B$=""
  60 RAND USR 32690
  70 IF B$="NN" THEN PRINT "OSOB
A NIEZNANA"
  75 IF B$="NN" THEN GOTO VAL "2
0"
  80 LET C$=""
  90 RAND USR 32690
 100 IF B$<>A$ THEN GOTO VAL "50
"
 110 PRINT A$;"=";C$
 120 GOTO VAL "20"
 130 REM JACEK,245-457,ZYGMUNT,8
7-03-21,ZOSIA,132384,
 150 REM NN,
1000 SAVE "KSIAZKA"
1050 GOTO VAL "20"
```

„WYKRESY" (WYDRUK 3) kreśli na ekranie wykresy różnych funkcji (zawartych w instrukcji REM) oraz pozwala na wprowadzenie własnej funkcji.

```
  10 LET F=SGN PI
  20 GOTO CODE " COPY "
  24 CLS
  25 IF F THEN PRINT "FUNKCJA"
  30 INPUT A$
  35 IF F THEN PRINT "DOLNA GRAN
ICA"
  40 INPUT X
  45 IF F THEN PRINT "GORNA GRAN
ICA"
  50 INPUT A
  60 LET K=X
  70 FAST
  80 CLS
  85 LET F=NOT PI
  90 LET DX=(A-X)/CODE "Z"
 100 LET H=VAL A$
 110 LET L=H
 120 FOR I=NOT PI TO CODE "Z"
 130 IF H<VAL A$ THEN LET H=VAL
A$
 140 IF L>VAL A$ THEN LET L=VAL
A$
 150 LET X=X+DX
 160 NEXT I
 170 LET X=K
 180 SLOW
 190 FOR I=NOT PI TO CODE "Z"
 200 PLOT I,VAL "43"*(VAL A$-L)/
(H-L)
 210 LET X=X+DX
 220 NEXT I
 230 GOSUB VAL "400"
 240 GOTO VAL "30"
 300 RAND USR 32648
 310 FOR A=VAL "-2" TO VAL "10"
 320 FOR B=NOT PI TO VAL "24" ST
EP VAL "8"
 330 LET A$=""
 340 IF B OR A<NOT PI THEN RAND
USR 32690
 350 IF NOT B AND A>-SGN PI THEN
 LET A$=STR$ A
 360 PRINT TAB B;A$;
 370 NEXT B
 380 NEXT A
 390 REM ***,WYBOR,FUNKCJA,***,N
UMER,FUNKCJA,ZAKRES,,WPROWADZ,SW
OJA,FUNKCJE,
 400 INPUT B
 420 IF NOT B THEN RETURN
 430 RAND USR 32648
 440 FOR A=SGN PI TO VAL "3*B+8"
 450 LET A$=""
 460 RAND USR 32690
 470 NEXT A
 480 FOR A=-SGN PI TO SGN PI
 490 LET B$=""
 500 RAND USR 32690
 510 IF A<NOT PI THEN LET A$=B$
 520 IF NOT A THEN LET X=VAL B$
 530 NEXT A
 540 LET A=VAL B$
 550 GOTO VAL "60"
1000 REM SIN X,0,2*PI,COS X,0,2*
PI,TAN X,-1,1,SIN X+SIN (2*X)+SI
N (3*X),0,4*PI,ASN X,-1,1,LN X,.
1,2.7,EXP X,-1,1,ABS (X-INT X-.5
),0,3,1/X,.1,2,EXP -(X*X),-2,2,
2005 SAVE "WYKRESY"
2010 GOTO VAL "10"
```

Oba programy napisane są w sposób wykorzystujący nowe instrukcje znajdujące się za RAMTOP dla wersji 64K i 16K. Aby programy działały z kodem w linii 1 REM, a także dla wersji 1K (za RAMTOP), należy wstawić odpowiednie adresy wywoływania nowych funkcji (wg tabeli lub wg wykazu komend na początku artykułu) w linie wykorzystujące kod maszynowy.

Dla wersji programu z kodem w linii „1" należy zmienić adresy wywoływania nowych instrukcji.

Ireneusz PARAFIAŃCZUK

PROGRAM 66

ZX-81

Skoki narciarskie

```
  1 REM ************************
  2 REM *  SKOKI NARCIARSKIE   *
  3 REM *       ZX-81          *
  4 REM ************************
 10 PRINT "SKOKI NARCIARSKIE"
 15 LET A=INT(RND*5)+100
 20 PRINT "ODLEGLOSC ";A;" METR
OW"
 25 PRINT "WYSOKOSC SKOKU ?"
 30 INPUT B
 35 PRINT "WAGA ?"
 40 INPUT C
 42 CLS
 45 LET D=INT
 50 PRINT " SZYBKOSC ";D;" KM./
GODZ."
 55 PRINT
 60 LET E=INT((D*(B/10)*(RND*5+
5))/10)
 65 IF E<A THEN GOTO 80
 70 PRINT "ZA DUZA SZYBKOSC"
 75 GOTO 160
 80 IF E<50 OR D<20 THEN GOTO 9
0
 85 GOTO 100
 90 PRINT "SKOK ZBYT WOLNY"
 95 GOTO 160
105 IF RND*5+1=1 THEN GOTO 115
110 GOTO 130
115 PRINT "SLABE LADOWANIE"
130 PRINT "MAX. WYSOKOSC ";A
135 PRINT "OSIAG ";E
140 LET F=INT(RND*11)
145 PRINT "POMYLKA ";INT(RND*1
1)
150 PRINT "PUNKTY ";300-A-(F*7
)
155 STOP
160 PRINT
165 PRINT "UPADEK"
```

Współpraca ATARI 800XL z drukarkami D-100, DZM-180 KSR i monitorem MERA-7952

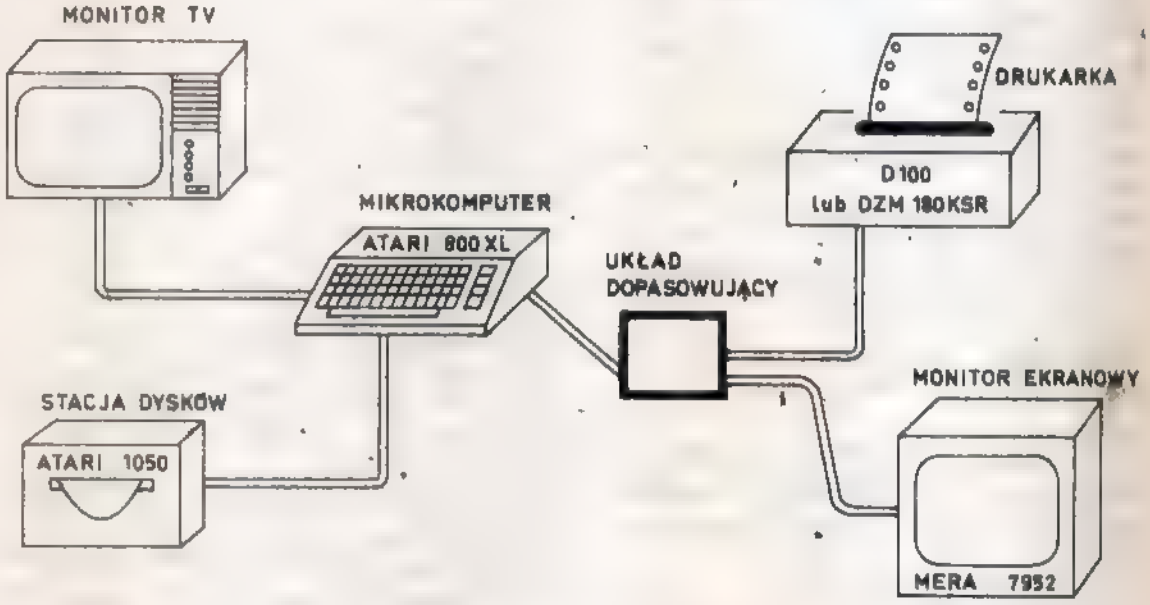
(CZĘŚĆ I)

Użytkownikom mikrokomputera ATARI 800XL (oraz kompatybilnych ATARI 130XE i 65XE) mającym również dostęp do takich urządzeń, jak monitor Mera-7952 i drukarka D-100 (lub DZM-180KSR), proponujemy stworzenie podręcznego i przydatnego zestawu komputerowego przedstawionego na rys. 1. Z punktu widzenia pokazanego połączenia w dalszej części artykułu ograniczono się do podania danych technicznych wykorzystywanych złącz standardowych, zwanych popularnie interfejsami. Znana i w ostatnich latach bardzo reklamowana w naszej literaturze drukarka D-100 produkcji polskiej wyposażona jest najczęściej w dwa złącza wejścia-wyjścia: CENTRONICS i V-24. W konfiguracji pokazanej na rys. 1 wykorzystany został interfejs V-24. Jest on przeznaczony do szeregowej transmisji informacji między komputerem a drukarką. Interfejs ten wykorzystuje 25-stykowe złącze szufladowe „Cannon". Poziomy i przeznaczenie sygnałów zamieszczono w tabelach 1 i 2.

Drukarka DZM-180KSR i monitor MERA-7952 również mają wspomniany interfejs V-24 i wykorzystują go do współpracy z komputerem. Czytelnicy i praktycy komputerowi znają dobrze wszechstronne zastosowanie tego interfejsu. Pomimo że istnieje obecnie wiele innych, często znacznie lepszych rozwiązań, nadal z powodzeniem stosowany jest standard V-24 i jego odpowiednik RS—232C. Składa się na to między innymi niski koszt realizacji, przydatność w różnych zastosowaniach i prosta konstrukcja. Jego uniwersalność pozwala na zredukowanie liczby linii (w całości wyróżnionych w tabeli 2), a nawet w minimalnych zestawach wykorzystanie tylko linii transmisji danych.

Mikrokomputer Atari 800XL nie ma w zespole wejścia-wyjścia interfejsu V-24. Dlatego też wykorzystano jedno z gniazd joysticków, które wewnątrz ATARI połączone są z programowalnym 8-bitowym portem wejścia-wyjścia tzw. PIA (układ 6520). Są to 9-stykowe gniazda typu „Cannon". Opis styków tych gniazd zawiera tabela 3. Porty układu PIA pracują w standardzie TTL, co uniemożliwia bezpośrednie podłączenie do nich linii interfejsu V-24.

W celu osiągnięcia połączenia wykonany został układ dopasowujący. Jego schemat ideowy przedstawia rys. 2. Układ dopasowu-



Rys. 1. Szkic sprzężeń podstawowych elementów mikrokomputera.

POZIOM	WARTOŚCI NAPIĘCIA	FUNKCJA STERUJĄCA
LOGICZNA „1"	−12V÷−3V	NIE /NEGACJA/
LOGICZNE „0"	+3V÷+12V	TAK /POZYCJA/

Tabela 1: Poziomy w standardzie V-24

NR STYKU	NR SYGNAŁU	NAZWA SYGNAŁU	OPIS SYGNAŁU
1	101	P.GND	Ziemia ochronna /ekran/
7	102	S.GND	Ziemia sygnałowa
2	103	TxD	Dane nadawane
3	104	RxD	Dane odbierane
4	105	RTS	Żądanie nadawania
5	106	CTS	Gotowość nadawania
6	107	DSR	Gotowość urządzenia transmisji danych.
20	108	DTR	Gotowość terminala
8	109	DCR	Poziom sygnału odbieranego przez urządzenie
25		LBRM	Zajętość bufora drukarki.

Tabela 2: Sygnały interfejsu V-24

jący zasadniczo spełnia dwa zadania: — dopasowuje poziomy logiczne sygnałów w standardzie TTL i V-24, — pozwala na równoległe podłączenie do jednego wyjścia mikrokomputera dwóch urządzeń zewnętrznych (drukarka, monitor).

Układ został uproszczony do minimum, zapewniając jednak możliwości wykorzystania wszystkich trybów pracy drukarki D-100. Dzięki temu uzyskano niski koszt układu oraz możliwość wykonania go przez średnio zaawansowanych amatorów. Sposób wykorzystania linii interfejsu V-24 i gniazda joysticka przedstawiono w tabeli 4. Od strony mikrokomputera do podłączenia układu dopasowującego wykorzystane zostały 4 styki pierwszego gniazda, z których dwa do transmisji (1 i 2 styk, tj. 0 i 1 bit portu A układu PIA). Pozostałe dwa styki to „ziemia sygnałowa” i napięcie +5V. Z urządzeniami zewnętrznymi układ dopasowujący współpracuje za pośrednictwem 7 styków (tabela 4). Na uwagę zasługują:

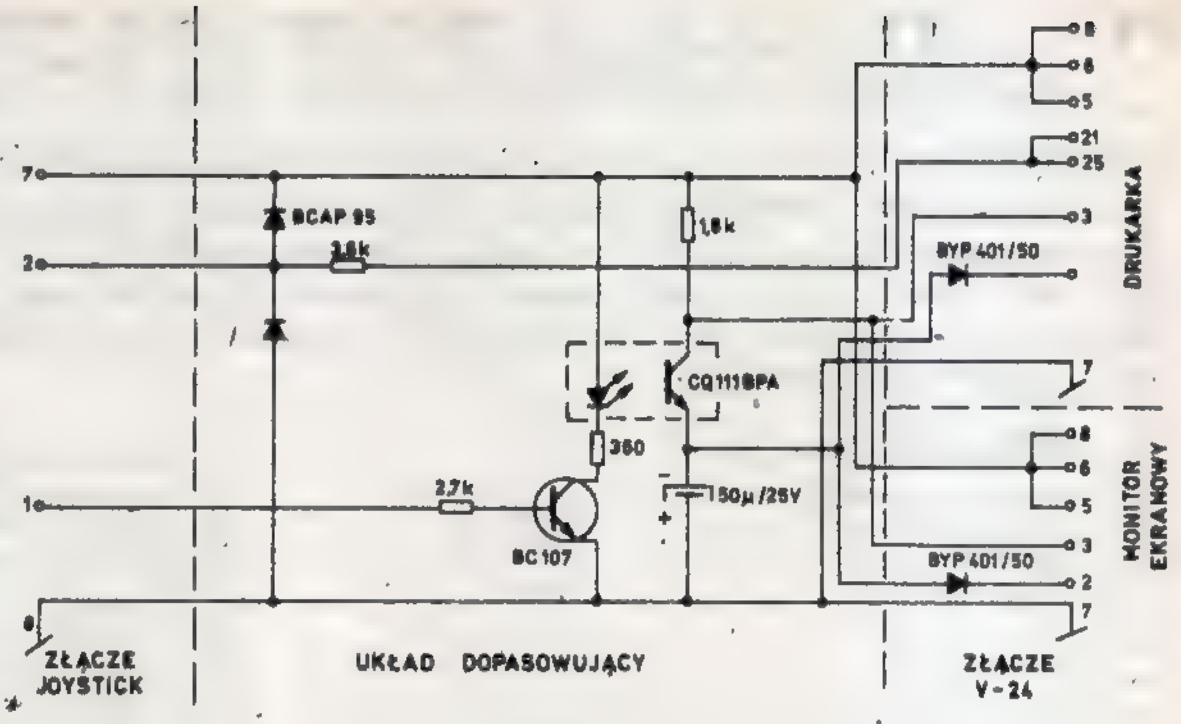
— LBRM — sygnał zajętości bufora, występujący tylko w drukarkach,

— CTS, DSR, DCR — sygnały informujące urządzenia zewnętrzne o operatywności komputera,

— TxD — sygnał wyjściowy urządzeń zewnętrznych; w tej konfiguracji jest on cały czas „1” logiczną, czyli ma napięcie ujemne, które wykorzystano do zasilania układu dopasowującego.

Oprócz układowej realizacji dopasowania, w celu sprzężenia mikrokomputera z urządzeniami, należało stworzyć aparat programowy, który będzie przedstawiony w drugiej części artykułu.

Zbigniew FIGUŁA
Artur KOŁOSOWSKI



Rys. 2

NAZWA STYKU	NAZWA SYGNAŁU
1	Joystic - Forward
2	Joystic - Back
3	Joystic - Left
4	Joystic - Right
5	Analog - B /Paddle/
6	Trigger
7	5 Volts
8	Gronnd
9	Analog - A /Paddle/

Tabela 3: Sygnały gniazda Joystick'a

ZŁĄCZE JOYSTICK MIKROKOMPUTERA		ZŁĄCZE V-24		URZĄDZENIA ZEWNĘTRZNE WYKORZYSTUJĄCE SYG.	KIERUNEK SYGNAŁU I OPIS
NR STYKU	NAZWA SYGNAŁU	NR STYKU	NAZWA SYGNAŁU		
1	FORWARD	3	R×D	Drukarki Monitor ekranowy	Przesyłanie danych z mikrokomputera do urządzeń
2	BACK	21 25	LBRM	Drukarki 21-DZM-180KSR 25-D100	Przesyłanie sygnału zajętości z drukarek do mikrokomputera
7	+5V	5 6 8	CTS DSR DCR	Drukarki Monitor ekranowy	Przesyłanie sygnału operatywności /gotowości/ z mikrokomput. do drukarek
8	GROUND	7	S.GND	Drukarki Monitor ekranowy	Ziemia sygnałowa
—	—	2	T×D	Układ dopasowujący	Ujemne napięcie zasilające układ dopasowujący

Tabela 4: Zestawienie sygnałów dopasowujących

„Ucieczka Orła"

W tegorocznym konkursie na scenariusz gry komputerowe, I miejsce zajęła „Ucieczka Orła". Dziś drukujemy ten scenariusz ogłaszając tym samym II etap konkursu.
Do 1 maja 1989 r. czekamy na program komputerowy (sprzęt dowolny) napisamy na podstawie zwycięskiego scenariusza, bądź też oparty na własnym, oryginalnym pomyśle.
Na zwycięzcę czeka komputer.

I. Ogólny opis gry

Gra składa się z siedmiu epizodów:
1) odwrócenie „Orła" dziobem w kierunku wyjścia z portu,
2) uniemożliwienie demontażu żyrokompasu,
3) sondowanie portu,
4) ucieczka,
5) przejście przez Bałtyk,
6) przejście przez pole minowe na Skagerraku,
7) wejście do portu w Rosyth.

W każdym epizodzie gracz ma prawo do wykonania trzech prób. Zaliczenie epizodu sygnalizowane jest pojawieniem się planszy z ręką zaciśniętą w pięść i kciukiem skierowanym w górę, niezaliczenie — podobną planszą, ale z kciukiem skierowanym w dół.

Po zakończeniu epizodu gracz powinien mieć możność:
1) rozpoczęcia gry od początku,
2) przejścia do następnego lub dowolnie wybranego epizodu.

Pozwala to na doskonalenie umiejętności w trudniejszych dla danego gracza epizodach bez konieczności powtarzania całej gry od początku, w pierwszym okresie poznawania gry oraz wybór tylko ulubionych epizodów, gdy gra jako całość już się znudzi.

Gra jest zakończona, gdy gracz pomyślnie zaliczy wszystkie epizody. Gracz jest dodatkowo punktowany, przy czym:
1) dostaje ujemne punkty za każdą nie zaliczoną próbę lub storpedowanie jednostki sojuszniczej,
2) dostaje dodatnie punkty za storpedowanie każdego statku lub okrętu nieprzyjacielskiego oraz niezauważenie przez nieprzyjacielskie jednostki

Podstawową umiejętnością, którą powinien opanować gracz, jest prowadzenie (manewrowanie) okrętu podwodnego. Komendy (dokładna lista komend do ustalenia w czasie programowania) na stery, silniki itp., powinny być wprowadzane z klawiatury w pełnym brzmieniu np.: „ster lewo 15", „oba motory naprzód" itp., wyświetlane na ekranie, a błędy sygnalizowane (błędnie wydana komenda nie może być wykonana).

Oczywiście, skala czasu (i pochodne, a przede wszystkim skala grafiki) powinna być tak dobrana, aby pozwalała na symulowanie rzeczywistego tempa rozwoju zdarzeń. W tym też celu w trakcie programowania należy uwzględnić podstawowe parametry ruchu okrętu podwodnego

Ważnym elementem gry powinien być dźwięk. Podjęciu przez gracza odpowiednich decyzji (wydaniu komend) powinny towarzyszyć określone sygnały dźwiękowe, na przykład klakson na zanurzenie czy dzwonki alarmowe. Należy również symulować: szum motorów (diesli), szum śrub jednostek przechodzących nad „Orłem".

Grę otwiera plansza ukazująca sylwetkę „Orła" (widok z boku z podniesioną banderą), słychać pierwsze takty piosenki „Morze nasze morze".

II. Przebieg gry

Epizod 1. Odwrócenie „Orła" dziobem w kierunku wyjścia z portu

Po internowaniu „Orła" zabrano z niego dziennik nawigacyjny i mapy, a po przeholowaniu go w głąb basenu portowego przystąpiono do jego rozbrajania: skonfiskowano znajdującą się na okręcie broń palną, wyjęto z dział zamki i rozpoczęto wyładowanie torped.

W momencie, gdy załoga podjęła decyzję o ucieczce, wyłoniło się kilka zasadniczych spraw, które warunkowały jej powodzenie. Przede wszystkim okręt musiał zostać obrócony dziobem w kierunku wyjścia z basenu portowego. Następnie należało uniemożliwić dalsze rozbrajanie okrętu, demontaż żyroskopu i radiostacji oraz rozłączenie wałów śrubowych. Należało również przeprowadzić pomiary głębokości basenu w celu znalezienia bezpiecznego przejścia. Trzy pierwsze epizody nawiązują do tych przygotowań.

Aby „Orzeł" został odwrócony dziobem w kierunku wyjścia z portu gracz musi wygrać (w ustalonej liczbie ruchów) grę typu „Master Mind".

Plansza podzielona jest na dwie części. Dolna część przedstawia widok z góry na „Orła" przycumowanego do nabrzeża, rufą do wyjścia z portu. W górnej części dwie sylwetki ludzkie dowódcy „Orła" i dowódcy portu stojącego obok tablicy. Po kilku sekundach dowódca portu ruchem ręki „rzuca" na tablicę cztery znaki zapytania. Zaczyna się gra. Po odgadnięciu liczby „Orzeł" zostaje obrócony dziobem w kierunku wyjścia z portu.

Epizod 2. Uniemożliwienie demontażu żyrokompasu

Uniemożliwienie demontażu żyrokompasu polega na wygraniu gry zręcznościowej. Plansza może przedstawiać labirynt stylizowany na przekrój łodzi podwodnej. W centralnym punkcie labiryntu znajduje się żyrokompas. Zadaniem gracza jest niedopuszczenie członków ekipy demontującej do żyrokompasu.

Epizod 3. Sondowanie portu

Zadaniem gracza jest znalezienie bezpiecznego przejścia przez basen portowy i na trasie wyjściowej z basenu na redę.

Gracz może znajdować się w dwóch stanach:
a) sondowanie (odpowiedni rysunek bączka), który jest stanem początkowym,
b) łowienie ryb (odpowiedni rysunek bączka).

(Przejście ze stanu a) do b) odbywa się na przykład przez podwójne naciśnięcie przycisku Fire).

Plansza przedstawia zarysy portu z przycumowanym „Orłem", obok którego stoi bączek. Sterując nim gracz szuka bezpiecznego przejścia: Basen portowy (i trasa wyjściowa) ma kilka poziomów głębokości (np. cztery), które sygnalizowane są dźwiękami o różnych wysokościach) i dodatkowo różnym stopniem zaczernienia specjalnego wskaźnika). Najniższy dźwięk świadczy o tym, że bączek znajduje się w punkcie o właściwej głębokości. W basenie (na trasie) jest kilka bezpiecznych przejść (labirynt) przedzielonych płyciznami. Gracz po znalezieniu się w punkcie o odpowiedniej głębokości zaznacza to miejsce (naciśnięcie odpowiedniego klawisza), co powoduje pojawienie się w tym miejscu krzyżyka, i płynie dalej wsłuchując się w dźwięk i zaznaczając kolejne miejsca. Jeżeli dwa kolejne punkty nie są przedzielone płycizną, to na ekranie pojawia się łącząca je podświetlona linia. Jeżeli linia (i krzyżyk) nie pojawi się, to gracz musi rozpocząć poszukiwanie od ostatnio zaznaczonego punktu. Trasę przejścia wyznacza się aż do ustalonej linii wytyczonej bojami na redzie.

W wykonaniu zadania przeszkadzają kręcące się po porcie motorówki. Gracz musi maskować swoje poczynania, albowiem jego misja może być rozpoznana, a tym samym ucieczka udaremniona. W tym celu gracz przechodzi w stan „łowienia ryb". Przejście w stan „sondowanie" wymaga pewnego czasu i umiejętności. Ściągnięcie wędek odbywa się przez 2—3-krotne powtórzenie (z klawiatury) określonej sekwencji cyfr, ukazujących się w wybranym miejscu ekranu. Pomyłka powoduje, rozpoczęcie czynności od nowa (inna sekwencja cyfr), a przy tym liczba powtórzeń wzrośnie.

Czas wykonania zadania jest ograniczony (od świtu do zmroku). Układ labiryntu powinien być losowo zmieniany po każdym rozpoczęciu gry.

Epizod 4. Ucieczka

Plansza przedstawia basen portowy z przycumowanym „Orłem" i wytyczoną trasą przejścia (widok z góry). Zadaniem gracza jest wyprowadzenie „Orła" z basenu po wytyczonej trasie z określonym, ale wąskim marginesem. Wyjście poza margines oznacza unieruchomienie okrętu na mieliźnie i koniec ucieczki. Ciekawym efektem zwiększającym atrakcyjność gry byłaby możliwość sprowadzenia okrętu z mielizny w wypadku, gdy wszedł na nią z niewielką prędkością. Ucieczkę w basenie portowym utrudniają jedy-

nie reflektory rozjaśniające linie trasy, przez co jest ona momentami słabo widoczna. Odgłosy serii z karabinów maszynowych i smugi pocisków są tylko dodatkowym efektem wizualnym. Dopiero po wyjściu z basenu portowego gracz trafia na dodatkową trudność — ostrza artylerii nabrzeżnej. Z lewej i prawej strony ukazują się pociski zmierzające w rejon znajdowania się okrętu. Gracz manewrując okrętem w granicach wytyczonego szlaku (margines większy niż w basenie) musi uniknąć trafienia pociskiem.

Ucieczka uwieńczona zostaje sukcesem, gdy okręt po osiągnięciu linii wyznaczonej bojami zacznie się zanurzać.

Epizod 5. Przejście przez Bałtyk

Po udanej ucieczce „Orła" z portu w Tallinie jego dowódca, kapitan Grudziński, podjął decyzję: pozostać na Bałtyku i tropić nieprzyjacielskie okręty i statki tak długo, jak na to pozwoli zapas paliwa, prowiantu i słodkiej wody do picia, następnie zaś przedostać się przez Cieśniny Duńskie i dotrzeć do Anglii. Epizod 5. Nawiązuje do bałtyckiej odysei „Orła".

Zadaniem gracza jest przeprowadzenie „Orła" przez Bałtyk i ewentualne tropienie nieprzyjacielskich okrętów i statków. Zagrożeniem są samoloty i okręty nieprzyjacielskie.

Epizod jest dzielony programowo na odcinki długie: dzień (kilka minut) i krótkie: noc (kilkadziesiąt sekund). Przejście odbywa się w dzień, noc jest przeznaczona na ładowanie akumulatorów i odnowienie zapasu powietrza. To założenie, niezgodne z rzeczywistością, służy uatrakcyjnieniu gry.

Okręt poruszać się może w położeniu nawodnym lub podwodnym (w tym na głębokości peryskopowej z wysuniętym peryskopem). W położeniu podwodnym okręt porusza się dwa razy wolniej niż w położeniu nawodnym. Zapas powietrza starcza na 2/3 dnia, a bateria akumulatorowa na ruch z prędkością optymalną przez połowę dnia. To zmusza gracza do stosowania odpowiedniej taktyki prowadzenia okrętu: zasadniczo w położeniu nawodnym, a tylko w chwili zagrożenia w położeniu podwodnym. Dodatkowo taktykę taką należy wymusić przez zwiększenie (zwiększanie) liczby zagrożeń po upływie określonej liczby dni.

Ekran podzielony jest na trzy części. W lewej górnej części wyświetlane są: komendy gracza, kurs okrętu, aktualny kąt obserwacji wzrokowej liczony względem osi podłużnej okrętu, zapas powietrza, stan baterii akumulatorów, wskaźnik „noc — dzień". W prawej, górnej części ekranu zobrazowana jest mapa Bałtyku z zaznaczoną linią brzegową, ważniejszymi miastami itp. W trakcie gry kreślona jest na niej trasa „Orła".

Dolna część ekranu może przedstawiać w zależności od sytuacji 3 obrazy:

a) widok morza w położeniu nawodnym okrętu,
b) widok morza przez peryskop,
c) sylwetkę okrętu (przekrój poprzeczny) na ciemnym tle (w zanurzeniu).

Uwaga! Ideałem byłoby zastosowanie w wariancie c) grafiki trójwymiarowej.

W położeniu nawodnym (i na głębokości peryskopowej) gracz musi w sposób ciągły prowadzić obserwację dookólną. Okręty i statki ukazywać się będą jako sylwetki widziane z boku lub przodu (wtedy idą kursem zbieżnym). Sylwetki muszą być charakterystyczne, umożliwiające rozpoznanie rodzaju jednostki przez gracza. Należy zaprogramować co najmniej cztery typy sylwetek z okresu II wojny światowej: ścigacz, niszczyciel, statek handlowy nieprzyjaciela, statek handlowy sojuszniczy.

Skuteczny atak torpedowy możliwy jest wyłącznie wówczas, gdy okręt (statek) widziany jest z boku. Po podjęciu decyzji o ataku na ekranie pojawia się celownik i dane o odległości celu; naprowadzanie na cel całym okrętem. W trakcie przeprowadzania ataku gracz powinien być informowany o zbliżaniu się innego niż atakowany obiekt.

Okręty i samoloty po zauważeniu „Orła" atakują go bombami głębinowymi. Należy przewidzieć cztery sytuacje, zależnie od szybkości reakcji gracza i zanurzenia okrętu po zauważeniu niebezpieczeństwa:

a) okręt (samolot) w ogóle nie zauważa „Orła" na skutek jego szybkiego zanurzenia (sytuacja punktowana),
b) „Orzeł" przed zanurzeniem zostaje zauważony z dużej odległości — rozrzut bomb przy ataku duży,
c) „Orzeł" przed zanurzeniem zostaje zauważony ze średniej odległości — rozrzut bomb mniejszy,
d) mała odległość — rozrzut bomb mały.

Odległości związane z powyższymi sytuacjami muszą być inne, gdy „Orzeł" znajduje się w położeniu nawodnym, i jeszcze inne, gdy znajduje się na głębokości peryskopowej.

Okręt w położeniu podwodnym odpiera ataki bombowe przez manewrowanie. W trakcie zmian głębokości zanurzenia należy uważać na dno i jego ukształtowanie (skały), aby nie rozbić okrętu. Opadające bomby widoczne są na ekranie, głębokość wybuchu ustawiona jest losowo.

Epizod 6. Przejście przez pole minowe

Po przybyciu na wody Skagerraku, w odległości 15 mil od okrętu latarniowego „Skaw", „Orzeł" trafia na pole minowe...

W tym epizodzie gracz musi się wykazać dobrym słuchem oraz umiejętnością szybkiego, precyzyjnego manewrowania okrętem.

Ekran jest prawie ciemny z lekko zarysowaną sylwetką okrętu (przekrój poprzeczny), niezmienny w czasie, jedynie w razie wybuchu miny rozjaśnia się pokazując gnieciony wybuchem okręt.

Trafienie na linę utrzymującą minę sygnalizowane jest dźwiękiem. Informacja o tym, po której stronie okrętu jest lina, zawarta jest w wysokości dźwięku. W miarę przesuwania się liny po burcie (wzdłuż podłużnej osi symetrii okrętu) wysokość dźwięku rośnie. Jeżeli lina zbyt długo przesuwa się wzdłuż burty, mina wybucha (przy określonej wysokości dźwięku). Oczywiście, im prędkość okrętu jest większa, tym szybciej wybuchnie mina, a więc reakcja gracza powinna być szybsza. Ten sam efekt (ale w znacznie szybszym tempie) powinno wywoływać „przemieszczenie" się liny (tej znajdującej się przy burcie, lub innej znajdującej się w pobliżu) wzdłuż poprzecznej linii symetrii okrętu; może się to zdarzyć w czasie gwałtownie wykonanego zwrotu lub zwrotu niewłaściwie wykonanego, tzn. na znajdującą się przy burcie linę. Czas wykonania zadania jest ograniczony.

Epizod 7. Wejście do portu w Rosyth

Po przejściu przez Morze Północne „Orzeł" znajduje się w pobliżu wybrzeży szkockich na wysokości Firth of Forth. Wobec braku szyfru okręt otwartym tekstem sygnalizuje swoje przybycie. Sytuacja jest w dalszym ciągu niebezpieczna, albowiem Anglicy mogą podejrzewać mistyfikację, a Niemcy po przechwyceniu radiogramu mogą skierować do miejsca, gdzie znajduje się „Orzeł", swoje jednostki.

Plansza przedstawia widok z góry na stojącego w prawej części ekranu „Orła", z lewej strony widoczne są cztery przylegające do siebie (dwoma bokami) kwadraty. Gracz musi nadać sygnałami Morse'a radiogram: „Przypuszczalna pozycja 0630 na wyznaczonym miejscu dla polskiej Marynarki Wojennej. Proszę o pozwolenie wejścia i o pilota, ale nie mam map. Orzeł."

Treść tekstu jest autentyczna, można ją skrócić. Jeżeli tekst zostanie nadany prawidłowo, przyjdzie zakodowana odpowiedź wskazująca, do którego kwadratu ma udać się okręt. „Rozszyfrowanie" kodu polega na odgadnięciu w grze typu „Master Mind" dwóch dwucyfrowych liczb (współrzędnych kwadratów). Po odgadnięciu współrzędnych kwadraty zostają opisane i „Orzeł" będzie atakowany torpedami wystrzelonymi przez nieprzyjacielski okręt podwodny. Uniknąć ataku można jedynie przez szybki manewr nawodny.

Gracz, który nie odgadnie współrzędnych w trzech próbach, ma jeszcze szansę na zakończenie gry, ale musi się wykazać doskonałym opanowaniem „manewrówki", prowadząc okręt pod zmasowanym ogniem artylerii nabrzeżnej do dowolnego kwadratu (tylko jedna próba).

Pomyślne zakończenie tego epizodu (i wszystkich poprzednich) jest równoznaczne z pomyślnym zakończeniem gry.

III. Zakończenie gry

Na zakończenie gry pojawia się plansza ukazująca sylwetkę „Orła" (widok z boku z podniesioną banderą) przesuwającą się na tle zabudowań portowych i słychać pierwsze takty Mazurka Dąbrowskiego. W dolnej części planszy podawana jest liczba nieudanych prób, liczba storpedowanych jednostek oraz liczba uzyskanych punktów. W wypadku gdy gra zostanie zakończona bez zaliczenia wszystkich epizodów, efekt końcowy ograniczony jest tylko do obrazu płynącego na tle portu „Orła".

Andrzej GRZESIAK

PROGRAM 67

SPECTRUM

Przesuwanka

Gra polega na przesuwaniu pasów planszy tak, aby nasze piony znalazły się w miejscu uprzednio uzgodnionym z przeciwnikiem (najczęściej jest to startowe pole przeciwnika). Każdy gracz ma dwa ruchy: przesunięcie wybranego pasa w pionie i w poziomie. Informacje o dostępnych klawiszach znajdują się na ekranie.

```
      PRZESUWANKA
   wersja komputerowa
opracowal K.Pozniak ©1988

Gra przeznaczona jest dla dwoch
graczy. Graczowi w jednej kolej-
ce przysluguja dwa ruchy: piono-
wy i poziomy. Gracze sami usta-
laja polozenie pionow, ktore
konczy gre

    dostepne klawisze:
ustalanie ruchu w pionie:
   klawisz <5> - w lewo
   klawisz <8> - w prawo
ustalanie ruchu w poziomie:
   klawisz <6> - w dol
   klawisz <7> - w gore
   klawisz <r> - wykonanie ruchu
   klawisz <s> - koniec gry

          wcisnij <s>
```

plansza informacyjna gry



**

podpis: początkowe ustawienie pionów. Wyróżniony pas planszy wskazują: **

Życzę przyjemnej zabawy

Krzysztof POŹNIAK

```
   1 REM ************************
   2 REM
   3 REM                   ©1988
   4 REM   Krzysztof Pozniak
   5 REM
   6 REM   P R Z E S U W A N K A
   7 REM       gra planszowa
   8 REM
   9 REM ************************
  10 REM ████████████
  15 PAPER 0: INK 7: BORDER 0: C
LS
  20 DIM x(6): DIM y(6): DIM n(8
,8)
 100 REM ████████████
 110 GO SUB 3000: GO SUB 900
 120 PRINT AT 21,0; FLASH 1; BRI
GHT 1;" DEMONSTRACJA - wcisnij k
lawisz"
 130 LET x=INT (6*RND)+2: GO SUB
 1400: LET y=INT (6*RND)+2: GO S
UB 1300: IF INKEY$="" THEN GO TO
 120
 200 REM ████████████
 210 CLS
 220 BEEP .1,10: BEEP .1,20: BEE
P .1,30
 230 GO SUB 4000
 240 IF INKEY$<>"s" THEN GO TO 2
40
 300 REM ████████
 310 CLS
 320 GO SUB 3000: GO SUB 900
 330 PRINT AT 3,25;"ruch";AT 5,2
4;" piona ";AT 10,23;"klawisz";A
T 16,23;"<r>-ruch";AT 18,23;"<s>
-stop"
 400 REM ████████████
 410 PRINT AT 7,26;" ▀ ";AT 8,26;
" ▀ ";AT 12,23;"<5>-lewo";AT 14,2
3;"<8>-prawo"
 420 GO SUB 1500: GO SUB 1400
 430 PRINT AT 12,23;"<6>-dol ";A
T 14,23;"<7>-gora "
 440 GO SUB 1600: GO SUB 1300
 500 REM ████████████
 510 PRINT AT 7,26;" ▄ ";AT 8,26;
" ▄ ";AT 12,23;"<5>-lewo";AT 14,2
3;"<8>-prawo"
 520 GO SUB 1500: GO SUB 1400
 530 PRINT AT 12,23;"<6>-dol ";A
T 14,23;"<7>-gora "
 540 GO SUB 1600: GO SUB 1300
 600 GO TO 400
 800 GO SUB 1500: GO SUB 1400: G
O SUB 1600: GO SUB 1300: GO TO 8
00
 900 REM ████████████
 910 FOR x=2 TO 7: GO SUB 1440:
NEXT x
 920 FOR y=2 TO 7: GO SUB 1340:
NEXT y
 930 RETURN
1000 REM ████████████
1010 IF n(y,x)=0 THEN PRINT AT 2
*y,2*x;"  ";AT 2*y+1,2*x;"  "
1020 IF n(y,x)=1 THEN PRINT AT 2
*y,2*x;" ▀";AT 2*y+1,2*x;" ▀"
1030 IF n(y,x)=2 THEN PRINT AT 2
*y,2*x;" ▄";AT 2*y+1,2*x;" ▄"
1040 PLOT 16*x,175-16*y: DRAW 16
,0: DRAW 0,-16: DRAW -16,0: DRAW
 0,16
1050 RETURN
1100 REM ████████████
1110 IF x(x-1)=0 THEN PRINT AT 0
,2*x;" █";AT 1,2*x;" █";AT 18,2*
x;"  ";AT 19,2*x;"  "
1120 IF x(x-1)=1 THEN PRINT AT 1
8,2*x;" █";AT 19,2*x;" █";AT 0,2
*x;"  ";AT 1,2*x;"  "
1130 PLOT 16*x,31: DRAW 16,0
1140 RETURN
1200 REM ████████████
1210 IF y(y-1)=0 THEN PRINT AT 2
*y,0;"▀▀";AT 2*y+1,0;"▀▀";AT 2*y
,18;"  ";AT 2*y+1,18;"  "
1220 IF y(y-1)=1 THEN PRINT AT 2
*y,18;"▀▀";AT 2*y+1,18;"▀▀";AT 2
*y,0;"  ";AT 2*y+1,0;"  "
1230 PLOT 144,175-16*y: DRAW 0,-
16
1240 RETURN
1300 REM ████████████
1310 LET y(y-1)=NOT y(y-1)
1320 IF y(y-1)=0 THEN FOR x=1 TO
 7: LET n(y,x)=n(y,x+1): NEXT x:
 LET n(y,8)=0
1330 IF y(y-1)=1 THEN FOR x=7 TO
 1 STEP -1: LET n(y,x+1)=n(y,x):
 NEXT x: LET n(y,1)=0
1340 FOR x=1 TO 8: GO SUB 1000:
NEXT x: GO SUB 1200
1350 RETURN
1400 REM ████████████
1410 LET x(x-1)=NOT x(x-1)
1420 IF x(x-1)=0 THEN FOR y=1 TO
 7: LET n(y,x)=n(y+1,x): NEXT y:
 LET n(8,x)=0
1430 IF x(x-1)=1 THEN FOR y=7 TO
 1 STEP -1: LET n(y+1,x)=n(y,x):
 NEXT y: LET n(1,x)=0
1440 FOR y=1 TO 8: GO SUB 1000:
NEXT y: GO SUB 1100
1450 RETURN
1500 REM ████████████
1510 LET x=2: GO TO 1540
1520 GO SUB 2000: PRINT AT 21,2*
x;"  "
1530 IF x$="5" AND x>2 THEN LET
x=x-1: GO TO 1540
1531 IF x$="5" AND x=2 THEN LET
x=7: GO TO 1540
1532 IF x$="8" AND x<7 THEN LET
x=x+1: GO TO 1540
1533 IF x$="8" AND x=7 THEN LET
x=2: GO TO 1540
1534 IF x$="r" THEN RETURN
1540 PRINT AT 21,2*x;"**": GO TO
 1520
1600 REM ████████████
1610 LET y=2: GO TO 1640
1620 GO SUB 2000: PRINT AT 2*y,2
1;" ";AT 2*y+1,21;" "
1630 IF x$="6" AND y<7 THEN LET
y=y+1: GO TO 1640
1631 IF x$="6" AND y=7 THEN LET
y=2: GO TO 1640
1632 IF x$="7" AND y>2 THEN LET
y=y-1: GO TO 1640
1633 IF x$="7" AND y=2 THEN LET
y=7: GO TO 1640
1634 IF x$="r" THEN RETURN
1640 PRINT AT 2*y,21;"*";AT 2*y+
1,21;"*": GO TO 1620
2000 REM ████████████
2010 PAUSE 1: PAUSE 0: LET x$=IN
KEY$: BEEP .05,30: IF x$<>"s" TH
EN RETURN
2020 RUN
3000 REM ████████████
3010 DIM x(6): DIM y(6): DIM n(8
,8)
3020 FOR k=4 TO 6: LET x(k)=1: L
ET y(k)=1: NEXT k
3030 FOR k=2 TO 4: FOR l=2 TO 6-
k: LET n(k,l)=1: LET n(9-k,9-l)=
2: NEXT l: NEXT k
3040 RETURN
4000 REM ████████████
4010 PRINT TAB 10; INVERSE 1;"PR
ZESUWANKA"; INVERSE 0'TAB 7;"wer
sja komputerowa"'TAB 3;"opracowa
l K.Pozniak ©1988"''" Gra przezn
aczona jest dla dwochgraczy. Gra
czowi w jednej kolej-ce przyslug
uja dwa ruchy: piono-wy i poziom
y. Gracze sami usta- laja poloze
nie pionow, ktore    konczy gre"
4020 PRINT 'TAB 7; INVERSE 1;"do
stepne klawisze:"; INVERSE 0;'"us
talanie ruchu w pionie:"'"  klaw
isz <5> - w lewo"'"  klawisz <8>
 - w prawo"'"ustalanie ruchu w p
oziomie:"'"  klawisz <6> - w dol
"'"  klawisz <7> - w gore"
4030 PRINT "  klawisz <r> - wyk
onanie ruchu"
4040 PRINT "  klawisz <s> - kon
iec gry"
4050 PRINT 'TAB 10; FLASH 1; BRI
GHT 1;"wcisnij <s>"
4060 RETURN
9000 REM ████████████
9010 CLEAR : SAVE "przesuw.KP" L
INE 1: VERIFY "przesuw.KP"
```

Odcyfrowywanie

Danuta KWASIŻUR

Duże podobieństwo do zadań polegających na rozszyfrowaniu spłowiałych rękopisów; prezentowanych w odcinku 9 „IKS" nr 10/88), mają dzisiaj prezentowane 3 przykłady. Ich istota sprowadza się do odgadnięcia, jakimi cyframi zakodowano poszczególne litery. Dokładną informację na ten temat otrzymuje użytkownik po zainicjowaniu pracy programu na ekranie, którego kopię przedstawia poniższy rysunek

```
Musisz rozwiazac 3 zadania. Litery ozna-
czaja cyfry. W kazdym zadaniu te same
litery oznaczaja te same cyfry, ale nie
we wszystkich razem wzietych.

STEROWANIE KLAWISZAMI:
 - ↑ , ↓    - przesuwanie kursora do
              litery, pod ktora chcemy
              podstawic cyfre
 - 'ENTER'  - koniec wykonywania zadania.


        NACISNIJ DOWOLNY KLAWISZ
```

W dalszej części program kolejno prezentuje 3 zadania. Przejście do następnego zadania odbywa się w 2 sytuacjach:
— poprawnego rozwiązania bieżącego przykładu;
— odpowiedzi twierdzącej na pytanie „POPEŁNIŁEŚ BŁĘDY. SPRÓBUJESZ JESZCZE RAZ?"

Przykładowa postać zadania na ekranie monitora jest przedstawiona na poniższym rysunku:





Wersja źródłowa programu w języku Basic jest następująca:

```
10 '
20 '          ekran informacyjny
30 '
40 MODE 1:INK 0,0:INK 1,26:INK 2,9:INK 3,6:BORDER 0:PAPER 0:CLS:PEN 1
50 PRINT "Musisz rozwiazac 3 zadania. Litery ozna-czaja cyfry. W kazdy
m zadaniu te same  litery oznaczaja te same cyfry, ale nie we
 wszystkich razem wzietych."
60 PRINT:PRINT:PRINT "STEROWANIE KLAWISZAMI: "
70 PRINT:PRINT " - ";CHR$(240);" , ";CHR$(241);"   - przesuwanie lurso
ra do"
80 PRINT SPC(13);"litery, pod ktora chcemy"
90 PRINT SPC(13);"podstawic cyfre"
100 PRINT " - 'ENTER' - koniec wykonywania zadania"
110 LOCATE 7,18:PRINT "NACISNIJ DOWOLNY KLAWISZ"
120 IF INKEY$="" THEN GOTO 120
130 '
140 '          segment sterujacy
150 '
160 MODE 1
170 FOR A=1 TO 3:READ ILL(A,1),ILL(A,2):NEXT A
180 DATA 5,15,6,15,6,21
190 DIM WSP(21,2):DIM LITERY$(21)
200 FOR I=1 TO 3
210 MODE 1
220 PAPER 1:PEN 0:GRAPHICS PEN 0
230 ON I GOSUB 720,740,760,
240 GOSUB 780
250 WINDOW £3,8,40,3,4:PAPER £2,1
260 FOR J=1 TO ILL(I,2)
270 LOCATE WSP(J,1),WSP(J,2):PRINT LITERY$(J)
280 NEXT J
290 FOR J=1 TO ILL(I,1)
300 LOCATE 24,9+J:PRINT LIT$(J)
310 NEXT J
320 k=10
330 LOCATE 23,k:PRINT CHR$(243)
340 a$=INKEY$
350 IF a$="" THEN GOTO 340
360 IF INKEY(6)=0 THEN GOTO 520
370 IF a$=CHR$(240) THEN GOTO 440
380 IF a$=CHR$(241) THEN GOTO 460
390 IF ASC(A$)<48 OR ASC(A$)>57 THEN GOTO 430
400 LOCATE 25,K:PRINT "=";A$
410 CYF1(K-9)=VAL(A$)
420 GOTO 480
430 SOUND 1,400:GOTO 340
440 IF k=10 THEN GOTO 430
450 LOCATE 23,k:PRINT " ":k=k-1:GOTO 330
460 IF k=ill(i,1)+9 THEN GOTO 430
470 LOCATE 23,k:PRINT " ":k=k+1:GOTO 330
480 FOR V=1 TO ill(I,2)
490 IF LITERY$(V)=LIT$(K-9) THEN LOCATE WSP(V,1),WSP(V,2):PRINT USING
"£";CYF1(K-9) ,
500 NEXT V
510 GOTO 340
520 GOSUB 980
530 IF bl=0 THEN GOTO 580
540 CLS £3:PRINT £3,"POPELNILES BLEDY.":PRINT £3,"SPROBUJESZ JESZCZE R
AZ? (T/N)";:INPUT £3," ",T$
550 IF T$="T" OR T$="t" THEN CLS £2:CLS £3:GOTO 260
560 IF T$="n" OR T$="N" THEN GOTO 630
570 GOTO 540
580 PRINT £3,"BRAWO!!!":PRINT £3,"ZADANIE WYKONANE POPRAWNIE"
590 FOR G=26 TO 0 STEP -1
600 BORDER G:FOR T=1 TO 200:NEXT T
610 NEXT G
620 IF I=3 THEN END
630 IF i=3 THEN END
640 CLS £3:INPUT £3,"NASTEPNE ZADANIE (T/N)? ",T$
650 IF T$="T" OR T$="t" THEN GOTO 680
660 IF T$="n" OR T$="N" THEN END
670 GOTO 630
680 NEXT I
```

W segmencie sterującym wykorzystuje się podprogramy, których rola jest następująca:

1. Trzy podprogramy realizujące wykonanie ekranów dla poszczególnych zadań (linie 690—770)

```
690 '
700 '  Podprogramy rysowania ekranow dla poszczegolnych zadan
710 '
720 WINDOW £1,11,18,7,17:PAPER £1,1:CLS £1::LOCATE 13,10:PRINT "X":PLO
```

```
T 192,232:DRAWR 64,0:PLOT 192,184:DRAWR 64,0
730 WINDOW £2,22,27,9,15:PAPER £2,1:CLS £2:RETURN
740 WINDOW £1,11,18,7,17:PAPER £1,1:CLS £1::LOCATE 13,10:PRINT "X":PLO
T 192,232:DRAWR 64,0:PLOT 192,184:DRAWR 64,0
750 WINDOW £2,22,27,9,16:PAPER £2,1:CLS £2:RETURN
760 WINDOW £1,11,20,7,20:PAPER £1,1:CLS £1::LOCATE 17,11:PRINT ":":PLO
T 192,248:DRAWR 64,0:PLOT 192,200:DRAWR 32,0:PLOT 208,152:DRAW
R 32,0:PLOT 208,104:DRAWR 48,0
770 WINDOW £2,22,27,9,16:PAPER £2,1:CLS £2:RETURN
```

2. Podprogram wczytywania danych organizacyjnych do poszczególnych zadań (linie 780—970)

```
780 '
790 '  Podprogram wczytywania danych do aktualnego zadania
800 '
810 FOR K=1 TO ili(I,1):READ LIT$(K):NEXT K
820 FOR K=1 TO ili(I,2):READ LITERY$(K):NEXT K
830 FOR K=1 TO ili(I,2):READ WSP(K,1),WSP(K,2):NEXT K
840 FOR K=1 TO ili(I,1):READ cyf(K):NEXT K
850 RETURN
860 DATA "A","C","I","N","T"
870 DATA "I","C","C","I","N","N","T","T","I","C","C","I","A","N","T"
880 DATA 14,9,15,9,16,9,15,10,16,10,14,12,15,12,16,12,13,13,14,13,15,1
3,13,15,14,15,15,15,16,15
890 DATA 7,4,1,2,8
900 DATA "I","L","N","O","S","U"
910 DATA "I","N","U","N","U","L","N","U","N","U","S","O","I","N","U"
920 DATA 14,9,15,9,16,9,15,10,16,10,14,12,15,12,16,12,13,13,14,13,15,1
3,13,15,14,15,15,13,16,15
930 DATA 1,6,2,3,0,5
940 DATA "A","E","N","S","T","U"
950 DATA "E","N","A","U","E","N","A","N","A","N","A","T","N","A","S","
E","N","A","E","N","A"
960 DATA 14,9,15,9,16,9,13,11,14,11,15,11,16,11,18,11,19,11,13,12,14,1
2,14,14,15,14,14,15,15,15,14,17,15,17,16,17,14,18,15,18,16,18
970 DATA 5,1,2,0,6,3
```

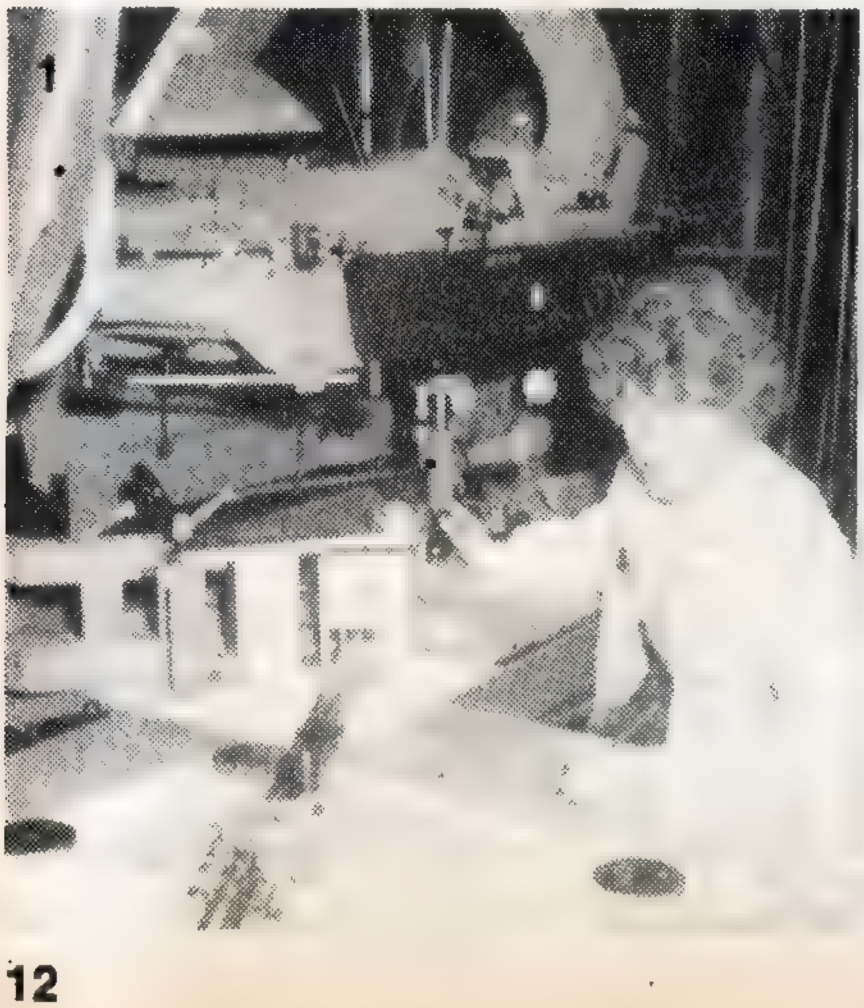
3. Program badania poprawnego rozwiązania zadania (linie 980—1060)

```
980 '
990 'Podprogram badania poprawnosci zadania
1000 '
1010 bl=0
1020 FOR z=1 TO ili(i,1)
1030 IF cyf(z)=cyf1(z) THEN GOTO 1050
1040 bl=1
1050 NEXT z
1060 RETURN
```

Danuta KWASIŻUR

Zanim powstanie mapa

1) **Izabela Grochala** na podstawie wykonanych uprzednio zdjęć lotniczych uaktualnia dane na istniejącej już mapie. To urządzenie, zainstalowane w pracowni autografów, jest w tym bardzo pomocne. Obserwując w okularze zdjęcia — z zastosowaniem metody stereoskopii — otrzymujemy przestrzenny obraz badanego terenu. Przed dobrym fachowcem nie ukryje się żaden element czy szczegół na obserwowanym obszarze. Oczywiście możliwe są nie tylko poprawki czy korekty na mapie, ale opracowanie jej od podstaw.





2) To urządzenie jest analogiczne do poprzedniego, ale znacznie nowocześniejsze, dużo szybsze i precyzyjniejsze. Wymaga jednak od operatora, mgr inż. **Barbary Komendackiej** doskonałego wzroku, który jest jakby punktem wyjścia. Resztę wykonuje komputer.



3) Dla przyrządów analitycznych, które prezentuje mgr inż. **Jerzy Saczuk**, nie ma rzeczy niemożliwych. W tej chwili na ploterze powstaje obraz lasu. Szybkość kreślenia jest wręcz nieporównywalna z tempem pracy kreślarza. Urządzenie też może grawerować. Dokładność kreślenia wręcz nieprawdopodobna: 0,02 mm! Cała mapa zapisana jest i zachowania na nośniku magnetycznym. W razie potrzeby może zostać przeniesiona na papier. W każdej postaci. Komputer można zaprogramować tak, że wyrysowany zostanie tylko konkretny fragment mapy lub na papier nanieść można wybrane elementy całości, na przykład sieć dróg, zalesienie czy zabudowania.



4) Na tych urządzeniach **Ewa Jesion** pracuje nie tylko przy mapach. Przygotowywany jest na nich także Magazyn Informatyczny Horyzontów Techniki — „Mikroklan". CR terminal służy do składu i edycji tekstów. Monitor o przekątnej 380 mm wyświetla znaki alfanumeryczne. Klawiatura wyposażona jest w 58 dodatkowych klawiszy funkcyjnych programowanych przez użytkownika. Z prawej strony Typeview 300 — urządzenie podglądowe, pokazujące na pionowym ekranie o formacie zbliżonym do A4 rzeczywisty obraz składu typograficznego.

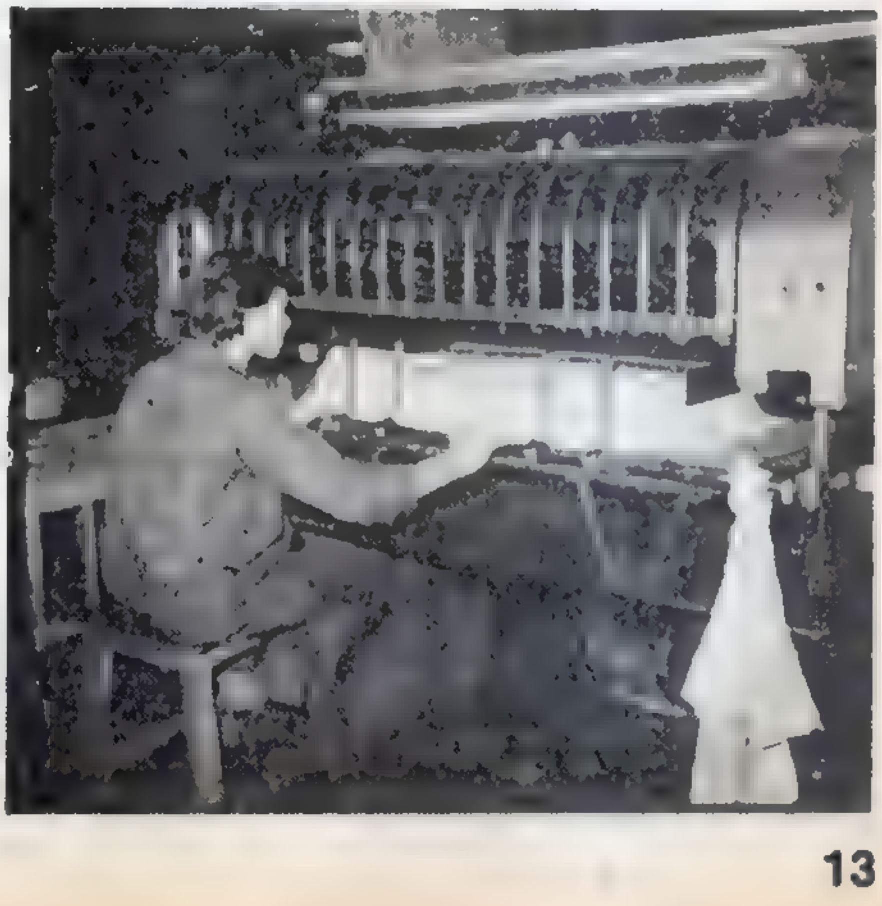
5) Logoscanner z komputerem IBM PC XT — wszystko firmy Linotype. To także urządzenia wielofunkcyjne. **Piotr Kurzec** opracowuje całe nazewnictwo do mapy.

6) Pominęliśmy wiele etapów powstawania mapy. Nie były tam bowiem aż tak bardzo wykorzystywane techniki komputerowe. A właśnie w fotoreportażu chodziło o pokazanie roli komputera we współczesnym tworzeniu mapy. Ostatecznie jednak wychodzą one spod tradycyjnych maszyn drukarskich.

J. RAJCH

(Foto: J. Zelman)





Chochlik znowu nam namieszał.

Tym razem „pomógł" w redagowaniu jednego z programów w numerze 11.

Oto prawidłowy wydruk drugiego programu:

KASETOWY SYSTEM OPERACYJNY 2T06 TURBO

```
10 REM PROGRAM 2
20 REM Zapisuje na dyskietce w pl
iku D:KOPIARKA.TUR
30 REM program przenoszacy pliki
miedzy systemem TURBO i dyskami
40 GRAPHICS 0:? "▶▶PROSZE CZEKAC"
50 READ LB
60 DIM A$(LB),B$(22)
70 LICZN=0:CHKSUM=0
80 FOR I=1 TO LB
90 IF LICZN=0 THEN READ B$
100 GOSUB 270:A$(I)=CHR$(BYTE)
110 CHKSUM=CHKSUM+BYTE:CHKSUM=CHK
SUM-256*(CHKSUM>255)
120 LICZN=LICZN+1
130 IF LICZN<>10 AND I<>LB THEN 1
70
140 GOSUB 270
150 IF CHKSUM<>BYTE THEN A=10*INT
((I-1)/10)+320:? "BLAD DANYCH W L
INII ";A:LIST A:END
160 LICZN=0
170 NEXT I
180 POKE 764,255:GRAPHICS 0
190 ? " Prosze wlozyç do stacji d
yskow dys- "
200 ? " kietke na ktorej ma byc z
apisany "
210 ? " program kopiujacy i nacis
nac"
220 ? " RETURN."
230 POKE 764,255:GET #16,A
240 OPEN #1,8,128,"D:KOPIARKA.TUR
"
250 ? #1;A$;:CLOSE #1
260 END
270 B=ASC(B$(LICZN*2+1)):GOSUB 30
0:BYTE=Y
280 B=ASC(B$(LICZN*2+2)):GOSUB 30
0:BYTE=Y+16*BYTE
290 RETURN
300 Y=(B-48)*(B<58)+(B-87)*(B>96)
:RETURN
310 DATA 1865
320 DATA ffff00209e224c8d2100d8
330 DATA 0d2135499199361bcad099
340 DATA fd60be0420200c20a900cd
350 DATA 8d00d3a9008d1ad0ad1f19
360 DATA d0f011be0520200c20a9c2
370 DATA df8d00d3a9088d1ad06089
380 DATA a0804c6221a20886300ae2
390 DATA 48a0009002a0022010204e
400 DATA 68c630f008a20c200c209e
410 DATA 4c3f2060a008ad1fd0f0dd
420 DATA d7ad00d330f6a9008d1aaa
430 DATA d0aae8301ead00d310f8e2
440 DATA a9088d1ad0ec0a20b00bdb
450 DATA 6688ec0b20263088d0d462
460 DATA 60248830cfa08c4c622168
470 DATA 209b21a900a010853098ea
480 DATA 48a00420102068a8c6302c
490 DATA d0f388d0f0a432a900853b
500 DATA 32a202200c209848b13220
510 DATA 481865318531682 03b20af
520 DATA 68a8c8f005c4004cce207a
530 DATA e633c434d009a535c53336
540 DATA d0064ce120eaeaea4cb417
550 DATA 20a531a201200c20203b57
560 DATA 20a0014c62214c362020a9
570 DATA 9b21a000a9ff8588a2005c
580 DATA ad1fd0f0edad00d330f67b
590 DATA a9008d1ad0e830ecad004c
600 DATA d310f8a9088d1ad0ec0a45
610 DATA 2090d4c8d0daa432a900ba
620 DATA 0532984820582068a8a59e
630 DATA 30ae0320d02691321865d5
640 DATA 318531c8d002e633c43467
650 DATA d0e2a533c535d0dc20580f
660 DATA 20a001a530c531f002a02d
670 DATA 8f4c6221d132f0d8a09187
680 DATA aec1219aa9408d0ed4a9b2
690 DATA 228d00d4a9388d02d3a921
700 DATA 008d00d3a93c8d02d35820
710 DATA 6086348435a2528632a241
720 DATA 278633ae0320d006aad042
730 DATA 034c90204cf320a9008dd6
740 DATA 0ed485318d00d4a9088d0d
750 DATA 1fd0a9388d02d3a9608dd5
760 DATA 00d3a93c8d02d378bae809
770 DATA e88ec1216000cf21b8228b
780 DATA 4023ed228f239123000063
790 DATA a9008d0320858385822a570
800 DATA 2a290cc904f035c908f082
810 DATA 0160a002b124c956f0258e
820 DATA a90420fcfd301ba251a032
830 DATA 2720fd23a900a25ea02709
840 DATA 2081212093239830 05a917
850 DATA 808583604c032338668291
860 DATA 186681a90320fcfd30f075
870 DATA a901a25ea027208121c068
880 DATA 80f0e39830f0a2aca02283
890 DATA 20fd23a200bdad22c92fe9
900 DATA f00dc92af016c940f00ae2
910 DATA dd5227d0d3e8e00cd0e766
920 DATA 209323a001848360a202e8
930 DATA 8a48bd522720b0f268aac4
940 DATA e8e00cd0f1a2008a48bd8a
950 DATA 992220b0f268aae8e006e7
960 DATA d0f1a9ff8dfc0220fdf2ea
970 DATA 295f09404820b0f2a99b09
980 DATA 20b0f268c954f0bc4c2068
990 DATA 222028542f4e29b92252f9
1000 DATA 27a482100a20412398108c
1010 DATA 3ec088d026a001a58310e1
1020 DATA 20a900a89184e684d002a3
1030 DATA e685a933c585d0efa954f0
1040 DATA c584d0e9a900a254a03364
1050 DATA 20812160a48210188dce2f
1060 DATA 21204123c000300dcdce6c
1070 DATA 21f008a091a900858285eb
1080 DATA 8360a4833003a08760a04f
1090 DATA 009184e684d002e685eef9
1100 DATA 5227d003ee5327ad5327d4
1110 DATA c90cf003a00160a900a2e8
1120 DATA 54a03320812198 1004a926
1130 DATA 008583209323 60a4838813
1140 DATA f003a08360ad52270d530f
1150 DATA 27d022a5811003a08860e9
1160 DATA a901a254a0332081212 03e
1170 DATA 9b2398309ca90ccd53275c
1180 DATA f005386681d0d5a000b167
1190 DATA 8448e684d002e68538adbf
1200 DATA 5227e9018d5227ad53274f
1210 DATA e9008d532768a00160a951
1220 DATA 008d52278d5327a92785b3
1230 DATA 85a954858460a20084cc90
1240 DATA 85cb1869029d4403986948
1250 DATA 009d4503a000b1cb9d482e
1260 DATA 03c8b1cb9d4903a90b9daf
1270 DATA 42034c56e485d484d5204c
1280 DATA aad920e6d8a90085cdf098
1290 DATA 0520b0f2e6cda4cdb980bc
1300 DATA 0510f4490020b0f24cf692
1310 DATA 23a92020b0f2e6cda906a2
1320 DATA c5cdd0f360868484 85a00a
1330 DATA 01a9009184a9ffc891844e
1340 DATA c8b124c99bf0089184c824
1350 DATA c00dd0f360a9209184c8ba
1360 DATA c00dd0f96030313233 34aa
1370 DATA 3536373839303132 3334b7
1380 DATA 35363738393030313233c0
1390 DATA 34353637383930313233cd
1400 DATA 34353637383930a972a0ff
1410 DATA 2520a423a200a9059d423a
1420 DATA 03a9249d4503a9254c7c85
1430 DATA 24a9a1a02520a423a20041
1440 DATA a9059d4203a9249d450383
1450 DATA a93a9d4403a9139d4803ee
1460 DATA a9009d49032056e41004ee
1470 DATA c089d02cbc490388f026d9
1480 DATA bd44038586bd45038587f9
1490 DATA a000b186c92ad0034cfee0
1500 DATA 2660a254a921a0c220862e
1510 DATA e4a9cea02520a423a9ffdd
1520 DATA 8dfc02204f2420672420c6
1530 DATA 4025a210a9039d4203a914
1540 DATA 049d4a03a9009d4b03a93f
1550 DATA 259d4403a9249d4503201a
1560 DATA 56e4303fa220a9039d4210
1570 DATA 03a9089d4a03a9009d4b3f
1580 DATA 03a93a9d4403a9249d45b8
1590 DATA 032056e4301f20eb2630c5
1600 DATA 0720d626301510f4c08879
1610 DATA d00f204025300aa9b4a014
1620 DATA 2520a4234cbe249048204e
1630 DATA 4025a955a02520a423a0fd
1640 DATA 006820cb234cbe24a21053
1650 DATA a90c9d42032056e4a22006
1660 DATA a90c9d42032056e4601b72
1670 DATA 009bfdfdfd20202020 20a4
1680 DATA 20426c61642074726 16e0c
1690 DATA 736d69736a693a202d0022
1700 DATA 9b2a6e20706f64616a65e8
1710 DATA 207a6177617274 6f7363e6
1720 DATA 206479736b75206e9b50af
1730 DATA 6c696b2077656a73636994
1740 DATA 6f7779203a11009b506cb5
1750 DATA 696b2077796a7363696fb1
1760 DATA 7779203a18009b4b6f70d8
1770 DATA 696f77616e6965207a61bf
1780 DATA 6b6f6e637a6f6e65209be1
1790 DATA 06019b7d202020202020c0
1800 DATA 4b6f706961726b61204456
1810 DATA 59534b3c3e54555242 4f53
1820 DATA 544150459b9b2070726f24
1830 DATA 6772616d206f7072616300
1840 DATA 6f77616e792070727a650f
1850 DATA 7a9b2020202020202020 24
1860 DATA 2020202020202 0576f6a34
1870 DATA 636965636861205a6162ce
1880 DATA 6f6c6f746e65676f9b9b6b
1890 DATA 20202020207765727 36a36
1900 DATA 6120322e36202050494167
1910 DATA 53544f57202031393838ce
1920 DATA 9b9b555741474120 2121db
1930 DATA 21204162792075 6e696b0f
1940 DATA 6e6163206b6f6c697a6af4
1950 DATA 69207a6520737461 2d9b8c
1960 DATA 636a612064797 36b6f777b
1970 DATA 206964656e747966696b62
1980 DATA 61746f72656d206d61673f
1990 DATA 6e65746f666f2d9b6e7575
2000 DATA 2054555242 4f206a657383
2010 DATA 7420543a2061206e696582
2020 DATA 20443a206a616b2077202d
2030 DATA 7379732d9b74656d696568
2040 DATA 204b2e532e4f2e205455c8
2050 DATA 52424f9b48a220a90b9da1
2060 DATA 4203a9009d48039d490360
2070 DATA 682056e460a210a9079d81
2080 DATA 4203a9009d48039d490340
2090 DATA 2056e460a001b1868d4dac
2100 DATA 27204025a903a2109d4295
2110 DATA 03a94c9d4403a9279d4523
2120 DATA 03a9069d4a03a9009d4b50
2130 DATA 032056e4301aa907a21059
2140 DATA 9d4203a9009d48039d49b2
2150 DATA 032056e4300620b0f24c53
2160 DATA 2827c088d0034cbe244c37
2170 DATA 2b25444d3a2a2e2a00e0b4
2180 DATA 02e102ae246b
```

INFORMATYKA W SZKOLE

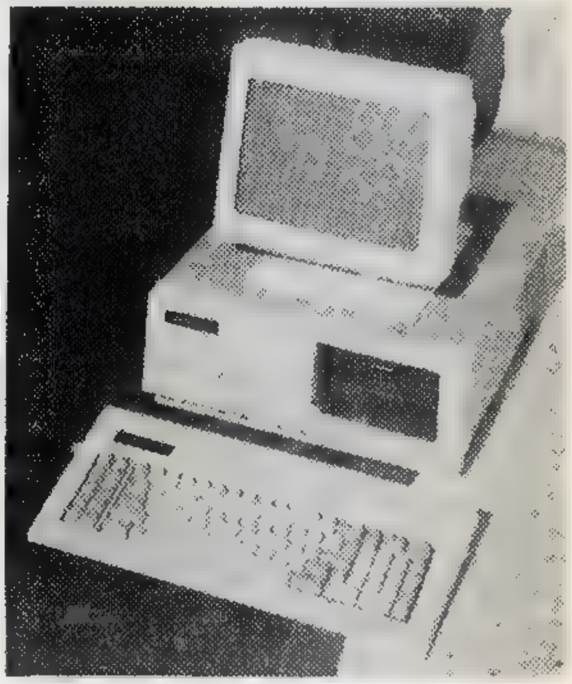
Usiłując pogodzić wymagania logiki przedmiotu z jednej strony, a kwestie metodologiczne z drugiej, trochę nieoczekiwanie przerwaliśmy analizę ciekawej problematyki budowy sprzętu informatycznego, na korzyść programowania. Było to niestety konieczne, przy założeniu, że chcemy równolegle uczestniczyć w ćwiczeniach praktycznych. Te zaś wymagają umiejętności budowy algorytmu oraz „opisywania go np.: w Basicu. Bo ile można poświęcić czasu na analizowanie budowy komputera podczas ćwiczeń?

Teraz mając materiały (fundamenty) na wiele ciekawych godzin przy komputerze (programowanie) możemy wrócić do budowy mikro, a ogólniej do budowy sprzętu informatycznego. Znajomość tych zagadnień ułatwi nam zrozumienie wielu podstawowych kwestii związanych z użytkowym wykorzystaniem maszyn w praktyce życia codziennego.

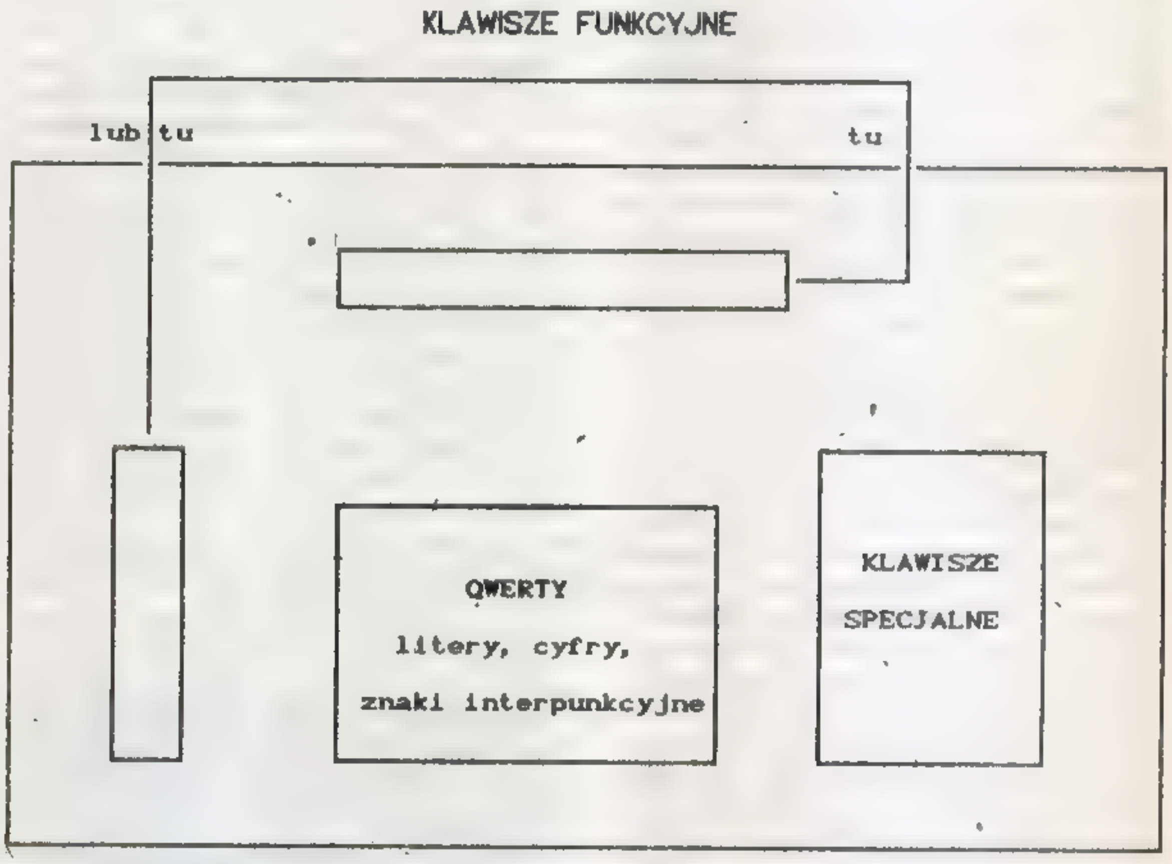
Wróćmy na chwilę do komputera, a w zasadzie do jednostki centralnej — pudełka gdzie mieszczą się podstawowe elementy tego urządzenia. Mówiliśmy już o pamięciach RAM i ROM, mikroprocesorze oraz „wszędobylskich" liczbach, które reprezentowane są tutaj w systemie dwójkowym, w postaci 8--bitowych kontenerków — bajtów. Pozwoliło to nam zrozumieć między innymi różnicę między 8- a 16-bitowymi procesorami, potrzebę dużej szybkości pamięci wewnętrznej i jej wielkości. Można powiedzieć, że w tej postaci nasza jednostka centralna może pracować sama na siebie — program steruje procesorem, ten zaś liczy i przekazuje wyniki do pamięci i tak w kółko. Podobnie jak 1000-osobowa administracja może pracować (bardzo ciężko) sama na siebie; polecenia, kontrole, zestawienia, aktualizacje, pensje... polecenia, kontrole. To błędne koło łatwo jest przerwać w wypadku naszego komputera — wystarczy wyłączyć prąd lub dalszą pracę maszyny uwarunkować informacjami z zewnątrz — podobnie jak miało to miejsce w przykładach z instrukcją INPUT. Oczekujemy także na wyniki (PRINT). Zatem w komputerze niezbędne są elementy elektroniczne, które pośredniczą w komunikacji komputera ze światem zewnętrznym — wprowadzanie i wyprowadzanie informacji. Są to specjalistyczne układy wejścia/ /wyjścia (we/wy). Umożliwiają one podłączenie bardzo wielu różnych urządzeń we/wy.

Zacznijny od podstawowego urządzenia — jest nim **klawiatura**, która mimo wielu wysiłków konstruktorów i programistów w stworzeniu lepszego sprzętu, pozostaje niezmiennie podstawowym pośrednikiem informacji adresowych do jednostki centralnej.

Sprzęt informatyczny



2. Komplet: jednostka centralna, monitor i klawiatura.



3. Szkic rozmieszczenia grup klawiszy w typowej klawiaturze.



1. Szkic sprzężeń podstawowych elementów mikrokomputera.

Warto pamiętać, iż klawiatura zbudowana jest zgodnie ze standardem nazywanym QWERTY od nazw pierwszych sześciu klawiszy liter w maszynach do pisania (w krajach anglosaskich). Niestety zazwyczaj, a w zasadzie prawie zawsze, nie ma tam liter polskich. Dostępne są za to, poza literami (duże i małe), cyframi i znakami interpunkcyjnymi jeszcze dwie grupy klawiszy: funkcyjne i specjalne. Podział ten jest umowny (podobnie zresztą jak ceny) i dotyczy klawiatur „zbliżonych" do sprzętu klasy IBM PC. Jednak nie odbiega zasadniczo np.: od Commodore, Amstrada, natomiast Spectrum w tym względzie jest **bardzo** nietypowym komputerem.

W chwili, gdy naciśniemy na przykład klawisz litery Q, układy elektroniczne klawiatury (w IBM PC jest to zazwyczaj mikroprocesor 8048) generują odpowiedni „zestaw" zer i jedynek, który, w znanym nam już kodzie ASCII, odpowiada właśnie temu znakowi. Podobnie rzecz się ma wobec wszystkich klawiszy.

Klawisze funkcyjne, mają bardzo ciekawą własność — można zaprogramować ich funkcje. Na przykład, jeśli komputer wykorzystywany byłby przez Fredrę, to przepisując swój rękopis, za pomocą komputera pan Aleksander mógłby znacznie ułatwić sobie pracę, gdyby naciskając klawisz (funkcyjny) np.: F1, komputer generował zwrot „mocium panie". Zatem naciskając tylko jeden klawisz — komputer drukuje w tym wypadku aż 12 znaków. Innymi słowy, zadanie każdego klawisza funkcyjnego może zależeć od nas, od tego jak je „oprogramujemy". Bywa, że niektóre interpretery BASIC-a przyporządkowują poszczególnym klawiszom klawiatury całe instrukcje, czyli zamiast naciskać pięć różnych klawiszy tworzących instrukcję PRINT, wystarczy nacisnąć F3 i na ekranie ukaże się owa instrukcja, itp. Idzie zatem o to, by ułatwić nam współpracę z komputerem, by móc poświęcić się sprawom ważniejszym niż wielokrotne, bezmyślne naciskanie klawiszy. Oczywiście jest to szczyt pływającej góry lodowej — możliwości, jakie stwarzają sobie klawisze funkcyjne. Wydaje się, że zasada ich stosowania jest zawsze taka sama — usprawnienie przekazywania informacji do komputera. Ilustruje to jeden z podstawowych obecnie celów twórców sprzętu i oprogramowania komputerów — ułatwienie jak najpełniejszego wykorzystania ogromnego potencjału sprzętu informatycznego.

„W parze" z klawiaturą występuje **monitor**. Jest to z kolei podstawowe urządzenie wyjściowe maszyny. Zdecydowaną ilość informacji wyjściowych komputer przekazuje użytkownikowi właśnie za pośrednictwem monitora. Urządzenie to obecnie mogło by stanowić przedmiot odrębnego wykładu — tak wiele można o nim powiedzieć.

Zacznijmy od początku. Ekran. Jest wiele technik tworzenia obrazu. Najpopularniejsze są ekrany podobne do telewizyjnego (na bazie lampy kineskopowej) — mają jednak co najmniej dwie poważne wady: — wymagają stosunkowo wydajnego źródła prądu (baterie nie wchodzą w rachubę) — mają ujemny wpływ na zdrowie użytkowników (głównie narażone są oczy). Alternatywą „usiłuje być" ekran ciekłokrystaliczny (LCD). Technologia ta od kilku lat zdaje egzamin (z różnym skutkiem) np. w niektórych zegarkach elektronicznych i kalkulatorach. Ekrany tego typu pobierają bardzo mało prądu i mogą pracować „na baterie", ale są to jednocześnie „ekrany dla jednej osoby", mało wyraźne, wymagające zewnętrznego źródła światła. Wadę tę wyeliminowano w „gas-plazma" ekranach. Technologia ta ma podobno same zalety?

We wszystkich typach ekranów obraz budowany jest z małych punktów — tak zwanych pikseli (elementów obrazu). Odpowiednio ułożone piksele różne od zera (zero — biel, jedynka — czerń) tworzą obraz na ekranie. Steruje tym odpowiedni podzespół w jednostce centralnej komputera. Oczywiście im więcej punktów składa się na obraz — tym jest on lepszej jakości. W monitorach monochromatycznych (o nich jest właśnie mowa) czyli niekolorowych — biel (ze względu na mniejszą szkodliwość dla oczu) zastępowana jest różnymi odcieniami zieleni, bursztynu i innych „bezpiecznych" dla oczu kolorów.

Jednym z podstawowych parametrów opisujących monitor (i związane z nim sterowanie) jest **rozdzielczość**. Typowa wartość tego

Wyraźną niedoskonałością takiego zestawu jest brak pamięci zewnętrznych, nazywanych także pamięciami masowymi.

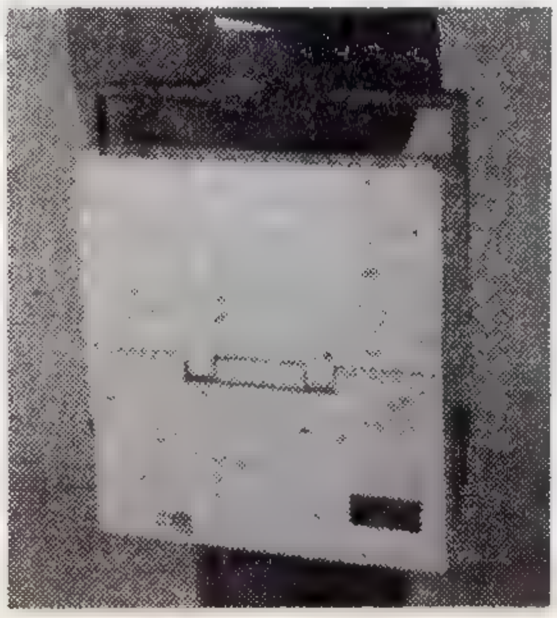
Najpowszechniejszymi pamięciami tego typu są różne „odmiany" dysków. Zacznijmy od najprostszych — **dysków miękkich,** nazywanych także dyskami **elastycznymi** lub po prostu „flopy". Jest to urządzenie, tak zwany „napęd" najczęściej zamontowane w jednostce centralnej komputera. Napęd umożliwia zapisywanie i odczytywanie informacji, której nośnikiem jest dyskietka.

Sprzęt informatyczny

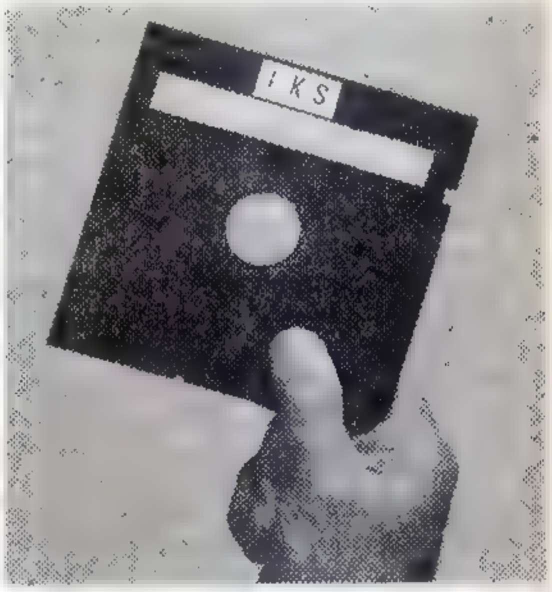
(Materiały do Podstaw Informatyki cz. 4)

parametru waha się w granicach: 700x600 pikseli. Należy pamiętać jednak, że w profesjonalnych zastosowaniach graficznych wykorzystywane są monitory o ogromnej rozdzielczości, np.: 4000x3000. pikseli, przy czym każdy z nich może przyjmować miliony różnych odcieni wszystkich barw tęczy. Mówiąc o barwach warto wspomnieć, że najczęściej są one tworzone z „mieszanki" odpowiednich proporcji barw podstawowych: czerwień, zieleń i barwa niebieska. Jest to tak zwany standard **RGB**. Barwy są uzyskiwane, jak dotąd, głównie na monitorach budowanych tradycyjną technologią. Piszę, że „głównie", widziałem bowiem na listopadowej wystawie Audio-Video telewizor firmy Hitachi wielkości pudełka zapałek (10 paczek) o ekranie 5 cali — LCD, piękne kolory!

Jednostka centralna, klawiatura i monitor stanowią podstawowy, minimalny zestaw komputera, który można wykorzystywać w praktyce. Przykładem komputera pracującego w takim właśnie zestawie jest „laptop", co w wolnym tłumaczeniu oznaczą komputer składany (walizkowy).



4. Przykład „składanego" (laptop) komputera.

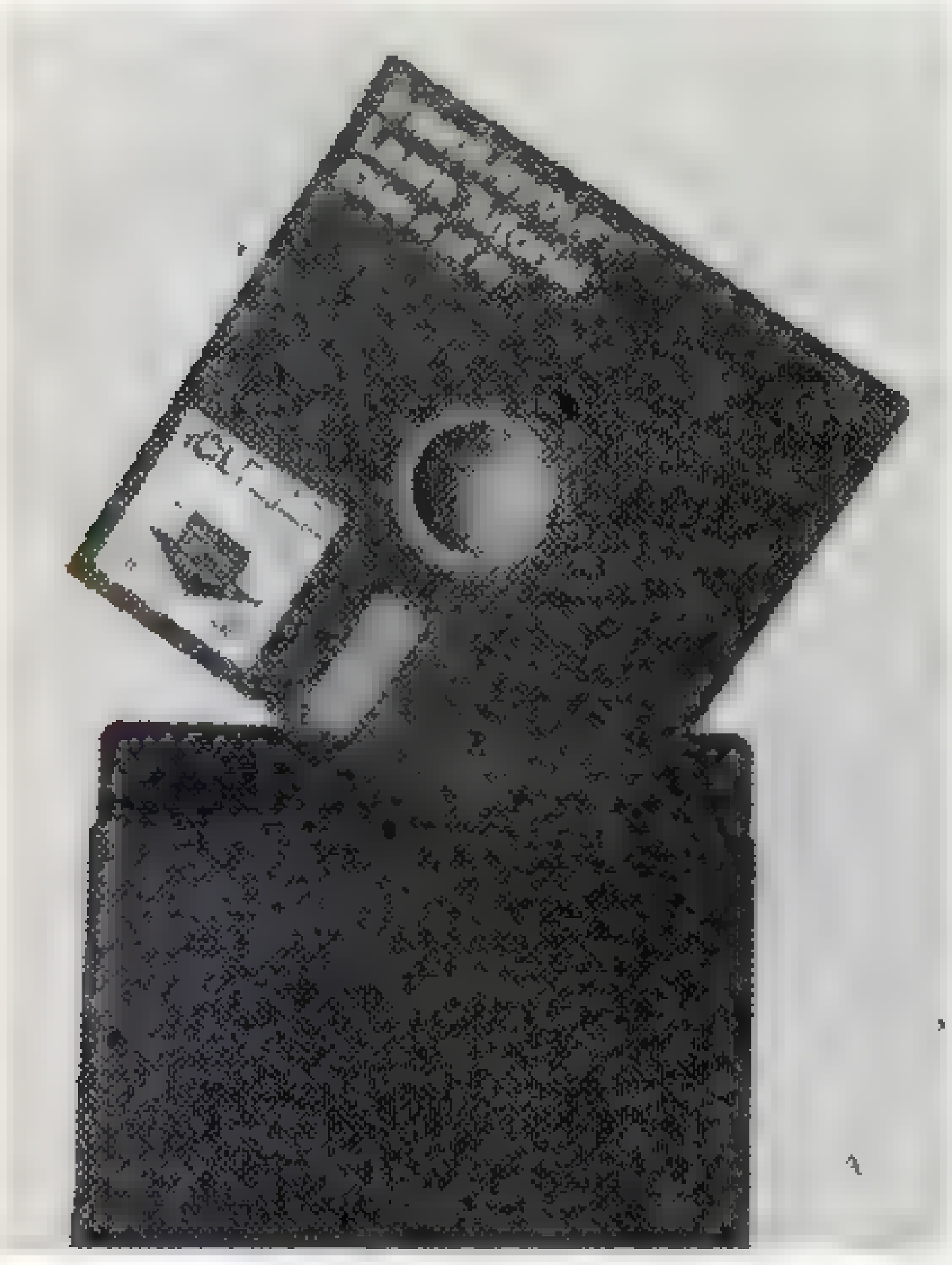


5. Dyskietka tuż przed włożeniem do napędu.

Dyskietka to magnetyczny nośnik informacji w postaci krążka (podobnie jak płyta), różnej wielkości, najczęściej 5,25 cala, 3,5 cala i rzadko 8 cali. W celu zwiększenia bezpieczeństwa przed fizycznym zniszczeniem dyskietki przechowywane są w dwóch kopertach, jedna w typowej postaci, druga natomiast jest na stałe „zalepiona" i krążka dyskietki z niej nie można wyjąć. Tak opakowaną dyskietkę wkładamy do napędu. Promienisty otwór w stałej kopercie pozwala dotrzeć głowicy zapisująco-odczytującej w każdy punkt owego krążka. Możliwe jest to dzięki temu, że napęd wprawia dyskietkę w ruch obrotowy wokół krążka.

Typowymi pojemnościami dyskietek są: 360 KB i 1,2 MB. Bywają także dyskietki o innych pojemnościach, nawet do 10 MB.

W ten sposób poznaliśmy uniwersalne urządzenie zewnętrzne komputera, może służyć do wyprowadzania, jak i wprowadzania informacji. Bardzo cenną zaletą dysku elastycznego jest jego wymienność. Oznacza to, że wszelkiego typu informacje (dane, programy)



możemy wczytywać z dyskietki zapisanej na innym komputerze. Dzięki tej własności, pracowicie wprowadzane dane do komputera możemy przepisać (stosunkowo szybko) na dyskietkę, po to, by za rok lub dwa bardzo szybko ponownie wprowadzić do maszyny przechowywane informacje w celu ich dalszego przetwarzania.

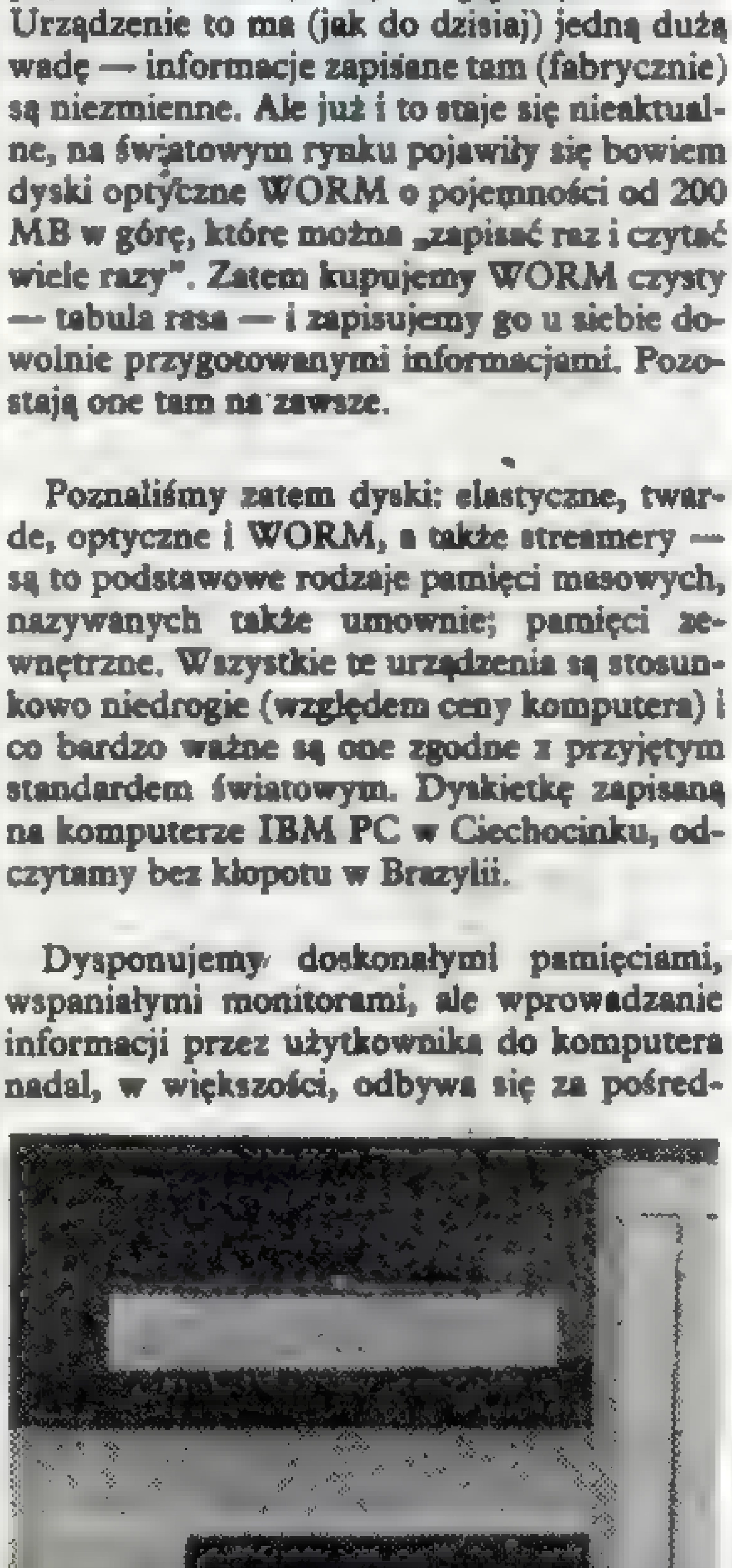
Niestety tej ogromnej zalety — wymienności — nie mają **dyski twarde**, tak zwane winchestery. Są to także napędy, najczęściej wbudowane do jednostki centralnej komputera. O ich użyteczności przesądziły dwie inne cechy: duża pojemność i szybki dostęp (czas odszukania) do informacji zapisanych „na twardym". Czas ten w zasadzie nie przekracza 20 ms (są dyski o czasie 10 ms i powyżej 50 ms). Zatem na twardym możemy tysiące razy zapisywać informacje, kasować je i odczytywać. Ogromna użyteczność tych pamięci spowodowała, że już coraz częściej się mówi o produkcji winchesterów wymiennych — rewelacja! Sądzę jednak, że niska cena dyskietek będzie długo konkurencyjna wobec tej nowości. Póki co, dysk twardy jest niewymienny (zamknięty szczelnie jak konserwa), a jego pojemność waha się od 20 MB do setek MB. Chwilę (kilka sekund) po włączeniu komputera mamy dostęp do pracowicie zgromadzonych tam informacji — zazwyczaj są to najczęściej używane programy, w tym oczywiście operacyjny, który będzie przedmiotem jednego z naszych kolejnych spotkań.

Co się stanie, gdy dysk twardy po prostu się zepsuje? Albo zostanie zarażony wirusem? Koniec. Wszystko co było tam zapisane przepada bezpowrotnie. Zważywszy na to, że częstokroć cena informacji zgromadzonych tam jest ogromna — skonstruowano streamer, kolejny typ pamięci (wolno stojące lub wbudowane do komputera). Jest to odmiana magnetofonu, który zapisuje zamiast muzyczki — „bajty z komputera", oczywiście na specjalnej taśmie (podobnej do stosowanej w wideo). Pojemność streamera (ilość informacji, jaką może zarejestrować) jest większa od pojemności winchestera. Wynika to z przeznaczenia tego „magnetofonu" — archiwizowanie wszystkich informacji zapisanych na dysku (na wszelki wypadek). Jest to pamięć sekwencyjna (zapisuje się i czyta po kolei) i służy wyłącznie do zabezpieczania informacji przed ich ewentualną stratą.

Bywa, że często korzystamy z ogromnych zbiorów stosunkowo niezmiennych informacji. Na przykład w czytelni zapewne przydał by się komputer dysponujący informacjami z Dużej Encyklopedii (50 tys. stron), Słownika Wyrazów Obcych (2 tys. stron), słowników: polsko-angielskiego, polsko-francuskiego, polsko-niemieckiego, polsko-rosyjskiego i odwrotnie, razem około 48 tys. stron. Łącznie około 100 tys. stron. Dane te zajęły by znacznie mniej miejsca niż wynosi pojemność połowy pamięci **dysku optycznego**, którego pojemność szacuje się w gigabajtach GB. Urządzenie to ma (jak do dzisiaj) jedną dużą wadę — informacje zapisane tam (fabrycznie) są niezmienne. Ale już i to staje się nieaktualne, na światowym rynku pojawiły się bowiem dyski optyczne WORM o pojemności od 200 MB w górę, które można „zapisać raz i czytać wiele razy". Zatem kupujemy WORM czysty — tabula rasa — i zapisujemy go u siebie dowolnie przygotowanymi informacjami. Pozostają one tam na zawsze.

Poznaliśmy zatem dyski: elastyczne, twarde, optyczne i WORM, a także streamery — są to podstawowe rodzaje pamięci masowych, nazywanych także umownie; pamięci zewnętrzne. Wszystkie te urządzenia są stosunkowo niedrogie (względem ceny komputera) i co bardzo ważne są one zgodne z przyjętym standardem światowym. Dyskietkę zapisaną na komputerze IBM PC w Ciechocinku, odczytamy bez kłopotu w Brazylii.

Dysponujemy doskonałymi pamięciami, wspaniałymi monitorami, ale wprowadzanie informacji przez użytkownika do komputera nadal, w większości, odbywa się za pośrednictwem klawiatury. Aby ograniczyć tę niedoskonałość skonstruowano wiele różnych urządzeń ułatwiających przekazywanie informacji, przez człowieka, do komputera. Służą temu między innymi: „myszy", pióra świetlne, digitizery, ekrany dotykowe, analizatory dźwięków mowy ludzkiej. Ograniczają one nieco konieczność posługiwania się ponad setką klawiszy mikrokomputera. Praktycznie wszystkie te urządzenia, poza analizatorem, służą do wskazywania wybranego punktu na ekranie monitora. Uzyskuje się to dzięki przesuwaniu znacznika (kursora, białej lub kolorowej, także migającej plamki) po ekranie. W tym czasie na monitorze wyświetlany jest jeden z zestawów poleceń, które komputer potrafi wykonywać samodzielnie (zagadnienie to stanowić będzie część odrębnego tematu naszych zajęć). Jest to tak zwane **menu** (w formie podobnej zresztą do tego z restauracji). Wracając do znacznika — gdy przesuniemy go tak, że „dotknie" wybrane przez nas polecenie, np.: wczytaj program XYZ z dysku, drukuj możliwości programu XYZ, uruchom program XYZ, maszyna natychmiast przystępuje do jego wykonywania (oczywiście wskazanego polecenia).



6. Mysz.

Pióro świetlne podobne jest do używanego przez nas na co dzień, tyle, że łączy je z maszyną kabel i oczywiście piszemy nie na papierze, a na ekranie monitora. Podobnie funkcjonuje **digitizer**, w tym wypadku miejscem pracy jest specjalna tabliczka umieszczona zazwyczaj na stole przed monitorem. Rysując na tabliczce powodujemy na ekranie wierne odwzorowanie naszych ruchów. Narzędziem pracy jest specjalny czujnik w postaci pióra mechanicznego lub lupa z celownikiem (by lepiej było widać kreski, po których przesuwamy ów czujnik). Urządzenie to ma szerokie zastosowanie wszędzie tam, gdzie istnieje konieczność przeniesienia do pamięci komputera skomplikowanego rysunku, np.: projektu architektonicznego mostu. Wodzimy wówczas „celownikiem" po kreskach schematu, który wcześniej mocujemy na wspomnianej tabliczce, w pamięci maszyny powstaje dokładne odzwierciedlenie oryginału.

Bardzo popularna już dzisiaj **mysz** — to pudełko z zamocowaną w jego dnie ruchomą kulką (jak kulka w czubku długopisu) lub ostatnio mechanizm ten zastąpiono układem optycznym. Przesuwając mysz po płaszczyźnie biurka, kulka się obraca — jest to źródłem sygnałów elektrycznych przekazywanych kablem do komputera. W efekcie powoduje to ruch kursora po ekranie (zgodny z ruchem myszy po stole). Gdy kursor „dotknie" wybraną przez nas pozycję menu — naciskamy „ucho" (ma ich zazwyczaj aż trzy) myszy i maszyna wykonuje wskazaną operację. Natomiast na **monitorze dotykowym**, wystarczy wybrany punkt ekranu dotknąć palcem lub długopisem czy papierosem (najlepiej zgaszonym), by przekazać pełną informację o umiejscowieniu (współrzędne X i Y) wyróżnionego przez nas obszaru ekranu.

Zasygnalizowane rozwiązania mają na celu uproszczenie czynności związanych z przekazywaniem informacji do mikrokomputera. Jednak niemal wszyscy, po cichu, oczekujemy,

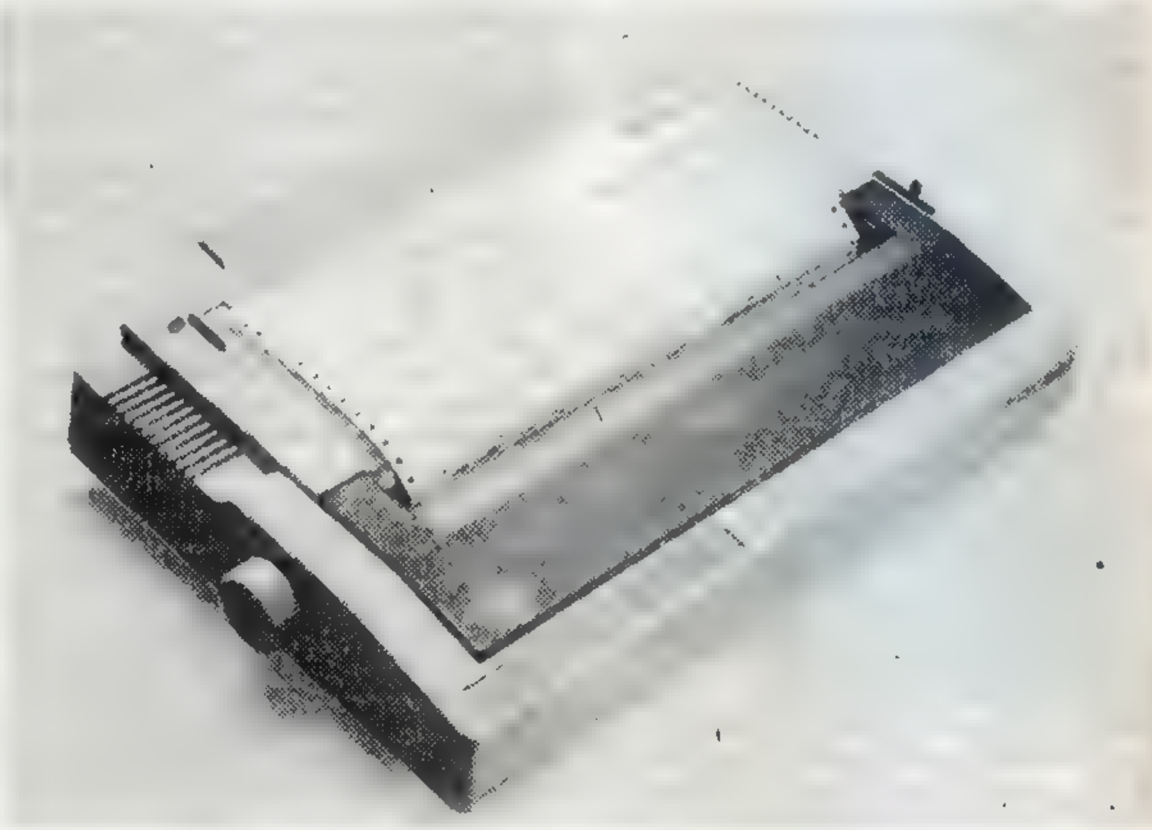
by komputer słuchał tego co mówimy, a co najważniejsze, by rozumiał nas. Przykładem próby takiego rozwiązania jest twór brytyjskiej firmy APRICOT — mikrokomputer, który rozpoznaje do 4000 słów. Nowy właściciel tego sprzętu musi określić, które komendy (słowa) mają być przez maszynę rozpoznawane i potem zobowiązany jest do wypowiedzenia kolejnych komend po pięć razy, aż komputer nauczy się je rozpoznawać. Jednak profesjonalny rynek oczekuje mikrokomputera ze słownikiem 20 000 słów, jednym słowem eksperyment się nie powiódł.

W uzupełnieniu informacji dotyczących urządzeń wejściowych komputera z przyjemnością (bo wiele lat spędziłem nad budową i eksploatacją tego typu urządzeń) pragnę przypomnieć sprzężenie kamery TV z komputerem. Są to, omawiane już wcześniej, **skanery**, obecnie dostępne nawet w wersji „ręcznej" — ręczne skanery, wraz z odpowiednim oprogramowaniem rozpoznającym pismo maszynowe (niestety tylko kodu ASCII). Przesuwamy takim skanerem po tekście gazety — jest on natychmiast odczytywany (rozpoznawane są znaki) i wpisywane do pamięci komputera. W sumie bardzo ciekawe urządzenie.

Świadomie pominąłem starsze typy urządzeń wejściowych maszyn cyfrowych, jak na przykład: czytniki kart perforowanych, czytniki tasiemki perforowanej oraz magnetyczne rejestratory danych. Sprzęt ten jest jeszcze u nas w kraju wykorzystywany, ale kontakt z nim mają tylko profesjonaliści, stąd też uznałem, że można sobie podarować te, w zasadzie już „wiekowe" urządzenia.

Pora wspomnieć o urządzeniach wyjściowych komputerów. Niezmiennie od początku historii mikrokomputerów są to **drukarki**. Obecnie urządzenia te, w wersji dla mikrokomputerów są małe, ciche, w miarę niezawodne i... bardzo wolne (w stosunku do drukarek wierszowych, które stosowane są na ogół do dużych komputerów). Zważywszy jednak, że większość informacji generowanych przez mikrokomputer użytkownikowi przekazywana jest za pośrednictwem monitora, małe drukarki są niezmiennie bardzo użyteczne. Zasada pracy tych drukarek polega na tym, iż każdy znak przez nie drukowany tworzony jest przez mozaikę odpowiednio ułożonych kropek, stąd popularna nazwa tych urządzeń — **drukarki mozaikowe**. W głowicy drukarki, która przesuwana jest odpowiednim mechanizmem wzdłuż kolejnych linii tworzonego tekstu na kartce papieru, mieści się zestaw igieł — prostopadle do papieru. Igły te uderzając przez taśmę barwiącą w papier pozostawiają ową mozaikę — znak lub fragment ryciny.

Usiłując podnieść jakość wydruków stosuje się coraz większą liczbę, coraz cieńszych igieł. Ostatnim osiągnięciem w tej dziedzinie jest zastąpienie igieł promieniami świetlnymi. Wykorzystywana jest do tego celu technika laserowa, stąd — **drukarki laserowe**. Obszar o wielkości 1 mm² „uderzany" jest równomiernie przez 12x12 (i więcej) „świetlnych igieł". Zatem jakość tworzonych tym sposobem wydruków stanowi nadzwyczajną konkurencję nawet dla profesjonalnego sprzętu wykorzystywanego w drukarniach. Poprawiła się także nieco szybkość — typowa wydajność drukarki laserowej — 8 stron A4 w czasie jednej minuty.

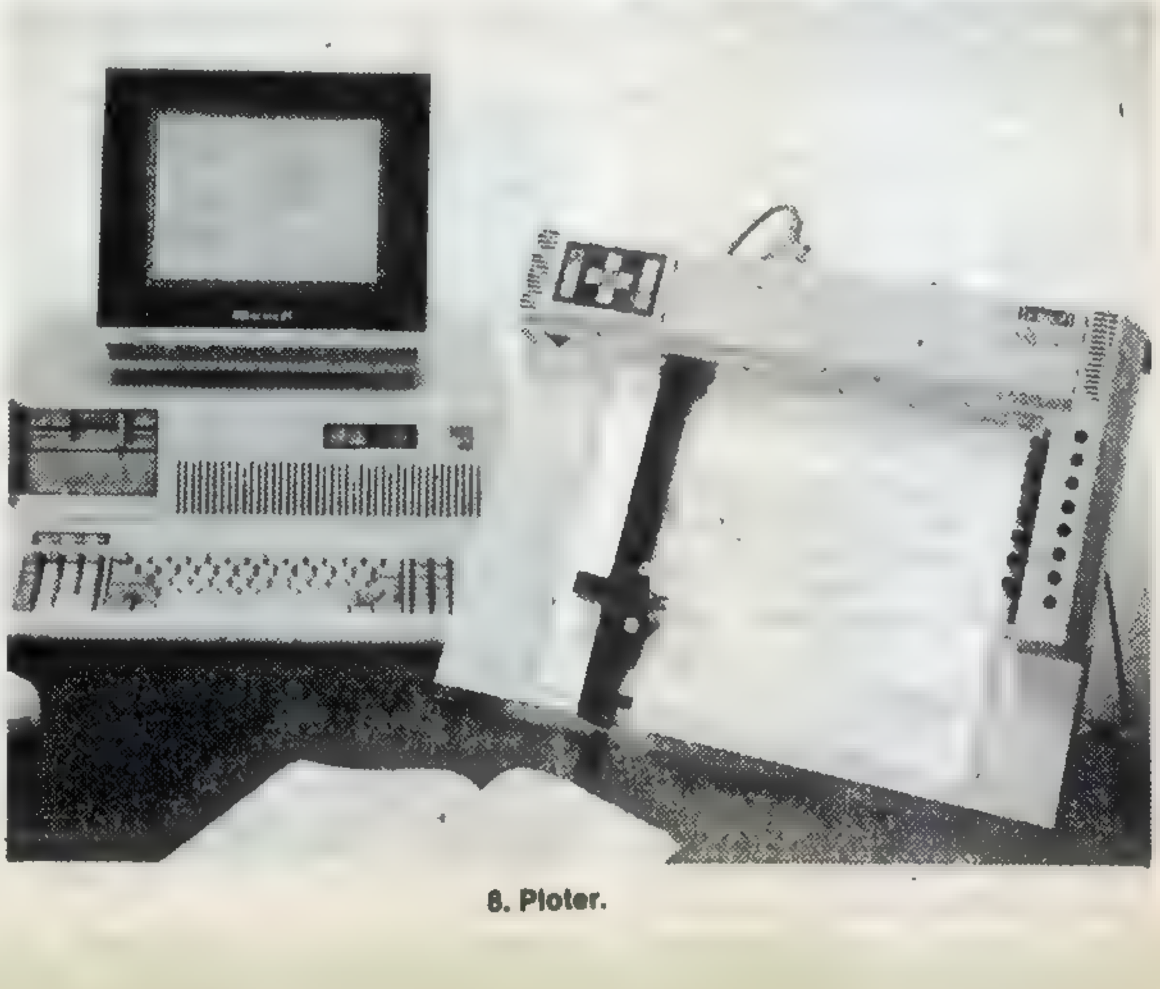


7. Przykład typowej drukarki do mikrokomputera.

Bardzo użytecznym uzupełnieniem drukarek są **plotery**. Są to urządzenia umożliwiające rysowanie ciągłych różnokolorowych linii na kartkach papieru od A5 do A0.

Tyle w wielkim skrócie o urządzeniach stanowiących „zmysły" i „narządy wykonawcze" naszego mikrokomputera. Sądzę zatem, że prezentację „idei" komunikowania się użytkownika z komputerem mamy już (nareszcie?) za sobą. Jeśli idzie o szczegóły — są one bardzo czytelnie przedstawione w bogatej literaturze, także we wcześniej polecanej pozycji „Komputery osobiste" D. Madeja i innych oraz niemal w każdym numerze „IKS-a", „Komputera", „Bajtka", czy „nobliwego" „Mikroklanu".

Włodzimierz GOGOŁEK



8. Ploter.

DYSKI TWARDE – CENY

dalszy ciąg ze str. 1.

MOUNTAIN

- DC020, 20MB 551 $
- DC030, 30MB 621 $
- DC050, 50MB, 54ms 898 $

MINISCRIBE

- 3053, 40MB, HH, 28ms 540 $
- 3650, 40MB, HH 347 $
- 3675, 60MB, RLL 373 $
- 6085, 70MB, FH 720 $
- 8425, 20MB, 3.5", HH 252 $
- 8438, 30MB, 3.5", HH 259 $
- 9380, 336MB ESDI, 16ms 2200 $

PRIAM

- E40 , 40MB ext., FH, ST506 1041 $
- E60 , 60MB ext., FH, ST506 1093 $
- 140PW , 40MB z kontr. 970 $
- 145AT2 , 45MB ST 506, FH 756 $
- 160PW , 60MB z kontr. 1093 $
- 162AT2 , 62MB ST 506, FH 969 $
- 175RF , 74MB, RLL z kontr. 1025 $
- 1100 PC, 103MB z kontr. 1164 $
- 1100 RC, 103MB RLL z kontr. 1200 $
- 1130 AT, 133MB ST506, FH, 13ms 2046 $
- 1230 RC, 233MB z kontr. FDD-HDD 2367 $
- 1230 RF, 233MB RLL z kontr. 2336 $
- 1330 EC, 237MB ESDI z kontr. 3528 $

SEAGATE

- 125 , 20MB, 3.5", HH MFM 267 $
- 125-1, 20MB, 3.5", MFM, 28ms 310 $
- 138 , 30MB, 3.5", MFM 344 $
- 138N , 32MB, 3.5", SCSI, 40ms 389 $
- 138R , 32MB, 3.5", RLL , 40ms 358 $
- 157R , 49MB, 3.5", RLL , HH 468 $
- 157N , 49MB, 3.5", SCSI, 40ms 519 $
- 225 , 20MB, HH MFM 239 $
- 238 , 30MB, HH RLL 255 $
- 251 , 40MB, HH MFM, 40ms 399 $
- 251-1, 40MB, HH MFM, 28ms 546 $
- 277R , 65MB, HH RLL, 40ms 576 $
- 296N , 80MB, SCSI HH 1096 $
- 4053 , 40MB FH 577 $
- 4096 , 80MB FH 863 $
- 4096N, 80MB SCSI FM, 17ms 1043 $

Ceny według firmy :
"JB TECHNOLOGIES, INC."
21011 ITASCA ST., NF CHATSWORTH, CA 91311

5 1/4" HARD DISK FH

- IMI 5018 H, 15MB 185 $
- Tandon TM503, 15MB 185 $
- CMI 6426, 15MB 175 $
- Rodime 202E, 15MB 275 $
- Seagate 4026, 20MB 295 $
- CMI 6425, 20MB 250 $
- Tandon TM703CXT, 26MB 275 $
- Rodime 203E, 30MB 415 $
- Tandon TM703AT, 30MB 365 $
- Seagate 4038, 30MB 375 $
- CDC BJ7D5, 30MB 425 $
- CDC 91455-36, 30MB 425 $
- CMI 6640, 32MB 350 $
- Quantum Q540, 32MB 695 $
- Micropolis 1333, 35MB 475 $
- Atasi 3046, 39MB 415 $
- Hitachi DK511-5, 42MB 515 $
- Fujitsu 2242AS, 42MB 495 $
- Miniscribe 6053, 44MB 525 $
- Rodime 204E, 44MB 495 $
- Priam V170, 60MB 820 $
- Brand Tech 8085, 72MB 795 $
- Miniscribe 6085, 72MB 740 $
- Micropolis 1325, 72MB 795 $
- CDC 94155-86, 72MB 805 $
- Priam V185, 72MB 876 $
- Brand Tech 8128 RLL, 108MB 995 $
- Maxtor XT 1140, 120MB 1625 $
- Newbury 1140 28ms, 120MB 1495 $
- Newbury 1140 25ms, 120MB 1849 $
- Maxtor XT2190 28ms, 150MB 1995 $
- Newbury 2190 25ms, 160 MB 1875 $

5 1/4" HARD DISK HH

- NEC 5126, 20MB 245 $
- Microscience HH725, 20MB 195 $
- CMI 3426, 20MB 195 $
- Olivetti 5220/2, 20MB 230 $
- Tandon TM702AT, 20MB 279 $
- Seagate ST 225, 20MB 225 $
- Seagate ST 238RLL, 30MB 245 $
- Epson HMD 860 RLL, 30MB 255 $
- NEC 5146, 42MB 425 $
- Seagate ST 251, 40MB 425 $
- Tandon TM 755, 42MB 395 $

5 1/4 SCSI HARD DISK FH

- Newbury 3170S, 147MB 1395 $
- Newbury 3280S, 244MB 1595 $
- Newbury 3380S, 320MB 1995 $

5 1/4 ESDI HARD DISK FH

- Micropolis 1355, 175MB 1495 $
- Newbury 4175, 175MB 1295 $
- Siemens Megafile 1300, 275MB 1495 $
- Newbury 4380, 380MB 1999 $
- Maxtor 4380, 380MB 2495 $

8" HARD DISK

- Quantum Q2010, 10MB 89 $
- Quantum Q2020, 20MB 235 $
- Mitsubishi 2860-1s, 20MB 295 $
- Quantum Q2040, 40MB 595 $
- Mitsubishi 2860-2s, 50MB 630 $
- Mitsubishi 2860-3s, 80MB 710 $
- Quantum Q2010, 10MB 725 $

Popularne u nas dyski firmy "Seagate" oferuje bardzo tanio na amerykańskim rynku firma "LYCOCOMPUTER Marketing & Consultants"

- ST 225 , 20MB, 65ms MFM 215 $
- ST 225N, 20MB, SCSI 290 $
- ST 238R, 30MB, RLL 230 $
- ST 251 , 40MB, 40ms MFM 345 $
- ST 251-1, 40MB, 28ms MFM 430 $
- ST 277R, 65MB, 40ms RLL 390 $
- ST 277N, 64MB, SCSI 440 $
- ST 125 , 3.5", 20MB, 40ms, MFM 235 $
- ST 125N, 3.5", 20MB, SCSI 300 $
- ST 138R, 3.5", 30MB, RLL 250 $
- ST 138N, 3.5", 30MB, SCSI 330 $
- ST 157R, 3.5", 40MB, RLL 400 $
- ST 157N, 3.5", 48MB, SCSI 440 $
- ST 4053, 40MB, 28ms, MFM 470 $
- ST 4096, 80MB, 28ms, MFM 570 $

A teraz dla porównania podamy ceny najtańszych firm w RFN. "PROSOFT GmBH" D-5400 Kobleuz-Goldgrube"

- ST 225 , 20MB, 65ms 277 $
- ST 238R, 30MB, 65ms RLL 302 $
- ST 251-1, 40MB, 28ms 514 $
- ST 277R, 65MB, 40ms RLL 508 $
- ST 4096, 80MB, 28ms 750 $
- ST 4144R, 122MB, 28ms RLL 1150 $
- ST 125-0, 3.5", 20MB, 40ms 302 $
- ST 125-1, 3.5", 20MB, 28ms 350 $
- ST 138R , 3.5", 30MB, 40ms 332 $
- ST 157-0, 3.5", 50MB, 40ms 520 $

Jak widać ceny w RFN są dużo wyższe niż w USA pomimo iż firma "PROSOFT GmBH" ma najniższe w RFN ceny dysków (oprócz oczywiście przedstawiciela firmy Seagate w Monachium). Gdzie zatem szukać powodzenia w dobrym i tanim zakupie twardych dysków, wliczając w to ryzyko embarga ?. Pozostaje tylko Singapur, który przy zwyżce waluty Taiwańskiej o 11% i mocnym kursie walut Korei Południowej i Hong Hongu staje się najtańszym źródłem zaopatrzenia. Do zakupu dysku w Singapurze skłania również fakt istnienia tam wielu montowni i fabryk jak np. Seagate, Priam, itp.

Matriały zostały przygotowane przez firmę BOIIE sp. z o.o. — Biuro Obsługi Importu i Exportu, 00—867 Warszawa ul. Chłodna 35/37 paw. 1A, tel. 247818, tlx 817073 BOIIE PL.

Atari symuluje zjawiska falowe

Jan GOLLA

Dobre efekty w zobrazowaniu ugięcia, odbicia i załamania fal za pomocą komputera można osiągnąć przez zapełnianie ekranów kolejnymi fazami ruchu, a następnie szybkim ich przełączaniu (page flipping). W czasie inicjacji komputer wprowadza kolejne obrazy, różniące się nieznacznie od siebie, do swojej pamięci, a następnie szybko wyświetla je kolejno na ekranie, co obserwatorowi daje złudzenie ruchu ciągłego (podobnie jak w kinematografii). Tak rozwiązano tu wszystkie zjawiska oprócz ruchu pojedynczej fali przed jej ugięciem. Za pomocą klawiszy S (szybciej) oraz W (wolniej) odbieramy najkorzystniejszą dla obserwacji prędkość rozchodzenia się fal. Najdłużej trwa inicjacja ekranów przy symulacji ugięcia ciągu fal (około 140 sek.), czas ten można wykorzystać w trakcie lekcji na omawianie tego zjawiska lub na powtórkę.

```
10 REM ***ODBICIE FALI***
20 GRAPHICS 2:POSITION 4,4:? #6;
"ODBICIE FALI":POKE 657,8
30 ? "NACISNIJ DOWOLNY KLAWISZ":
POKE 764,255:POKE 755,0:SETCOLOR
 2,0,0
40 IF PEEK(764)=255 THEN 40
60 GRAPHICS 0:SETCOLOR 2,0,0:POK
E 82,2:POKE 83,38
70 POSITION 2,4:? "PROGRAM POKAZ
UJE ODBICIE FALI POD    WYBRANYM
 KATEM"
80 TRAP 60
90 POSITION 8,7:? "PODAJ KAT PAD
ANIA (0-70)",:INPUT K:DEG :OP=10
95 IF K>70 OR K<0 THEN 60
100 POSITION 7,10:? "PROSZE CZEK
AC OK.30 SEKUND":POKE 755,0
110 FOR O=1 TO 1000:NEXT O
120 POKE 106,160
130 FOR E=1 TO 5
140 POKE 106,160-E*16
150 GRAPHICS 6:COLOR 1:POKE 559,
0
160 SI=SIN(K):CO=COS(K)
170 POKE 82,8:POKE 755,0:POKE 65
7,8
180 ? "KLAWISZ S-SZYBCIEJ":? "KL
AWISZ W-WOLNIEJ"
190 ? "KLAWISZ ESC-NOWY KAT"
200 PLOT 80,1:DRAWTO 80,80:SETCO
LOR 2,0,0
210 FOR PZ=-10 TO 110 STEP 5
220 TRAP 250
230 PLOT 80+(PZ+E)*SI-12*CO,80-(
PZ+E)*CO-12*SI
240 DRAWTO 80+(PZ+E)*SI+12*CO,80
-(PZ+E)*CO+12*SI
250 NEXT PZ
260 FOR PZ=-10 TO 80 STEP 5
270 TRAP 300
280 PLOT 80-(PZ-E)*SI+12*CO,80-(
PZ-E)*CO-12*SI
290 DRAWTO 80-(PZ-E)*SI-12*CO,80
```

```
-(PZ-E)*CO+12*SI
300 NEXT PZ
310 FOR FA=78 TO 83 STEP 0.05:TR
AP 340
320 PLOT 80-FA*SI+14*CO,80-FA*CO
-14*SI
330 DRAWTO 80-FA*SI-14*CO,80-FA*
CO+14*SI
340 NEXT FA
350 NEXT E
360 GRAPHICS 6 POKE 559,34
370 FOR EK=135 TO 87 STEP -16:PO
KE 561,EK
380 FOR AS=1 TO OP
390 IF PEEK(764)<255 THEN GOTO 4
40
400 NEXT AS
410 NEXT EK
420 GOTO 370
430 REM ***REGULACJA PREDKOSCI**
*
440 IF PEEK(764)=28 THEN CLR :GO
TO 60
450 IF PEEK(764)=46 THEN OP=OP+1
460 IF PEEK(764)=62 AND OP>2 THE
N OP=OP-1
470 POKE 764,255:GOTO 380
```

* SUMA KONTROLNA / ETYKIETA *

FT	10	UP	20	NN	30	QQ	40
OX	50	PX	60	RY	70	RU	80
UF	90	VP	95	UJ	100	YB	110
PM	120	MJ	130	WV	140	IM	150
AZ	160	TX	170	JS	180	BJ	190
BT	200	ZY	210	MI	220	VV	230
EL	240	UW	250	BG	260	KZ	270
ZJ	280	LZ	290	UN	300	DL	310
BY	320	FU	330	GK	340	EP	350
RB	360	PS	370	AT	380	ZX	390
LV	400	KB	410	PG	420	EC	430
BT	440	WS	450	PK	460	YF	470

```
10 REM ***UGIECIE FAL***
20 DIM MH(60),AK(60):POKE 566,15
8:POKE 764,255:POKE 83,37
30 GRAPHICS 2:SETCOLOR 2,0,0:POK
E 755,0:POSITION 4,4
40 ? #6,"UGIECIE FALI"
50 POKE 657,8:? "NACISNIJ DOWOLN
Y KLAWISZ"
60 IF PEEK(764)=255 THEN 60
70 GRAPHICS 0:SETCOLOR 2,0,0:POS
ITION 2,5:POKE 755,0
80 POSITION 2,1:? "PROGRAM SKLAD
A SIE Z DWOCH CZESCI:"
90 ? :? "CZESC 1:UGIECIE JEDNEJ
FALI":? "CZESC 2:UGIECIE CIAGU F
AL"
100 POSITION 2,7:? "KLAWISZ ESC
POWODUJE PRZEJSCIE DO   CZESCI 2
"
110 POSITION 9,10:? "TRWA INICJA
CJA CZESCI 1."
120 POSITION 7,12:? "PROSZE CZEK
AC OK.30 SEKUND."
130 FOR MT=1 TO 6000:NEXT MT
140 REM ***UGIECIE POJEDYNCZEJ F
ALI***
150 FOR KO=6.3 TO 8 STEP 0.03:N=
N+1:MH(N)=SIN(KO):AK(N)=COS(KO):
NEXT KO
160 BS=PEEK(106)-16:POKE 106,BS:
POKE 559,0:POKE 54279,BS
170 MS=BS*256+1024+72
180 FOR X=MS TO MS+109:POKE X,12
8:NEXT X:FOR ST=0 TO 5
190 POKE 106,160-ST*16:GRAPHICS
6+16:COLOR 1:POKE 559,0:GOSUB 31
0
200 RK=10+ST*10:FOR N=2 TO 52:XO
=MH(N)*RK+100:YO=AK(N)*RK+47
210 TRAP 230
220 PLOT XO,YO:PLOT XO,95-YO
230 NEXT N:NEXT ST
240 GRAPHICS 6+16:POKE 53277,14:
POKE 559,62:GOSUB 310
250 POKE 704,8:FOR PK=60 TO 148
STEP 0.8:POKE 53248,PK
260 FOR RM=1 TO 7:IF PEEK(764)=2
8 THEN POKE 53248,0:GOTO 330:NEX
T RM
270 NEXT PK:POKE 704,0
280 FOR GD=151 TO 64 STEP -16:PO
KE 561,GD
290 FOR PZ=1 TO 14:IF PEEK(764)=
28 THEN POKE 53248,0:GOTO 330
300 NEXT PZ:NEXT GD:GOTO 250
310 FOR P=95 TO 100:PLOT P,1:DRA
WTO P,45:PLOT P,50:DRAWTO P,95:N
EXT P
320 FOR KZ=10 TO 15:PLOT KZ,20:D
RAWTO KZ,75:NEXT KZ:RETURN
330 GRAPHICS 0:SETCOLOR 2,0,0:PO
KE 755,0:OP1=20:POSITION 9,4
340 ? "CZESC 2:RUCH CIAGU FAL"
350 ? :? "W CELU ZWIEKSZENIA PRE
DKOSCI FAL NA CISKAJ KLAWISZ S".
360 ? "W CELU ZMNIEJSZENIA PREDK
OSCI FAL NACISKAJ KLAWISZ W"
370 ? "W CELU ZATRZYMANIA PROGRA
MU NACISNIJ KLAWISZ ESC"
380 POSITION 9,14:? "TRWA INICJA
CJA CZESCI 2.".
390 POSITION 7,16:? "PROSZE CZEK
AC OK.140 SEKUND."
400 FOR RB=1 TO 6000:NEXT RB:FOR
ST=1 TO 6
410 REM ***UGIECIE CIAGU FAL***
420 FOR ST=1 TO 5
430 GRAPHICS 6+16:SETCOLOR 2,0,0
:COLOR 1:POKE 559,0:POKE 106,160
-ST*16
440 FOR G=10 TO 15:PLOT G,20:DRA
WTO G,75:NEXT G
450 FOR P=95 TO 100:PLOT P,1:DRA
WTO P,45:PLOT P,50:DRAWTO P,95:N
EXT P
460 FOR F=9+ST TO 94+ST STEP 5:P
LOT F,20:DRAWTO F,75 NEXT F
470 FOR RK=1+ST TO 72 STEP.6
480 FOR N=2 TO 57:XO=MH(N)*RK+10
0:YO=AK(N)*RK+47
490 TRAP 500:PLOT XO,YO PLOT XO,
95-YO
500 NEXT N:NEXT RK:NEXT ST
510 POKE 764,255:POKE 77,0:POKE
764,255
520 POKE 708,40:POKE 709,202:POK
E 559,34
530 FOR EK=135 TO 64 STEP -16:PO
KE 561,EK
540 FOR OP=1 TO OP1:IF PEEK(764)
<255 THEN 580
550 NEXT OP:NEXT EK
560 GOTO 530
570 REM ***REGULACJA PREDKOSCI C
IAGU FAL***
580 IF PEEK(764)=46 OR OP1<0 THE
N OP1=OP1+0.4
590 IF PEEK(764)=28 THEN END
600 IF PEEK(764)=62 OR OP1>50 TH
EN OP1=OP1-0.4
610 POKE 764,255:GOTO 550
```

* SUMA KONTROLNA / ETYKIETA *

WK	10	RW	20	AY	30	UL	40
OU	50	SS	60	GW	70	YI	80
IX	90	TC	100	VC	110	BB	120
EQ	130	JA	140	PC	150	FY	160
AO	170	CF	180	NA	190	XR	200
LK	210	RZ	220	SW	230	MM	240
LI	250	MD	260	DY	270	AW	280
KU	290	RF	300	FU	310	SZ	320
UZ	330	WV	340	UD	350	LW	360
NF	370	AA	380	BB	390	ED	400
ZF	410	MD	420	DQ	430	UF	440
GD	450	PP	460	PU	470	IA	480
XV	490	SC	500	PY	510	NC	520
LS	530	II	540	HG	550	OR	560
QS	570	WT	580	WH	590	UR	600
WY	610						

```
10 REM ***ZALAMANIE FALI ***
20 GRAPHICS 1:SETCOLOR 0,2,12:PO
SITION 3,8
30 ? #6;"ZALAMANIE FALI"
40 ? CHR$(253):? "ZATRZYMAJ MAGN
ETOFON."
50 ? "NACISNIJ DOWOLNY KLAWISZ."
60 POKE 764,255:TRAP 80
70 IF PEEK(764)=255 THEN 70
80 POKE 764,255:GRAPHICS 0:POSIT
ION 2,10:POKE 106,160:M=5
90 ? "PODAJ KAT PADANIA A,(SIN(A
)<n)";:INPUT A:POSITION 2,12:TRA
P 100
100 ? "PODAJ WSPOLCZ. ZALAMANIA"
;:INPUT N:DEG :GRAPHICS 0:POSITI
ON 8,10
110 POKE 755,0 .
120 ? "PROSZE CHWILE POCZEKAC."
130 FOR OP=1 TO 500:NEXT OP
140 FOR EK=1 TO 5
150 B=ATN((SIN(A)/N)/SQR(-(SIN(A
)/N)*(SIN(A)/N)+1))
160 SIA=SIN(A):COA=COS(A):SIB=SI
N(B):COB=COS(B)
170 REM *** ZAPELNIANIE EKRANOW
***
180 GRAPHICS 6:POKE 559,0:SETCOL
OR 2,0,0:COLOR 1
190 POKE 106,160-EK*16
200 ? "DO REGULACJI PREDKOSCI SL
UZA KLAWISZE:"
210 ? "S-SZYBCIEJ ";:? " W-WOLNI
EJ ";:POKE 755,0
220 ? " ESC-NOWA FALA"
230 FOR G=42 TO 79:COLOR 1:PLOT
1,G:DRAWTO 159,G:NEXT G
240 FOR D=-20 TO 120 STEP 4
250 TRAP 290
260 COLOR 1
270 PLOT 80-(D-EK)*SIA+10*COA,40
-(D-EK)*COA-10*SIA
280 DRAWTO 80-(D-EK)*SIA-10*COA,
40-(D-EK)*COA+10*SIA

300 PLOT 80+(D+EK)*SIB-10*COB/CO
A*COB,40+(D+EK)*COB+10*COB/COA*S
IB
310 DRAWTO 80+(D+EK)*SIB+10*COB/
COA*COB,40+(D+EK)*COB-10*COB/COA
*SIB
320 NEXT D
330 NEXT EK
340 POKE 559,34
350 REM *** PETLA ANIMACYJNA ***
360 FOR X=151 TO 87 STEP -16
370 POKE 561,X
380 FOR Q=1 TO M
390 IF PEEK(764)=46 THEN GOSUB 4
60
400 IF PEEK(764)=62 THEN GOSUB 4
70
410 IF PEEK(764)=28 THEN 80
420 NEXT Q
430 NEXT X
440 GOTO 360
450 REM *** REGULACJA PREDKOSCI
***
460 M=M+0.1:POKE 764,225:RETURN
470 M=M-0.1:POKE 764,255:RETURN
```

* SUMA KONTROLNA / ETYKIETA *

KC	10	OU	20	RX	30	YD	40
TG	50	XB	60	TT	70	JB	80
TG	90	AQ	100	KD	110	ER	120
UD	130	EH	140	BY	150	RL	160
HQ	170	SZ	180	FP	190	JZ	200
EH	210	QA	220	PL	230	AT	240
OG	250	XW	260	TI	270	VA	280
PW	290	PL	300	VL	310	DZ	320
KE	330	RD	340	EN	350	DL	360
EA	370	FT	380	UB	390	TD	400
YO	410	JA	420	LU	430	OZ	440
QM	450	MS	460	QB	470		

Zamiana liczb podanych cyfrowo na postać słowną

Danuta KWASIŻUR

Przy wielu problemach rozwiązanych przez mikrokomputery występuje potrzeba rozkodowywania liczb zadanych cyfrowo na postać alfanumeryczną. Algorytm zamiany wbrew pozorom nie jest prosty. Trzeba bowiem analizować końcówki (np. tysiąc, tysiące, tysięcy). Stopień trudności rośnie proporcjonalnie do górnej granicy podziału, który podlega zamianie.

Poniższy program demonstracyjny obsługuje przedział (1,999999). Jego wersja źródłowa jest następująca:

```
10 ' Program demonstracyjny
20 GOSUB 330
30 ' Deklaracja kolorow i okien
40 MODE 1
50 INK 0,26:INK 1,0
60 PAPER 1:CLS:BORDER 0
70 WINDOW £1,1,40,1,4
80 PAPER £1,0:CLS £1:PEN £1,1
90 WINDOW £2,1,40,23,25
100 PAPER £2,0:CLS £2:PEN £2,1
110 ' wypisanie komunikatu w oknie 1
120 LOCATE £1,5,2:PRINT£1,"Zamiana liczby pod
anej cyfrowo"
130 LOCATE £1,3,3:PRINT£1,"na postac slowna z
 zakresu 1-999999"
140 ' Wprowadzenie liczby
150 PEN 0:LOCATE 1,7
160 PRINT"Podaj liczbe: "
170 INPUT "",dana
180 ' Kontrola dopuszczalnosci liczby
190 IF dana>0 AND dana<1000000 GOTO 230
200 LOCATE £2,7,2:PRINT£2,"Niedopuszczalna li
czba !!!"
210 FOR t=1 TO 2000:NEXT
220 GOTO 60
230 GOSUB 440
240 ' Wypisanie postaci rozkodowanej
250 LOCATE 1,11
260 PRINT"Postac rozkodowana:"
270 PRINT sl$
280 LOCATE £2,2,2:PRINT £2,"Nastepna liczba (
T/N):";
290 INPUT £2,"",d$
300 IF d$="t" OR d$="T" THEN 60
310 CLS:LOCATE 10,10:PRINT"KONIEC PROGRAMU"
320 GOTO 320
```

W trakcie realizacji tego programu są wykorzystywane 2 podprogramy o zasadniczym znaczeniu dla zamiany liczb na postać słowną:

1. Podprogram wczytywania słów do tablicy

W czterdziestoelementowej tablicy znakowej t1$ przechowywane są stałe znakowe, które służą do sklejania rozkładowej liczby w postaci słownej. Wersja źródłowa tego podprogramu jest następująca:

```
330 ' Podprogram wczytywania slow do tablicy
340 DIM t1$(40)
350 FOR i=1 TO 40
360 READ t1$(i)
370 NEXT
380 DATA " ","jeden","dwa","trzy","cztery","p
iec","szesc","siedem","osiem","dziewiec","dzi
esiec"
390 DATA "jedenascie","dwanascie","trzynascie
","czternascie","pietnascie","szesnascie","si
edemnascie","osiemnascie","dziewietnascie"
400 DATA "dwadziescia","trzydziesci","czterdz
iesci","piecdziesiat","szescdziesiat","siedem
dziesiat","osiemdziesiat","dziewiecdziesia
t"
410 DATA "sto","dwiescie","trzysta","czteryst
a","piecset","szescset","siedemset","osiemset
","dziewietset"
420 DATA "tysiac","tysiace","tysiecy"
430 RETURN
```

2. Podprogram rozkodowywania i sklejania liczb

Zmienne pomocnicze wykorzystywane w tym podprogramie odpowiednio reprezentują:

St — rząd tysięcy;
dt — rząd dziesiątek tysięcy;
ts — rząd jedności tysięcy;
s — rząd setek;
d — rząd dziesiątek;
j — rząd jedności;

Zmienna St$ po wyjściu z tego programu zawiera rozkodowaną liczbę. Wersja źródłowa tego podprogramu jest następująca:

```
440 ' Podprogram rozkodowywania liczby
450 st=FIX(dana/100000):dan1=dana-st*100000
460 dt=FIX(dan1/10000):dan2=dan1-dt*10000
470 ts=FIX(dan2/1000):dan3=dan2-ts*1000
480 s=FIX(dan3/100):dan4=dan3-s*100
490 d=FIX(dan4/10):j=dan4-d*10
500 sl$=""
510 IF st=0 THEN 530
520 sl$=sl$+t1$(st+28)+" "
530 IF dt=0 THEN 570
540 IF dt=1 THEN 560
550 sl$=sl$+t1$(19+dt)+" ":GOTO 570
560 sl$=sl$+t1$(11+ts)+" ":GOTO 620
570 IF ts=0 THEN 620
580 sl$=sl$+t1$(1+ts)+" "
590 IF (ts=1 AND dana>2000) OR ts>4 THEN sl$=
sl$+t1$(40)+" ":GOTO 630
600 IF ts=1 THEN sl$=sl$+t1$(38)+" ":GOTO 630
610 sl$=sl$+t1$(39)+" ":GOTO 630
620 IF DANA>999 THEN sl$=sl$+t1$(40)+" "
630 IF s=0 THEN 650
640 sl$=sl$+t1$(28+s)+" "
650 IF d=1 THEN 680
660 IF d=0 THEN 690
670 sl$=sl$+t1$(19+d)+" ":GOTO 690
680 sl$=sl$+t1$(11+j)+" ":RETURN
690 IF j=0 THEN RETURN
700 sl$=sl$+t1$(1+j):RETURN
```

Stosując dość proste modyfikacje można zwiększyć górną granicę przedziału liczb dopuszczalnych — co pozostawiam zainteresowanym.

Jeżeli wybrednego użytkownika razi w postaci rozkodowanej brak liter charakterystycznych dla języka polskiego (ś, ć, ą, itp.), to można dołączyć do wersji źródłowej jeden z licznych podprogramów obsługi dodatkowego zestawu tych liter. Wystarczy wtedy zmodyfikować stałe znakowe w pierwszym podprogramie i problem jest rozwiązany.



Krzysztof WALCZAK

PROGRAM ATARI 68

Telegrafista

Programy telegraficzne zwykle realizowane są przez zapisanie kodów znaków alfabetu Morse'a instrukcją DATA, a następnie odczytywanie ich instrukcją READ. Gdy zastosujemy automatyczne nadawanie, pojawia się jednak wada (szczególnie widoczna przy większych tempach) polegająca na nierównych odstępach pomiędzy czytanymi znakami z początku listy DATA, a tymi z jej końca.

Poniższy program, nie mający tej wady, spełnia szereg innych funkcji, które mogą być przydatne zarówno dla krótkofalowców operatorów, jak i dla osób uczących się telegrafii. Dzięki możliwości uzyskania dużych prędkości nadawania program może być pomocny przy nawiązywaniu łączności meteorytowych i zorzowych.

Program realizuje:

— nadawanie bezpośrednio z klawiatury,

— zapis w pamięci tekstu o długości do 899 znaków,

— nagranie tak zredagowanego tekstu na taśmie oraz jego odczyt,

Ponadto ma możliwość

— nadania jednego z 20 tekstów o długości do 120 znaków umieszczonych na stałe w programie,

— wybór prędkości nadawania i jej szybkiej zmiany.

Nadawany automatycznie tekst może być zatrzymany w dowolnym momencie, a następnie odtwarzany od miejsca przerwania (OPTION — przerwanie, SELECT — kontynuacja). Zmiany prędkości nadawania mogą być dokonywane klawiszami START i HELP. Wyjście z trybu nadawania uzyskuje się po naciśnięciu klawisza „*". Po wciśnięciu ESC w trybie nadawania z klawiatury zostanie uruchomione nadawanie tekstu z pamięci.

Komputer można podłączyć do nadajnika za pośrednictwem specjalnego interfejsu dołączonego do wyjścia AUDIO lub do szyny systemu. W tym drugim wypadku przy instrukcjach **AOUND** 0,70,10,15 i **SOUND** 0,0,0,0 należy odpowiednio umieścić instrukcje **POKE ADR,1** i **POKE ADR,0**, gdzie ADR jest adresem interfejsu. Należy pamiętać o sprzężeniu transoptorowym w celu odizolowania komputera od wysokich napięć indukujących się na obudowie nadajnika.

M. DUDA (SP5NOD)

```
5 DIM TX(1000),SK$(120),SKD$(2)
10 OPEN #1,4,0,"K":POKE 710,192:
POKE 82,0:POKE 731,1
20 ? CHR$(125):POSITION 7,8:? "P
ROSZE CHWILE POCZEKAC ":POSITION
 7,10:? "PROGRAM JEST INICJOWANY
":GOSUB 1000
30 GOTO 1710
40 REM **** GENERACJA ZNAKOW ***
50 ? CHR$(LIT);
55 AD=LIT+1495:D=PEEK(AD):AD=158
6+D:DL=PEEK(AD):AD2=AD+1:AD1=AD2
+DL-1
60 FOR I=AD2 TO AD1:TON=PEEK(I):
TON1=TON*FAST:SOUND 0,70,10,15:F
OR J=1 TO TON1
70 NEXT J:SOUND 0,0,0,0:FOR J=1
TO FAST:NEXT J:NEXT I
80 FOR O=0 TO FAST/4:NEXT O
90 ON N GOTO 310,1310,1590,1730
100 REM ***  MENU GLOWNE  ***
110 POKE 755,0:? CHR$(125):POSIT
ION 0,0:? CHR$(17);:FOR LI=1 TO
38:? CHR$(18);:NEXT LI:? CHR$(5)
115 POSITION 0,1:? "|           T
E L E G R A F I A              |"
120 POSITION 0,2:? "|          ---
-----------------------         ||
    by SP5NOD      (C)1987
   |":POSITION 0,4
130 ? CHR$(26);:FOR LI=1 TO 38:?
 CHR$(18);:NEXT LI:? CHR$(3)
135 POSITION 9,7:? "M E N U   G
L O W N E"
140 POSITION 8,8:? "''''''''''''''
''''''''''''''"
145 POSITION 8,10:? CHR$(177);".
Nadawanie":POSITION 8,12:? CHR$(
178);".Wpis tekstu do pamieci"
150 POSITION 8,16:? CHR$(180);".
Ladowanie z kasety"
151 POSITION 8,14:? CHR$(179);".
Zapis na kasecie"
155 POSITION 8,18:? CHR$(181);".
Wyswietlenie zawartosci pam."
156 POSITION 8,20:? CHR$(182);".
Istrukcja"
160 POKE 755,2:POSITION 7,22:? "
 ";:GET #1,M:M=M-48:IF M<1 OR M>
6 THEN,160
170 ON M GOTO 200,1100,1145,1220
,1910,1800
200 REM *** MENU NADAWANIA ***
205 POKE 755,0
210 ? CHR$(125):POSITION 5,5:? "
M E N U   N A D A W A N I A"
215 POSITION 4,6:? "''''''''''''''
''''''''''''''''''":POSITION 4,8
220 ? CHR$(177);".Nadawanie z kl
awiatury"
221 POSITION 4,10:? CHR$(178);".
Nadawanie z pamieci":POSITION 4,
12:? CHR$(179);".Ustalenie predk
osci"
230 POSITION 4,14:? CHR$(180);".
Powrot do MENU GLOWNEGO"
240 POKE 755,2:POSITION 3,16:? "
 ";:GET #1,N:N=N-48:IF N<1 OR N>
4 THEN 240
250 ON N GOTO 300,1270,430,100
290 REM *** OBSLUGA KLAWIATURY *
**
300 ? CHR$(125):GOSUB 420
310 IF PEEK(764)<>255 THEN 360
320 IF PEEK(53279)=6 THEN FAST=F
AST+1:POKE 53279,7:GOSUB 420
330 IF PEEK(732)=17 THEN FAST=FA
ST-1:POKE 732,0:GOSUB 420
340 IF PEEK(53279)=5 THEN X=X-1:
X1=X1-1:GOTO SBR
350 GOTO 310
360 GET #1,LIT:IF LIT=32 THEN ?
" ";:GOTO 310
365 IF LIT=27 THEN N=2:GOTO 1270
370 IF LIT<20 THEN 1350
380 IF LIT=42 THEN 200
390 IF LIT>90 THEN 360
400 IF PEEK(94)=215 AND PEEK(95)
=159 THEN ? CHR$(125):GOSUB 420
410 GOTO 50
420 POSITION 1,0:? "SPEED:";FAST
;"   ":RETURN
430 TRAP 440:? CHR$(125):POSITIO
N 0,4:? " PODAJ PREDKOSC NADAWAN
IA(1-100):";:INPUT FAST:GOTO 200
440 ? :? " BLAD:PODAJ LICZBE !!
":SOUND 0,255,10,15:FOR BL=0 TO
500:NEXT BL:SOUND 0,0,0,0:GOTO 4
30
1000 REM **** KODY ZNAKOW ****
1010 FOR I=1528 TO 1786:READ P:P
OKE I,P:NEXT I:RETURN
1020 DATA 0,0,0,0,0,0,0,0,0,0,0,
0,0,7,14,20,26,32,38,44,50,56,62
,68,74,0,0,0,80,0,86,0,93,96,101
,106
1030 DATA 110,112,117,121,126,12
9,134,138,143,146,149,153,158,16
3,167,171,173,177,182,186,191,19
6
1040 DATA 6,3,3,1,1,3,3,6,1,3,1,
3,1,3,5,3,1,1,3,1,5,3,3,3,3,3,5,
1,3,3,3,3,5,1,1,3,3,3,5,1,1,1,3,
3,5,1,1,1,1,3
1050 DATA 5,1,1,1,1,1,5,3,1,1,1,
1,5,3,3,1,1,1,5,3,3,3,1,1,5,3,3,
3,3,1,5,3,1,1,1,3,5,1,1,3,3,1,1
1060 DATA 2,1,3,4,3,1,1,1,4,3,1,
3,1,3,3,1,1,1,1,4,1,1,3,1,3,3,3,
1,4,1,1,1,1,2,1,1,4,1,3,3,3,3,3,
1,3,4,1,3,1,1
1070 DATA 2,3,3,2,3,1,3,3,3,3,4,
1,3,3,1,4,3,3,1,3,3,1,3,1,3,1,1,
1,1,3,3,1,1,3,4,1,1,1,3,3,1,3,3,
4,3,1,1,3
1080 DATA 4,3,1,3,3,4,3,3,1,1
1090 REM *** WPISYWANIE TEKSTU *
**
1100 ? CHR$(125):POSITION 1,0:?
" MOZESZ PISAC TEKST :":X=0
1110 GET #1,TX:? CHR$(TX);:TX(X)
=TX:IF TX=155 THEN 100
1120 X=X+1:IF X>899 THEN ? "PRZY
KRO MI:KOMIEC PAMIECI DLA TEKSTU
":TX(900)=155:GET #1,BYK GOTO 10
0
1130 GOTO 1110
1140 REM *** ZAPIS NA KASECIE **
*
1145 ? CHR$(125):POSITION 1,0:?
" WCISNIJ 2 RAZY 'RETURN' "
1146 GET #1,LOA:? CHR$(125):POSI
TION 1,0:? " TEKST JEST NAGRYWAN
Y "
1150 IF TX(0)=0 OR TX(1)=0 THEN
POSITION 2,5:? "Zbior pusty ; wc
isnij '*'.":GET #1,BYK GOTO 100
1160 X=0:OPEN #2,8,0,"C"
1170 TX=TX(X):PUT #2,TX:IF TX=15
5 THEN GOTO 1190
```

```
1180 X=X+1:GOTO 1170
1190 CLOSE #2:GOTO 100
1210 REM *** ODCZYT Z KASETY ***
1220 X=0:? CHR$(125):POSITION 1,
0:? " WCISNIJ 'RETURN' "
1221 OPEN #2,4,0,"C":? CHR$(125)
:POSITION 1,0:? " CZYTANY TEKST
:":TRAP 1255
1230 GET #2,TX:? CHR$(TX);:TX(X)
=TX:IF TX=155 THEN 1250
1240 X=X+1:GOTO 1230
1250 CLOSE #2:?:? " WCISNIJ DOW
OLNY KLAWISZ";:GET #1,PDM:GOTO 1
00
1255 CLOSE #2:?:? " BLAD LADOWA
NIA.SPROBOJ JESZCZE RAZ. ":? " N
ACISNIJ DOWOLNY KLAWISZ";:GET #1
,BYK:GOTO 1220
1260 REM ** CZYTANIE ZAW. PAMIEC
I ***
1270 X=0:SBR=1310:? CHR$(125):GO
SUB 420
1280 LIT=TX(X):IF LIT=155 THEN S
BR=310:N=1:GOTO 310
1290 IF PEEK(53279)=3 THEN N=1:G
OTO 310
1295 IF LIT=32 THEN GOTO 1330
1300 GOTO 50
1310 X=X+1:N=2:GOTO 1280
1330 FOR I=1 TO FAST:NEXT I:? "
";:GOTO 1310
1340 REM *** 20*TEKST ***
1350 ON LIT GOTO 1360,1370,1380,
1390,1400,1410,1420,1430,1440,14
50,1460,1470,1480,1490,1500,1510
,1520,1530,1540
1360 SK$="CQ CQ CQ DE SP5NOD SP5
NOD SP5NOD CQ CQ CQ DE SP5NOD SP
5NOD SP5NOD PSE K":GOTO 1550
1370 SK$="QRZ ?":GOTO 1550
1380 SK$="MY NAME NAME NAME IS I
S IS MARK MARK MARK ES MY QTH QT
H QTH IS IS IS PLOCK PLOCK PLOCK
 =":GOTO 1550
1390 SK$="MY RIG IS TX ABT 50 WT
TS = ANT IS DIPOLE G5RV = COMPUT
ER ATARI 800 XL =":GOTO 1550
1400 SK$="UR RST RST RST I8":GOT
O 1550
1410 SK$="NW VY TNX FOR OUR NICE
 QSO = VB 73 73 73 ES MN1 BEST D
XS = GB GL ES HPE CUAGN =":GOTO
1550
1420 SK$=" ":GOTO 1550
1430 SK$=" ":GOTO 1550
1440 SK$=" ":GOTO 1550
1450 SK$=" ":GOTO 1550
1460 SK$=" ":GOTO 1550
1470 SK$=" ":GOTO 1550
1480 SK$=" ":GOTO 1550
1490 SK$=" ":GOTO 1550
1500 SK$=" ":GOTO 1550
1510 SK$=" ":GOTO 1550
1520 SK$=" ":GOTO 1550
1530 SK$=" ":GOTO 1550
1540 SK$="CQ CQ CQ TEST TEST TES
T DE SP5NOD SP5NOD SP5NOD TEST T
EST TEST DE SP5NOD SP5NOD SP5NOD
 KK":GOTO 1550
1550 SBR=1590:N=3:X1=1:IF LEN(SK
$)<2 THEN M=M1:SBR=310:GOTO 310
1560 SKD$=SK$(X1,X1):LIT=ASC(SKD
$):IF SKD$=" " THEN ? " ";:GOTO
1610
1570 IF PEEK(53279)=3 THEN N=1:G
OTO 310
1580 GOTO 50
1590 IF X1=LEN(SK$) THEN N=1:SBR
=310:GOTO 310
1600 GOTO 1620
1610 FOR U=0 TO FAST:NEXT U
1620 X1=X1+1:N=3:GOTO 1560
1700 REM *** CZOLOWKA ***
1710 SK$="TELEGRAFIA":GRAPHICS 2
+16:FAST=30:N=4:POSITION 4,5:X=1
1720 SKD$=SK$(X,X):LIT=ASC(SKD$)
:? #6;CHR$(LIT);:GOTO 55
1730 IF X=10 THEN 1750
1740 X=X+1:GOTO 1720
1750 IF PEEK(53279)=6 THEN X=0:G
RAPHICS 0:POKE 710,192:GOTO 100
1760 GOTO 1750
1800 REM *** INSTRUKCJE ***
1810 ? CHR$(125):POSITION 12,1:?
 " OBJASNIENIA :"
1811 POSITION 2,4:? "Klawisze z
[CONTROL] uruchamiaja 20 w-pisan
ych na";
1820 ? " dole programu tekstow (
nume-ry ich linii:1360";
1821 ? "-1540). Mozna je dowol-n
ie zmieniac, lecz ich dlugosc  n
ie moze";
1830 ? "przekroczyc 120 znakow (
ze spacjami).   Czytanie pamieci
 mozna zatrzymac  w kaz-dej chwi
li poprzez ";
1840 ? "przytrzymanie w odpo-wie
dnim momencie klawisza";
1841 ? " [OPTION],a na-stepnie m
ozna je wznowic od  momentu za-t
rzyma";
1850 ? "nia.Predkosc nadawania m
ozna zwie-kszac";
1851 ? " (tylko w opcji nadawani
a z klawia-tury)  za  pomoca  kl
awisza  [HELP]";
1860 ? " oraz zmniejszac za pomo
ca klawisza [START]."
1870 POSITION 10,20:? "MILEJ ZAB
AWY !!!":POSITION 6,22:? " WCISN
IJ DOWOLNY KLAWISZ";:GET #1,INS:
GOTO 100
1900 REM *** WYSW.ZAW.PAM. ***
1910 ? CHR$(125):POSITION 1,0:?
" ZAWARTOSC PAMIECI :":X=0
1920 ? CHR$(TX(X));:IF TX(X)=155
 THEN ? :? " WCISNIJ DOWOLNY KLA
WISZ";:GET #1,CON:GOTO 100
1925 IF PEEK(53279)=3 THEN ? :?
" WCISNIJ DOWOLNY KLAWISZ";:GET
#1,CON:GOTO 100
1930 X=X+1:GOTO 1920
```

*** SUMA KONTROLNA / ETYKIETA ***

TI	5	YY	10	XC	20	KZ	30
KU	40	RE	50	US	55	ZX	60
VA	70	RB	80	ZV	90	HM	100
MJ	110	TG	115	XR	120	SZ	130
IP	135	HZ	140	DC	145	FQ	150
AI	151	QJ	155	BG	156	BP	160
HG	170	NW	200	KR	205	IK	210
JP	215	QZ	220	JY	221	UG	230
CY	240	LB	250	BS	290	OI	300
WH	310	EU	320	WM	330	TM	340
MX	350	BD	360	FS	365	JP	370
PR	380	XC	390	CO	400	QV	410
VK	420	CH	430	KX	440	WD	1000
IC	1010	ZG	1020	ZA	1030	EN	1040
FY	1050	FT	1060	VU	1070	GD	1080
AX	1090	YW	1100	CC	1110	SJ	1120
NY	1130	OL	1140	ZT	1145	SS	1146
GU	1150	TW	1160	KU	1170	FP	1180
UW	1190	RX	1210	CO	1220	KP	1221
OV	1230	CZ	1240	AB	1250	NK	1255
JI	1260	SC	1270	WM	1280	LE	1290
NX	1295	QQ	1300	JQ	1310	WV	1330
DZ	1340	FC	1350	CJ	1360	NM	1370
VQ	1380	VI	1390	WC	1400	XB	1410
RJ	1420	RM	1430	RP	1440	RS	1450
RV	1460	RY	1470	SB	1480	SE	1490
RF	1500	RI	1510	RL	1520	RO	1530
IN	1540	EQ	1550	CX	1560	LE	1570
RS	1580	[illegible]	1590	QU	1600	QI	1610
RV	1620	WY	1700	JO	1710	NL	1720
BV	1730	GC	1740	CW	1750	TN	1760
JB	1800	BF	1810	TM	1811	HC	1820
RY	1821	EF	1830	NT	1840	OE	1841
CE	1850	VG	1851	BC	1860	AZ	1870
OM	1900	HM	1910	XU	1920	JS	1925
HN	1930						

Tajemnicze w swej istocie przypadki, które przez kilka tygodni trzymały w napięciu miasto Artemidos, zarazem intrygując mieszkańców całego kontynentu, dziś miały zostać wyjaśnione. Nic więc dziwnego, że w kierunku kopulastej, wykładanej biało-zielonymi płytami rotundy policji miejskiej, gdzie miała się odbyć konferencja zmierzało wiele pojazdów. Raz po raz na lotnisku lądowały małe awionetki i wirowce.

Sala konferencyjna szybko się zapełniała. Pięć minut przed rozpoczęciem spotkania z rzecznikiem policji i przewodniczącym Rady Obywatelskiej pięćset miejsc przed pulpitami fotoramów było zajętych. Na stanowiskach kwadrotelewizji czuwali operatorzy sześciu stacji kontynentalnych. Niektórzy zdążyli zaprogramować zapis przekazu, teraz spokojnie popijali dostane przez urządzenia podawcze soki i inne zamówione napoje. Zapóźnieni szybko programowali fotoramy wciskając klawisz. Większość miała czas rozglądać się po sali, wymienić uwagi z sąsiadem. Podniecenie powoli zaczęło narastać.

Trzy minuty przed rozpoczęciem spotkania zapaliła się kontrolna lampka na ekranach fotoramów i miły kobiecy głos zapowiedział rozpoczęcie konferencji.

W centralnym miejscu sali otworzyła się podłoga i na powierzchnię wyjechał pulpit sterowniczy, stół prezydialny i fotele. Bezszelestnie otworzyły się drzwi i do sali weszło trzech mężczyzn. Zajęli miejsca za stołem, za ich plecami odsłonił się centralny ekran fotoramu.

Ubrany w beżowy garnitur i takąż koszulę siwowłosy mężczyzna Calla Wertena, przewodniczący Rady Obywatelskiej, wcisnął guzik mikrofonu i zaczął konferencję.

— Szanowni państwo! — zaczął — wydarzenia, jakie rozegrały się w naszym mieście, nie są bliżej znane naszej społeczności, dlatego pojawiły się domniemania i plotki. Padło i takie stwierdzenie, iż staliśmy się przedmiotem ataku innej cywilizacji. Zanim udało się nam wyświetlić całą sprawę, upłynęło trochę czasu.

Myślę, powiedział, że najlepiej sprawę wyjaśni — rzecznik policji, Ed Cartt, mówiąc to wskazał ręką na ubranego w mundur policji miejskiej wysokiego mężczyznę o suchym obliczu i przesłoniętych ciemnymi okularami oczach. Ten lekko skłonił głowę.

— Aby, jednak w pełni sprawa była znana, poprosiliśmy też pana profesora Legera, specjalistę od metapsychologii i psychiki.

— Pierwszy ma głos pan Cartt. Ten zdjął okulary i spojrzał na salę. Zaczął: — Wydaje się, że wszyscy państwo wiecie, że chodzi o całą serię wydarzeń

związanych ze znikaniem przedmiotów i rzeczy. Z myślą o porządkowaniu przebiegu wydarzeń przypomnę podstawowe fakty.

25 marca z naszego miejskiego muzeum techniki ginie bez śladu lawator. Nieobeznanym z historią techniki przypomnę, że był to pojazd wielokrotnego wykorzystania, mogący poruszać się po wodzie, lądzie, i w powietrzu. Mógł na krótko zagłębić się do 200 metrów pod wodą i wznosić się do 20 nad

GENY

ziemię. Modele wykorzystywane były na początku naszego wieku do badania Galaktyki Oriona. Jego wygląd, powiedział wciskając przycisk na pulpicie, zobaczycie państwo na ekranie. Godzi się przypomnieć, że legendarny badacz, Dean Deder spenetrował za jego pomocą planetę Olfar.

Widząc obraz wyświetlany na ekranach kontrolnych swoich pulpitów wielu z uczestników konferencji nacisnęło przyciski robiąc powiększenie, aby mieć wyraźniejszą dokumentację.

W tym czasie rzecznik policji mówił dalej. — Jak wiadomo konwencja 25 z Warszawy zobowiązywała Rady Obywatelskie do likwidacji silników atomowych na całym naszym globie. Ponieważ lawator był nimi napędzany, a nowy silnik parafotonowy niezbyt pasował i zwiększał koszty, zapadła decyzja zniszczenia wszystkich pojazdów, poza tymi, które po uprzednim zdemontowaniu silnika przeznaczone były do muzeów oczywiście. Ten egzemplarz, który zaginął, należał, jak już powiedziałem, do legendarnego już za życia Deana Dedera. Muszę państwu zwrócić uwagę, że był pewnym szczeblem do dzisiejszych pojazdów Alfa. Sama idea zachowania w muzeum lawatora sprowadzała się do tego, aby nasza młodzież mogła w żywym oglądzie poznawać starą technikę.

Zniknięcie pojazdu nikogo by nie zaniepokoiło, gdyby... Były wypadki, iż ten czy ów miłośnik techniki wypożyczał eksponaty, aby odtworzyć ten czy inny pojazd, zawsze jednak po kilku dniach pojazd wracał. Tym razem było inaczej. Zmuszeni byliśmy do ogłoszenia zdarzenia w środkach masowej informacji i prosić o zwrot eksponatu, ale to nie nastąpiło.

To było pierwsze wydarzenie.

Gdzieś około dwóch tygodni później, z Kompleksu Budowy Silników do Alfa — zaginął eksperymentalny najnowszy model silnika Carbo Ax.

Dzień później z Kompleksu Paliw giną baterie zalitowe przeznaczone do małego promu kosmicznego. Rex—1. Ich czas pracy jest praktycznie nieograniczony.

Do czwartego wypadku doszło w Agrokompleksie—1, produkującym żywność dla załóg promów. Tym razem znika i to bez śladu 2-letni zestaw posiłków dla załóg kosmolotów.

Pojawiło się równocześnie kilkanaście rozbieżnych hipotez. Zresztą zaczęliśmy je sprawdzać po kolei, jedna po drugiej. Być może trwało to długo i w końcu byśmy doszli do istoty, ale zdecydował przypadek i dzięki niemu sprawa została przyspieszona. — Właśnie w Agrokompleksie zarządzona została przerwa rekreacyjna, wszystkie urządzenia i roboty miały być sprawdzone i przejść okres odpoczynku. Inżynier Waren, prowadzący sprawdzanie robotów typu Goliat, przeznaczonych do najcięższych prac, znalazł u jednego z nich wmontowany dodatkowy obwód, który uczynił robota posłusznym innemu nieznanemu decydentowi. Szef policji polecił wtedy dokładną kontrolę wszystkich robotów tego typu. Było ich więcej — jeszcze dwa podobne modele mające wmontowane układy nadsterownicze odkryto tam, gdzie wydarzyły się wcześniejsze wypadki. Układy bardzo prymitywne, ale skuteczne. Nasi znawcy ocenili je. Urządzenia wykonał ich zdaniem człowiek niezbyt obeznany z techniką.

Nie będę państwa zanudzał szczegółami dalszych naszych poszukiwań. W końcu znaleźliśmy sprawcę. Okazał się nim dwunastoletni syn naczelnego konstruktora Kompleksu Rozwiniętej Techniki — Rexa Dedera — Adaś. On to wmontował układy robotom. I one na jego życzenie dostarczyły mu tych przedmiotów.

Jak to się stało, że urządzenia, w czasie przerw regeneracyjnych, pozbawione dopływu energii na okres dwugodzinny, zdołały wykonać polecenia chłopca, pozostanie na razie tajemnicą.

On sam zapytany o przyczyny podjęcia takiego działania, stwierdził, że oglądając stare filmy zapragnął być nowym Gregiem Berry — legendarnym zdobywcą planety Arfa 4 w galaktyce Fixa.

Z tą myślą rozpoczął przygotowanie własnego pojazdu, aby wyruszyć na nieznaną planetę.

Teraz głos oddałbym profesorowi Legerowi, który postara się wyświetlić samo zjawisko.

Sam problem, który chcę państwu przybliżyć, faktycznie nie jest znany naszej cywilizacji, nie oznacza bowiem pojęcia kradzieży, albo inaczej zaboru mienia społeczności. Jest to pojęcie historyczne wykreślone z języka przed wielu, wielu laty.

Wyobraźnię chłopca opanował fikcyjny bohater, który doznawał wielu przygód i zawsze zwalczał przeciwności, zabijał mityczne potwory, uwalniał przyjaciół z różnego potrzasków losu.

Tu dopatrujemy się poważnego błędu wychowawczego. Jeżeli ojciec chłopca interesował się starymi filmami, to do ich wyświetlania nie powinien dopuszczać chłopca. Zgodnie bowiem z naszym założeniem wychowawczym stare filmy mogą oglądać ludzie dorośli.

Po trzecie właśnie, wykryliśmy w genach chłopca zakodowaną chęć przygód i zdobywania. Sądzę osobiście, że wypływa to z dziedziczenia. Jest on bowiem spokrewniony w linii prostej z Deanem Dederem. Z badań wynika, że jest dzieckiem w niczym nie odbiegającym od normy, może tylko niekiedy przejawia zbyt dużo inicjatywy. Teraz przebywa na specjalistycznym leczeniu, aby zatrzeć w nim cechy nieprzydatne w naszym życiu społecznym.

Dyskusja ciągnęła się jeszcze dłuższy czas. Na koniec głos zabrał Cali Werten.

— Nasze prawo nie przewiduje odpowiedzialności za chłopięce marzenia. Toteż po zwróceniu uwagi rodzicom, sprawa, jeżeli można to nazwać sprawą, została zamknięta.

Martin CARR



Krzysztof WALCZAK

GIEŁDA POMYSŁÓW

Gra zręcznościowa
B A R M A N

Mikrokomputer Meritum ma niewielkie oprogramowanie. Jego słaba grafika, brak kolorów i niewielkie możliwości generowania dźwięku zniechęcają do tworzenia programów, a szczególnie gier.

Być może program „Barman" pozwoli użytkownikom tego mikrokomputera nieco się zabawić. Jednakże dobra zabawa w robota podającego piwo okupiona jest czasem potrzebnym na wpisanie dosyć długiego programu.

Program najlepiej jest wpisywać częściami, dopisując poszczególne linie. Należy pamiętać o utrwalaniu na taśmie magnetofonowej stopniowo rozbudowującego się programu. Ze względu na możliwość przypadkowego przekasowania pamięci. Linie 4 i 6 blokują przerwanie BREAK i NMI, dlatego powinny być wpisane dopiero po uruchomieniu programu.

Piotr BIEDRZYCKI

```
3     CLEAR 200  :GOSUB 5000
4     POKE 16396,175: POKE 16397,201
5     DIM R(5), R$(5): A=25350
6     OUT 250,0  'BLOKADA NMI
9    '------------------------------
10    GOSUB  1000
15    POKE 16526,115: POKE 16527,102
17    U=USR(0)  'SKASOWANIE EKRANU
18    GOSUB 2005
19    POKE 16526,159: POKE 16527,101
20    U=USR(0)  'URUCHOMIENIE PROGRAMU
25    NW=PEEK(A)+256*PEEK(A+1)
30    GOSUB 600  : GOTO   10
35   '------------------------------
599  'WYSWIETLEMIE WYNIKU
600   PRINT@ 525,    "------------------------";
605   PRINT@ 589,USING"       ZAROBILES   ####$  ";NW;
607   PRINT@ 653,    "------------------------";
610   FOR F=0 TO 900: NEXT F
619    'SPRAWDZANIE WYNIKU
620      FOR H=1 TO 5
625      IF NW > R(H) THEN 655
630      NEXT H
635    RETURN
654    'WYNIK JEDEN Z PIECIU NAJLEPSZYCH
655      FOR J=5 TO H STEP -1
660      R(J)=R(J-1): R$(J)=R$(J-1)
665      NEXT J
668      CLS: PRINT CHR$(23)
670      PRINT@ 512,"   TWOJE IMIE";
672      INPUT I$
673      I$=I$+"
675      R(H)=NW      : R$(H)=LEFT$(I$,6)
680    RETURN
999   'PIERWSZY EKRAN
1000    CLS: PRINT CHR$(23)
1005    PRINT: PRINT"            BARMAN   "
1010    PRINT
1015    PRINT"          'W'-W GORE"
1018    PRINT"     PODAJ KUFEL-'ENTER'"
1020    PRINT"          'S'-W DOL"
1025    PRINT: PRINT"   LISTA NAJLEPSZYCH WYPLAT"
1026    PRINT: FOR H=1 TO 5
1030    PRINT "     ";H;" - ";R$(H);USING"####$";R(H)
1035    NEXT H
1036    A$=INKEY$ 'KASOWANIE KODU OSTATNIEGO KL.
1037    IF INKEY$="" THEN 1037  ELSE CLS
1040   RETURN
1999  'PLANSZA GRY
2005   FOR H=0 TO 99
2010   SET (H,10): SET (H,28): SET (H,46)
2025   NEXT H
2030   FOR L=0 TO 47
2035   SET (127,L): SET (120,L)
2040   NEXT L
2042   PRINT@ 189,CHR$(191);CHR$(154);
2045   PRINT@ 573,CHR$(191);CHR$(154);
2047   PRINT@ 957,CHR$(191);CHR$(154);
2050   RETURN
2888 'KOD WEWNETRZNY ZAJMUJE OBSZAR
2889 'OD 6400H (25600)   DO 696FH (26991)
3999     'PODPROGRAM LADUJACY
4000     FOR Z=A  TO  A+P*16-1
4005     READ A$:PRINTA$;"  ";
4010     N=ASC( LEFT$(A$,1))-48
4015     M=ASC(RIGHT$(A$,1))-48
4020     N=N-INT(N/12)*7
4025     M=M-INT(M/12)*7
4030     B=N*16+M: S=S+B
```

ZX-81
Trafianie kwadratów

```
1 REM ***************************
2 REM *  TRAFIANIE KWADRATOW    *
3 REM *          ZX-81          *
4 REM ***************************
10 LET S=0
15 CLS
20 PRINT AT 18,0; "-------------
-------------------"
25 FOR F=1 TO 30
30 PRINT AT RND*15+2,RND*18; "■"
35 NEXT F
40 LET L=10
45 FOR F=1 TO 17
50 LET L=L+(INKEY$="8")-(INKEY$=
"5" AND L>0)
55 PRINT AT F,L;
60 LET P=PEEK(256*PEEK 16399+PEE
K 16398)
65 PRINT "Y"
70 IF P=180 THEN GOTO 115
75 IF P=128 THEN GOTO 95
80 PRINT AT F,L; " "
85 NEXT F
90 GOTO 115
95 PRINT AT F,L; "*"
100 LET S=S+1
105 IF S=20 OR S=40 OR S=60 OR S
=80 THEN GOTO 15
110 GOTO 45
115 PRINT AT 19,0; "PUNKTY:";S
```

```
4035        POKE Z,B : NEXT Z
4040       RETURN
4999 'LADOWANIE KODU MASZYNOWEGO
5000 A=25600 :P=16 :S=0 :GOSUB4000 :
     IF S()31417 THEN PRINT"BLAD BLOK 1":END
5001 DATA 80,80,B6,B3,BC,B3,B9,80,80,80,80,B2,B3,BF,B3,B1
5002 DATA 80,80,BB,BE,B1,8B,9F,BF,8B,BD,B0,8C,8C,8C,8C,BC
5003 DATA BC,8C,8C,8C,80,80,B8,BF,80,BF,B4,90,80,00,00,00
5004 DATA 80,80,A0,B9,B6,90,80,80,80,B0,8B,AE,9D,87,B0,80
5005 DATA 8A,95,BE,BF,BF,BD,AA,85,8C,BC,8C,8C,8C,BC,BC,8C
5006 DATA 80,80,BF,9F,AF,BF,80,80,80,80,BB,91,A2,B7,80,80
5007 DATA A4,A8,BC,B4,90,82,AF,24,9D,85,00,00,00,00,00,50
5008 DATA 21,00,72,5E,23,56,7A,A2,C8,2B,EB,2B,7D,FE,00,CA
5009 DATA 04,69,FE,80,CA,04,69,D3,FE,36,9A,2B,36,BF,23,EB
5010 DATA 73,23,72,23,18,DD,21,00,72,5E,23,56,23,7A,A2,CB
5011 DATA EB,36,80,2B,36,80,EB,18,F0,21,00,75,5E,23,56,7A
5012 DATA A2,C8,2B,EB,23,7D,FE,33,28,11,FE,B3,28,0D,36,BA
5013 DATA 2B,36,A5,23,EB,73,23,72,23,18,E1,D5,D5,D5,E5,11
5014 DATA 80,00,2A,00,63,19,E5,D1,E1,CD,39,0A,A7,C2,04,69
5015 DATA D1,CD,B9,68,E1,23,23,01,00,01,ED,B0,2A,06,63,23
5016 DATA 22,06,63,E1,18,B6,21,00,75,5E,23,56,23,7A,A2,C8
5050 PRINT :A=Z :S=0 :GOSUB 4000 :
     IF S()22940 THEN PRINT"BLAD BLOK 2":END
5051 DATA EB,36,80,2B,36,80,EB,18,F0,2A,00,63,A7,CB,FE,08
5052 DATA 20,0E,11,40,00,ED,52,06,08,36,80,23,10,FB,18,11
5053 DATA 00,11,80,01,19,11,00,40,CD,39,0A,FE,FF,20,02,18
5054 DATA E6,2A,00,63,11,30,64,0E,05,E5,06,08,1A,77,13,23
5055 DATA 10,FA,E1,D5,11,40,00,19,D1,CB,74,28,05,CD,63,65
5056 DATA 18,11,0D,20,E4,CD,63,65,CD,5B,65,06,04,1A,77,23
5057 DATA 13,10,FA,3A,10,63,FE,80,20,21,D5,E5,2A,00,63,11
5058 DATA C0,00,19,36,8C,06,06,23,36,BF,23,10,FB,36,8C,11
5059 DATA 83,00,ED,52,36,9F,2B,36,AF,E1,D1,0E,0A,D3,FE,3C
5060 DATA 06,0A,10,FE,D3,FE,06,0D,10,FE,3C,0D,20,EF,C9,E5
5061 DATA CD,A5,65,E1,C9,21,33,3C,22,00,63,CD,27,69,22,02
5062 DATA 63,22,00,70,22,A0,70,22,00,72,22,00,75,22,06,63
5063 DATA 22,0C,63,3E,02,32,AF,40,32,0E,63,ED,73,04,63,CD
5064 DATA 31,65,18,23,CD,20,69,06,02,C5,3A,10,63,CD,3D,66
5065 DATA CD,09,65,3A,10,63,A7,28,0A,06,09,C5,06,00,10,FE
5066 DATA C1,10,F8,C1,10,E3,C9,00,3A,02,63,FE,09,FC,B3,67
5100 PRINT :A=Z :S=0 :GOSUB 4000 :
     IF S()27142 THEN PRINT"BLAD BLOK 3":END
5101 DATA CD,51,68,CD,96,64,CD,F6,64,CD,31,69,CD,83,66,CD
5102 DATA A9,64,CD,70,64,CD,FB,67,CD,D4,65,06,92,C5,10,FE
5103 DATA C1,10,FA,CD,11,68,CD,F6,64,CD,96,64,CD,A9,64,CD
5104 DATA 70,64,CD,FB,67,CD,77,68,18,BD,3A,04,38,11,40,00
5105 DATA 2A,00,63,FE,80,20,10,A7,ED,52,7C,FE,3B,C8,22,00
5106 DATA 63,3E,80,32,10,63,C9,FE,08,20,13,19,11,40,3F,CD
5107 DATA 39,0A,FE,FF,C0,22,00,63,3E,08,32,10,63,C9,AF,32
5108 DATA 10,63,C9,E5,01,FF,03,11,01,3C,21,00,3C,36,80,ED
5109 DATA B0,E1,C9,01,00,70,0A,A7,C8,CD,E5,67,03,0A,3C,FE
5110 DATA 32,CA,04,69,02,C5,85,6F,11,80,00,A7,ED,52,CD,02
5111 DATA 67,11,80,00,19,7D,FE,7F,28,06,FE,FF,28,02,36,80
5112 DATA C1,03,18,D2,01,A0,70,0A,A7,C8,CD,E5,67,03,0A,3D
5113 DATA FE,FF,28,24,02,B5,6F,E5,C5,23,36,80,A7,11,81,00
5114 DATA ED,52,CD,02,67,C1,E1,00,00,00,00,00,00,00,00,00
5115 DATA 00,00,00,00,00,03,18,CF,36,80,0B,C5,C5,C5,D1,E1
5116 DATA 23,23,01,40,00,ED,B0,C1,0B,3A,02,63,3C,32,02,63
5150 PRINT :A=Z :S=0 :GOSUB 4000 :
     IF S()26748 THEN PRINT"BLAD BLOK 4":END
5151 DATA 18,E3,11,08,64,06,09,C5,D5,E5,06,05,C5,01,40,00
5152 DATA 1A,77,09,7B,C6,09,5F,CB,74,20,14,C1,10,EE,E1,D1
5153 DATA 2B,1B,7D,FE,7F,28,0B,FE,FF,28,07,C1,10,D9,C9,C1
5154 DATA 18,EC,C1,C9,E5,0A,FE,06,FA,53,67,7E,FE,BF,28,06
5155 DATA 2B,7E,FE,BF,20,0D,23,CD,55,67,CD,80,67,CD,70,67
5156 DATA CD,E7,68,E1,C9,E5,C5,C5,01,40,00,11,C0,70,21,BE
5157 DATA 70,ED,B8,C1,0A,32,A1,70,0B,0A,32,A0,70,C1,E1,C9
5158 DATA 0B,C5,C5,C5,D1,E1,23,23,01,40,00,ED,B0,C1,0B,C9
5159 DATA C5,0E,7E,11,00,72,ED,53,0A,63,E5,E5,2A,0A,63,5E
5160 DATA 23,56,23,22,0A,63,E1,CD,39,0A,A7,28,06,E1,0D,20
5161 DATA E9,C1,C9,01,00,01,2A,0A,63,E5,D1,1B,1B,ED,B0,0E
5162 DATA 01,18,EA,21,10,00,CD,9A,0A,CD,C9,14,CD,37,0B,3A
5163 DATA 21,41,FE,04,F0,32,03,63,CD,E5,67,7E,FE,80,C0,01
5164 DATA 40,00,11,40,70,21,3E,70,ED,B8,3A,03,63,32,00,70
5165 DATA AF,32,01,70,C9,FE,01,20,05,21,80,3C,18,0C,FE,02
5166 DATA 20,05,21,00,3E,18,03,21,80,3F,C9,00,01,00,70,0A
5200 PRINT :A=Z :S=0 :GOSUB 4000 :
     IF S()25902 THEN PRINT"BLAD BLOK 5":END
5201 DATA A7,C8,CD,E5,67,03,0A,85,D6,04,6F,CD,34,67,03,18
5202 DATA EE,CD,E3,03,FE,0D,C0,2A,00,63,11,7E,00,19,11,B1
5203 DATA 3C,CD,39,0A,A7,28,13,11,31,3E,CD,39,0A,A7,28,0A
5204 DATA 11,B1,3F,CD,39,0A,A7,28,01,C9,E5,CD,D0,68,01,00
5205 DATA 01,11,00,73,21,FE,72,ED,B8,D1,21,00,72,73,23,72
5206 DATA C9,00,3A,B4,67,FE,09,F8,2A,06,63,ED,5B,0C,63,ED
5207 DATA 52,11,06,00,CD,39,0A,A7,FE,01,C0,2A,06,63,22,0C
5208 DATA 63,21,B4,67,35,35,C9,00,3A,0E,63,3D,32,0E,63,A7
5209 DATA C0,3E,08,32,0E,63,3A,02,63,A7,C8,21,08,00,CD,9A
5210 DATA 0A,CD,C9,14,CD,37,0B,3A,21,41,FE,04,F0,CD,E5,67
5211 DATA 23,E5,01,00,01,11,00,76,21,FE,75,ED,B8,D1,21,00
5212 DATA 75,73,23,72,21,02,63,35,C9,C5,0E,23,3E,78,D3,FE
5213 DATA 47,10,FE,3D,3D,3D,D3,FE,47,10,FE,0D,20,F0,C1,C9
5214 DATA C5,0E,23,3E,0C,D3,FE,47,10,FE,3C,3C,3C,D3,FE,47
5215 DATA 10,FE,0D,20,F0,C1,C9,C5,3A,02,63,07,07,4F,3E,60
5216 DATA 91,0E,14,47,D3,FE,10,FE,3D,47,D3,FE,47,10,FE,0D
5250 PRINT :A=Z :S=0 :P=7 :GOSUB 4000 :
     IF S()14326 THEN PRINT"BLAD BLOK 6":END
5251 DATA 20,F1,C1,C9,ED,7B,04,63,21,00,00,0E,FE,3E,32,ED
5252 DATA B3,ED,B3,ED,B3,ED,B3,3D,20,F5,C9,54,52,55,44,B0
5253 DATA CD,3A,66,CD,09,65,C9,21,18,00,22,B4,67,21,00,00
5254 DATA C9,21,49,69,22,03,67,CD,B4,66,21,08,64,22,03,67
5255 DATA C9,80,80,B6,B3,BC,B3,B9,80,80,80,80,B2,B3,BF,B3
5256 DATA B1,80,80,A0,BF,9F,B5,BF,BA,BF,BF,B4,8C,8C,8C,8C
5257 DATA 8C,8C,8C,8C,8C,80,A0,B8,BF,80,BF,B4,80,80,50,2F
5999 '----------- KONIEC DANYCH ------------------
6000 RETURN  'POWROT Z PODPROGRAMU LADUJACEGO KODY -----
```

ZX-81 GOLF

```
 2 REM *         GOLF          *
 3 REM *         ZX-81         *
 4 REM *********************
 5 LET C=NOT PI
 6 Z=C
 10 LET E=INT(RND*120+200)
 15 LET Z=Z+1
 20 PRINT "ODLEGLOSC DO L. ";Z;
":";E
 23 LET C=C+1
 25 IF E<10 THEN GOTO 150
 30 FOR I=1 TO E/10
 35 PRINT AT 12,I;"-"
 40 NEXT I
 45 PRINT AT 4,0;"WYBIERAJ (1-20
)?"
 55 IF S<1 OR S>20 THEN GOTO 50
 60 CLS
 65 LET E=E-INT(RND*S*10+S)
 70 LET E=ABS E
 IF E>1 THEN GOTO 20
 80 PRINT "TRAFILES DO ";Z;" DZ
IURKI"
 90 IF Z<5 THEN GOTO 10
 95 CLS
 100 PRINT AT 10,0;"MASZ ";C;" U
DERZEN PILECZKI"
 110 STOP
 150 PRINT AT 7,NOT PI;"█"
 155 PRINT "█","█","█","█"
,"█","█","█","█"
 160 PRINT AT 14,E+1; 0
 165 GOTO 45
```

Programy narzędziowe na CPC 6128

(cz. II)

Program DYSKMAPA sporządza mapę dyskietki. W wyniku uruchomienia programu dowiemy się, jakie ścieżki i jakie sektory zajmują programy (zbiory) zapisane na dyskietce. Są to niezbędne informacje przy dokonywaniu jakichkolwiek zmian na poziomie pojedynczego sektora.

OPIS DZIAŁANIA PROGRAMU

Po uruchomieniu programu komendą RUN następuje zaczytanie wszystkich ścieżek i sektorów z dyskietki. Następnie, wciskając klawisz ENTER, wyświetlimy na ekranie monitora nazwy wszystkich programów (zbiorów), znajdujących się na dyskietce wraz z opisem, który sektor, na której ścieżce zajmują. Ścieżki opisują liczby od 0 do 39, a wielkości od C1 do C9 oznaczają sektory (gdyż na każdej ścieżce znajduje się 9 sektorów po 512 bajtów każdy).

Wyświetlanie mapy dyskietki możemy w każdej chwili przerwać przyciskając dowolny klawisz (wznowienie wyświetlania następuje także po wciśnięciu dowolnego klawisza). Po wyświetleniu nazw wszystkich programów (zbiorów) wraz z ich rozmieszczeniem na dyskietce możemy wyświetlić mapę innej dyskietki.

Chcąc wydrukować mapę dyskietki na drukarce należy wcisnąć klawisz P.

```
10 POKE &BDEE,&C9:KEY DEF 66,0,0,0,0:ON ERROR GOTO 350
20 IF HIMEM<>&FFF THEN OPENOUT"d":MEMORY &FFF:CLOSEOUT:
LOAD"progbin.bin",&1000
30 MODE 2:INK 0,13:INK 1,0:BORDER 13:ZONE 21:PRINT"DYSK
MAPA - (C) PROGRAM WYSWIETLA MAPE DYSKIETKI":WINDOW 1,8
0,3,25:WIDTH 255
40 DIM t(39):xp=POS(#0):yp=VPOS(#0)
50 LOCATE xp,yp:PRINT"WLOZ DYSKIETKE DO STACJI A: I NAC
ISNIJ DOWOLNY KLAWISZ":WHILE INKEY$="":WEND
60 CLS
70 FOR x=0 TO 39:LOCATE xp,yp:PRINT"CZYTANIE DYSKIETKI
 -  SCIEZKA "x:CALL &1000,x
80 t(x)=256:buff=&100D
90 IF PEEK(&100C)<>0 THEN tx=0:b=buff:GOTO 130
100 y=PEEK(buff):IF y=t(x) THEN 130
110 IF y<t(x) THEN t(x)=y:b=buff
120 buff=buff+1:GOTO 100
130 IF x=0 THEN spt=buff-b
140 NEXT
150 IF t(0)=&C1 THEN t=0:s=&C1
160 IF t(1)=1 THEN t=1:s=1
170 IF t(2)=&41 THEN t=2:s=&41
180 st=t:ss=s+4
190 buff=&1100:FOR x=0 TO 3:CALL &1003,buff,s+x,t:buff=
buff+512:NEXT
200 LOCATE xp,yp:PRINT"WCISNIJ  ENTER  DLA WYSWIETLENIA
 MAPY DYSKIETKI, DOWOLNY KLAWISZ = PAUZA"
210 i$="":WHILE i$="":i$=INKEY$:WEND:i$=UPPER$(i$):IF i
$=CHR$(13) THEN pri=0:GOTO 220 ELSE IF i$="P" THEN pri=
-1 ELSE 210
220 CLS:FOR x=&1101 TO &18FF STEP 32:IF PEEK(x-1)=&E5 T
HEN 370
230 FOR y=x TO x+7:PRINT CHR$(1)CHR$(PEEK(y));: IF pri
THEN IF PEEK(y)>31 THEN PRINT#8,CHR$(PEEK(y)); ELSE PRI
NT#8,"?";
240 NEXT:PRINT".";:IF pri THEN PRINT#8,".";
250 FOR y=x+8 TO x+10:q=PEEK(y):IF q>127 THEN q=q-128
260 PRINT CHR$(1)CHR$(q);:IF pri THEN PRINT#8,CHR$(q);:
270 NEXT
280 PRINT"  SCIEZKA/SEKTOR ";:y=x+15:lc=0:IF pri THEN P
RINT#8,"  SCIEZKA/SEKTOR ";
290 r=PEEK(y):IF r=0 THEN 350
300 track=INT(r*2/spt):sect=(r*2)-(track*spt)+t(track):
track=track+st
310 GOSUB 400
320 sect=sect+1:IF sect>t(track)+spt-1 THEN track=track
+1:sect=t(track)
330 GOSUB 400
340 y=y+1:IF y<x+31 THEN 290
350 IF INKEY$<>"" THEN WHILE INKEY$<>"":WEND:WHILE INKE
Y$="":WEND
360 PRINT:PRINT:IF pri THEN PRINT#8:PRINT#8
370 NEXT:IF pri THEN PRINT#8,CHR$(12);
380 PRINT"MAPA INNEJ DYSKIETKI ? (T/N)";:i$="":WHILE i$=
"":i$=INKEY$:WEND:i$=UPPER$(i$):IF i$="N" THEN CALL 0
390 RUN
400 t$=STR$(track)+"/":IF track<10 THEN t$=" "+t$
410 PRINT t$;HEX$(sect,2);:IF pri THEN PRINT#8,t$;HEX$(
sect,2);
420 lc=lc+1:IF lc=8 THEN IF y<>x+30 THEN IF PEEK(y+1)<>
0 THEN PRINT TAB(30);:lc=0:IF pri THEN PRINT#8,TAB(30);
430 RETURN
```

Program SEKTORED pozwala na dokonywanie wszelkich zmian na poziomie sektora (poprzez zmianę dowolnych bajtów danego sektora) wybranej ścieżki dyskietki.

Chcąc poprawić program (zbiór) znajdujący się na dyskietce musimy wiedzieć, jakie ścieżki i sektory zajmuje. Informacje te uzyskamy uruchamiając, prezentowany wcześniej program DYSKMAPA.

OPIS DZIAŁANIA PROGRAMU

Z chwilą uruchomienia programu SEKTORED, u dołu ekranu zostanie wyświetlony opis używanych przy edycji klawiszy. Następnie wybieramy interesujący nas numer ścieżki (od 0 do 39) i sektora (od C1 do C9). Na ekranie monitora, wybrany sektor wyświetlany jest jednocześnie w kodzie hexadecymalnym (lewa część ekranu) oraz w kodzie ASCII (prawa część ekranu). Każdy sektor zajmuje 512 bajtów. Na ekranie monitora przedstawionych jest 256 bajtów wybranego sektora. Drugie 256 bajtów możemy wyświetlić po wciśnięciu klawisza funkcyjnego F2.

Mając programy DYSKMAPA i SEKTORED jesteśmy w stanie poprawić nasz język polski, obcojęzyczne opisy i komunikaty większości gier komputerowych.

```
1 REM ***************************************************
2 REM * UWAGA !                                          *
3 REM * W liniach 80, 100 w miejsce znaku '+' nalezy *

4 REM * wpisac znak, wciskajac jednoczesnie klawisz  *
5 REM * CONTROL oraz klawisz ze znakiem '@',          *
6 REM * a w miejsce znaku '$' nalezy wpisac znak,     *
7 REM * wciskajac jednoczesnie klawisz CONTROL oraz   *
8 REM * klawisz z litera 'R'.                         *
9 REM ***************************************************
10 POKE &BDEE,&C9:KEY DEF 66,0,0,0,0:ON ERROR GOTO 740
20 IF HIMEM<>&FFF THEN OPENOUT"d":MEMORY &FFF:CLOSEOUT
:LOAD"progbin.bin",&1000
30 FOR x=1 TO 3:KEY x,CHR$(x):NEXT
40 MODE 2:INK 0,13:INK 1,0:BORDER 13:ZONE 21:PRINT"SEK
TORED (C) - PROGRAM UMOZLIWIA EDYCJE ZBIORU NA POZIOMI
E SEKTOROW NA DYSKIETCE":WINDOW 6,80,3,20:WINDOW#1,6,8
0,22,25
50 PRINT"WLOZ DYSK DO STACJI A: i NACISNIJ DOWOLNY KLA
WISZ"
60 WHILE INKEY$="":WEND
70 CALL &1000,0:IF PEEK(&100C)<> 0 THEN PRINT CHR$(7):
GOTO 60
80 INPUT;"+$Ktora sciezka (0-39) ",t:IF t<0 OR t>39 TH
EN 80
90 CALL &1000,t
100 PRINT CHR$(13)"+$Sciezka"t"  ";:IF t<10 THEN PRINT
" ";
110 PRINT"Sektory   ";
120 mi=256:y=&100D:yy=y
130 p=PEEK(y):IF p=mi THEN 160
140 IF p<mi THEN mi=p:yy=y
150 y=y+1:GOTO 130
160 n=y-yy:IF n<8 OR n>9 THEN n=9
170 FOR x=yy TO yy+n-1:PRINT HEX$(PEEK(x),2)" ";:NEXT:
IF n=8 THEN PRINT"   ";
180 INPUT;"   Ktory sektor ",s$:ON ERROR GOTO 750
190 s=VAL("&"+s$)
200 ON ERROR GOTO 760
210 page=1
220 WINDOW SWAP 0,1:CLS:CALL &1003,&1100,s,t:CLS:WINDO
W SWAP 0,1:menu=-1
230 CLS:PRINT"Sciezka"t"  Sektor "UPPER$(s$)"   Strona
"page
240 PRINT:start=&1100+(page-1)*256
250 CALL &1300,start:IF menu THEN GOSUB 630
260 x=1:y=3:buff=start
270 LOCATE x,y
280 i$="":WHILE i$="":CALL &BB81:i$=INKEY$:CALL &BB84:
WEND:i=ASC(i$):i$=UPPER$(i$)
290 IF i=&E0 THEN GOSUB 700:GOTO 280
300 IF i>31 THEN 350
310 IF i=1 THEN 480
320 IF i=0 THEN CALL 0
330 IF i=2 THEN page=3-page:GOTO 230
340 IF i=3 THEN 100
350 IF i$<"0" OR i$>"F" OR (i$>"9" AND i$<"A") THEN 42
0
360 PRINT i$;
370 buff=start+16*(y-3)+INT(x/3):p$=HEX$(PEEK(buff),2)
380 p=3*(x/3-INT(x/3)):MID$(p$,p)=i$
390 pn=VAL("&"+p$):POKE buff,pn:LOCATE 52+INT(x/3),y
400 IF pn<32 THEN pn=46
410 PRINT CHR$(pn);:x=x+1:x=x-(x/3=INT(x/3)):GOTO 430
420 x=x+3*(i=&F2)-3*(i=&F3):y=y-(i=&F1)+(i=&F0):IF (x+
1)/3=INT((x+1)/3) THEN x=x-1
430 IF x<1 THEN y=y-1:x=46
440 IF x>47 THEN x=1:y=y+1
450 IF y<3 THEN y=18
460 IF y>18 THEN y=3
470 GOTO 270
480 x=52+INT(x/3)
490 LOCATE x,y
500 i$="":WHILE i$="":CALL &BB81:i$=INKEY$:CALL &BB84:
WEND:i=ASC(i$)
510 IF i=&E0 THEN GOSUB 700:GOTO 500
520 IF i>3 THEN 540
530 IF i=1 THEN x=1+(x-52)*3:GOTO 270 ELSE 320
540 IF i>&EF AND i<&F4 THEN 570
550 PRINT i$;:buff=start+16*(y-3)+x-52:POKE buff,i
560 LOCATE 1+(x-52)*3,y:PRINT HEX$(i,2);:x=x+1
570 x=x+(i=&F2)-(i=&F3):y=y-(i=&F1)+(i=&F0)
580 IF x<52 THEN y=y-1:x=67
590 IF x>67 THEN x=52:y=y+1
600 IF y<3 THEN y=18
610 IF y>18 THEN y=3
620 GOTO 490
630 WINDOW SWAP 0,1
640 PRINT CHR$(240)".....Przesun do gory o bajt
f1.....Zmiana kodu HEX na ASCII"
650 PRINT CHR$(241)".....Przesun w  dol  o bajt
f2.....Zmiana strony sektorow"
660 PRINT CHR$(242)".....Przesun w  lewo o bajt
f3.....Zmiana sektora na sciezce"
670 PRINT CHR$(243)".....Przesun w prawo o bajt
COPY...Kopiuj sektor na dysk"
680 menu=0
690 WINDOW SWAP 0,1:RETURN
700 WINDOW SWAP 0,1:CLS
710 PRINT TAB(21)"NAGRANIE sciezki"t"sektora "HEX$(s,2
);
720 PRINT TAB(28)"OK. (T/N) ? ";:q$="":WHILE q$<>"T" A
ND q$<>"N":q$=INKEY$:q$=UPPER$(q$):WEND:PRINT q$" ";
730 IF q$="Y" THEN CALL &1009,&1100,s,0,t
740 CLS:GOTO 640
750 RESUME 80
760 RUN
```

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

HOME TERMINAL - terminal domowy,
HOMOGENEOUS - jednorodny,
HOMOGENEOUS COMPUTER NETWORK - jednorodna (homogeniczna) sieć komputerowa,
HOPPER - zbiornik,
HORIZONTAL DRIVING - synchronizacja pozioma,
HORIZONTAL MICROCODING - mikrokodowanie poziome,
HORIZONTAL MICROPROGRAMMING - mikroprogramowanie poziome,
HORIZONTAL PARITY - patrz: HORIZONTAL PARITY CHECK,
HORIZONTAL PARITY CHECK - poprzeczna kontrola parzystości,
HORIZONTAL REDUNDANCY CHECK - kontrola poprzeczna,
HORIZONTAL TAB - znak tabulacji poziomej,
HOST - główny, macierzysty, także: gospodarz,
HOST COMPUTER - komputer główny, komputer gościnny, komputer macierzysty, komputer obliczeniowy,
HOST INTERFACE - interfejs komputera obliczeniowego,
HOST LANGUAGE - język uzupełnieniowy,
HOST SYSTEM - 1. system główny, 2. komputer główny,
HOT BACKUP - "gorące" składowanie (tworzenie kopii),
HOT POTATO ROUTING - metoda najszybszej transmisji (kolejne węzły sieci starają się przekazać paczkę danych jak najszybciej dalej),
HOT SPARE - patrz: HOT BACKUP,
HOT STANDBY - patrz: HOT BACKUP,
HOUSE - mieścić (w sobie), chować, przechowywać,
HOUSEKEEPING - operacje porządkowe,
HOUSEKEEPING INFORMATION - informacja organizacyjna, informacja porządkowa,
HOUSEKEEPING INSTRUCTION - rozkaz organizacyjny, rozkaz porządkowy,
HOUSEKEEPING OPERATION - operacja porządkowa, operacja organizacyjna,
HOUSEKEEPING OVERHEAD - systemowe koszty, systemowe nakłady,
HOUSEKEEPING ROUTINE - program organizacyjny,
HOUSTON AUTOMATIC SPOOLING PROGRAM - system HASP do sterowania zdalnym przetwarzaniem (pakietowy system operacyjny dla komputerów serii IBM/360),
HOWEVER - natomiast,
HRDWARE LOCKOUT - konstrukcyjne zabezpieczenie,
HSAM - patrz: HIERARCHICAL SEQUENTIAL ACCESS METHOD,
HSV MODEL - model "barwa-nasycenie-wielkość" (w grafice komputerowej charakterystyka koloru za pomocą trzech parametrów: barwy, nasycenia i wartości jaskrawości),
HT - patrz: HORIZONTAL TAB,
HUB - gniazdo (krosownicy),
HUB RING - pierścień wzmacniający krawędź osiowego otworu (np. w dyskietce),
HUE - 1. odcień barwy, ton barwy, 2. barwa,
HUFFMAN CODE - kod Huffmana,
HUMAN ENGINEERING - ergonomia,
HUMIDITY - wilgotność (powietrza), wilgoć,
HUMIDITY CONTROL EQUIPMENT - urządzenie do regulowania wilgotności,
HUNTING - szukanie błędu,
HYBRID COMPUTATION - obliczenia hybrydowe,
HYBRID COMPUTER - komputer hybrydowy, maszyna hybrydowa,
HYBRID INTEGRATED CIRCUIT - hybrydowy układ scalony,
HYPERBOLIC EQUATION - równanie hiperboliczne,
HYPERBOLIC LOGARITHM - logarytm naturalny,
HYPHEN - łącznik (kreska), znak myślnika, znak "-",
HYPHENATION - przeniesienie, rozdzielenie słów w celu przeniesienia,

I

IA - patrz: INSTRUCTION ADDRESS,
IAD - patrz: IMAGE ACQUISITION AND DISPLAY,
IAS - patrz: IMMEDIATE ACCESS STORAGE,
IBM - patrz: INTERNATIONAL BUSINESS MACHINES CORPORATION,
IBM COMPATIBLE - kompatybilny (wymienny) z komputerami firmy IBM,
IBM PC - komputer osobisty firmy IBM, oparty na 16-to bitowym mikroprocesorze Intel 8088 lub jego modyfikacjach (IBM PC XT z dyskiem stałym, IBM PC AT na bazie mikroprocesora Intel 80286),
IBM PC RT - 32-bitowy komputer osobisty firmy IBM, z mikroprocesorem zbudowanym w oparciu o architekturę RISC,
IBM PS - częściowo zgodna z IBM PC seria mikrokomputerów charakteryzująca się podwyższonym stopniem integracji oraz grafiką o wysokiej rozdzielczości (wykorzystuje mikroprocesory 8086, 80286 i 80386),
IC - patrz: 1. INSTRUCTION COUNTER, 2. INTEGRATED CIRCUIT, 3. INFORMATION CONTENT,
ICA - patrz: INTRA-APPLICATION COMMUNICATIONS AREA,
ICAI - patrz: INTELLIGENT COMPUTER-ASSISTED INSTRUCTION,
ICAND - mnożna,
IC FAMILY - rodzina układów scalonych,
ICL - patrz: INTERNATIONAL COMPUTERS LTD.,
IC LAYOUT - topologia układów scalonych,
ICON - piktogram (służy do wybierania rodzaju operacji lub jej argumentu),
IC SOCKET - gniazdko układu scalonego, podstawka układu scalonego,
ID - patrz: 1. IDENTIFICATION, 2. IDENTIFIER, 3. IDENTIFY MARKER,
IDAS - patrz: INDUSTRIAL DATA ACQUISITION SYSTEM,
IDEAL - 1. idealny, doskonały, 2. ideał,
IDENTICAL - identyczny,
IDENTIFICATION - identyfikacja,
IDENTIFICATIONS BURST - ciąg identyfikacji systemu,
IDENTIFICATION DIVISION - część identyfikująca,
IDENTIFICATION MARK - znak tożsamości, znak rozpoznawczy,
IDENTIFIER - symbol identyfikujący, identyfikator, oznacznik,
IDENTIFY - identyfikować, oznaczać, oznakować, rozpoznawać,
IDENTIFY MARKER - znacznik identyfikacyjny,
IDENTITY - tożsamość, równanie tożsamościowe, także: nazwa (umieszczana na etykiecie dla określenia zawartości),
IDENTITY CHECK - kontrola zgodności, kontrola identyczności,
IDENTITY MATRIX - macierz jednostkowa,
IDENTYFYING MARKER - znacznik identyfikacyjny.
IDLE - jałowy, nieobciążony, nieczynny.
IDLE CAPACITY - niewykorzystana wydajność urządzenia,
IDLE CHARACTER - ślepy symbol, symbol przesyłany poprzez łącze w przypadku braku informacji,
IDLE LIGHT - wskaźnik postoju urządzenia,
IDLING CYCLE - pusty cykl maszyny,
IDP - patrz: INTEGRATED DATA PROCESSING,
IDS - patrz: INTEGRATED DATA STORE,
IEEE - patrz: INSTITUTE OF ELECTRICAL AND ELECTRONICAL ENGINEERS,
IER - mnożnik,

IF AND ONLY IF - 1. wtedy i tylko wtedy, 2. równoważność,
IFF - patrz: IF AND ONLY IF,
IFIP - patrz: INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING,
IF-STATEMENT - instrukcja (dyrektywa) warunkowa, operator warunkowy,
IF-THEN-ELSE - patrz: IF-STATEMENT,
IGNORE - nie brać pod uwagę, ignorować,
ILL-ADAPTED - nieprzystosowany,
ILLEGAL - nielegalny, niedopuszczalny, niedozwolony, nieprzepisowy, bezprawny,
ILLEGAL CHARACTER - znak nieprzepisowy, znak niedozwolony,
ILLEGAL INSTRUCTION - rozkaz nieprzepisowy, rozkaz nie dający się wykonać,
ILLEGAL OPERATION - operacja nielegalna, operacja niedozwolona,
ILLEGAL SYMBOL - symbol niedopuszczalny, symbol niedozwolony,
ILLEGIBLE - nieczytelny,
ILLOGICAL - nielogiczny,
ILLUMINATE - oświetlać, iluminować, także: podświetlać,
ILLUSTRATE - ilustrować,
ILLUSTRATION - ilustracja,
IMAGE - 1. informacja wizyjna, 2. obraz eksponowany, 3. odtwarzać, odrysować (np. na ekranie monitora),
IMAGE ACQUISITION AND DISPLAY - gromadzenie i wyświetlanie informacji wizyjnej,
IMAGE FILE - 1. moduł sterujący, 2. zbiór obrazu zadania,
IMAGE GENERATION - zestawianie obrazu, formowanie obrazu,
IMAGE GRAPHIC - grafika rastrowa (obrazowa),
IMAGE MEMORY - pamięć obrazu (pamięć, w której zachowany jest obraz z ekranu monitora),
IMAGE POSITIONING - ustawianie położenia obrazu,
IMAGE PROCESSING - przetwarzanie obrazów,
IMAGE REGENERATION - regeneracja obrazu, odtwarzanie obrazu (odtwarzanie zachowanego w pamięci obrazu ekranu monitora),
IMAGE SET - zbiór obrazów,
IMAGE SPACE - przestrzeń obrazowania,
IMAGE UNDERSTANDING - rozpoznawanie obrazów,

IMAGINARY - rzekomy, pozorny, urojony,
IMIS - patrz: INTEGRATED MANAGEMENT INFORMATION SYSTEM,
IMMATERIAL WARE - oprogramowanie, software,
IMMEDIATE - bezpośredni, natychmiastowy,
IMMEDIATE ACCESS MEMORY - szybkodziałające urządzenie pamięciowe (urządzenie, w którym czas dostępu do komórki pamięci nie zależy od adresu, a cykl pracy jest równy cyklowi pracy procesora),
IMMEDIATE-ACCESS STORAGE - pamięć szybka, pamięć operacyjna,
IMMEDIATE ACTION - działanie bezpośrednie, działanie natychmiastowe,
IMMEDIATE ADDRESS - adres bezpośredni,
IMMEDIATE ADDRESSING - adresowanie bezpośrednie, adresowanie proste, adresowanie natychmiastowe (z argumentem bezpośrednim),
IMMEDIATE DATA - 1. dana bezpośrednia, 2. operand bezpośredni,

IMMEDIATE MODE - tryb bezpośredni (używany do wprowadzania i wykonywania kolejnych poleceń z monitora),
IMMEDIATE OPERAND - operand bezpośredni,
IMMOBILIZE - unieruchomić,
IMMOVABLE - nieruchomy, stały,
IMMUNITY - odporność, niewrażliwość,
IMP - patrz: INTERFACE MESSAGE PROCESSOR,
IMPACT - 1. uderzenie, zderzenie, udar, 2. wpływ, 3. patrz: IMPLEMENTATION PLANNING AND CONTROL TECHNIQUE,
IMPACT PRINTER - drukarka uderzeniowa,
IMPERATIVE - imperatyw,
IMPERATIVE LANGUAGE - język rozkazujący,
IMPERATIVE STATEMENT - instrukcja rozkazująca,
IMPERFECT - niedoskonały, niezupełny,
IMPERFECTION - niedoskonałość,
IMPLEMENT - wprowadzić (w życie), wdrażać,
IMPLEMENTATION - 1. wdrażanie, wprowadzanie, wprowadzanie w życie, 2. realizacja,
IMPLEMENTATION MODULE - moduł realizacji,
IMPLEMENTATION PLANNING AND CONTROL TECHNIQUE - zastosowanie metody planowania i sterowania,
IMPLEMENTATION SPECIFICATION - opis realizacji,
IMPLEMENTS - narzędzia,
IMPLICATION - implikacja, wyniki,
IMPLICIT - niejawny, domniemany, ukryty, dający się wywnioskować,
IMPLIED ADDRESS - adres niejawny,
IMPLIED ADDRESSING - adresowanie niejawne, adresowanie implikowane,
IMPLY - dawać do zrozumienia, implikować, wysuwać wniosek, wynikać,
IMPORTED - importowany,
IMPORT LIST - lista importu, spis importu (w opisie modułu lista zadeklarowanych w innych modułach nazw wykorzystywanych w danym module),
IMPRESSION ROLLER - wałek dociskowy, wałek drukujący,
IMPROPER - niewłaściwy (np. znak),
IMPROVE - ulepszać, udoskonalać,
IMPROVEMENT - ulepszenie, udoskonalenie,
IMPURE DATA - dane zmieniane, dane wymienne,
IMPURE FUNCTION - funkcja z ubocznym efektem,
IMR - patrz: INTERRUPT-MASK REGISTER,
IMS - patrz: INFORMATION MANAGEMENT SYSTEM,
INACCESSIBLE - niedostępny,
IN ACCORDANCE WITH .... - zgodnie z ....,
INACCURACY - niedokładność,
INACCURATE - niedokładny, nieścisły,
INADEQUATE - niedostateczny, niewystarczający,
INADMISSIBLE - niedopuszczalny,
IN BAD REPAIR - w złym stanie,
IN BROAD TERMS - w szerokim ujęciu,
INBUILT - wbudowany,
IN CHRONOLOGICAL ORDER - w porządku chronologicznym,
INCIDENT - przyległy, incydentny (o wierzchołkach sąsiednich grafu),
INCIDENTAL ERROR - błąd przypadkowy,
INCIDENTAL INPUT - wejście przypadkowe,
IN-CIRCUIT EMULATOR - wewnątrzobwodowy emulator,
INCLUDE - włączać, zawierać,
INCLUDING - włącznie,
INCLUSION - zawarcie, uwzględnienie, dołączenie, włączenie,
INCLUSIVE OR OPERATION - suma logiczna, alternatywa,
INCLUSIVE PROPERTY - cecha dopuszczająca,

INCOMING DATA - dane wejściowe,
IN COMMISSION - w eksploatacji,
IN COMMON USE - powszechnie używany, powszechnie stosowany,
INCOMPARABLE - nieporównywalny,
INCOMPATIBILITY - niezgodność, niezdolność dopasowania się, niekompatybilność,
INCOMPATIBLE - niezgodny, niedający się pogodzić, także: niemieszalny, nie mieszający się,
INCONCLUSIVE RESULTS - wyniki nieprzekonujące,

INCONNECTOR - łącznik, wskazujący (na schemacie blokowym) dalszy ciąg przerwanej linii,
IN CONSEQUENCE - w wyniku,
INCONSISTENCY - sprzeczność, niekonsekwencja, nielogiczność,
INCONSISTENT - sprzeczny, także: nielogiczny,
INCONSISTENT COMPILATION - kompilacja nieskoordynowana, kompilacja sprzeczna,
INCORPORATE - włączyć, wprowadzić, wbudować, także: zawierać (w sobie), obejmować,
INCORRECT - niepoprawny, błędny,
INCREASING SEQUENCE - ciąg rosnący,
INCREMENT - przyrost, powiększenie, inkrement, także: powiększać, zwiększać,
INCREMENTAL COMPILER - kompilacja przyrostowa,
INCREMENTAL COMPUTER - komputer przyrostowy,
INCREMENTAL DISPLAY - zobrazowanie przyrostowe,
INCREMENTAL REFINEMENT - oczyszczanie przyrostów,
INCREMENTAL TAPE STORAGE - pamięć taśmowa krokowa,
INCREMENTAL VECTOR - wektor względny, wektor przyrostowy,
INCREMENTER - rejestr następnikowy,
INCREMENT OPERATION - operacja przyrostu,
INCREMENT SIZE - 1. rozmiar przyrostu, 2. w grafice komputerowej odstęp pomiędzy sąsiednimi adresowanymi punktami powierzchni zobrazowania),
IN CYCLES - cyklicznie,
INDELIBLE - nieścieralny, niezmywalny, nieusuwalny,
INDENT - patrz: INDENTATION,
INDENTATION - 1. wcięcie (tekstu), przesunięcie (przemieszczenie) w prawo, 2. wcinać, przesuwać, przemieszczać,
INDEPENDENCE - niezależność,
INDEPENDENT - niezależny,
INDEPENDENTLY - niezależnie,
INDEPENDENT TASKS - zadania niezależne,
IN DETAIL - szczegółowo,
INDETERMINATE ERROR - błąd nieoznaczony,
INDEX - wskaźnik, indeks, wykładnik (potęgi), także: skorowidz, sporządzać wykaz,
INDEXABLE INSTRUCTION - rozkaz indeksowalny,
INDEX BUCKET - porcja (blok) z tablicą indeksów, porcja indeksowa,
INDEX BUCKET INDICATOR - wskaźnik porcji indeksowej,
INDEX BUFFER - bufor indeksowy, bufor na tablice indeksowe,
INDEX CARD - karta kartotekowa,
INDEXED ACCESS METHOD - metoda dostępu indeksowego,
INDEXED ADDRESSING - adresacja indeksowa,
INDEXED FILE - plik (zbiór) indeksowany,
INDEXED SEQUENTIAL ACCESS METHOD - metoda dostępu indeksowo-sekwencyjna,
INDEXED-SEQUENTIAL DATA SET - zbiór danych indeksowo-sekwencyjnych,
INDEX ENTRY - pozycja w skorowidzu,
INDEX FIELD - pole indeksowe,

INDEX FILE - plik (zbiór) indeksowy,
INDEXING - indeksowanie,
INDEXING LANGUAGE - język indeksowania, język informacyjno-wyszukiwawczy,
INDEXING PROCEDURE - procedura indeksowania,
INDEXING REGISTER - rejestr indeksowy,
INDEXING UNIT - blok modyfikacji,
INDEX MODE - tryb indeksacji (warunki indeksacji),
INDEX NUMBER - wskaźnik,
INDEX OF A RADICAL - stopień pierwiastka,
INDEX PART - część indeksowa,
INDEX REGISTER - rejestr modyfikacji, rejestr B,
INDEX-SEQUENTIAL ACCESS - dostęp indeksowo-sekwencyjny,
INDEX TABLE - tablica indeksowa, tablica danych ułożona w określonym porządku,
INDEX WORD - słowo indeksowe, modyfikator,
INDICATE - wskazywać, zaznaczać, oznaczać, pokazywać (na rysunku),
INDICATION DEVICE - urządzenie wskazujące,
INDICATION ERROR - błąd wskazania,
INDICATIVE DATA - dane charakterystyczne (dane identyfikujące lub opisujące obiekt),
INDICATOR - wskazówka (przyrządu), wskaźnik, indykator,
INDICATOR LIGHT - światło sygnalizacyjne,
INDICATOR ROUTING - wskaźnik wyboru trasy,
INDIRECT ADDRESS - adres pośredni,
INDIRECT ADDRESSING - adresowanie pośrednie,
INDIRECT FILE - 1. plik (zbiór) pośredni, 2. plik (zbiór) zależny,
INDIRECTION - 1. pośredniość, 2. wykorzystanie adresacji pośredniej,
INDIRECTION LABEL - poziom pośredniości,
INDIRECTION OPERATOR - operacja zależności, operacja pośredniości,
INDIRECT JUMP - 1. skok pośredni, 2. rozkaz skoku pośredniego,
INDISCRIMINATE - bezładny, bez wyboru, przypadkowy,

LIGA MYŚLĄCYCH

ZADANIE 1

Liczba L jest czterocyfrowa i jest kwadratem innej liczby całkowitej 1. Dwie pierwsze cyfry liczby L są jednakowe i dwie ostatnie cyfry są również jednakowe. Znaleźć liczby L i 1.

ZADANIE 2

Miniaturowa wskazówka zegara elektrycznego porusza się skokami w końcu każdej minuty. Jakie jest prawdopodobieństwo, że w losowo wybranej chwili zegar wskazuje godzinę różniącą się od rzeczywistej nie więcej niż o 20 sekund?

ZADANIE 3

Duże meteoryty upadają na Ziemię przeciętnie raz na miesiąc, przy czym prawdopodobieństwo, że meteoryt upadnie na dowolny obszar, jest proporcjonalne do pola tego obszaru. Obliczyć prawdopodobieństwo zdarzenia, że w ciągu 10 lat spadną na terytorium Polski co najmniej dwa duże meteoryty.

ZADANIE 4

Jacek i Tomek grają w szachy tak długo, aż jeden z nich wygra dwie kolejne partie. Prawdopodobieństwo, że w pojedynczej partii wygra Jacek jest równe p, a odpowiednie prawdopodobieństwo, że wygra Tomek?

ZADANIE 5

Ani nigdy nie wykipiało gotujące się mleko, gdyż przegotowuje je w specjalnym garnku. Na czym polega szczególna właściwość specjalnego garnka, z którego mleko nie kipi, mimo że temperatura mleka jest dokładnie równa temperaturze wrzenia wody?

Rozwiązania zadań prosimy przesyłać do redakcji do końca lutego 1989 r. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród czytelników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

Dokończenie ze str. 32

wiem założeniem jest, iż wszystkie „ważniejsze komputery" są podłączone do sieci telefonicznej) i można rozpocząć odgadywanie hasła. Oczywiście musimy dysponować komputerem i telefonem. Przy odrobinie szczęścia i pomysłowości podobno musi się udać.

Jaka jest zatem na to rada? W zasadzie najpewniej jest podłączać komputer do sieci telefonicznej, tylko wtedy, gdy zamierzamy pracować w domu via telefon z firmą. Ewentualni „włamywacze", z angielskiego „hakerzy", słyszą wówczas sygnał zajętości. W praktyce komputerowej, metoda ta jest do przyjęcia tylko w sporadycznych wypadkach, i propozycję tę należy traktować jako udany żart.

Wiele dziwacznych błędów popełniono w starszych systemach na przykład: hasła były „instalowane" w nich na stałe. Można je było oczywiście zmienić, ale użytkownikom po prostu się nie chciało. Nie przestrzegali podstawowej zasady: wszystkie sposoby zwiększania bezpieczeństwa baz danych są dobre. Na przykład IBM dysponuje siecią NOSS z 190 000 użytkowników. System nadzorujący tę sieć automatycznie wymusza co dwa tygodnie zmianę hasła (bezwarunkowo). Nowe hasło nie mogło być użyte w ostatnich 6 miesiącach. Odrzucane jest także hasło zbyt banalne, jak np.: system, guest (gość), secret, help (pomoc), demo, manager, test, password (hasło) i inne. Tworzone są odpowiednie listy złych haseł, łącznie z tworzeniem „top ten".

Komputerowi włamywacze

W walce przeciw hakerom nie można polegać ani na organach ścigania, ani na prawie. Po pierwsze dlatego, iż gdy powiadomimy policję o „włamaniu do komputera" dowiadują się o tym nasi klienci, a to jest bardziej kosztowne od strat wynikłych w efekcie włamania. W Wielkiej Brytanii straty te szacuje się na miliony funtów rocznie. Należy także pamiętać, że policja ma bardzo małe doświadczenie w tej problematyce. Na przykład „komputerowa komórka" londyńskiej Metropolitan Police, liczy zaledwie 5 osób. Zresztą w państwie tym hakerstwo nie jest traktowane jako postępowanie sprzeczne z prawem. Natomiast w Teksasie (tam zawsze coś muszą wymyśleć) uznano takie działania jako przestępstwo. Od 1985 roku przyjęto, iż czynności polegające na „szkodliwym korzystaniu z komputera" lub „działaniu mającym na celu wpłynięcie na normalne użytkowanie komputera" — kwalifikowane są jako przestępstwa. Aktualnie, toczy się sprawa przeciw hakerowi (w Ameryce), który wymazał w jednej z firm 168 000 rekordów związanych z płacami. Grozi mu za to 10 lat więzienia! Ale gdy piszący te słowa telefonicznie z Warszawy spowoduje podobne szkody w... Teksasie, jakie prawo mnie dotyczy?

Pora na wnioski. Po pierwsze: pilnować haseł, nie notować ich (ponoć magnez poprawia pamięć), nie mówić kolegom i żonom (bez względu na to, czy własnym, czy cudzym). Po drugie, ci których zawodem jest dbanie o bezpieczeństwo danych zgromadzonych w pamięciach komputerów powinni mieć odpowiednie przygotowanie. Należy bowiem pamiętać, że „legitymowanie" użytkownika nie może pozostać jedynym zabezpieczeniem przed hakerami.

Więcej szczegółów w: How secure is your computer? Global Business, Autumm 1988.

W. G.

KRZYŻÓWKA NR 11



Litery z kratek ponumerowanych od 1 do 19 dadzą hasło, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji, na kartach pocztowych, w terminie do końca stycznia, naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe



Komputerowi włamywacze

Jeszcze niedawno omawiając przestępstwa komputerowe mieliśmy te, które odbywały się „tradycyjnymi" sposobami. Na przykład pewien zdolny (bardzo zdolny) programista, wykonując oprogramowanie finansowe dla dużej firmy, był to zapewne bank, dołączył do tego program własny, prywatny fragmencik. Dzięki temu zabiegowi, „resztówki" operacji przeliczeniowych, np.: 1/3 centa, także 1/20 centa, których nikt nigdzie nie zauważał, przesyłane były „całkiem spokojnie" na prywatne konto autora tego ciekawego pomysłu. Owo konto rosło niespodziewanie szybko aż do...

Ale czasy się zmieniły, a z nimi rodzaje przestępstw. Dowiedli tego dwaj angielscy dziennikarze: Robert Schifreen i Steve Gold. Pokazali oni, jak niepewne są zabezpieczenia komputerów „włamując się" do jednej z najbardziej zabezpieczonych elektronicznych poczt Wielkiej Brytanii. Oczywiście „wytrychem" był komputer, a dokładniej hasło otwierające drogę, via telefon do sekretnych zbiorów owej poczty. Izba Lordów uniewinniła „włamywaczy".

Już ponad 20 lat temu dostrzeżono ogromne korzyści płynące z przechowywania danych, zamiast na papierze, w komputerowych pamięciach. Niestety po 10 latach, a może nawet szybciej, pojawiły się szalone telefony do zastrzeżonych baz danych. Dzwoniący usiłowali je odczytać lub co gorsze zmienić ich treść. Pouczający jest w tym względzie film sprzed paru lat pt.: „Gry wojenne". Bohater owego filmu nie tylko odczytał tajną bazę danych ocen w szkole, ale przez telefon zmienił zawartość komputerowego dziennika! W 1985 roku nasi dziennikarze, jako pierwsi w Wielkiej Brytanii zostali aresztowani pod zarzutem „włamania się" do komputerowego systemu PRESTEL. Faktycznie uczynili to, by dowieść, jak mało bezpieczne są zgromadzone tam dane — nawet te najbardziej tajne. Nasi bohaterowie od połowy 1984 roku do wiosny 1985, starali się oszukać system, tak by móc dotrzeć do wszystkich zgromadzonych tam informacji. O finale sprawy już pisałem — uniewinnienie, „póki co"!

W zasadzie nielegalne korzystanie z komputerów (także ze zgromadzonych w ich pamięciach informacji) nie wymaga wielkich nakładów, trochę gotówki na połączenia telefoniczne (milczącym bo-

Dokończenie na str. 31

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 2032. Nr ind. 361682.

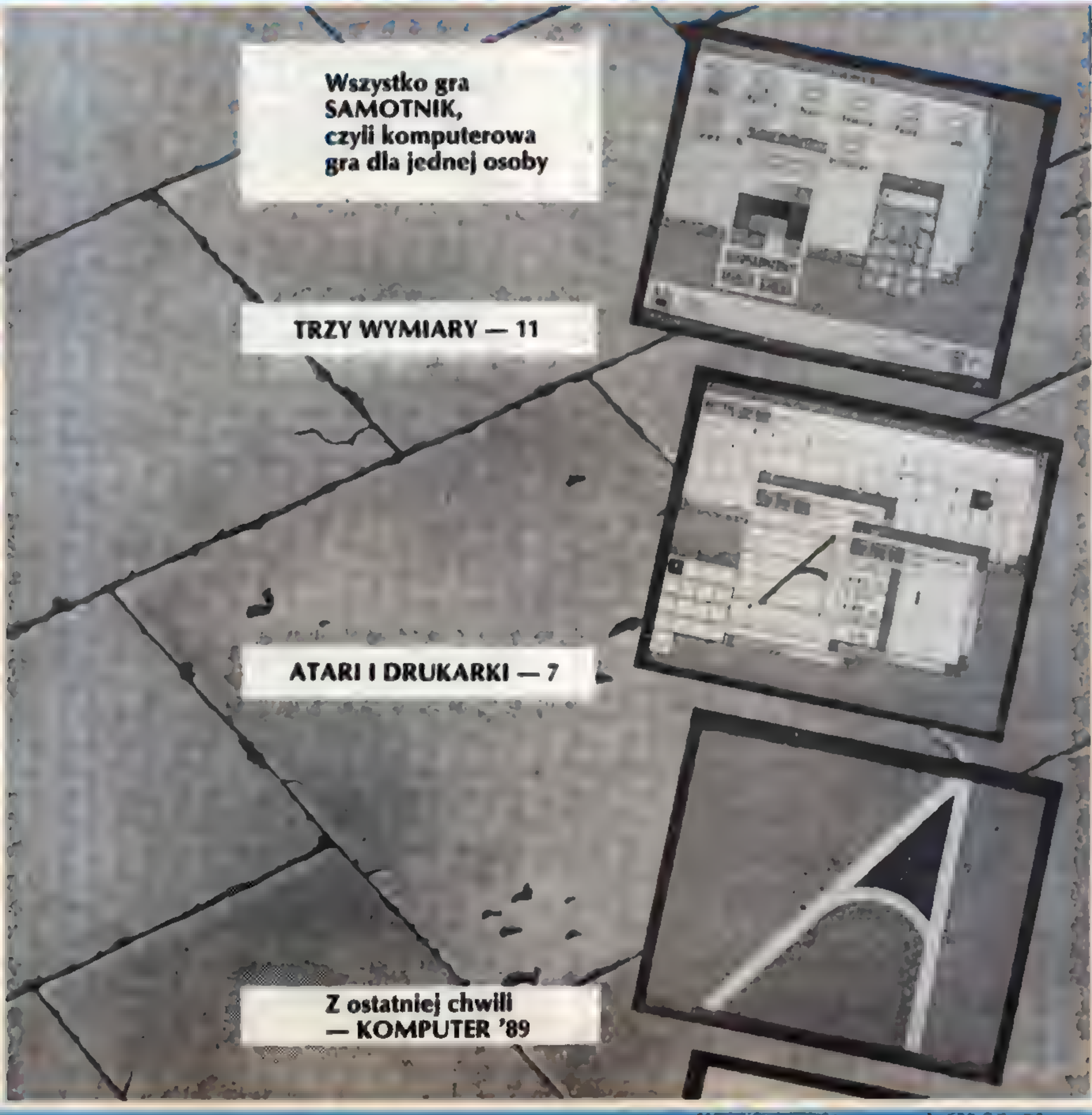
U-36

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 zł

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" NR 1/1989 ISSN 0860-2794



Wszystko gra SAMOTNIK, czyli komputerowa gra dla jednej osoby

TRZY WYMIARY — 11

ATARI I DRUKARKI — 7

Z ostatniej chwili — KOMPUTER '89

Oby nam się...
czyli zróbmy wreszcie coś

Zaczynamy swobodny lot w stronę nowej gospodarki, miejmy zatem nadzieję, że rządzącej się racjonalnymi prawami, surowymi dla wszelkiego pustosłowia i jednocześnie popierającymi ekonomiczne efekty działania.

Gospodarka bez gorsetu w nowym, szytym na miarę ambicji i wyobraźni fraku ustawy o działalności gospodarczej powinna być po prostu lepsza. Łatwiej będzie też nam, informatykom, którzy przywykli realizować funkcję celu w surowych kryteriach. Teraz tych ostatnich jakby mniej, a i cel nieco jaśniejszy.

Mamy okazję, której stracić nie można, aby udowodnić, że informatyka jest koniecznością i to nie koniecznością propagandową. Tak się bowiem złożyło, że jeśli ktoś chce powiedzieć coś naprawdę dobrego natychmiast dopowiada, że „komputery, to owszem nasza rzeczywistość...” itd. Tymczasem sam ledwo odróżnia telewizor od monitora, a przecież dostrzeganie odmienności fortepianu i komputerowej klawiatury nie daje jeszcze talentu mistrza Pendereckiego, czy choćby fizycznego daru słuchu.

Tak i z wiedzą bywa. Bo teraz nastał czas nie tylko dobrych pomysłów, ale ich realizacji, a nic jak działanie nie weryfikuje możliwości.

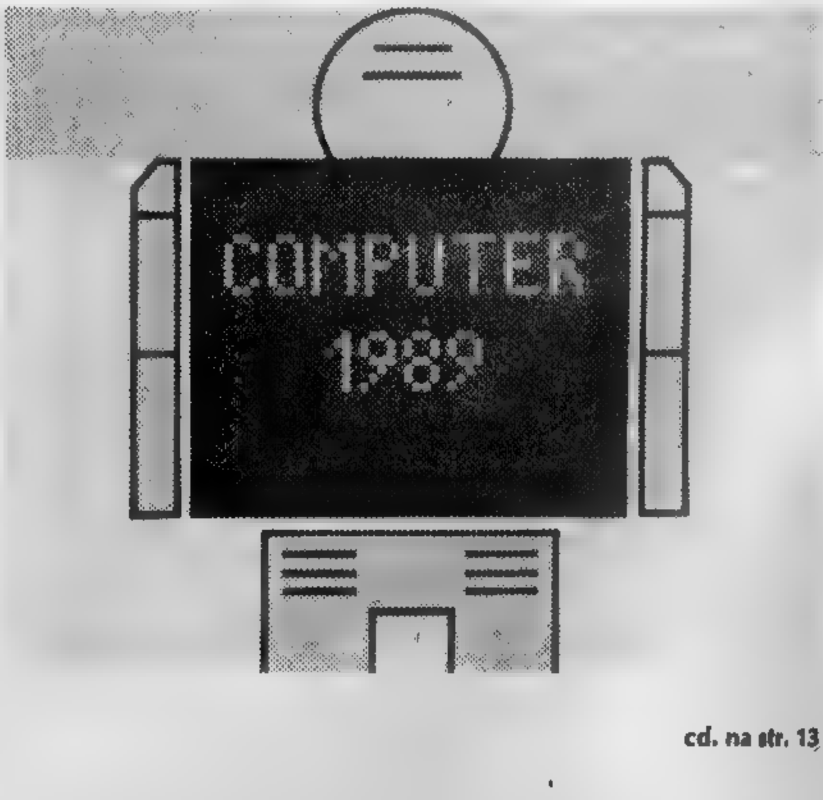
Pokonywać zatem trzeba kolejne bariery: jedną z nich są stale rosnące ceny sprzętu, które dyktuje kurs zielonej waluty. Na komputery pieniądze się ma lub nie. Tu nie ma żadnego wyboru — ZX nie zastąpi PC, zatem albo go się kupi, albo nie.

Nieco lepiej jest z oprogramowaniem. Programy standardowe bezczelnie kopiowane bez jakiejkolwiek licencji i ochrony praw autorskich krążą w spolszczonych wersjach i różnorakich cenach. Trzeba przyznać, że nie są to ceny wersji oryginalnych i to co „tam” kosztuje np. 1500$, u nas można „nabyć” za 200 tysięcy. Tak, tak, dwieście tysięcy i choć byłby to najkorzystniejszy przelicznik, o jakim nawet minister finansów nie śni, nie myśli się o eksporcie tej „oryginalnej” twórczości, bo szacunek do praw autorskich „tam” jakby większy, a i obawa przed byle czym niemała.

Mamy też własne, naprawdę własne i oryginalne oprogramowanie. Tu jednak też możemy zadziwić świat. Na ogół myśl ludzka kosztuje więcej niż „żelastwo”. U nas natomiast informatyczną twórczość można kupić za ułamek ceny komputera i część ta jest niemal z dnia na dzień mniejsza.

To tylko niektóre ekonomiczne szczegóły wstępu do informatyzacji. Dalej droga wcale nie usłana różami, bo jakiż rachunek zastosowań? Jeden komputer IBM PC, który w większości naszych firm jest ostoją informatyki, kosztuje w USA czy w RFN mniej więcej tyle ile wynosi miesięczna gaża jego operatora — u nas (uwzględniając niekorzystny przelicznik przy zakupie sprzętu za złotówki) tyle ile... roczne zarobki trzech informatyków. Nieźle zatem trzeba się nagimnastykować, aby te elektroniczne cacka zaczęły na siebie zarabiać — czego wszystkim i sobie życzę.

WIESŁAW CETERA



cd. na str. 13

SPIS TREŚCI

Małe jest piękne — 3
Indyjska Krzemowa Dolina — 4
Samotnik — gra — 5
Tajemniczy rozkaz INPUT — 6
Gra w kości (ZX-81) — 6
Atari i drukarki (II) — 7
Rozlewanie wina — 9
Trzy wymiary — 11
Procedura Gauss — 12
Informatyka w szkole — 14
Moc w obwodach prądu sinusoidalnego — 18
Obcy (ZX-81) — 19
Zmiany zawartości ekranu — 20
Streamery — notowania giełdowe — 21
Tendencje rozwojowe (Atari) — 22
Komputer '89 — 23
CPC 6128 — programy narzędziowe — 24
SF — 26
Kasetowy system operacyjny — schemat interfejsu — 28
Słownik — 29

małe JEST PIĘKNE

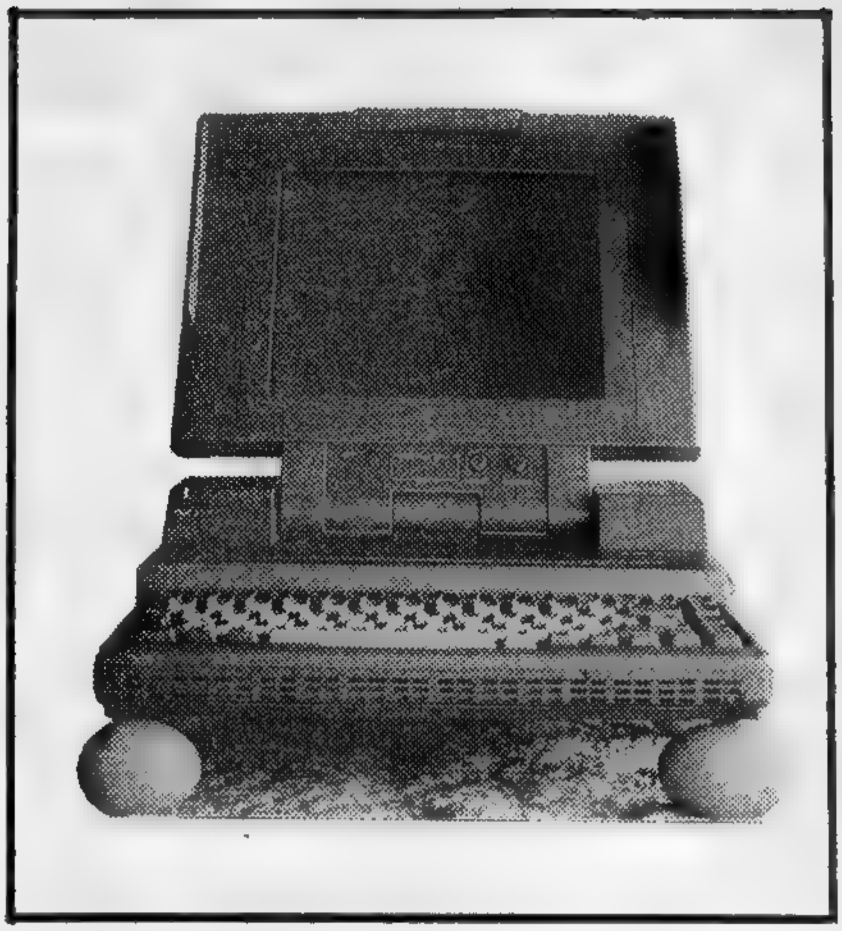
Pierwszy przenośny komputer powstał w 1978 roku, był to OSBORNE 1. Początkowo wątpiono w jego przydatność, tym bardziej że komputery tego typu wymagały zasilania z sieci. TANDY 100 (wielkości A4) był już na baterie. Maszynę tę „wyposażono" w programy do przetwarzania tekstów, „notatnik elektroniczny" oraz programy komunikacyjne. Dzięki temu komputer ten można było podłączyć do sieci telefonicznej i bez względu na odległość przesyłać teksty lub inne zbiory danych z naszej przenośnej maszynki np. do komputera w macierzystej firmie. Zresztą jako pierwsi dziennikarze zaczęli z powodzeniem wykorzystywać tego typu sprzęt właśnie jako „notatniki elektroniczne" połączone z redakcyjnym komputerem.

Przez pewien czas była wyraźnie widoczna różnica między prawdziwymi „laptopami" — tak właśnie nazywają się owe przenośne komputery — a tradycyjnymi komputerami klasy IBM PC. Laptopy bowiem, poza zasilaniem bateryjnym, wyróżniały się niestety szeregiem uproszczeń (u nas nazwano by to „zubożona wersja"), szczególnie odczuwalny był brak pamięci masowych i bardzo mały, słabo czytelny ekran ciekłokrystaliczny.

Jak można było się spodziewać, nieustanny postęp w technologii produkcji sprzętu elektronicznego, zaciera wspomnianą różnicę. Ekrany LCD stały się większe i bardziej czytelne, do tego stopnia, że wyświetlają tę samą liczbę znaków co tradycyjne stołowe monitory. Ponadto pojawiły się mniejsze, mniej energochłonne napędy dysków elastycznych (flopy), które mogły być z nieco większym powodzeniem instalowane w komputerach mniejszych od teczki. Niestety baterie umożliwiają tylko kilkugodzinną pracę tego sprzętu.

Pozwoliło to jednak wykorzystywać programy prosto „z biura", które były pisane na maszyny typu IBM PC. Mimo to, utrzymujące się kłopoty z zasilaniem (baterie pozostały nadal mało pojemne) przesądzały o opinii, iż sprzęt jest po prostu kiepski. Opinii tej nie zmieniły nawet odpowiednio przystosowane dyski twarde, energia baterii bowiem starczała nadal tylko na kilka godzin pracy laptopa. Spowodowało to pojawienie się nowej klasy sprzętu, oferowanego przez Toshibę — „stołowe-przenośne". Maszyny te są laptopami w wielkości i funkcji, ale mają zasilanie sieciowe, co w efekcie pozwoliło zastosować wysokiej jakości ekrany, pojemne i szybkie dyski dyski twarde oraz większą moc obliczeniową. Toshiba 3100 — pierwszy komputer tej kategorii, jest tak szybki jak najszybsze „PO-ty". Wyposażony jest w „gas-plasma" ekran, który wyświetla obraz podobnie do lampy neonowej. Może być zatem używany przy każdym świetle zewnętrznym w przeciwieństwie do ekranu ciekłokrystalicznego — obraz wyświetlany na nim widoczny jest tylko wówczas gdy otaczające go światło jest dostatecznie mocne, chyba że mamy pod ręką latarkę.

Istnieje jednak nadal zapotrzebowanie na komputery bardzo lekkie, przenośne, z których można korzystać w samolocie, w samochodzie (oczywiście będąc pasażerem) lub innym miejscu oddalonym od zasilającej sieci.



Wydaje się, że brytyjski komputer Z88 stanowi pewien kompromis, jak każdy Tandy 100 ma bowiem wbudowane oprogramowanie i mały ekran LCD (wyświetlający tylko kilka linii tekstu). Posiada jednak pewne istotne udoskonalenie — wymienne pamięci półprzewodnikowe, które z powodzeniem, w wielu zastosowaniach zastępują energochłonne napędy dyskowe.

Generalnie, na „rynku laptopów" toczy się walka między zapotrzebowaniem na energię, ze względu na „mobilnych użytkowników", a rozmiarem, wagą i możliwościami: przetwarzania, wyświetlania i składowania danych.

W dużej mierze o zapotrzebowaniu na laptopy decydują ludzie, którzy dużą część roku spędzają w podróżach służbowych — w samolocie, pociągu i samochodzie, w hotelu, także w domu.

Szczęśliwie problemy te nie dotyczą nas jeszcze bezpośrednio, aczkolwiek przydał by się taki laptop choćby podczas pisania tego tekstu (koniecznie ze słownikiem amerykańsko-angielskim). Pozwoliło by to także natychmiast zmniejszyć zużycie papieru do celów komputerowych. Zamiast bowiem setek stron wydruku odbieranego codziennie przez tysiące (to już miliony stron!) „korespondencyjnych" użytkowników komputerów, wręczano by im dyskietkę, może dwie z całością oczekiwanej informacji, np.: z wynikami obliczeń skomplikowanych problemów. Użytkownik ten może potem w dowolnej chwili odczytać owe dane, korzystając oczywiście z laptopa! Może być zasilany z sieci.

W. G.

Skorzystano z informacji zawartych w: „Small is beautiful", P. Jackson, Global Business, Autumn 1988.

Indyjska Krzemowa Dolina

Indie, ze względu na swą wielkość, są siódmym krajem świata, drugim natomiast, jeśli idzie o liczbę obywateli. Ponad 70 proc. z nich pracuje w rolnictwie. W ostatnim czasie i ten kraj postawił na nowoczesną technologię, w tym także na informatykę.

Od 1974 roku pracuje tam elektrownia atomowa, na orbicie okołoziemskiej umieszczonych jest już sporo indyjskich sputników, a 3 kwietnia 1984 roku pierwszy Hindus oglądał Ziemię z przestrzeni kosmicznej, dzięki radziecko-indyjskiej wyprawie.

Trudno byłoby nie dostrzec kwitnącego przemysłu komputerowego, którego centrum znajduje się w indyjskiej „Krzemowej dolinie" — mieście Bangalore w stanie Karnataka. W centrum tym między innymi ponad setka ekspertów pracuje nad badaniami podstawowymi, które dotyczą sztucznej inteligencji. O kompleksowości tych przedsięwzięć świadczy także fakt, iż przywódcy Indii otworzyli dwa lata temu granice dla obcego oprogramowania. Wiąże się to również z oczekiwaniami znacznego zwiększenia eksportu. Pierwsze znaczne efekty już są dostrzegalne — Amerykanie kupują programy „made in India".

Niestety podstawę „eksportu" Indii stanowią ludzie. Ten subkontynent ma trzeci największy na świecie potencjał doskonale przygotowanych naukowców. Każdego roku blisko 12 000 doktorów i inżynierów wyjeżdża do pracy w Ameryce i Europie. Zważywszy na wspomniane zmiany, stan ten nieco się zmienia — wielu z potencjalnych emigrantów znajduje odpowiednią pracę u siebie w kraju. Dzieje się tak za sprawą wspomnianej „Doliny", w której poza badaniami, pracami programowymi, produkuje się trzecią część sprzętu elektronicznego. Miasto rośnie szybciej niż inne — staje się autentycznym centrum badań i przemysłu. Znajduje się tutaj ponad 30 instytutów, między innymi Indyjski Instytut Nauki i Narodowe Aeronautyczne Laboratorium (NAL). W czerwcu 1988 roku NAL ogłosił uruchomienie pierwszego indyjskiego komputera z równoległym przetwarzaniem danych.

Oczekuje się, iż Bangalore stanowić będzie także łatwo dostępne źródło nowoczesnej technologii dla całego przemysłu Indii. Jednym z pierwszych tematów prowadzonych w tym kierunku prac jest telefoniczna łączność krajowa. Indie mają 3,8 miliona linii telefonicznych dla populacji przekraczającej 700 milionów ludzi. Dla porównania w Wielkiej Brytanii przypadają 42 telefony na 100 mieszkańców (danych z Warszawy niestety nie mam). W Indiach proporcja ta wynosi 0,54 na 100 mieszkańców. Problemem tym, i wieloma innymi, zajął się wykształcony (i wzbogacony) w USA Hindus Pitroda. Średnia wieku pracujących nad tym problemem wynosi 23 lata! Wprowadzane są, budowane według własnych projektów, cyfrowe centralne odporne na specyficzne warunki klimatyczne tej części świata.

W dużej części postęp techniczny jest wynikiem potrzeby aktywnego i praktycznego wykorzystywania kosmosu (15 000 zatrudnionych). Potrzebę tę uzasadnia wielkość państwa — ponad 3,25 miliona kilometrów kwadratowych. Rozwiązuje się w ten sposób problem łączności. Dzięki satelitom pozwala ona na efektywną walkę z analfabetyzmem — programy edukacyjne docierają do najodleglejszych miejsc subkontynentu.



Urządzenia te służą także do oceny stanu (w tym aktualnych potrzeb) ziemi rolnej, pogody (ostrzeżenia przed huraganami), ułatwiają szukanie wody i bogactw naturalnych. Prowadzone przez Hindusów kalkulacje wskazują, że sputnik spłaca się już po trzech latach eksploatacji. Według oceny Światowego Banku, w połowie przyszłego wieku Indie będą miały ponad 1,6 miliarda ludzi. W tej sytuacji tylko „wysoka technologia" wydaje się być jedyną narodową szansą Hindusów. Umiejętne jej wykorzystanie powinno pomóc rozwiązać podstawowe problemy życia — żywność, zdrowie i pracę.

Zważywszy na to, że „wysoka technologia", to także, a może przede wszystkim powszechne wykorzystanie komputerów (w rolnictwie, w badaniach kosmicznych i w szkole), warto pamiętać, iż nie jest to modą, ale koniecznością, także w naszych warunkach.

Opracowane na podstawie: India: old world, new technology, Global Business, Autumn 1988.

W. G.

SAMOTNIK

gra dla jednej osoby

Gra znana jest chyba wszystkim, ale na wszelki wypadek przypomnę jej zasady: na specjalnej planszy ustawia się 37 pionów. Przed rozpoczęciem gry usuwa się jeden pion. Pionami ruszamy się skacząc jednym przez drugi. Usuwamy z planszy pion, przez który się przeskoczyło. Celem gry jest pozostawienie na planszy jednego niezbitego piona lub osiągnięcie z góry założonego układu pionów.

Obsługa programu:

1. Program zgłasza się planszą informacyjną, na której przypomniane są podstawowe zasady gry (rys. 1). Naciskając dowolny klawisz wywołujemy planszę gry.

```
Gra jest przeznaczona dla jed-
nej osoby.

Gra polega na zbijaniu pionow,
przez ktore sie przeskakuje. Na
poczatku usuwa sie z planszy je-
den pion (klawisz <r>).

Pion przeznaczony do przeskoku
ustala sie klawiszem <r> ( pole
mruga). Nastepnie ustala sie
kierunek przeskoku.

Celem gry jest pozbycie sie
wszystkich pionow, z wyjatkiem
ostatniego, lub uzyskanie z gory
zalozonego ukladu pionow.
```

Rys. 1

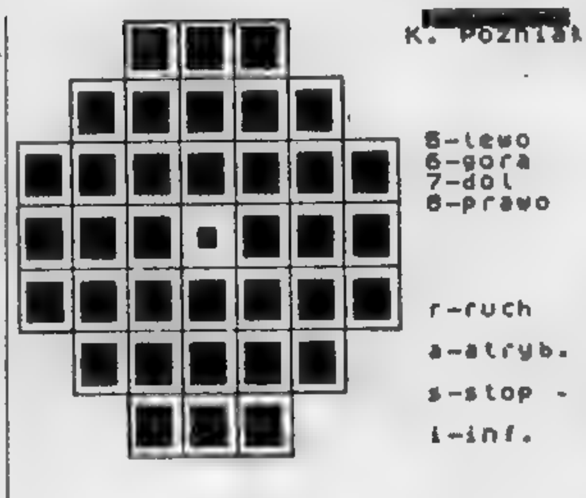
2. Na rysunku 2 pokazana jest plansza z początkowym rozstawieniem pionów. Zgodnie z zasadami gry należy usunąć z planszy jeden, wybrany przez siebie pion. Pion wybieramy za pomocą klawiszy:

5 — ruch znacznika w lewo
6 — ruch znacznika w prawo
7 — ruch znacznika w dół
8 — ruch znacznika w górę

(na rys. 2 oznaczone pole znajduje się w środku planszy)

Naciskając klawisz „r" usuwamy oznaczony pion z planszy i przystępujemy do rozgrywki.

3. Poruszamy się po planszy skacząc przez piony. Ustalamy pion, którym zamierzamy skoczyć, ustawiając na jego polu znacznik. Następnie naciskamy klawisz „r". Pole zaczyna mrugać za pomocą klawiszy 5, ..., 8 ustalamy kierunek skoku zaznaczonego piona. Komputer kontroluje poprawność skoku i nie dopuszcza do ruchów niezgodnych z zasadami gry.



Rys. 2

```
KOLORY:

linii:
pionow:

wybor pozycji: klawisze 6,7
wybor koloru: klawisze 5,8

k-koniec ustalania
```

Rys. 3

Funkcje dodatkowe programu:

Ustalenie atrybutów koloru: klawisz „a"

Na ekranie ukazuje się plansza pokazana na rys. 3. Program pozwala ustalić kolory: planszy (tło), linii planszy i pionów. Standardowo został przyjęty tryb czarno-biały. Pozycję wybieramy za pomocą klawiszy 6 i 7. Jest ona zaznaczona trybem INVERSE (np. na rys. 3 wybrana jest *plansza*). Kolor dobieramy za pomocą klawiszy 5 i 8. Powrót do gry: klawisz „k".

Przerwanie gry: klawisz „s"

Wywołanie planszy informacyjnej: klawisz „i"

Krzysztof POŹNIAK

```
   1 REM *************************
   2 REM
   3 REM
   4 REM
   5 REM Gra dla jednej osoby
   6 REM
   7 REM         ©1988 ,
   8 REM    Krzysztof Pozniak
   9 REM
  10 REM *************************
 100 REM
 110 DIM a(3): LET a(1)=0: LET a
(2)=7: LET a(3)=7
 120 GO SUB 9000
 130 GO SUB 9050
 140 GO SUB 9070
 150 GO SUB 3000
 200 GO SUB 9100
 210 LET y=4: LET x=4: LET ruch=
0
 500 GO SUB 8100
1000 REM
1010 LET x$=INKEY$: IF x$="" THE
N GO TO 1010
1020 IF x$="5" AND t(y+1,x)>0 AN
D ruch<2 THEN GO SUB 8000: LET x
=x-1: GO SUB 8100: GO TO 1010
1021 IF x$="6" AND t(y+2,x+1)>0
AND ruch<2 THEN GO SUB 8000: LET
 y=y+1: GO SUB 8100: GO TO 1010
1022 IF x$="7" AND t(y,x+1)>0 AN
D ruch<2 THEN GO SUB 8000: LET y
=y-1: GO SUB 8100: GO TO 1010
1023 IF x$="8" AND t(y+1,x+2)>0
AND ruch<2 THEN GO SUB 8000: LET
 x=x+1: GO SUB 8100: GO TO 1010
1030 IF x$="r" AND ruch=0 THEN L
ET t(y+1,x+1)=2: LET ruch=1: GO
SUB 8100: GO TO 1010
1031 IF x$="r" AND ruch=1 AND t(
y+1,x+1)=1 THEN LET ruch=2: GO S
UB 8200: GO TO 1010
1032 IF x$="r" AND ruch=2 THEN L
ET ruch=1: GO SUB 8100: GO TO 10
10
1040 IF x$="5" AND x>2 AND ruch=
2 THEN IF t(y+1,x-1)=2 AND t(y+1
,x)=1 THEN LET t(y+1,x+1)=2: LET
 t(y+1,x)=2: GO SUB 8000: LET t(
y+1,x-1)=1: LET x=x-1: GO SUB 80
00: LET x=x-1: GO SUB 8100: LET
ruch=1: GO TO 1010
1041 IF x$="6" AND y<7 AND ruch=
2 THEN IF t(y+3,x+1)=2 AND t(y+2
,x+1)=1 THEN LET t(y+1,x+1)=2: L
ET t(y+2,x+1)=2: GO SUB 8000: LE
T t(y+3,x+1)=1: LET y=y+1: GO SU
B 8000: LET y=y+1: GO SUB 8100:
LET ruch=1: GO TO 1010
1042 IF x$="7" AND y>2 AND ruch=
2 THEN IF t(y-1,x+1)=2 AND t(y,x
+1)=1 THEN LET t(y+1,x+1)=2: GO
SUB 8000: LET t(y-1,x+1)=1: LET
t(y,x+1)=2: LET y=y-1: GO SUB 80
00: LET y=y-1: GO SUB 8100: LET
ruch=1: GO TO 1010

1043 IF x$="8" AND x<7 AND ruch=
2 THEN IF t(y+1,x+3)=2 AND t(y+1
,x+2)=1 THEN LET t(y+1,x+1)=2: G
O SUB 8000: LET t(y+1,x+3)=1: LE
T t(y+1,x+2)=2: LET x=x+1: GO SU
B 8000: LET x=x+1: GO SUB 8100:
LET ruch=1: GO TO 1010
1050 IF x$="a" THEN GO SUB 2000
1060 IF x$="s" THEN CLS : GO SUB
 9000: GO TO 120
1070 IF x$="i" THEN GO SUB 3000:
 GO SUB 2060: GO TO 1010
1090 BEEP .1,20: GO TO 1010
2000 REM
2010 INK 7: PAPER 0: BORDER 0: C
LS
2020 PRINT TAB 7; INVERSE 1;"S A
 M O T N I K"; INVERSE 0;AT 3,10
;"KOLORY:";AT 15,3;"wybor pozycj
i: klawisze 6,7";AT 17,4;"wybor
koloru: klawisze 5,8",AT 20,7;"k
-koniec ustalania"

2030 FOR l=1 TO 3: PRINT AT 4+2*
l,8;t$(l): PLOT 143,144-l*16: DR
AW 17,0: DRAW 0,-9: DRAW -17,0:
DRAW 0,9: PRINT AT 4+2*l,18; PAP
ER a(l);"  ": NEXT l
2040 LET l=3: LET x$="6": GO TO
2060
2050 LET x$=INKEY$: IF x$="" THE
N GO TO 2050
2060 IF x$="6" OR x$="7" THEN PR
INT AT 4+2*l,8;t$(l): LET l=l+1-
2*(x$="7"): LET l=l-3*(l=4)+3*(l
=0): PRINT AT 4+2*l,8; INVERSE 1
;t$(l)
2061 IF x$="5" OR x$="8" THEN LE
T a(l)=a(l)+1-8*(a(l)=7): PRINT
AT 4+2*l,18; PAPER a(l);"  "
2070 BEEP .1,20: IF x$<>"k" THEN
 GO TO 2050
2080 LET yp=y: LET xp=x: GO SUB
9100: LET y=yp: LET x=xp: IF ruc
h<2 THEN GO TO 8100

2090 GO TO 8200
2099 RETURN
3000 REM
3010 INK 7: PAPER 0: BORDER 0: C
LS
3020 PRINT TAB 8; INVERSE 1;"S A
 M O T N I K"
3030 PRINT AT 2,2;"Gra jest prze
znaczona dla jed-nej osoby."
3040 PRINT AT 5,2,"Gra polega na
 zbijaniu pionow,przez ktore sie
 przeskakuje. Na poczatku usuwa
```

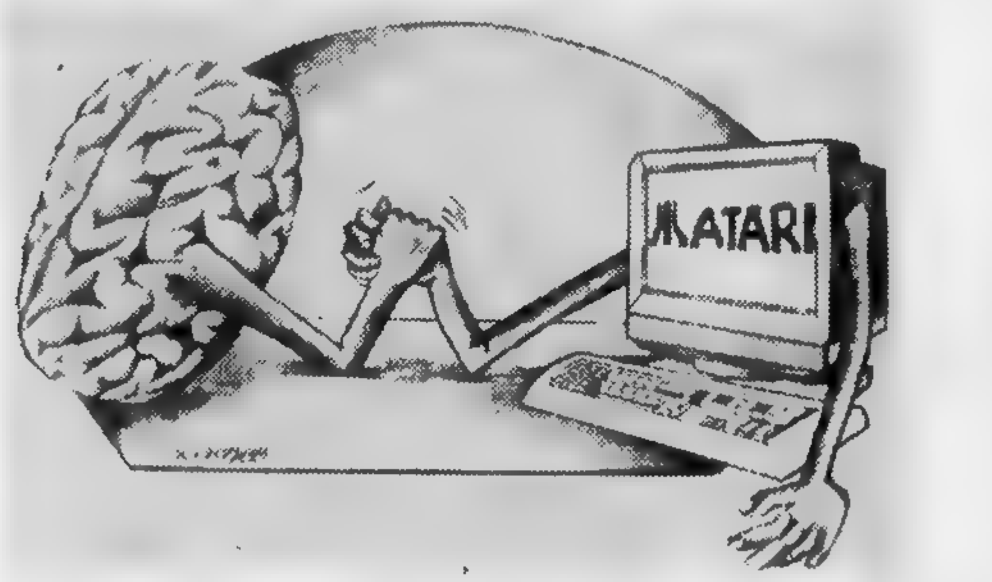
```
sie z planszy je-den pion (klawi
sz (r))."
3050 PRINT 'TAB 2;"Pion przeznac
zony do przeskokuustala sie klaw
iszem (r) ( pole mryga). Nastepn
ie ustala sie    kierunek przesk
oku."
3060 PRINT 'TAB 2;"Celem gry jes
t pozbycie sie   wszystkich pion
ow, z wyjatkiem  ostatniego, lub

 uzyskanie z goryzalozonego ukla
du pionow."
3070 PRINT ''TAB 4; INVERSE 1;"N
ACISNIJ DOWOLNY KLAWISZ"
3080 PAUSE 0
3090 RETURN
8000 REM
8010 PRINT INK a(3), PAPER a(1);
 INVERSE t(y+1,x+1)=1;AT 3*y-2,3
*x-2,"  ";AT 3*y-1,3*x-2,"  ": R
ETURN
8100 REM
8110 PRINT INK a(2)*(t(y+1,x+1)<
>1)+a(1)*(t(y+1,x+1)=1); PAPER a
(1)*(t(y+1,x+1)<>1)+a(3)*(t(y+1,
x+1)=1),AT 3*y-2,3*x-2;"▄",AT 3
*y-1,3*x-2,"▀"  RETURN
8200 REM
8210 FLASH 1  GO SUB 8100  FLASH
 0  RETURN
9000 REM
9010 DIM t(9,9): RESTORE 9900

9020 FOR y=1 TO 9: FOR x=1 TO 9
9030 READ t(y,x)
9040 NEXT x  NEXT y  RETURN
9050 REM
9060 DIM p(3,8)  FOR x=1 TO 8: F
OR y=1 TO 3: READ p(y,x): NEXT y
: NEXT x: RETURN
9070 REM
9080 DIM t$(3,8)  FOR l=1 TO 3
READ t$(l)  NEXT l  RETURN
9100 REM
9110 PAPER a(1)  INK a(2)  BORDE
R a(1): CLS : FOR l=1 TO 8
9120 PLOT p(1,l),p(2,l): DRAW p(
3,l),0
9130 PLOT 176-p(2,l),176-p(1,l):
 DRAW 0,-p(3,l)
9140 NEXT l
9150 LET poz=0  FOR y=1 TO 7: FO
R x=1 TO 7
9160 IF t(y+1,x+1)=1 THEN GO SUB
 8000


9170 NEXT x: NEXT y
9180 PRINT AT 0,23; INVERSE 1;"S
AMOTNIK"; INVERSE 0;AT 1,22;"K.
Pozniak";AT 6,23,"5-lewo",AT 7,2
3;"6-gora";AT 8,23;"7-dol";AT 9,
23;"8-prawo";AT 14,23;"r-ruch";A
T 16,23;"a-atryb.";AT 18,23;"s-s
top";AT 20,23;"i-inf."
9290 RETURN
9900 REM
9910 DATA 0,0,0,0,0,0,0,0,0
9911 DATA 0,0,0,1,1,1,0,0,0
9912 DATA 0,0,1,1,1,1,1,0,0
9913 DATA 0,1,1,1,1,1,1,1,0
9914 DATA 0,1,1,1,1,1,1,1,0
9915 DATA 0,1,1,1,1,1,1,1,0
9916 DATA 0,0,1,1,1,1,1,0,0
9917 DATA 0,0,0,1,1,1,0,0,0
9918 DATA 0,0,0,0,0,0,0,0,0
9920 REM
9930 DATA  52,172, 72
9931 DATA  28,148,120

9932 DATA   4,124,168
9933 DATA   4,100,168
9934 DATA   4, 76,168
9935 DATA   4, 52,168
9936 DATA  28, 28,120
9937 DATA  52,  4, 72
9940 REM
9950 DATA "planszy:"
9951 DATA "   linii:"
9952 DATA " pionow:"
```



Repr. J. Zelman

TAJEMNICZY ROZKAZ: INPUT dla komputera ZX Spectrum

Jest to rozkaz, który zgodnie z prawidłami języka BASIC **nic nie robi!** Wywołujemy bowiem rozkaz, który ani nie drukuje tekstu, ani nie wczytuje danych. Ma on jednak dwie bardzo przydatne własności:

po pierwsze: kasuje licznik scroll'i i pozwala na ciągłe przewijanie ekranu. Nie musimy dzięki temu „półkować" do komórki systemowej 23692 wartości 255. Często błąd podczas POKE kończy się zawieszeniem systemu. Przykładowy program znajduje się w liniach: 100—150

po drugie: czyści okno systemowe nie wtrącając się w okno programu. Po wykonaniu INPUT okno systemu ponownie zajmuje 2 linie na dole ekranu (odpowiada rozkazowi: **„CLS#0"**). Przykładowy program znajduje się w liniach 200—280

Krzysztof POŹNIAK

```
  1 REM **************************
  2 REM *                        *
  3 REM *                © 1988  *
  4 REM *                        *
  5 REM *  Krzysztof Poźniak     *
  6 REM *                        *
  7 REM *  Rozkaz:   INPUT ""    *
  8 REM *                        *
  9 REM **************************

100 REM ciagly SCROLL
110 FOR k=1 TO 10000
120 PRINT "liczba: ";k
130 IF INKEY$<>"" THEN GO TO 20
0
140 INPUT ""
150 NEXT k
200 REM czyszczenie okna syste-
        mowego
210 CLS
220 PRINT "ten tekst znajduje s
ie w oknie programu"

230 FOR k=1 TO 20
240 PRINT #0;"liczba: ";k;" w o
knie systemu"
250 NEXT k
260 PAUSE 10
270 INPUT ""
280 IF INKEY$="" THEN GO TO 230
```

PROGRAM ZX-81

```
2 REM *****************
3 REM *               *
4 REM *  GRA W KOSCI  *
5 REM *     ZX-81     *
6 REM *               *
7 REM *****************
10 CLS
15 FOR N=0 TO 20
20 PLOT 45,N+10
25 PLOT N+25,30
30 PLOT 25,N+10
35 PLOT N+25,10
40 NEXT N
43 LET Z=0
44 FOR C=1 TO 5
45 LET A=1+INT RND 6
46 LET Z=Z+A
47 PRINT AT 3,15;"
48 PRINT AT 13,0;"
"

50 UNPLOT 30,15
55 UNPLOT 30,20
60 UNPLOT 30,25
65 UNPLOT 40,15
70 UNPLOT 40,20
75 UNPLOT 40,25
76 UNPLOT 35,20
80 IF A=1 THEN PLOT 35,20
81 IF A=1 THEN GOTO 165
85 IF A=2 THEN GOTO 150
90 IF A=3 THEN PLOT 35,20
91 IF A=3 THEN GOTO 150
95 IF A=4 THEN GOTO 130
100 IF A=5 THEN GOTO 120
105 IF A=6 THEN GOTO 110
110 PLOT 30,15
120 PLOT 40,25
130 PLOT 30,25
140 PLOT 40,15
150 PLOT 40,20
160 PLOT 30,20
165 PRINT AT 11,0;A
166 IF C=5 THEN PRINT AT 1
3,0;"SUMA=";Z
180 NEXT C
181 IF Z=20 THEN PRINT AT
3,15;"NIEZLE"
185 INPUT A$
190 GOTO 43
```

J. JANIEC

Współpraca ATARI 800 XL z drukarkami D-100, DZM-180 KSR z monitorem MERA-7952

Część II

W pierwszej części artykułu opisana została układowa realizacja sprzężenia mikrokomputera ATARI 800 XL z drukarkami DZM-180 KSR, D-100 i monitorem ekranowym MERA-7952. Jak było wspomniane, w celu zapewnienia poprawnego funkcjonowania tak powstałego zestawu konieczne jest także stworzenie właściwego aparatu programowego. Rolę taką spełnia program sterujący, tzw. sterownik (ang. handler), napisany w asemblerze (MAC 65). Sterownik zapewnia programową obsługę portu A układu PIA, realizując następujące zadania:

— zmiana bajtu otrzymanego do wysłania, z postaci równoległej na szeregową,
— podanie kolejno stanu wszystkich bitów na bit 0 portu A układu PIA,
— odliczanie czasu trwania poszczególnych bitów.

Sterownik zajmuje około 250 bajtów pamięci RAM i w celu wykorzystania go przez programy napisane w BASIC-u umieszczono go na stronie 6 pamięci. Istnieje również możliwość łączenia go z popularnymi programami użytkowymi, jak edytory tekstu, bazy danych, arkusze elektroniczne itp., ale wówczas może nastąpić kolizja między „naszym" sterownikiem a programem zasadniczym. Konieczne będzie wówczas umieszczenie sterownika w innym, wolnym miejscu pamięci. Program niestety nie jest relokowalny ze względu na zastosowanie rozkazów skoku do podprogramu JSR i w celu przeniesienia go w inne miejsce pamięci należy przed *asemblacją* programu źródłowego nadać zmiennej START wartość żądanego adresu umieszczenia sterownika w pamięci (aktualnie Ø600, tj. początek 6 strony). Asemblacji można dokonać za pomocą makroasemblera MAC 65 lub asemblera ATARI Editor/Assembler.

Po omówieniu układowej realizacji połączenia warto byłoby podać ogólną zasadę funkcjonowania stworzonej konfiguracji oraz jej możliwości.

Głównym elementem zestawu jest mikrokomputer sterujący przetwarzaniem informacji. Programy do komputera wprowadzane są ze stacji dysków. Na dyskietkach gromadzi się również dane wejściowe oraz wyniki przetwarzania. Monitor telewizyjny, jak w każdym standardowym systemie mikrokomputerowym umożliwia bieżące kontrolowanie pracy systemu przez operatora. Drukarki są zasadniczymi urządzeniami przeznaczonymi do wyprowadzania wyników programów i danych wyjściowych z przetwarzania. Praktyka wykazała, że dla uzyskania ostatecznego układu graficznego wyprowadzanych wyników często niezbędne jest wielokrotne uruchamianie programu i kolejne korekty kształtujące wydruk. Wiąże się to z dużym zużyciem papieru. Można temu skutecznie zapobiec używając do próbnych wydruków monitora ekranowego. Możliwość taka wynika stąd, że układ graficzny wyprowadzanych wyników na monitorze ekranowym jest identyczny jak na drukarce (przy ograniczeniu szerokości strony do 80 znaków). W transmisji danych z mikrokomputera do urządzenia wyjściowego pośredniczy układ dopasowujący. W czasie wyprowadzania danych przez mikrokomputer układ dopasowujący wysyła je równocześnie do obu urządzeń. Zablokowanie wydruku na papier uzyskuje się przez oddzielenie drukarki, co pozwala na próbne wydruki na monitorze ekranowym.

Stworzona konfiguracja daje wiele możliwości:

— stanowi automatyczną maszynę do pisania (korzystanie z pełnego zestawu polskich liter jest możliwe po zastosowaniu dodatkowej pamięci EPROM w drukarkach); możliwe jest przechowywanie wielu różnych tekstów na dyskietkach w celu późniejszego ich wykorzystania,
— umożliwia wyprowadzanie wyników programów użytkowych na drukarkę lub monitor ekranowy w zależności od potrzeb,
— pozwala na edycję tekstów z mozliwością włączania rysunków.

Istnieje jeszcze wiele innych możliwości zastosowania przedstawionego zestawu w wojsku np. do:

— rozwiązywania wielu zagadnień i problemów finansowych, ewidencyjnych i kadrowych,
— sporządzania list, planów, wykresów i schematów,
— pisania i redagowania pism, meldunków i rozkazów z drukowaniem ich w dowolnej ilości egzemplarzy.

Istnieje jeszcze wiele innych możliwości, szczególnie po rozszerzeniu oprogramowania mikrokomputera o programy graficzne. W wypadku posiadania przez użytkownika tylko jednego z podłączonych urządzeń zewnętrznych (drukarka lub monitor) można zbudować zestaw w oparciu o mikrokomputer i posiadane urządzenie wg przedstawionego opisu.

Zbigniew FIGUŁA
Artur KOŁOSOWSKI

```
PAGE 1

               0101          .OPT LIST
               0102 ********************************
               0103 *                              *
               0104 *      D-100  INTERFACE        *
               0105 *                              *
               0106 ********************************
               0107 ;
               0108 ;
               0109 ; DEKLARACJA STALYCH
               0110 ;
=0600          0111 START   =     $0600
=000C          0112 DOSINI  =     $0C
=031B          0113 PRTVEC  =     $031B
=D302          0114 PACTL   =     $D302
=D300          0115 PORTA   =     $D300
=D40E          0116 NMIEN   =     $D40E
=D400          0117 DMACTL  =     $D400
```

```
        =D01F        0118 CONSOL  =    $D01F
        =009B        0119 EOL     =    $9B
        =000D        0120 CR      =    $0D
        =000A        0121 LF      =    [illegible]
        =0083        0122 K       =    131
                     0123 ;
                     0124 ;*****************************
                     0125 ;
0000                 0126         *=   START
                     0127 ;
                     0128 ; TABLICA OPERACJI DRUKARKI
                     0129 ;
0600                 0130 HANDLTAB
0600 0F06            0131         .WORD OPEN-1
0602 2606            0132         .WORD CLOSE-1
0604 2906            0133         .WORD GETBYTE-1
0606 2C06            0134         .WORD PUTBYTE-1
0608 2606            0135         .WORD STATUS-1
060A 2906            [illegible]  .WORD SPECJAL-1
060C 00000000        [illegible]  .BYTE 0,0,0,0
                     [illegible]38 ;
                     [illegible]39 ;*****************************
                     0140 ;
0610                 0141 OPEN
0610 A930            0142         LDA  [illegible]
0612 8D02D3          0143         STA  PACTL
0615 A901            0144         LDA  #$01
0617 8D00D3          0145         STA  PORTA
061A A934            0146         LDA  #$34
061C 8D02D3          0147         STA  PACTL
061F A901            0148         LDA  #$01
0621 8D00D3          0149         STA  PORTA
0624 20CC06          0150         JSR  PRWAIT
                     0151 ;
0627 A001            0152 SUCCES  LDY  #1
0629 60              0153         RTS
                     0154 ;
062A A092            0155 NOTIMPL LDY  #146
062C 60              0156         RTS
                     0157 ;
        =0627        0158 CLOSE   =    SUCCES
        =0627        0159 STATUS  =    SUCCES
        =062A        0160 GETBYTE =    NOTIMPL
        =062A        0161 SPECJAL =    NOTIMPL
                     0162 ;
                     0163 ;*****************************
                     0164 ;
                     0165 ; ROZKAZ PUTBYTE
                     0166 ;
                     0167 ; BAJT DO WYSLANIA POWINIEN
                     0168 ; ZNAJDOWAC SIE W AKUMULATORZE.
                     0169 ;
                     0170 ;*****************************
                     0171 ;
062D                 0172 PUTBYTE
062D 48              0173         PHA
062E C90D            0174         CMP  #CR
0630 F004            0175         BEQ  NL
0632 C99B            0176         CMP  #EOL
0634 D031            0177         BNE  NOEOL
                     0178 ;
0636 A90D            0179 NL      LDA  #CR
0638 206E06          0180         JSR  SEROUT
063B A90A            0181         LDA  #LF
063D 206E06          0182         JSR  SEROUT
                     0183 ;
                     0184 ; SPR. KLAW. FUNKC.
                     0185 ;
0640 ADFB06          0186         LDA  PAM
0643 D00F            0187         [illegible]  NOPAM
0645 AE1FD0          0188         LDX  CONSOL
0648 E005            0189         CPX  #5
064A D01E            0190         BNE  OUT
064C A901            0191         LDA  #1
064[illegible] 8DFB06          0192         STA  PAM
065[illegible] 4C6A06          0193         JMP  OUT
0654 AE1FD0          0194 NOPAM   LDX  CONSOL
0657 E007            0195         CPX  #7
0659 F0F9            0196         BEQ  NOPAM
065B E005            0197         CPX  #5
065D F00B            0198         BEQ  OUT
065F A900            0199         LDA  #0
0661 8DFB06          0200         STA  PAM
0664 4C6A06          0201         JMP  OUT
0667 206F06          0202 NOEOL   JSR  SEROUT
066A 68              0203 OUT     PLA
066B A001            0204         LDY  #1
066D 60              0205         RTS
                     0206 ;
                     0207 ;*****************************
                     0208 ;
                     0209 ;
066E                 0210 SEROUT
066E 8DF806          0211         STA  BUFFER
                     0212 ;
                     0213 ; WYL. PRZERWAN
                     0214 ;
0671 78              0215         SEI
0672 A900            0216         LDA  #0
0674 8D0ED4          0217         STA  NMIEN
0677 8D00D4          0218         STA  DMACTL
067A 8DFA06          0219         STA  BP          ;ZEROWANIE BP
067D A908            0220         LDA  #8
067F 8DF906          0221         STA  COUNT       ;LICZNIK BITOW =8
                     0222 ;
                     0223 ; SPR.CZY WOLNA DRUK.
                     0224 ;
0682 AD00D3          0225 BUFZ    LDA  PORTA
0685 6A              0226         ROR  A
0686 6A              0227         ROR  A
0687 9006            0228         BCC  BUFW
0689 20CC06          0229         JSR  PRWAIT
068C 4C8206          0230         JMP  BUFZ
                     0231 ;
                     0232 ; BIT STARTU NA WYJSCIE
                     0233 ;
068F A900            0234 BUFW    LDA  #0
0691 8D00D3          0235         STA  PORTA
0694 20C606          0236         JSR  BITWAIT
                     0237 ;
                     0238 ; BITY INF 0-7 NA WYJSCIE
                     0239 ;
0697 ADF806          0240 SENDBITE LDA BUFFER
069A 8D00D3          0241         STA  PORTA
069D 6A              0242         ROR  A
069E 8DF806          0243         STA  BUFFER
06A1 20C606          0244         JSR  BITWAIT
06A4 9003            0245         BCC  BIT0
06A6 EEFA06          0246         INC  BP
06A9 CEF906          0247 BIT0    DEC  COUNT
06AC D0E9            0248         BNE  SENDBITE
                     0249 ;
                     0250 ; BIT PARZYST. NA WYJSCIE
                     0251 ;
06AE ADFA06          0252         LDA  BP
06B1 8D00D3          0253         STA  PORTA
06B4 20C606          0254         JSR  BITWAIT
                     0255 ;
                     0256 ; BITY STOP NA WYJSCIE
                     0257 ;
06B7 A901            0258         LDA  #1
06B9 8D00D3          0259         STA  PORTA
06BC 20CC06          0260         JSR  PRWAIT
                     0261 ;
                     0262 ; WLACZENIE PRZERWAN
                     0263 ;
06BF A9FF            0264         LDA  #$FF
06C1 8D0ED4          0265         STA  NMIEN
06C4 58              0266         CLI
06C5 60              0267         RTS
                     0268 ;
                     0269 ;
                     0270 ;*****************************
                     0271 ;
                     0272 ;
                     0273 ; PODPROGRAM OPOZNIAJACY
                     0274 ; O CZAS  1 BITU.
                     0275 ;
06C6 A283            0276 BITWAIT LDX  #K
06C8 CA              0277 LOOPK   DEX
06C9 D0FD            0278         BNE  LOOPK
06CB 60              0279         RTS
                     0280 ;
                     0281 ;
                     0282 ; PODPROGRAM OPOZNIAJACY
                     0283 ; O CZAS 4 BITOW.
                     0284 ;
06CC A004            0285 PRWAIT  LDY  #4
06CE 20C606          0286 LOOPF   JSR  BITWAIT
06D1 88              0287         DEY
06D2 D0FA            0288         BNE  LOOPF
06D4 60              0289         RTS
                     0290 ;
                     0291 ;*****************************
                     0292 ;
                     0293 ;
                     0294 ; PODPROGRAM DO AUTOMATYCZNEJ
                     0295 ; AKTUALIZACJI ADRESU STEROWNIKA
                     0296 ; DRUKARKI W TABLICY URZADZEN PO
                     0297 ; "SYSTEM RESET".
                     0298 ;
                     0299 ;
06D5                 0300 INITDOS
06D5 20E206          0301         JSR  TROJAN
                     0302 ;
[illegible]          [illegible] INITMAN
06D8 A900            [illegible]  LDA  # <HANDLTAB
06DA 8D1B03          0305         STA  PRTVEC
06DD A906            [illegible]  LDA  # >HANDLTAB
06DF 8D1C03          0307         STA  PRTVEC+1
06E2                 [illegible] TROJAN
06E2 60              0309         RTS
                     0310 ;
06E3                 0311 INSTALLDOSINI
06E3 A50C            0312         LDA  DOSINI
06E5 8DD606          0313         STA  INITDOS+1
06E8 A50D            0314         LDA  DOSINI+1
06EA 8DD706          0315         STA  INITDOS+2
06ED A9D5            0316         LDA  # <INITDOS
06EF 850C            0317         STA  DOSINI
06F1 A906            0318         LDA  # >INITDOS
06F3 850D            0319         STA  DOSINI+1
```

```
06F5 4CD806     0320          JMP  INITHAN
                0321 ;
                0322 ;********************************
                0323 ;
                0324 ; TABLICA ZMIENNYCH
                0325 ;
06F8 00         0326 BUFFER  .BYTE 0
06F9 00         0327 COUNT   .BYTE 0
06FA 00         0328 BP      .BYTE 0
06FB 00         0329 PAM     .BYTE 0
                0330 ;
                0331 ;********************************
                0332 ;
                0333 ; ADRES STARTU ( DOS LOAD&RUN)
                0334 ;
06FC            0335         *=   $02E2
02E2 E306       0336         .WORD INSTALLDOSINI
                0337 ;
                0338 ;********************************
                0339 ;

*** ASSEMBLY ERRORS: 0   17265 BYTES FREE
```

Lilavati (12)

ROZLEWANIE WINA

DANUTA KWASIŻUR, MIECZYSŁAW SKONIECZNY

W rozdziale 1 zatytułowanym „Anegdoty matematyczne i zadania anegdotyczne" znajdujemy zadanie nr 48 o rozlewaniu wina. Mikrokomputerowe rozwiązanie tego problemu przedstawiamy w dzisiejszym odcinku.

Po uruchomieniu programu użytkownik zapoznaje się z treścią zadania, która jest wyświetlona na ekranie monitora w następującej postaci:

```
Dwaj przyjaciele sprowadzili wspólnie
ośmiolitrową beczułkę wina. Ten, który
wino sprowadził, ma zatrzymać beczułkę
z połową jej zawartości. Drugi jednak ma
tylko dwie beczułki, w których stale wi-
no przechowuje:
 - jedna o pojemności 5 litrów
 - druga o pojemności 3 litrów.

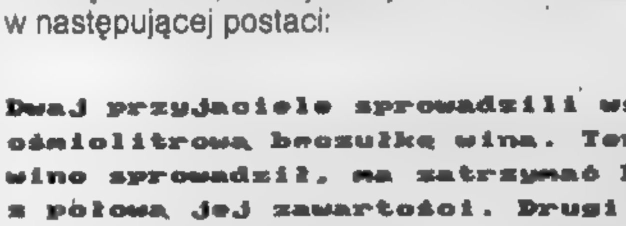
W jaki sposób można dokonać podziału po-
sługując się tymi beczułkami?

          [illegible]
```

Po naciśnięciu dowolnego klawisza wyświetla się plansza gry i program oczekuje na wprowadzenie danych w następującej kolejności:

— nr beczułki, z której przelewamy wino,
— nr beczułki, do której przelewamy wino,
— ilość litrów przelewanego wina.

Poniższy wydruk pokazuje sytuacje po 1 przelaniu wina:



```
1 PRZELEWANIE WINA
1. Beczulka 8-litrowa.
2. Beczulka 5-litrowa.
3. Beczulka 3-litrowa.
```

```
PRZELEWAM WINO Z BECZULKI NR  1
DO BECZULKI NR
ILE LITROW WINA?
```

Sukces w grze jest sygnalizowany komunikatem:
„BRAWO!!! WINO PRAWIDŁOWO PRZELANE
PO 7 PRZELANIACH
SPRÓBUJESZ JESZCZE RAZ? (T/N):"
teraz można ponownie rozpocząć grę. Wersja źródłowa programu opracowanego w BASIC-u jest następująca:

```
10 '
20 '     Ekran informacyjny
30 '
40 GOSUB 1030
50 MODE 1:INK 0,26:INK 1,0:INK 2,6:INK 3,9
60 p$="     ":p1$=STRING$(5,CHR$(143))
70 PAPER 0:BORDER 26:CLS
80 PEN 1:PRINT "Dwaj przyjaciele sprowadzili wspolnie    "
90 PRINT "osmiolitrowa beczulke wina. Ten, ktory  "
100 PRINT"wino sprowadzil, ma zatrzymac beczulke   "
110 PRINT"z polowa jej zawartosci. Drugi jednak ma "
120 PRINT"tylko dwie beczulki, w ktorych stale wi- "
130 PRINT"no przechowuje:                          "
140 PRINT " - jedna o pojemnosci 5 litrow          "
150 PRINT" - druga o pojemnosci 3 litrow.          "
160 PRINT:PRINT "W jaki sposob mozna dokonac podzialu po- "
170 PRINT"slugujac sie tymi beczulkami?"
180 LOCATE 6,25:PRINT"NACISNIJ DOWOLNY KLAWISZ"
190 IF INKEY$="" THEN GOTO 190
200 '
210 ' Rysowanie planszy gry
220 '
230 MODE 1
240 wsp(1)=2:wsp(2)=5:wsp(3)=7
250 FOR a=1 TO 3
[illegible]
270 NEXT a
280 FOR K=2 TO 9:LOCATE 5,K:PRINT CHR$(138);:PEN 2:PRINT STRING$(5,CHR$(143));:P
EN 1:PRINT CHR$(133):NEXT K
290 GRAPHICS PEN 3:PLOT 48,400:DRAWR 0,-146:PLOT 50,400:DRAWR 0,-146
300 FOR k=0 TO 8:PLOT 46,382-k*16:DRAWR -4,0:NEXT k
310 GRAPHICS PEN 1:TAG:FOR k=8 TO 1 STEP -1:MOVE 8,276+(k-1)*16:PRINT USING "£";
k;:PRINT "1";:NEXT k
320 MOVE 24,260:PRINT "0";:TAGOFF
330 WINDOW £1,5,30,12,18:PAPER £1,3:CLS £1
340 lp=1:b(1)=8:b(2)=0:b(3)=0
350 ILW(1)=8:ILW(2)=5:ILW(3)=3
360 PEN £1,1:LOCATE £1,2,2:PRINT £1,USING "£";lp;:PRINT £1," PRZELEWANIE WINA"
370 PEN £1,0:LOCATE £1,2,4:PRINT £1,"1. Beczulka 8-litrowa.":PRINT £1," 2. Beczu
lka 5-litrowa.":PRINT £1," 3. Beczulka 3-litrowa."
380 WINDOW £2,2,35,20,21:PAPER £2,1:CLS £2
```

```
390 WINDOW £3,2,35,22,22:PAPER £3,1:CLS £3
400 WINDOW £4,2,35,23,25:PAPER £4,1:CLS £4:PEN £4,0
410 '
420 '   Segment sterujacy
430 '
440 CLS £2:PEN £2,0:LOCATE £2,2,2:INPUT £2,"PRZELEWAM WINO Z BECZULKI NR  ",NR
450 IF NR=1 OR NR=2 OR NR=3 THEN GOTO 470
460 GOTO 410
470 IF B(NR)>0 THEN GOTO 490
480 GOSUB 960:GOTO 410
490 CLS £3:PEN £3,0:LOCATE £3,2,1:INPUT £3,"DO BECZULKI NR
500 IF NR=NR1 THEN GOTO 490
510 IF b(nr1)=ilw(nr1) THEN GOSUB 1000:GOTO 490
520 IF NR1=1 OR NR1=2 OR NR1=3 THEN GOTO 540
530 GOTO 490
540 CLS £4:PEN £4,0:LOCATE £4,2,1:INPUT £4,"ILE LITROW WINA?
550 IF LW<0 OR LW>5 THEN GOSUB 990:GOTO 540
560 IF B(NR)-LW=0 OR LW+B(NR1)=ILW(NR1) THEN GOTO 580
570 GOSUB 1010:GOTO 540
580 IF B(NR)>=LW THEN GOTO 600
590 GOSUB 970:GOTO 540
600 IF B(NR1)+LW<=ILW(NR1) THEN GOTO 620
610 GOSUB 980:GOTO 540
620 FOR K=1 TO LW
630 LOCATE 6+(NR-1)*8,9-b(nr)+K:PRINT p$
640 NEXT k
650 PEN 2
660 FOR K=1 TO LW
670 LOCATE 6+(NR1-1)*8,10-b(nr1)-K:PRINT p1$
680 NEXT k
690 b(nr)=b(nr)-lw:b(nr1)=b(nr1)+lw
700 lp=lp+1
710 IF b(1)=4 AND b(3)+b(2)=4 THEN GOTO 740
720 PEN £1,1:LOCATE £1,1,2:PRINT £1,USING "££";lp
730 CLS £2:CLS £3:CLS £4:GOTO 410
740 CLS £2:CLS £3:CLS £4
750 LP=LP-1
760 IF LP<8 THEN GOTO 830
770 PRINT £2:PRINT £2," WINO PRAWIDLOWO PRZELANE":PRINT £3," PO";LP;" PRZELEWANI
ACH. (MIN. 7)."
780 FOR A=0 TO 26:BORDER A:FOR T=1 TO 200:NEXT T:NEXT A
790 PEN £4,3:PRINT £4:INPUT £4," SPROBUJESZ JESZCZE RAZ? (T/N) ",T$
800 IF T$="T" OR T$="t" THEN GOTO 230
810 IF T$="N" OR T$="n" THEN END
820 GOTO 790
830 PRINT £2:PRINT £2," BRAWO!!! WINO PRAWIDLOWO":PRINT £3," PRZELANE PO";LP;" P
RZELEWANIACH.":GOTO [illegible]
```

W głównej części programu wykorzystuje się 3 podprogramy, które realizują następujące funkcje:

1. Podprogram rysowania dowolnej beczki.
(instrukcje 840—920)

```
840 '
850 '  Podprogram rysowania dowolnej beczki
860 '
870 PEN 1:LOCATE 5+(a-1)*8,wsp(a)-1:PRINT CHR$(169);SPC(5);CHR$(170)
880 FOR K=wsp(a) TO 9
890 LOCATE 5+(a-1)*8,K:PRINT CHR$(138);STRING$(5," ");CHR$(133)
900 NEXT k
910 LOCATE 5+(a-1)*8,10:PRINT CHR$(167);STRING$(5,CHR$(131));CHR$(168)
920 RETURN
```

2. Podprogramy z komunikatami o błędach.
(instrukcje 930—1020)

```
930 '
940 '  Podprogramy z komunikatami o bledach
950 '
960 CLS £2:PEN £2,2:LOCATE £2,2,2:PRINT £2,"W TEJ BECZCE NIE MA WINA":
GOTO 1020
970 CLS £4:PEN £4,2:LOCATE £4,2,1:PRINT £4,"W BECZCE NR ";USING "£";NR
;:PRINT £4," NIE MA TYLE WINA":GOTO 1020
980 CLS £4:PEN £4,2:LOCATE £4,2,1:PRINT £4,"W BECZCE NR ";USING "£";NR
1;:PRINT £4," NIE ZMIESCI SIE":PRINT £4," TYLE WINA!!":GOTO 10
20
990 CLS £4:PEN £4,2:LOCATE £4,2,1:PRINT £4,"NIE MOZNA PRZELEWAC TYLE W
INA":GOTO 1020
1000 CLS £3:PEN £3,2:LOCATE £3,2,1:PRINT £3,"TA BECZKA JEST JUZ PELNA"
:GOTO 1020
1010 CLS £4:PEN £4,2:LOCATE £4,2,1:PRINT £4,"NIE MOZESZ ODMIERZYC TAKI
EJ":PRINT £4," ILOSCI WINA!!!":GOTO 1020
1020 SOUND 1,300:SOUND 1,200:FOR T=1 TO 3000:NEXT T:RETURN
```

3. Podprogram definiowania polskich liter i symboli własnych.
(instrukcja 1030—1400)

```
1030 '
1040 '  Podrogram definiowania polskich liter i symboli wlasnych '
1050 '
1060 SYMBOL AFTER 167
1070 SYMBOL 167,7,7,3
1080 SYMBOL 168,224,224,192
1090 SYMBOL 169,0,0,0,0,0,3,7,7
1100 SYMBOL 170,0,0,0,0,0,192,224,224
1110 SYMBOL 171,0,0,120,12,124,204,118,3
1120 SYMBOL 172,24,60,102,102,126,102,102,3
1130 SYMBOL 173,24,0,60,102,96,102,60,0
1140 SYMBOL 174,24,60,102,192,192,102,60,0
1150 SYMBOL 175,0,0,60,102,126,96,60,6
1160 SYMBOL 176,254,98,104,120,104,98,254,3
1170 SYMBOL 177,56,24,28,24,56,24,60,0
1180 SYMBOL 178,240,96,112,96,226,102,254,0
1190 SYMBOL 179,24,0,220,102,102,102,102,0
1200 SYMBOL 180,24,198,230,246,222,206,198,0
1210 SYMBOL 181,24,0,60,102,102,102,60,0
1220 SYMBOL 182,12,56,108,198,198,108,56,0
1230 SYMBOL 183,24,0,60,96,60,6,124,0
1240 SYMBOL 184,24,60,96,60,6,102,60,0
1250 SYMBOL 185,24,0,126,76,24,48,126,0
1260 SYMBOL 186,254,198,140,126,48,102,254,0
1270 SYMBOL 187,12,24,126,76,24,48,126,0
1280 SYMBOL 188,48,254,172,24,50,102,254,0
1290 SYMBOL 189,0,0,0,0,0,12,12,24
1300 KEY DEF 10,1,55,171,172
1310 KEY DEF 11,1,56,173,174
1320 KEY DEF 3,1,57,175,176
1330 KEY DEF 20,1,52,177,178
1340 KEY DEF 12,1,53,179,180
1350 KEY DEF 4,1,54,181,182
1360 KEY DEF 13,1,49,183,184
1370 KEY DEF 14,1,50,185,186
1380 KEY DEF 5,1,51,187,188
1390 KEY DEF 15,1,189
1400 RETURN
```

DANUTA KWASIŻUR
MIECZYSŁAW SKONIECZNY

TRZY WYMIARY

W programie pokazano prosty sposób kontynuowania obrazu trójwymiarowego na bazie zobrazowania dwuwymiarowego. Zdefiniowane na płaszczyźnie figury program traktuje jako podstawy graniastosłupów, które następnie zobrazowuje w postaci trójwymiarowej z ustalonymi wcześniej wysokościami. Graniastosłupy są przedstawione na wspólnej płaszczyźnie (o elewacji zerowej).

Podczas wprowadzania wysokości poszczególnych figur należy zwrócić uwagę na fakt, że na ekranie będą całkowicie widoczne tylko te, których wysokość jest mniejsza niż ilość jednostek na osi y. Wysokości graniastosłupów, które nie zmieściły się na ekranie można zmienić podczas edycji obrazu. Omawiane zależności można rozpoznać doświadczalnie.

Program, oprócz zmiany wysokości, umożliwia modyfikację obrazu, poprzez zmianę kąta obserwacji (perspektywy) i dorysowywanie nowych figur.

Definiowanie figur realizowane jest poprzez wprowadzanie współrzędnych (x, y) kolejnych punktów, które wyznaczają odcinki (boki figury). Współrzędne wprowadzane są w jednostkach umownych rysunku, które ustala się na początku działania programu. Zdefiniowanie jednostek powoduje wyświetlenie osi wraz z podziałką, co ułatwia orientację figur na ekranie. **Zakończenie definiowania pojedynczej figury dokonuje się poprzez wprowadzenie punktu o współrzędnej** $x = 998$ oraz dowolnej wartości współrzędnej y.

Poniżej zamieszczamy przebieg i obsługę operatorską programu:

Komunikat	Operacja
1. *Podaj ilość jednostek wsp. y — max. 30*	*Wprowadzić ilość jednostek, maksymalnie 30, która ma być odmierzona na osi y.*
2. *Kąt obserwacji*	*Wprowadź w stopniach kąt obserwacji (perspektywę).*
3. *Wysokość figury*	*Wprowadź, w jednostkach wysokość definiowanej figury.*
4. *Pierwszy punkt*	*Wprowadź współrzędne (x, y) pierwszego punktu definiowanej figury.*
5. *Następny punkt*	*Wprowadź współrzędne (x, y) kolejnego punktu definiowanej figury lub współrzędne kończące definiowanie figury.*
6. *Koniec rysowania*	*Wprowadzenie litery: „t" — zobrazowanie trójwymiarowe, „n" — powrót do komunikatu 3.*
7. *Dorysowywanie figur (t/n)*	*Wprowadzenie litery: „t" — powrót do zobrazowania dwuwymiarowego i komunikatu 3.*
8. *Edycja (t/n)*	*Wprowadzenie litery: „t" — następny komunikat, „N" — komunikat 12.*
9. *Kąt obserwacji*	*Wprowadź kąt obserwacji w stopniach.*
10. *Podaj numer figury*	*Wprowadź numer figury wynikający z kolejności definiowania. Numer większy niż liczba zdefiniowanych figur powoduje wyświetlenie zmodyfikowanego zobrazowania trójwymiarowego i powrót do komunikatu 7.*
11. *Nowa wysokość*	*Wprowadź wysokość w jednostkach rysunku. Powrót programu do komunikatu 10.*
12. *Początek programu (t/n)*	*Wprowadzenie litery: „t" — realizacja programu od początku, „N" — zakończenie działania programu.*

Zenon MAJCHRZAK

```
10 REM*******************************************
20 REM*                                         *
30 REM*            GRANIASTOSLUPY               *
40 REM*                                         *
50 REM*******************************************
60 DIM wsp(501,2),nwsp(501,2,2),opis(50,3)
70 MODE 1:INK 0,6:INK 1,13:INK 2,1:INK 3,24
80 BORDER 0:PEN 1:PAPER 2
90 CLS
100 GRAPHICS PAPER 1:REM Tlo grafiki
110 ORIGIN 0,16
120 CLEAR INPUT
130 WINDOW 1,40,25.25
140 ikz=0
150 INPUT"Podaj ilosc jednostek wsp. y-max.30";skal
160 IF skal>30 THEN GOTO 150
170 skal=384/skal
180 GOSUB 1180:REM Rysowanie wspolrzednych
190 i1=1
200 j=0
210 INPUT"Kat obserwacji";ak
220 ak=ak*PI/180
230 INPUT"Wysokosc figury";z
240 z=z*skal
250 opis(i1,1)=z
260 INPUT"Pierwszy punkt";x,y
270 GOTO 300
280 INPUT"Nastepny punkt";x,y
290 IF x>998 THEN GOTO 360
300 j=j+1
310 IF j>1000 THEN GOTO 360
320 wsp(j,1)=x*skal:wsp(j,2)=y*skal
330 GOSUB 1060:REM Rysuj odcinek
340 ikz=1
350 GOTO 280
360 ikz=0
370 opis(i1,3)=j
380 INPUT"Koniec rysowania (t/n)";nk$
390 IF nk$="t" THEN GOTO 440
400 IF nk$="n" THEN GOTO 420
410 GOTO 380
420 i1=i1+1
430 GOTO 230
440 CLG 1
450 nf=1
460 FOR kkk=1 TO i1
470 z=opis(kkk,1)
480 i=opis(kkk,3)
490 cak=COS(ak)
500 sak=SIN(ak)
510 FOR k=nf TO i
520 nwsp(k,1,1)=wsp(k,1)+wsp(k,2)*cak
530 nwsp(k,2,1)=wsp(k,2)*sak
540 nwsp(k,1,2)=nwsp(k,1,1)
550 nwsp(k,2,2)=z+nwsp(k,2,1)
560 NEXT k
570 FOR l=1 TO 2
580 FOR k=nf TO i-1
590 x=nwsp(k,l,1)
600 y=nwsp(k,2,l)
610 x1=nwsp(k+1,l,l)
620 y1=nwsp(k+1,2,l)
630 GOSUB 1110
640 NEXT k
650 NEXT l
660 FOR k=nf TO i
670 x=nwsp(k,1,1)
680 y=nwsp(k,2,1)
690 x1=nwsp(k,1,2)
700 y1=nwsp(k,2,2)
710 GOSUB 1110
720 NEXT k
730 nf=i+1
740 NEXT kkk
750 INPUT"Dorysowywanie figur (t/n)";nk$
760 IF nk$="t" THEN GOTO 790
770 IF nk$="n" THEN GOTO 830
780 GOTO 750
790 GOSUB 1180
800 GOSUB 1400
810 i1=i1+1
820 GOTO 230
830 INPUT"Edycja (t/n)";nk$
840 IF nk$="t" THEN GOTO 910
850 IF nk$="n" THEN GOTO 870
860 GOTO 830
870 INPUT"Poczatek programu (t/n)";nk$
880 IF nk$="t" THEN GOTO 70
890 IF nk$="n" THEN GOTO 1530
900 GOTO 870
910 INPUT"Kat obserwacji";ak
920 ak=ak*PI/180
930 INPUT"Podaj nr figury";nr
940 IF opis(nr,3)=0 THEN GOTO 440
950 INPUT"Nowa wysokosc";z
960 opis(nr,1)=z*skal
970 GOTO 930
980 REM *****************************************
990 REM *                                       *
1000 REM*             PODPROGRAMY               *
1010 REM*                                       *
1020 REM*****************************************
1030 REM
1040 REM Rysowanie definiowanego odcinka
1050 REM
1060 GRAPHICS PEN 3:REM Stare wspolrzedne
1070 IF ikz=0 THEN PLOT wsp(j,1),wsp(j,2)
```

```
1080 DRAW wsp(j,1),wsp(j,2)
1090 RETURN
1100 REM
1110 REM Rysowanie krawedzi graniastoslupa
1120 REM
1130 GRAPHICS PEN 0:REM Nowe wspolrzedne
1140 PLOT x,y
1150 DRAW x1,y1
1160 RETURN
1170 REM
1180 REM Rysowanie ukladu wspolrzednych
1190 REM
1200 CLG 1
1210 GRAPHICS PEN 3:REM pioro wspolrzednych
1220 PLOT 0,400
1230 DRAW 0,0
1240 DRAW 640,0
1250 wy=400/skal
1260 wx=640/skal
1270 sk=skal
1280 FOR ii=1 TO wy:REM Podzialka y
1290 PLOT 0,sk
1300 DRAW 4,sk
1310 sk=sk+skal
1320 NEXT ii
1330 sk=skal
1340 FOR ii=1 TO wx:REM Podzialka x
1350 PLOT sk,0
1360 DRAW sk,4
1370 sk=sk+skal
1380 NEXT ii
1390 RETURN
1400 REM
1410 REM rysowanie pierwotnego obrazu
1420 REM
1430 lx=1
1440 FOR kk=1 TO il
1450 kl=opis(kk,3):REM Ilosc punktow figury
1460 PLOT wsp(lx,1),wsp(lx,2)
1470 FOR px=lx TO kl-1
1480 DRAW wsp(px+1,1),wsp(px+1,2)
1490 NEXT px
1500 lx=kl+1
1510 NEXT kk
1520 RETURN
1521 REM
1522 REM  koniec
1523 REM
1530 WINDOW 1,40,1,25
1540 CLS
1550 CLG 1
1560 END
```

```
1 REM ********************
2 REM *                  *
3 REM *  BOMBARDOWANIE   *
4 REM *      ZX-81       *
5 REM *                  *
6 REM ********************
7 LET E=0
8 LET F=0
9 LET C=RND*15+5
10 LET D=RND*8+20
15 PRINT AT 19,C;"Ō"
20 PRINT AT 19,D;"Ō"
25 INPUT A$
30 FOR A=0 TO 27
32 PRINT AT 0,A-1;"      ";
TAB A+1;" "
35 PRINT AT 0,A;"▄▄▖ ";TA
B A+2;"▄"
40 IF INKEY$<>"" THEN GOSU
B 100
50 NEXT A
100 FOR B=2 TO 19
105 PRINT AT B-1,A+3;" "
110 PRINT AT B,A+3;"."
115 IF B=19 THEN PRINT AT
B,A+3;"*"
120 NEXT B
130 RETURN
```

J. J.

Procedura GAUSS

KRZYSZTOF POŹNIAK

```
procedura rozwiazuje uklad rownan liniowych n-tego rzedu.
naglowek funkcji: int gauss(n,a,b,x)
   gdzie: n-stopien rownania:                    int n;
          a-tablica wspolczynnikow:              float a[n][n];
          b-tablica wyrazow wolnych:             float b[n];
          x-tablica wektora niewiadomych:        float x[n];
*/

#include <stdlib.h>

void main() /*program testowy*/
   #define stopien 40 /*przykladowy stopien maksymalny*/
{
   int n; /*przykladowy stopien rzeczywisty*/
   int y,x1;
   double a[stopien][stopien],b[stopien],x[stopien],zmpom;

   do
   {
      do
      {
         printf("podaj stopien ukladu <1,%d>: n=",stopien);
         scanf("%d",&n);
         printf("\n");
      }
      while(n<1||n>40);

      /*tworzenie tablicy a[j][j], przy pomocy
           generatora liczb pseudolosowych*/

      for(y=0;y<n;y++)
      {
         zmpom=0.0;
         for(x1=0;x1<n;x1++)
         {
            a[y][x1]=random(100);
            zmpom=zmpom+a[y][x1]*(1+x1);
         }
         b[y]=zmpom;
      }
      printf("rozwiazywanie\n");
      if(gauss(n,a,b,x))
      {
         printf("uklad rozwiazany:\n");
         for(y=0;y<n;)
            printf("x[%d]=%f\n",++y,x[y]);
      }
      else
         printf("brak rozwiazan\n");
      printf("czy dalej testujesz? (t/n)\n");
   }
   while(getch()=='t');
}

gauss(n,a,b,w) /*procedura rozwiazywania ukladu
                 n-rownan liniowych metoda
                 eliminacji Gaussa                  */

int n;
double a[stopien][stopien],b[stopien],w[stopien];

int k,x,y,wspx,wspy,przes[stopien];
double zmpom;

for(y=0;y<n;y++)
   przes[y]=y;
for(k=0;k<n;k++)
{
   zmpom=0.0;
   for(y=k;y<n;y++)
      for(x=k;x<n;x++)
      {
```

```
        if(abs(a[y][x])>zmpom)
        {
            zmpom=abs(a[y][x]);
            wspy=y;
            wspx=x;
        }
    }
    if(zmpom==0.0)
        return(0);
    else
        for (x=k;x<n;x++)
        {
            zmpom=a[k][x];
            a[k][x]=a[wspy][x];
            a[wspy][x]=zmpom;
        }
        zmpom=b[k];
        b[k]=b[wspy];
        b[wspy]=zmpom;
        for (y=0;y<n;y++)
        {
            zmpom=a[y][k];
            a[y][k]=a[y][wspx];
            a[y][wspx]=zmpom;
        }
        zmpom=przes[k];
        przes[k]=przes[wspx];
        przes[wspx]=zmpom;
        for (y=k+1;y<n;y++)
        {
            zmpom=a[y][k]/a[k][k];
            for (x=k+1;x<n;x++)
                a[y][x]-=a[k][x]*zmpom;
            b[y]-=b[k]*zmpom;
        }
    }
    for (y=n-1;y>=0;y--)
    {
        zmpom=0;
        for (x=y+1;x<n;x++)
            zmpom+=w[przes[x]]*a[y][x];
        w[przes[y]]=(b[y]-zmpom)/a[y][y];
    }
    return(1);
}
```



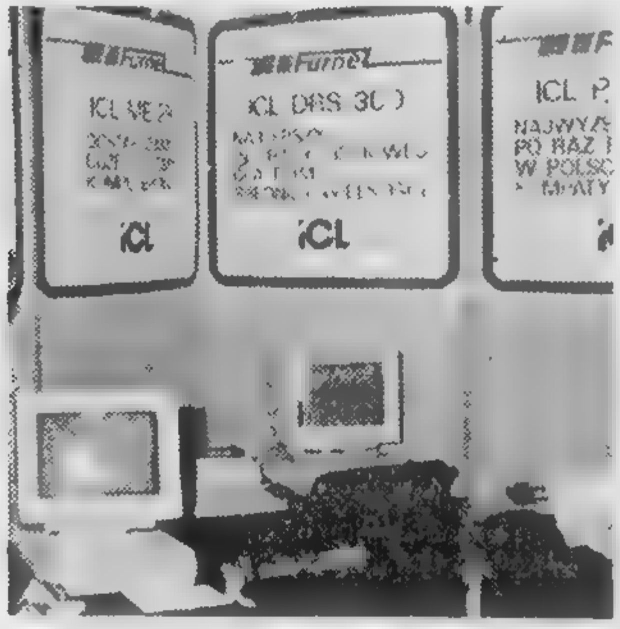


FOTO J. ZELMAN





PROFESJONALNE KOMPUTERY

(materiały do podstaw informatyki część 5)

Poznane dotychczas „cegiełki gmachu", powiedzmy „altanki" naszej wiedzy informatycznej, to już w zasadzie cały komputer. Wiemy bowiem co to jest pamięć, procesor, program, urządzenia wejścia/wyjścia. Nie są nam obce takie pojęcia jak bit, bajt i inne wymyślne słowa typu RAM, flop, ploter. Informacje te miały charakter uniwersalny, dotyczą bowiem w zasadzie wszystkich komputerów. Począwszy od domowych poprzez personalne do dużych maszyn. Wszędzie mamy do czynienia ze sprzętem (pamięci, procesory, drukarki, monitory, itp.) oraz z oprogramowaniem (algorytmy, języki, instrukcje). Sądzę, że uniwersalność owa długo będzie się broniła przed nieustannym postępem technologii. Szybkiej modyfikacji ulegają bowiem głównie parametry sprzętu informatycznego, a nie istota jego konstrukcji i użytkowania. Podobnie sprawa wygląda w kwestii oprogramowania. Nadal informacje muszą być wprowadzane do komputera; tam przetwarzane i następnie wyprowadzane w czytelnej (zazwyczaj), dla nas uzytkowników, formie.

Oczywiście zmienia się zakres i sposób wprowadzania informacji do komputerów, szybkość przetwarzania oraz sposób prezentacji wyników. Programy są coraz łatwiejsze w eksploatacji, ich użytkownikami stają się przedstawiciele niemal wszystkich zawodów, także i dzieci. Maszyny wspomagają pracę księgowego, „tworzenie" gazety, samochodu i butów, lecą w kosmos i malują obrazy, bawią i uczą. Eksplozja zastosowań komputerów wydaje się nie podlegać żadnej ogólnej prawidłowości. Po prostu jak ogień pochłania wszystko wokoło. Nienaruszone pozostają tylko stare (najczęściej) twarde kamienie i inne głazy...

Jednak nadal niezmiennie komputer odbiera informacje (gdy my tego chcemy, a nie maszyna), przetwarza je i wyprowadza efekty swojej pracy. Kolejność tych czynności (WPN — Wprowadzanie, Przetwarzanie, Wyprowadzanie) jest taka sama w Spectrum, Amstradzie, Commodore, Atari, a także gdy korzystamy z mikrokomputerów klasy IBM PC.

Komputery IBM PC są najbardziej reprezentatywnym tworem informatyków ostatnich 20 lat. Im poświęcimy zatem nieco więcej uwagi. Stanowią one bowiem pewien ogólnoświatowy standard komputerów osobistych, które przeznaczone są do zastosowań zawodowych. Stosunkowo niska cena tego sprzętu, a co najważniejsze całkowita autonomia działania, stworzyły sytuację dotychczas niemożliwą do osiągnięcia — „SS" — „Święty Spokój". Oznacza to, że komputer ten możemy sobie włączyć w dowolnej chwili, a co najważniejsze, gdy jesteśmy sami (SŚ!). Nikt nie podgląda naszych (często nieudolnych) zmagań z klawiaturą, popełnionych błędów. Maszyna ma „anielską cierpliwość" niemal jak papier, który przecież nigdy nie protestuje, nawet gdy piszemy na nim brzydkie słowa. Pozornie tylko jestem daleki od tematu, pragnę bowiem zwrócić uwagę na „intymność" naszego kontaktu z komputerem. Odpada ogromna bariera, tak wielka jak z Ciechocinka do Legionowa, wstydu, nieśmiałości, obawy przed kompromitacją. Nikt nie śledzi naszej mozolnej nauki. Potem możemy śmiało powiedzieć — „komputer? Przecież to bardzo proste. Posiedziałem dwie godziny i wszystko jasne!" By popracować te kilka nocy nie potrzebowaliśmy żadnego personelu technicznego, operatorów (czytaj — świadków), zgody przełożonych — po prostu pstryk i komputer jest TWÓJ. Wszystko to stało się dzięki „PCtom" — personalnym komputerom, których parametry noszą pewne znamiona profesjonalizmu (jak na razie). Oznacza to po prostu, iż ich szybkość, pamięć, są dostatecznie duże, by w każdej szkole lub małym przedsiębiorstwie można je było efektywnie wykorzystać. Szczęśliwie się stało, że PCty stanowią pewien standard, który jak już kiedyś wspomniałem, pozwala nam „pożyczać" napisane w Meksyku programy i z powodzeniem je wykorzystywać w ciechocińskim liceum podczas zajęć z podstaw informatyki. Po prostu komputery te, podobnie jak samochody, którymi każdy (prawie każdy) może jeździć, mają od lat niezmienną konstrukcję (silnik, cztery koła, piąte przed siedzeniem kierowcy, a szóste w bagażniku). Natomiast mikrokomputer IBM PC ma płytę główną, kilka kart rozszerzających możliwości komputera, dysk twardy, flop, zasilacz, monitor i klawiaturę. Ten nadzwyczajny twór geniuszu ludzkiego powstał stosunkowo niedawno (na pewno za życia każdego z czytających te słowa) bo w 1981 roku. Początkowo traktowany nieufnie, „z rezerwą", obecnie stanowi niekwestionowany światowy standard mikrokomputera personalnego o ogromnym oprogramowaniu użytkowym (kilkadziesiąt tysięcy) — stanowić to będzie przedmiot naszego odrębnego spotkania.

O tym, że komputery IBM są nie tylko bardzo popularne, ale także nadzwyczaj opłacalne w produkcji szybko zorientowali się „ludzie interesu" na Dalekim Wschodzie. Przyjrzeli się dokładnie owej „maskotce made in USA", zakupili nieco technologii i... rozpoczęli produkcję własnych PCtów („klonów", „komputerów kompatybilnych"), które wcale nie są gorsze od, o wiele droższych, oryginałów zza Oceanu. Z zapartym tchem słuchałem wrażeń kolegi z podróży (służbowej) do Singapuru i Tajwanu, który widział tamtejszą produkcję, kontrolę jakości i „dodatki" do PCtów. Zbudowano tam takie karty rozszerzające do IBMów, których w Ameryce nigdy wcześniej nie produkowano. Przykładem mogą być karty doskonałych skanerów, układy współpracy z innymi urządzeniami, np. z instrumentami muzycznymi lub z monitorami ogromnej rozdzielczości. Ten fajerwerk pomysłów (praktycznie zrealizowanych!) ściągnął do Singapuru przedstawicieli zza Oceanu (do górnej połowy jednej ze stupiętrowych „pagód"), którzy patrzą bardzo uważnie co też tam ONI robią?

Podobno następca IBM PC — PS/2, nowe dziecko IBM'a ma być między innymi sposobem na oderwanie się od dalekowschodniej konkurencji. Wynika to z nadziei, że z PS/2, klona nie da się zrobić (?).

Jakkolwiek by było, IBM PC jest standardem i pozostanie nim w naszych warunkach zapewne jeszcze ładnych parę lat. Nie uda się nam zatem uniknąć tego tematu. Co to jest IBM PC? Pudło, dokładnie pudło wielkości małej walizki (lub jak kto woli dużej teczki) z jednym, czasami dwoma lub trzema pstrykami i parą lampek. To wszystko. Nierozerwal-

ne uzupełnienie naszego pudełka stanowi monitor (znany!) i klawiatura (też znana!). W owym opakowaniu mieszczą się wspomniane wcześniej składowe:
P — płyta główna (systemowa, „matka"),
K — karty rozszerzające,
C — dysk twardy (winchester),
A, B — dysk elastyczny (jeden lub dwa),
Z — zasilacz (130—200 W).

Poza włącznikiem zasilania pozostałe „pstryki", to czasami instalowany przełącznik szybkości pracy komputera, a dokładniej zegara, który „napędza" procesor wolniej lub szybciej (turbo), na przykład 4,77 MHz lub 8 MHz. Inny „pstryk", to także tylko czasami instalowane, zerowanie procesora. Natomiast lampki najczęściej sygnalizują stan dysku twardego (pracuje czy nie) oraz wybraną szybkość pracy komputera (turbo czy normalną).

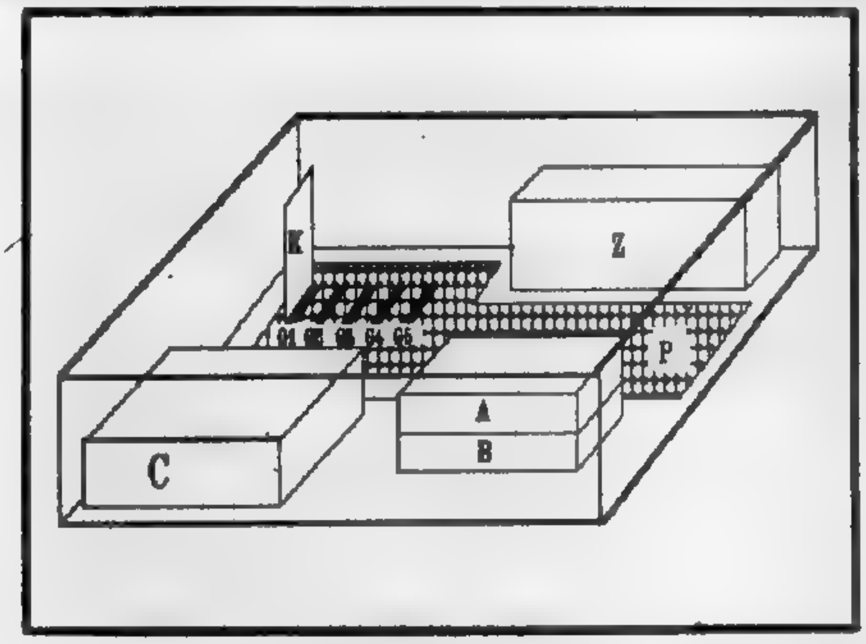
Płyta główna usytuowana jest w pozycji horyzontalnej na dole naszego pudła. Zamontowano na niej same ważne elementy: procesor (8088 dla PC XT, 80286 w PC AT), kostki pamięci: **RAM** (od 512 KB do 1 MB w komputerach XT do kilku megabajtów w AT, **ROM** — zazwyczaj z interpreterem BASIC'a oraz BIOS'em (podstawowymi programami obsługi urządzeń zewnętrznych, w PC AT BIOS pamiętany jest w wyróżnionej, zasilanej bateriami, dlaczego?, pamięci RAM), **gniazda szczelinowe** (na rysunku oznaczone są G1 — G5) — najczęściej jest ich od 5 do 8. Wkłada się do nich (pionowo) karty rozszerzające.

Na płycie głównej jest także bateryjny zegarek, który można regulować i odczytywać programowo. Zainstalowano tam także jeszcze kilka kostek dbających o współpracę z pamięciami masowymi (dyski) i urządzeniami zewnętrznymi. Jest także głośnik, który podobnie jak zegar obsługiwany jest programowo.

Wymienione elementy są w zasadzie niezmienne (jak silnik w samochodzie) i w znacznym stopniu decydują o parametrach komputera. Mówiąc o „silniku" warto zwrócić uwagę na „przyspieszenie" różnych typów — układy mikroprocesorowe (RISC). Po zamocowaniu w specjalnym gnieździe płyty głównej, powodują one znaczne zwiększenie szybkości działania komputera. Na przykład stosując NDP — 80-bitowy procesor! można nawet 100-krotnie przyspieszyć wykonywanie niektórych działań matematycznych. Należy podkreślić, iż omawiany efekt nie ma nic wspólnego z pracą „turbo", są to dwa odrębne sposoby skracania czasu wykonywania przez komputer operacji.

Pozostałe składowe IBM'a PC można w ciągu kilku minut wymienić (po prostu wyjąć i włożyć nowe, lepsze jak akumulator lub świece w samochodzie). Dotyczy to także twardego dysku, który zamocowany jest do obudowy kilkoma śrubkami, a łączy go z maszyną parę przewodów (pasm). Podobnie rzecz się ma z flopami. Przy okazji omawiania dysków, warto pamiętać, że w zasadzie „dysk twardy" oznacza sam nośnik informacji — talerze (krążki) pokryte odpowiednim materiałem. Także „dysk elastyczny", to nośnik (dyskietka), a nie urządzenie. Stąd też poprawnie należy mówić **napęd** dysku twardego/elastycznego.

Zatrzymajmy się chwilę dłużej przy kartach rozszerzających. Na każdej z nich mieści się ponad setka układów scalonych (lub mniej, są karty duże i małe), które sterują pracą napędów (są to tak zwane kontrolery), monitorów (adaptery) oraz wykonują wiele innych funkcji, na przykład komunikowanie się z otoczeniem za pośrednictwem tak zwanych **portów**. Do nich to wpływają i wypływają „towary" w postaci informacji. Jeśli port jest „wąski" — pośredniczy w przesyłaniu informacji jedna za drugą („gęsiego") jest to port szeregowy. Porty te obsługują urządzenia stosunkowo wolne, między innymi myszy (znane!). Gdy informacje muszą przepływać szybko — „osiem szeregów jednocześnie" stosowane są porty równoległe. Najczęściej dopina się do nich drukarki.



Rys. 1. Szkic budowy mikrokomputera IBM PC

Na kartach rozszerzających montowane są najróżniejsze urządzenia, które umożliwiają podłączanie kamer (skanery), mikrofonów, głośników (synteza dźwięków mowy ludzkiej), linii telefonicznej (do łączenia się z innymi komputerami — do tego rozwiązania jeszcze wrócimy). Wśród wielu innych typów kart wyróżniane są karty graficzne, które stanowią elektronikę do monitorów. Zważywszy na fakt, iż obraz na monitorze komputera jest podstawowym pośrednikiem informacji przekazywanych przez maszynę człowiekowi, karty graficzne zasługują na szczególną uwagę. Od nich zależy między innymi jakość obrazu, która ma istotny wpływ na efektywność (także komfort) korzystania ze sprzętu informatycznego. Wyróżniane są dwa umowne typy kart: do monitorów monochromatycznych oraz do kolorowych.

Karta o sympatycznej nazwie HERCULES pozwala tworzyć na ekranie monochromatycznego monitora obraz składający się aż z 720 razy 348 pikseli. Ta imponująca rozdzielczość maleje nieco w „kartach kolorowych". Karty te po ich zainstalowaniu w komputerze (włożeniu do gniazda), umożliwiają korzystanie z monitorów kolorowych. Na przykład karta CGA wyróżnia 640 X 200 pikseli (gdy wykorzystywane są tylko dwa kolory), 320 X 200 (cztery kolory). Gdy wyświetlamy tekst: 25 wierszy po 80 znaków każdy, dostępne jest 16 kolorów. Znacznie lepsze parametry kolorowego obrazu gwarantuje użycie, obecnie bardzo powszechnej karty EGA. Obraz tworzony przez tę kartę zbudowany jest z 640 X 350 pikseli dla 16 kolorów!

Zważywszy na rangę grafiki komputerowej, należy zwrócić uwagę, że dostępny jest szereg innych kart wraz ze specjalnymi monitorami, które mają o wiele lepsze (od typowych) parametry (oczywiście poza ceną, która jest gorsza). Są to karty do zastosowań profesjonalnych. Na przykład karta PGC pozwala operować 256 barwami z palety 4096 kolorów, przy czym rozdzielczość wynosi 640 X 480 pikseli, karta GS-1280 gwarantuje rozdzielczość 1280 X 960 pikseli plus 16 barw z palety 4096 barw, natomiast GS-800 ma nieco niższą rozdzielczość: 800 X 600 pikseli, ale sprzęt ▶

ten pozwala malować obrazki korzystając z 256 barw wybranych z palety... 16 milionów barw!

Kończąc temat „karty rozszerzające", chcę zwrócić uwagę na oryginalny typ pamięci dodatkowych, pamięci, które nazywane są RAMdysk. Sama nazwa wskazuje istotę owej konstrukcji. Na karcie zainstalowano bowiem pewną liczbę kostek pamięci typu RAM, które komputer, dzięki specjalnemu oprogramowaniu, widzi jako dodatkową umowną pamięć dyskową o niebagatelnej pojemności wielu megabajtów. W pierwszej chwili pojawiają się wątpliwości — po co komu taka pamięć, która zaraz po wyłączeniu komputera gubi wszystkie zapisane tam wcześniej informacje? Bywa, iż podczas pracy program bardzo często wykorzystuje (odczytuje lub zapisuje) dane zapisane na dysku twardym. Czas odszukania każdej informacji wynosi około 20 milisekund, to jest 0,02 s (bardzo dobry dysk). Jeśli jednak wcześniej niezbędne nam dane przepiszemy do RAMdysku, nasz program będzie wyszukiwał potrzebne mu dane wiele tysięcy razy szybciej, czas dostępu w pamięci RAM wynosi bowiem około 200 nanosekund, to jest 0,0000002 s. Imponujące, prawda?

Efektem walki o szybkość pracy komputera jest IBM PC AT. Różni się on od XT znacznie lepszym mikroprocesorem i nowocześniejszymi układami scalonymi (szczegóły sobie podarujemy). Ważne jest także to, iż AT „widzi" znacznie większą pamięć RAMowską niż komputer klasy XT. Bardziej dociekliwym czytelnikom pragnę zwrócić uwagę, że problem owego „widzenia większej pamięci" wiąże się z liczbą „drutów" (tak zwana szyna adresowa), które można podłączyć do mikroprocesora. Na nich podawana jest informacja (oczywiście w kodzie binarnym) o adresie wyróżnionej części pamięci RAM. IBM PC AT ma tych „drutów" aż 24, XT natomiast tyle, iż można adresować zaledwie 1024 KB (zadanie domowe: ile bitów ma adres w XT?).

Mówiąc o różnicach między komputerami XT a AT należy wspomnieć o napędach dysków elastycznych. W IBM PC AT zainstalowano bowiem nieco precyzyjniejsze napędy, które umożliwiają zapisywanie (odczytywanie do 1,2 MB na jednej dyskietce). Najczęściej w praktyce XT ma jeden lub dwa napędy flopów 360 KB, natomiast AT jeden napęd 360 KB i jeden 1,2 MB.

Bardzo istotne uzupełnienie podanych informacji o komputerach IBM PC powinien stanowić opis oprogramowania. Ono także zadecydowało, w stopniu porównywalnym do sprzętu (a może nawet większym) o użyteczności tej klasy maszyn. Temat ten pozostawmy sobie jednak „na potem".

Komputery personalne, PCty, są bardzo dobre dla zastosowań o stosunkowo niewielkich „wymaganiach". To znaczy takich, gdzie ilość przetwarzanych informacji jest niewielka (np.: szkoła, mała firma, dom...). Wobec tego stwierdzenia można by sądzić, że rozwiązaniem może być (dla nieco większej fabryki) kupienie następnego PCta, może dwóch, dziesięciu... Okazuje się to nie takie proste. Najczęściej bowiem przetwarzane dane są wspólne dla różnych użytkowników w tej samej firmie. Efekty pracy na jednym komputerze powinny być natychmiast dostępne na innym. Odpowiedzią na ten zarzut może być połączenie komputerów między sobą — tworzące tak zwaną sieć. Zatem aktualizując listę pracowników na komputerze zainstalowanym w kadrach (Abackiego w końcu wyrzucili z pracy), informacje te, przez sprzęgający (łączący) komputery „drut" dostępne są natychmiast w maszynie eksploatowanej przez finanse (Abackiego skreślić z listy płac).

Co zatem ostatecznie decyduje o potrzebie budowy większych komputerów? Odpowiedź jest niezmienna — zbyt mała moc tej klasy sprzętu. Oznacza to, że PCty potrzebują zbyt dużo czasu na wykonywanie prostych operacji arytmetycznych na dużych zbiorach danych. Szybkość nawet kilku milionów operacji na sekundę okazuje się niewystarczająca. Ma to miejsce w każdym poważnym zastosowaniu, gdzie liczba przetwarzanych informacji jest ogromna. Dotyczy to między innymi: wspomagania prac konstrukcyjnych (mostów, samolotów, samochodów, itp.), badań nad nowymi lekami, obsługiwania baz danych (dostępnych w sieciach komputerowych), tworzenia specjalnych efektów do filmów i oczywiście nie można pominąć prac nad sztuczną inteligencją. Służą temu tak zwane **superkomputery**. Jest oczywiście pośredni sprzęt w postaci minikomputerów, np.: WANG, pozostańmy jednak przy gigantach. Najbardziej znane to wytwór Seymour'a Cray'a: komputery CRAY1, CRAY2, CRAY X-MP, CRAY Y-MP oraz superkomputery (z innej „stajni") serii ETA. Wszystkie je łączy ogromna moc obliczeniowa. Wyraża się ją liczbą operacji na sekundę, ale operacji szczególnych — bardziej złożonych, tych które wykonywane są na tak zwanych liczbach zmiennoprzecinkowych. Dzięki logarytmicznemu zapisowi (cecha i mantysa) liczby te zapisywane są w bardzo ekonomiczny sposób. Ideę tę prezentuje następujący przykład: aby zapisać liczbę 1 000 000 musimy wykorzystać 7 znaków, lub... tylko trzy znaki — 1.6, gdzie 1 oznacza wartość, a 6 liczbę zer. W praktyce jest to oczywiście bardziej złożona operacja, którą stosuje się zresztą także w kilku krajowych pokoleniach komputerowych.

Typowa szybkość superkomputera — to miliardy FLOPSów — operacja na liczbach zmiennoprzecinkowych. Oczekuje się już wkrótce, że jeden superkomputer (jest ich na świecie około 300) zastąpi 10 milionów PCtów. Obrazuje to skalę mocy obliczeniowych, które mimo wszystko okazują się nadal, dla części użytkowników, zbyt małe. Proste przeliczenie dowodzi, że superkomputer wykonuje jedną operację w czasie, gdy światło pokona odległość równą zaledwie kilku centymetrom!

Podarujemy sobie oczywiście próbę analizy budowy tego sprzętu, aczkolwiek warto wiedzieć, że komputery tego typu muszą być podczas pracy chłodzone, a ogromne (coraz większe) szybkości uzyskuje się przez wykorzystywanie wielu, pracujących równolegle, procesorów. Wydaje się, że sprzęt ten znacznie przybliża moment, gdy komputer poza „mocą" dysponował będzie inteligencją o której dotychczas poważnie pisali tylko autorzy opowiadań fantastycznych.

Póki co wracamy do naszych PCtów, ich oprogramowania, ale to już podczas naszego kolejnego spotkania.

Warszawa 18 grudnia 1988
Włodzimierz GOGOŁEK

Literatura

1. Najlepiej napisaną, z dostępnych na naszym rynku, pozycją jest VADEMECUM nabywcy i użytkownika mikrokomputera profesjonalnego, NOT, Koło użytkowników mikrokomputerów profesjonalnych. Niestety owa dostępność jest tylko teoretyczna, zatem.
2. Mikrokomputery IBM PC, sześć części autorstwa Jacka Wojtal (dla bardziej wnikliwych czytelników) w następujących numerach naszego pisma: 3, 5, 6, 9, 11, 12/1987.
3. Wiele innych artykułów w konkurencyjnych pismach.





A W SZKOLE





MPUTERY I CIEKŁE KRYSZTAŁY W WAT

FOTO J. CHMURZYŃSKI





Moc w obwodach prądu sinusoidalnego

Poniższy program może być wykorzystywany w zajęciach dydaktycznych z elektrotechniki w szkołach ponadpodstawowych i pomaturalnych o profilu elektrycznym oraz w zajęciach dydaktycznych z fizyki w szkołach średnich.

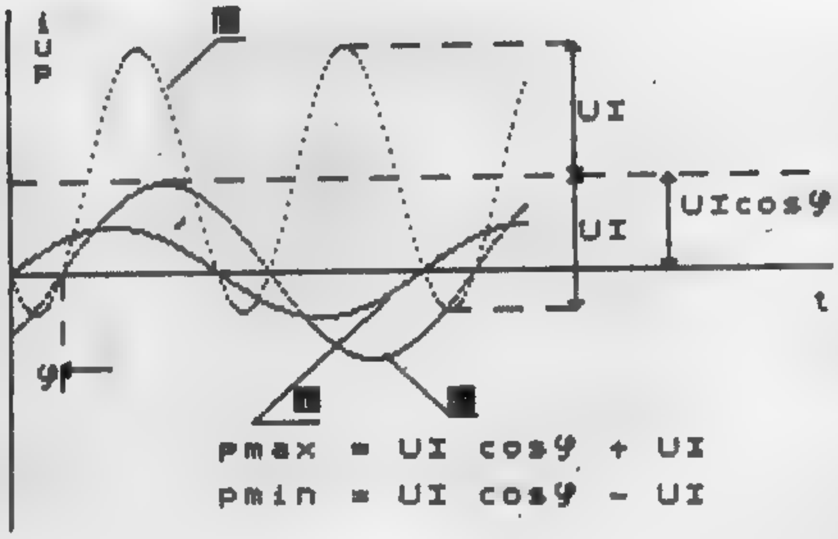
Z praktyki wiadomo, że wiadomości, dla zobrazowania których należy posłużyć się wykresami zawierającymi wiele krzywych w funkcji tej samej zmiennej, przekazywane przy wykorzystaniu rzutnika są mało czytelne. Zastosowanie komputera pozwala na rysowanie wykresów etapami (kolejne charakterystyki) i ich wzajemne nakładanie się. Uzyskuje się przez to możliwość śledzenia przebiegu powstawania każdej z charakterystyk i łatwo można wówczas zrozumieć występujące zjawiska fizyczne. Konstrukcja programu umożliwia zobrazowanie przebiegów czasowych mocy chwilowej w obwodach prądu sinusoidalnego zmiennego przy dowolnie założonym kącie przesunięcia fazowego pomiędzy prądem a napięciem, a zatem dla elementów idealnych i rzeczywistych (cewka indukcyjna kondensator, rezystor). Założony kąt przesunięcia fazowego wprowadza się w linii 165 programu.

W celu napisania programu należy w pierwszej kolejności wprowadzić do pamięci komputera podprogram 1 (linie 1000 do 1260) i po uruchomieniu i sprawdzeniu zapisać na taśmie magnetycznej lub dyskietce (komenda SAVE „mocs" SCREEN). Rysunek ten wywoływany jest później w programie w linii 20.

Program stwarza możliwość narysowania w trakcie jego realizacji 5 wykresów, które można wyświetlić w każdej chwili na ekranie monitora (linie 900—930). W tym celu należy wprowadzić linię **1 REM** i 35 dowolnych znaków. Następnie należy wykonać następującą procedurę: FOR i = 23760 To 23794:READ k:POKE i, k: NEXT i : DATA 33,88,147,24,13,33,88,174,24,8,33,88,201, 24,3,33, 88,228,17,0,64,26,78,119,121,18, 35,19,122,254,91,32,244,201,33. Po wykonaniu tego polecenia można przystąpić do zapisania właściwego programu (linie 2—930).

Po uruchomieniu i sprawdzeniu prawidłowości działania należy program zapamiętać (komenda SAVE „moc"). Linia 925 (naciśnięcie klawisza p) pozwala powrócić do tej części programu, w której tworzone są wykresy i ponownie sporządzić ich nowe wersje.

Eligiusz TARCZYŃSKI



PODPROGRAM 1

```
000 BORDER 0: PAPER 0: INK 7: CLS
1005 PLOT 0,0: DRAW 0,175: PLOT 0,87: DRAW 255,0
1010 PRINT AT 0,1;"i";AT 1,1;"u";AT 2,1;"p";AT 12,31;"t"
1020 LET y=87: LET w=4*PI/256
1030 FOR x=0 TO 160
1040 PLOT x,y+15*SIN (w*x)
1050 PLOT x,y+30*SIN (w*x-.25*PI)
1060 PLOT x,y+88*SIN (w*x)*SIN (w*x-.25*PI)
1070 NEXT x
1080 LET t=44*COS (.25*PI)+87
1090 FOR a=0 TO 253 STEP 16
1100 PLOT a,t: DRAW 8,0
1110 NEXT a
1130 FOR a=103 TO 177 STEP 16
1140 PLOT a,t+44: DRAW 8,0
1150 NEXT a
1160 FOR a=134 TO 177 STEP 16
1170 PLOT a,t-44: DRAW 8,0
1180 NEXT a
1190 PLOT 175,t-44: DRAW 0,88: PLOT 174,161: DRAW 2,0: PLOT 173,160: DRAW 4,0: P
LOT 173,t+2: DRAW 4,0: PLOT 174,t+1: DRAW 2,0: PLOT 174,t-1: DRAW 2,0: PLOT 173,
t-2: DRAW 4,0: PLOT 173,76: DRAW 4,0: PLOT 174,75: DRAW 2,0
1200 PLOT 204,87: DRAW 0,t-87: PLOT 203,88: DRAW 2,0: PLOT 202,89: DRAW 4,0: PLO
T 202,t-2: DRAW 4,0: PLOT 203,t-1: DRAW 2,0
1210 FOR a=87 TO 47 STEP -16
1220 PLOT 16,a: DRAW 0,-8
1230 NEXT a
1240 PLOT 16,54: DRAW 15,0: PLOT 17,53: DRAW 0,2: PLOT 18,52: DRAW 0,4
1250 PLOT 50,150: DRAW 15,15: DRAW 8,0: PLOT 117,37: DRAW 19,-19: DRAW 8,0: PLOT
 117,78: DRAW -42,-42: DRAW 20,0
1260 PRINT AT 0,8; INVERSE 1;"p"; INVERSE 0;AT 15,1;"φ";AT 16,11; INVERSE 1;"i";
 INVERSE 0;AT 16,17; INVERSE 1;"u"; INVERSE 0;AT 4,22;"UI";AT 9,22;"UI";AT 8,26;
"UIcosφ";AT 18,8; PAPER 2;"pmax = UI cosφ + UI";AT 20,8; PAPER 2;"pmin = UI cosφ
 - UI"
```

PROGRAM W.ABCIWY

```
  2 CLEAR 37719:LOAD *"MOCS"CODE 37720:LOAD *"UDG.COD"CODE
  3 PAPER 6: INK 1: BORDER 4: CLS
  5 FOR i=65304 TO 65304+63: POKE i,7: NEXT i
  7
  8 INK 0: BORDER 1: CLS
 10 PRINT AT 8,1;"PREZENTOWANY PROGRAM POZWALA    PRZYSWOIC SOBIE ZAGADNIENIA
  DOTYCZACE WYDZIELANIA SIE MOCY  CHWILOWEJ W OBWODACH PRADU      SINUSOIDALNIE
 ZMIENNEGO": PAUSE 400
 12 CLS : PRINT "Moc chwilowa w obwodach  prądu  sinusoidalnego definiowana jes
t jako iloczyn wartości chwilowychnapięcia i prądu
                          "; PAPER 1; INK 7;" p = u i "; PAPER 6; INK 0;"
                                 Ponieważ prąd i napięcie          zmieniaj
q w funkcji czasu swą   wartość bezwzględną jak również i znak, zatem moc chwilo
wa takżezmienia się w czasie zarówno    co do wartości bezwzględnej jak i co do
znaku. ": PAUSE 0
 13 CLS : PRINT "Jeżeli w danej chwili znaki      prądu i napięcia są zgodne
  wówczas "; INVERSE 1;"p>0"; INVERSE 0;" - energia jest       dostarczana ze źró
dła do        odbiornika. Jeżeli natomiast    znaki prądu i napięcia są       pr
zeciwne wówczas "; INVERSE 1;"p<0"; INVERSE 0;" - energia jest zwracana z odbior
nika do   źródła. Zmiana znaku mocy       chwilowej występuje tylko w     obwoda
ch zawierających elementy zachowawcze (L,C).": PAUSE 0
 15 CLS : PRINT "W liniowym obwodzie elektrycznymwystępujące napięcie oraz prąd
  mają tę samą częstotliwość lecz są przesunięte w fazie o kąt φ. Przyjmując faz
ę początkową     prądu równą zero moc chwilowa  "
```

```
  16 PRINT "wyniesie:                                                        "; P
APER 1; INK 7;"             p = u i =          UmIm sin2*PI*f*t sin(2*PI*f*t-φ)"
; PAPER 6; INK 0;"                                 Po przekształceniach:
                                  "; PAPER 1; INK 7;"p = UI cosφ - UI cos(4*PI*f
*t-φ)"; PAPER 6; INK 0;"                                     przy czym:"
  18 PRINT  INVERSE 1;"UI cosφ"; INVERSE 0;" - składowa stała
                           "; INVERSE 1;"UI cos(4*PI*f*t-φ)"; INVERSE 0;" - składowa
 zmienna o częstotliwości          dwukrotnie większej od           częstotliwości
prądu i napięcia"; PAUSE 0
  20 RANDOMIZE USR 23760
  21 PAUSE 0
  22: BORDER 0: PAPER 0: CLS : PAUSE 0
  40 LET s=23755
  45 LET s=s+5: BORDER 0: PAPER 0: INK 7: CLS
  50 PLOT 0,0: DRAW 0,175
  55 PLOT 0,87: DRAW 255,0
  60 PRINT AT 0,1;"i"
  61 PRINT AT 1,1;"u"
  62 PRINT AT 2,1;"p"
  63 PRINT AT 12,31;"t"
 125 LET y=87
 127 LET w=4*PI/256
 130 PRINT AT 17,27; FLASH 1;"i(t)"
 140 FOR x=0 TO 254
 145 PLOT x,y+15*SIN (w*x)
 150 NEXT x
 155 PRINT AT 17,27;"i(t)"
 160 PRINT AT 18,27; FLASH 1;"u(t)"

 165 INPUT "Podaj kąt przesunięcia fazowego φ=...PI od -0.5 do 0.5  ";k
 166 IF k<-.5 OR k>.5 THEN  GO TO 165
 168 GO SUB 500
 170 FOR x=0 TO 254
 175 PLOT x,y+30*SIN (w*x+k*PI)
 180 NEXT x
 185 PRINT AT 18,27;"u(t)"
 190 PRINT AT 19,27; FLASH 1;"p(t)"
 195 FOR x=0 TO 254
 200 PLOT x,y+80*SIN (w*x)*SIN (w*x+k*PI)
 205 NEXT x
 210 PRINT AT 19,27;"p(t)"
 215 PAUSE 0
 217 IF k>-.5 AND k<.5 THEN  GO SUB 600
 218 BORDER 1: PAPER 6: INK 0
 219 IF s=23780 THEN  GO TO 900
 220 RANDOMIZE USR s: CLS : PRINT AT 9,1;"Dla zobrazowania przebiegów      czasow
ych o innych prametrach   należy nacisnąć klawisz "; INVERSE 1;"T"; INVERSE 0;"

      natomiast naciśnięcie klawisza  "; INVERSE 1;"N"; INVERSE 0;" zakończy dzi
ałanie programu"
 225 IF INKEY$="t" OR INKEY$="T" THEN  GO TO 45
 230 IF INKEY$="n" OR INKEY$="N" THEN  PAPER 0: CLS : GO TO 900
 235 GO TO 225
 500 IF k=-.5 THEN  PRINT AT 18,3; PAPER 6; INK 1;"Przebiegi czasowe dla ";AT 19
,3;"idealnego kondensatora"
 505 IF k>-.5 AND k<0 THEN  PRINT AT 18,5; PAPER 6; INK 1;"Przebiegi czasowe ";A
T 19,5;"dla rzeczywistego ";AT 20,5;"  kondensatora     "
 510 IF k=0 THEN  PRINT AT 18,5; PAPER 6; INK 1;"Przebiegi czasowe ";AT 19,5;"
dla rezystora    "
 515 IF k>0 AND k<.5 THEN  PRINT AT 18,3; PAPER 6; INK 1;"  Przebiegi czasowe
";AT 19,3;"dla rzeczywistej cewki"
 520 IF k=.5 THEN  PRINT AT 18,3; PAPER 6; INK 1;"Przebiegi czasowe dla";AT 19,3
;"    idealnej cewki    "
 525 RETURN
 600 LET t=44*COS (k*PI)+87
 605 LET t1=INT (t/8)
 610 FOR a=0 TO 253 STEP 16
 615 PLOT a,t: DRAW 8,0
 620 NEXT a
 625 PRINT AT (21-t1),26;"UIcosφ"
 630 PAUSE 0
 635 RETURN
 900 PAUSE 0: RANDOMIZE USR 23760
 910 PAUSE 0: RANDOMIZE USR 23765
 915 PAUSE 0: RANDOMIZE USR 23770
 920 PAUSE 0: RANDOMIZE USR 23775
 925 IF INKEY$="p" THEN  GO TO 40
 930 GO TO 900
```

```
2 REM *******************
3 REM *                 *
4 REM * TEST NA REFLEKS *
5 REM *     ZX-81       *
6 REM *                 *
7 REM *******************
8 DIM A(4)
9 LET A=0
10 FOR I=1 TO 4
20 PRINT
25 FAST
30 FOR J=1 TO 100*RND+100
35 NEXT J
37 IF INKEY$<>"" THEN GOTO
 37
40 SLOW
45 POKE 16437,255
50 POKE 16436,255
55 PRINT "WCISNIJ TERAZ KLA
WISZ"
60 IF INKEY$="" THEN GOTO 6
0
65 LET A(I)=(255-PEEK 16437
)*5+(255-PEEK 16436)/50
67 IF A(I)>A THEN LET A=A(I
)
70 PRINT
75 PRINT "SZYBKOSC REAKCJI"
80 PRINT A(I);" SEKUNDY"
85 PRINT
90 PRINT "WCISNIJ ENTER"
95 INPUT A$
100 CLS
105 NEXT I
110 CLS
115 PRINT "WYNIK"
120 PRINT
125 PRINT "TEST --> CZAS"
130 FOR I=1 TO 4
135 PRINT I; " ";
140 FOR J=1 TO A(I)*30 STEP
 32/(A*100)
145 PRINT "▓";
150 NEXT J
155 PRINT
160 NEXT I
```

J. J.

```
1 REM *******************
2 REM *                 *
3 REM *    O B C Y      *
4 REM *     ZX-81       *
4 REM *                 *
5 REM *******************

15 PAUSE 50
20 LET T=0
30 LET W=INT(RND*7)*2
40 LET X=1
50 LET Y=X
60 LET Z=W
70 LET J=11
80 PRINT AT W+1,X-1;"▀▀▀"
90 PRINT AT Y,Z;"▀▄"
100 IF W=Y AND X=Z THEN GO
TO 280
110 PRINT AT J,10;"▚ ▞"
115 PRINT AT J,11;"▚ ▞"
130 IF J=Y AND J=Z THEN PR
INT T; "*** OBCY ZNISZCZON
Y ***    ZIEMIA URATOWA
NA"
135 IF J=Y AND J=Z THEN ST
OP
136 LET T=T+5
140 IF W=Y AND X=Z THEN PR
INT T; "*** ZIEMIA ZNISZCZ
ONA ***";K
170 LET X=X+2*(INKEY$="8")
-2*(INKEY$="5")
180 LET W=W+2*(INKEY$="6")
-2*(INKEY$="7")
190 IF Y<W THEN LET Y=Y+1
200 IF Y>W THEN LET Y=Y-1
210 IF Y>W THEN LET Y=Y-1
220 IF Z>X THEN LET Z=Z-1
230 IF Z<X THEN LET Z=Z+1
240 CLS
260 GOTO 80
280 GOTO 140
```

Atakuje Ciebie statek z innej galaktyki. Niestety, nie posiadasz juz amunicji. Pozostaje tylko ucieczka. Im dluzej bedzie ona trwala, tym wiecej zgromadzisz punktow. Jesli znuzy Cie ucieczka, wprowadz statek przeciwnika do wnetrza figury na srodku ekranu. Statek jego zostanie zniszczony a Ty zostaniesz ocalony. Sterowanie statkiem nastepuje za pomoca klawiszy:"5","6","7","8".

Ciekawe sposoby zmian zawartości ekranu mikrokomputera AMSTRAD CPC 6128 z wykorzystaniem dodatkowych 64 kB pamięci RAM

MIECZYSŁAW SKONIECZNY, PIOTR RAKOWSKI

Przedstawione poniżej programy umożliwiają uzyskanie na monitorze dodatkowych efektów graficznych, polegających na przesuwaniu fragmentu ekranu i wsuwaniu części obrazka zapamiętanego w drugim banku pamięci oraz na ciekawym sposobie przesyłania obrazu z dodatkowej pamięci na ekran.

Pierwszy z programów o nazwie „PRZESPIO" umożliwia przesuwanie w pionie fragmentów ekranu oraz jednoczesne wsuwanie części obrazka znajdującego się na monitorze lub w drugim banku pamięci.

Program ten uruchamiamy instrukcją:

CALL adres, KIER, CON, KROK, ILE, X1, X2, SZ, NR,

gdzie:

Adres = 38800.

KIER — jeśli przesuwamy z góry na dół to KIER = 1, jeśli przesuwamy z dołu do góry to KIER = 0.

CON — jeśli chcemy, aby wsuwanie było kontynuowane od miejsca, w którym się zatrzymaliśmy (jeśli nie przesunęliśmy o cał wysokości ekranu) to CONT = 1, w przeciwnym wypadku — jeśli wsuwamy fragment od początku to CONT = 0.

KROK — co ile linii ma być przesuwany fragment ekranu, KROK = 1, 2, 4, 5, 8, 10, 20, 40, 50, 100, (im KROK jest większy, tym szybciej się przesuwa).

ILE — ile razy mamy dokonać przesunięcia; ILE dobieramy tak, aby iloczyn KROK×ILE < 200.

X1 — współrzędna lewej części paska na ekranie (gdzie ma być wsuwany inny fragment), od 1 do 80.

X2 — podobnie jak X1 tylko, że dotyczy wsuwanego fragmentu z ekranu bądź innego obrazka zawartego w drugim banku pamięci.

SZ — szerokość przesuwanego paska, od 1 do 80.
Uwaga!
X1+SZ lub X2+SZ musi być mniejsze lub równe 80.

NR — skąd ma być brany fragment obrazka,
NR = 1 do 4 — kolejne bloki z drugiego banku pamięci,
NR = 0 — widoczny obrazek na ekranie.
Uwaga!
Aby uzyskać efekt cyklicznego przesuwania fragmentu na ekranie to NR = 0 i X1 = X2. Jeśli NR = 0 i X1 < > X2 to należy X1 i X2 tak dobrać, aby paski były rozłączne.
Jeśli NR = 0 i X1 = X2 to za program musimy mieć KROK×SZ wolnych bajtów pamięci.

Prezentowane niżej progamy w BASIC-u mają za zadanie załadować kody programów opracowanych w asemblerze pod wskazane adresy i nagrać je na dyskietkę.

Pierwszy z nich, ładujący program „PRZESPIO" jest następujący:

```
...            "przespio"
...  38800:ilosclinii=20
...
170 POKE adres,a
180 s=s+a:adres=adres+1
190 NEXT j
200 IF s<>VAL("&"+w$) THEN PRINT"Blad w linii nr ";220+i*10:STOP
210 NEXT i
215 STOP
220 SAVE"przespio",b,38800,585
230 DATA 1B0E00C00000141414000101010001000D7E003299970D7E0232969700D7E,07FA
240 DATA 04329897DD7E06329797DD7E08329A97DD7E0A329C97DD7E0C329B97DD7E,0DDC
250 DATA 0E329F97B7284321807F2297982180FF22A79821C39922D69822F4982218,0D02
260 DATA 99221F99222F9922879921B0FF22EA9822ED982100C0273A9921AE992244,0C2E
270 DATA 9921C6992282993E4832809918412100402297982100C022A79821AE9922,0AFE
280 DATA D69822F4982218992 21F99222F99228799 2100C022EA9872ED982180FF22,0CE6
290 DATA 3A9921C39922449921B1992282993E7832809 93A97972198 97BE20043E00,0BD0
300 DATA 18023E01329E973A9C97470E003EC8900CB720FB793D329D973A99978728,095B
310 DATA 1A01FF7F57153EC482ED793A9B97B728052A9297181D21807F18183A9B97,0B83
320 DATA B728052A929718 0D2180FF3A98974F0600092294973A9A97E52199973435,0B1A
330 DATA E1202811D9992A9497F5E53A9C97F5E53A96974F0600EDB0E1CDC399F13D,10B3
340 DATA 20EE3A9E97B72803229497C1F1F5E51180FF2180FF3A9C97F5CDC399F13D,1121
350 DATA 20F83A97974F060009EB09EB3A9D97E5E5F53A9C97F5E5D53A96974F0600,0EC2
360 DATA EDB0D1E1CDC399EBCDC399EBF13D20E7F1D1E1F53A9C97F5CDC399F13D20,131D
370 DATA F8F13D20CE2100C03A9C973D2808F5CDAE99F13D20F83A97974F060009EB,0DCF
380 DATA E1E53A98974F0600093A9997B7200321D9993A9C97F5D5E53A96974F0600,0D3C
390 DATA 3A9997B72005E1EDB01809EDB0E17CFE48CDC699D1EBCDC399EBF13D20D9,1248
400 DATA 229297E12A9297AF3A98974F0600ED4222929 7F13DC2B69801FF7F3EC0ED,0F0E
410 DATA 79C97CFEF8280238 09D63867015000091804010008 09C97CFEC8300AC638,0A5F
420 DATA 67015000AFED42C9AF01000 8ED42C900000000000000000000000000000,060F
```

Drugi z programów nazywa się „PRZESPOZ". Umożliwia on uzyskanie podobnych efektów — ale w poziomie.

Program uruchamiamy instrukcją:

CALL adres, KIER, CONT, KROK, ILE, Y1, Y2, SZ NR,

gdzie:

adres = 37750.

KIER — KIER = 1, gdy przesuwamy w prawo lub KIER = 0, gdy przesuwamy w lewo.

CONT — podobnie jak w poprzednim programie.

KROK — KROK = 0 to przesuwamy na ekranie co punkt, KROK = 1 to przesuwamy co bajt, czyli w poszczególnych trybach:
a) tryb 2 co 8 punktów,
b) tryb 1 co 4 punkty,
c) tryb 0 co 2 punkty.

ILE — ile razy mamy dokonać przesunięcia,
— jeśli KROK = 1 to ILE od 1 do 80,
— jeśli KROK = 0 to ILE od 1 do: a) 640 dla trybu 2,
b) 320 dla trybu 1,
c) 160 dla trybu 0.

Y1 — współrzędna górnej części paska przesuwanego na ekranie od 0 do 199.
Uwaga! Numer 199 ma linia na górze ekranu, a numer 0 na dole.

Y2 — współrzędna górnej części paska wsuwanego na ekran, bądź z innego obrazka z drugiego banku pamięci.

SZ — szerokość przesuwanego paska, od 1 do 200.

NR — podobnie jak w poprzednim programie.

Postać źródłowa tego programu jest następująca:

```
100 'kod maszynowy  "przespoz"
110 MEMORY 36299:adres=37750:ilosclinii=25
120 RESTORE 230
130 FOR i=1 TO ilosclinii
140 s=0:READ a$,w$
150 FOR j=1 TO 59 STEP 2
160 a=VAL("&"+MID$(a$,j,2))
170 POKE adres,a
180 s=s+a:adres=adres+1
190 NEXT j
200 IF s<>VAL("&"+w$) THEN PRINT"Blad w linii nr ";220+i*10:STOP
210 NEXT i
215 STOP
```

```
220 SAVE"przespoz",b,37750,778
230 DATA 1B0E000000000000000000000000000DD7E0032D7D93DD7E02327E93DD7E,06BE
240 DATA 04327E93DD7E06328093DD7E0A328193DD7E09328293DD7E0A328393DD7E,0D4A
250 DATA 0C328493DD7E0E328593B72B5D[illegible]EAF[illegible]0F953EED[illegible]0953E42[illegible]11953E2[illegible],0AC2
260 DATA 322D953E2B324D953E4F3292953 2A4953E1F32C69532DD9532EE953E1E32,0893
270 DATA CB95[illegible]F17[illegible]2CB95[illegible]2D[illegible]95[illegible]D495[illegible]2D5953 2E6953E[illegible]8[illegible]2D[illegible]953E[illegible]DF953E,0EBF
280 DATA AA[illegible]2EB95[illegible]F953[illegible]2F[illegible]953E[illegible]E[illegible]21F[illegible]618597E[illegible]32[illegible]F953211953E[illegible]210953E,098F
290 DATA 2B322D953E23324D953E00329295 32A4953E1732C69532DD95326E953E16,0825
300 DATA 32C8953E1F32C89532D39532D49532D59532E6953E11320795 3EEE32DF95,0EBB
310 DATA 3E5532E8953EAA32F0953E06321F96210B96222695CD11BCE603327893FE,0C69
320 DATA 00[illegible]B18FE01[illegible]80A[illegible]C[illegible]95[illegible]B[illegible]081812[illegible]1D095[illegible]B953E 418[illegible]1E395,0B9F
330 DATA 22[illegible]B953E[illegible]147[illegible]AB49[illegible]B[illegible]20[illegible]7B[illegible]2[illegible]C9[illegible]21[illegible]AB[illegible]93FE[illegible]2ADE21F49522,0A7B
340 DATA 2B953E[illegible]1[illegible]77C[illegible]296[illegible]A7D9[illegible]B72816[illegible]D01FF7F57[illegible]EC48[illegible]FD[illegible]93A349[illegible]B7,0C5E
350 DATA 2806DD2A7A931B203A7D93B7700E21FF952226953A7F93CD9195180 63A7F,0B51
360 DATA 9[illegible]CDA[illegible]95E5DDF1CD2196[illegible]AB[illegible]D919[illegible]4E0000090[illegible]D4HB19[illegible]C9E5DDE5,1D10
370 DATA DD214E963A7E9357E50000CDFF950E50CD8B952B0D20F9DD23E1CD649515,0E4F
380 DATA [illegible]0FBDE1[illegible]A7D[illegible]B7CC[illegible]196[illegible]A7[illegible]9[illegible]D[illegible]C93[illegible]05DD2[illegible]CD2196E1DD[illegible]A93,0EA7
390 DATA C10B7BB1208A01FF7F3EC0ED79C97CFCF828023809D6386701500091804,0C47
400 DATA 01[illegible]9C9[illegible]CFE1828[illegible]2[illegible]0AC6[illegible]8C[illegible]15[illegible]AFED4[illegible]C[illegible]AF01[illegible]BED42C921,0AAF
410 DATA 00C0473EC790B7CBF5CD6495F13020F8C921004047 3EC790B7CBF57CFE78,1C8B
420 DATA CD6795F13D20F5C93A7B93FE0024FED12B0D7A79931 7CB163E001F3279,0BDE
430 DATA 9[illegible]A799[illegible]1E1[illegible]1FE[illegible]114[illegible]9717[illegible]C8[illegible]C9[illegible]A799[illegible]554[illegible]E[illegible]2,0D6B
440 DATA 799317E6AAB077C93A7993477E327993787 7C9DD7E003279933EFF327C93,0EBB
450 DATA [illegible]A7[illegible]B7[illegible]E0B[illegible]C93DD7E[illegible]2799[illegible]D[illegible]CB[illegible]AC9F5DDE[illegible]3A7E9[illegible]DD,0E54
460 DATA F5E1DD 14E9646DD7[illegible]DD1Z3E53A7D9[illegible]B[illegible]2[illegible]5CD649518[illegible]A7CFE78CD6795,0EF5
470 DATA F1[illegible]D2[illegible]E4DDE1E[illegible]C90000000000000000000000000000000000,059A
```

Trzeci program, o nazwie „ROL", umożliwia przesłanie obrazka z drugiego banku pamięci na ekran, kolejno zmieniając linie. Na monitorze obserwujemy efekty nakładania nowego obrazka na starą zawartość ekranu.

Wywołanie:

CALL adres, KIER, NR,

gdzie:

adres = 36300,

KIER — KIER = 1, jeśli linie mają się zmieniać z dołu do góry lub KIER = 0 w przeciwnym wypadku.

NR — NR = 0—3 numer bloku drugiego banku pamięci, w którym znajduje się obrazek.

Postać źródłowa tego programu jest następująca:

```
100 'kod maszynowy "rol"
110 MEMORY 36299;adres=36300:iloscl inii=4
120 RESTORE 230
130 FOR i=1 TO ilosclinii
140 s=0:READ a$,w$
150 FOR j=1 TO 59 STEP 2
160 a=VAL("&"+MID$(a$,j,2))
170 POKE adres,a
180 s=s+a:adres=adres+1
190 NEXT J
200 IF s<>VAL("&"+w$) THEN PRINT"Blad w linii nr ";220+i*10:STOP
210 NEXT i
215 STOP
220 SAVE"rol",b,36300,110
230 DATA DD7E02B728172126BC271D8E22218E210040220B8E2100C0220E8E181521,07FD
240 DATA 29BC221D8E22218E21807F220B8E2180FF220E8EDD7E0001FF7F573EC482,0B71
250 DATA ED7921004011 00C03EC8E5D5015000EDB0D1E1F5CD26BCEBCD26BCEBE521,1027
260 DATA EB032B7CB520FBE1F13D20E001FF7F3EC0ED79C9000000000000000000000,0B1D
```

Program demonstracyjny zawiera przykłady zastosowania omówionych wyżej programów. Umożliwi on Czytelnikom dokładniejsze poznanie możliwości opisanych programów. Pokazuje on także, jak we własnych programach korzystać z programów: „PRZESPIO", „PRZESPOZ", i „ROL".

Wersja źródłowa programu demonstracyjnego jest następująca:

```
1 'program demonstracyjny
10 MEMORY 36299
20 MODE 1:LOAD"przespoz.bin":LOAD"przespio.bin":LOAD"rol.bin"
30 poz=37750:pio=38800:rol=36300
35 FOR i=1 TO 23 STEP 2
40 LOCATE 4,i:PRINT"Przyklad cyklicznego przesuwania"
42 LOCATE 12,i+1:PRINT"fragmentu ekranu"
45 NEXT
50 kier=1:con=0:krok=0:ile=320:y1=111:y2=y1:sz=16:nr=0
60 GOSUB 1000
70 kier=0:krok=1:ile=80
80 GOSUB 1000
90 kier=0:krok=1:ile=200:x1=30:x2=x1:sz=10
100 GOSUB 2000
110 kier=1:krok=2:ile=100:x1=50:x2=x1:sz=10
120 GOSUB 2000
130 kier=0:krok=8:ile=25:x1=20:x2=x1:sz=40
140 GOSUB 2000
150 FOR i=1 TO 23 STEP 2
160 LOCATE 2,i:PRINT"Przyklad wsuwania fragmentow obrazkow"
165 LOCATE 8,i+1:PRINT"z drugiego banku pamieci"
170 NEXT
180 kier=1:con=0:krok=0:ile=320:y1=111:y2=199:sz=16:nr=1
190 GOSUB 1000
200 kier=0:krok=1:ile=80:y1=50:y2=199:sz=20:nr=2
210 GOSUB 1000
220 kier=0:krok=4:ile=50:x1=50:x2=1:sz=20:nr=3
230 GOSUB 2000
240 kier=1:krok=2:ile=50:x1=10:x2=70:sz=10:nr=4
250 GOSUB 2000
255 CLS
260 FOR i=1 TO 23 STEP 2
270 LOCATE 5,i:PRINT"Przyklad wymiany obrazka przez"
275 LOCATE 10,i+1:PRINT"kolejne zmiany linii"
280 NEXT
290 kier=0:nr=0
300 GOSUB 3000
310 kier=1:nr=3:CLS
320 GOSUB 3000
330 FOR i=1 TO 500:NEXT
340 CLS:LOCATE 3,12:PRINT"Czy powtorzyc demonstracje ? (t/n)"
350 IF INKEY(51)<>0 AND INKEY(46)<>0 THEN 350
355 IF INKEY(46)=0 THEN STOP
360 CLS:GOTO 30
999 STOP
1000 CALL poz,kier,con,krok,ile,y1,y2,sz,nr
1001 RETURN
2000 CALL pio,kier,con,krok,ile,x1,x2,sz,nr
2001 RETURN
3000 CALL rol,kier,nr
3001 RETURN
```

STREAMERY

Kontynuacją rozpoczętego w zeszłym numerze cyklu artykułów pokazujących ceny dolarowe sprzętu komputerowego i jego peryferii jest zamieszczony dzisiaj cennik taśmowych pamięci magnetycznych czyli streamerów.

ceny zamieszczone poniżej opracowane zostały na podstawie cenników firm:

DISCOUNT MICRO PRODUCT

QIC REASEARCH INCORPORATED

BULLDOG COMPUTER PRODUCTS

Streamery:

Streamer	Cena
20 MB wewnętrzny firmy "IRVINE" do PC XT/AT	320 $
20 MB zewnętrzny firmy "IRVINE" do PC XT/AT	330 $
20 MB wewnętrzny firmy "WANGTEK" do PC XT/AT	380 $
20 MB zewnętrzny firmy "WANGTEK" do PC XT/AT	410 $
40 MB wewnętrzny firmy "ARCHIVE" do PC XT/AT	300 $
40 MB wewnętrzny firmy "IRVINE" do PC XT/AT	320 $
40 MB zewnętrzny firmy "IRVINE" do PC XT/AT	430 $
40 MB wewnętrzny firmy "MOUNTAIN" do PC XT/AT i PS2	440 $
40 MB zewnętrzny firmy "MOUNTAIN" do PC XT/AT i PS2	550 $
60 MB wewnętrzny firmy "EVEREX" do PC XT/AT	550 $
60 MB zewnętrzny firmy "ARCHIVE" do PC XT	620 $
60 MB wewnętrzny firmy "ARCHIVE" do Compaq	630 $
60 MB zewnętrzny firmy "EVEREX" do PC XT/AT	640 $
60 MB wewnętrzny firmy "TEAC" do PC XT/AT	650 $
60 MB zewnętrzny firmy "ARCHIVE" do PC AT	670 $
60 MB wewnętrzny firmy "ARCHIVE" do PC AT	680 $
60 MB zewnętrzny firmy "TEAC" do PC XT/AT	690 $
60 MB wewnętrzny firmy "WANGTEK" do PC XT/AT	700 $
60 MB wewnętrzny firmy "SYSGEN" do PC XT/AT	700 $
60 MB wewnętrzny firmy "GENOA GALAXY" do PC XT/AT	720 $
60 MB wewnętrzny firmy "WANGTEK" do PC XT/AT i PS2	750 $
60 MB zewnętrzny firmy "SYSGEN" do PC XT/AT	750 $
60 MB zewnętrzny firmy "ARCHIVE" do PS2	770 $
60 MB zewnętrzny firmy "GENOA GALAXY" do PC XT/AT	800 $
60 MB zewnętrzny firmy "WANGTEK" do PC XT/AT	800 $
60 MB zewnętrzny firmy "MAYNESTREAM" do PS2	870 $
120MB wewnętrzny firmy "WANGTEK" do PC XT/AT	990 $
120MB zewnętrzny firmy "WANGTEK" do PC XT/AT	1190 $
120MB zewnętrzny firmy "MOUNTAIN" novell compatibly	1700 $
150MB zewnętrzny firmy "MOUNTAIN" do PS/2	1780 $
300MB zewnętrzny firmy "MOUNTAIN" do PC XT/AT PS/2	2500 $
600MB zewnętrzny firmy "MOUNTAIN" do PC AT	1100 $

Materiały zostały przygotowane przez firmę BOIiE sp. z o.o. - Biuro Obsługi Importu i Exportu, 00-867 Warszawa ul. Chłodna 35/37 paw. iA, tel.247818 tlx 817073 BOIE PL

WYODRĘBNIONE TENDENCJE ROZWOJOWE

ANDRZEJ GRZESIAK

Program służy do wyodrębniania tendencji rozwojowej (trendu) metodą analitycznego wyrównywania szeregów czasowych, polegającą na tym, że tendencję rozwojową szeregu wyraża się za pomocą funkcji matematycznej. W programie można aproksymować równanie trendu następującymi funkcjami: liniową, parabolą drugiego stopnia, parabolą trzeciego stopnia, wykładniczą, hiperboliczną. Równania, według których oblicza się parametry funkcji trendu otrzymuje się stosując klasyczną metodę najmniejszych kwadratów (patrz: K. Zając Zarys metod statystycznych, PWN, Warszawa 1971). Oprócz parametrów równania trendu program oblicza wartości równania dla wszystkich wartości argumentu (dane teoretyczne) oraz średni błąd szacowania, który jest pierwiastkiem kwadratowym ze średniej arytmetycznej kwadratów odchyleń danych empirycznych od danych teoretycznych.

Podstawową zaletą programu jest to, że pozwala na szybkie dobranie najwłaściwszej postaci funkcji trendu, to znaczy takiej, która zapewnia najmniejszą wartość średniego błędu szacowania.

Program zbudowano w taki sposób, aby po jednorazowym wprowadzeniu danych (dane empiryczne) pozwalał na wielokrotne powtarzanie obliczeń przy różnych funkcjach aproksymujących. Dane wprowadza się instrukcjami DATA poczynając od linii 1600. Jeżeli krok dyskretyzacji czasu równy jest 1, to wystarczy wprowadzić tylko wartości szeregu czasowego, a program sam uzupełni brakujące wartości czasu (zmienna T). Jeżeli krok dyskretyzacji czasu jest różny od jedności, to program należy uzupełnić instrukcjami powodującymi wczytywanie czasu (linie 160—210). Wczytywane wartości danych są pokazywane na ekranie monitora, co pozwala na sprawdzenie poprawności procesu wczytywania. Proces można przerwać w dowolnym momencie, naciskając jednocześnie klawisze CTRL i 1 (wymaga to pewnej wprawy), powtórne naciśnięcie klawiszy wznawia proces.

Maksymalna liczba prób N wynosi 100. Można ją zwiększyć przez zmianę deklaracji zmiennych w linii 110.

```
10 GOTO 80
20 ? :? "UWAGA!":? :?
30 ? "Dane wpisujemy instrukcjam
i DATA poczynajac od linii 1600.
Jezeli krok dyskretyzacji czasu
rowny jest 1";
40 ? ",to wpisuje-my tylko warto
sci szeregu czasowego.Jezeli kro
k dyskretyzacji czasu jest rozny
 od 1,to";
50 ? " nalezy dokonac zmian w li
- niach 160-210(wczytywanie czas
u:T(F))."
60 ? :? "Wpisz dane!"
70 CLR :STOP
80 GRAPHICS 1:SETCOLOR 2,0,0:POS
ITION 3,9
90 ? #6;"Wyodrebnianie"
100 POSITION 0,12:? #6;"tendencj
i rozwojowej":FOR ZMP=1 TO 1500:
NEXT ZMP
110 DIM T(100),Y(100),A(4,4),B(4
),Q$(1),X(4),YTR(100)
120 ? CHR$(125):? "Liczba prob N
=";:INPUT N:?
130 TRAP 20
140 GRAPHICS 0
150 ZMP=1
160 FOR F=1 TO N
170 READ L
180 T(F)=ZMP:? "T(";F;")=";T(F)
190 Y(F)=L:? "Y(";F;")=";Y(F)
200 ZMP=ZMP+1
210 NEXT F
220 ? :? "Koniec wczytywania dan
ych":? :? "c.d. po nac. RET.":IN
PUT Q$
230 ? CHR$(125)
240 S1=0:S2=0:S3=0:S4=0:S5=0:S6=
0:S7=0:S8=0:S9=0:S10=0:S11=0:S12
=0:S13=0:S14=0:S15=0:S16=0:S17=0
:S18=0
250 FOR F=1 TO N
260 S1=S1+Y(F):S2=S2+Y(F)*T(F)
270 S3=S3+Y(F)*T(F)^2:S4=S4+Y(F)
*T(F)^3
280 S5=S5+T(F):S6=S6+T(F)^2
290 S7=S7+T(F)^3:S8=S8+T(F)^4
300 S9=S9+T(F)^5:S10=S10+T(F)^6
310 S12=S12+CLOG(Y(F)):S13=S13+T
(F)*CLOG(Y(F))
320 S14=S14+(1/T(F))*Y(F):S15=S1
5+(1/T(F))
330 S16=S16+(1/T(F)^2)
340 NEXT F
350 B(1)=S1:B(2)=S2:B(3)=S3:B(4)
=S4
360 A(1,1)=N:A(1,2)=S5:A(1,3)=S6
:A(1,4)=S7
370 A(2,1)=S5:A(2,2)=S6:A(2,3)=S
7:A(2,4)=S8
380 A(3,1)=S6:A(3,2)=S7:A(3,3)=S
8:A(3,4)=S9
390 A(4,1)=S7:A(4,2)=S8:A(4,3)=S
9:A(4,4)=S10
400 ? CHR$(125):? :? "Jaka funkc
ja trendu?":? :? "1.liniowa":? :
? "2.kwadratowa":? :? "3.parabol
a III stopnia"
410 ? :? "4.wykladnicza":? :? "5
.hiperboliczna":?
420 INPUT FT:ON FT GOTO 430,570,
710,860,1030
430 REM liniowa funkcja trendu
440 LR=2
450 GOSUB 1180
460 ? CHR$(125):POSITION 8,8:? "
LINIOWA FUNKCJA TRENDU"
470 POSITION 7,11:? "y=";"(";X(1
);")+(";X(2);")*t"
480 ? :? :? :? "c.d. po nac. RET
.":INPUT Q$
490 ? CHR$(125):? "Wartosci funk
ji trendu":?
500 ZMP=1
510 FOR F=1 TO N
520 YTR=X(1)+X(2)*T(F):YTR(F)=YT
R
530 ? "YTR(";F;")=";YTR(F)
540 ZMP=ZMP+1:IF ZMP=16 THEN ZMP
=0:? "c.d. po nac. RET.":INPUT Q
$
550 NEXT F
560 GOTO 1480
570 REM kwadratowa funkcja trend
u
580 LR=3
590 GOSUB 1180
600 ? CHR$(125):POSITION 7,8:? "
KWADRATOWA FUNKCJA TRENDU"
610 POSITION 2,11:? "y=";"(";X(1
),")+(",X(2);")*t","+(";X(3);")*
t^2"
620 ? :? :? :? "c.d. po nac. RET
.":INPUT Q$
630 ? CHR$(125):? "Wartosci funk
cji trendu":?
640 ZMP=1
650 FOR F=1 TO N
660 YTR=X(1)+X(2)*T(F)+X(3)*T(F)
^2:YTR(F)=YTR
670 ? "YTR(";F;")=";YTR(F)
680 ZMP=ZMP+1:IF ZMP=16 THEN ZMP
=0:? :? "c.d. po nac. RET.":INPU
T Q$
690 NEXT F
700 GOTO 1480
710 REM parabola III stopnia
720 LR=4
730 GOSUB 1180
740 ? CHR$(125):POSITION 10,8:?
"PARABOLA III STOPNIA"
750 POSITION 2,11:? "y=";"(";X(1
);")+(";X(2);")*t";"+(";X(3);")*
t^2";"+"
760 POSITION 12,13:? "+(";X(4);"
)*t^3"
770 ? :? :? :? "c.d. po nac. RET
.":INPUT Q$
780 ? CHR$(125):? "Wartosci funk
cji trendu":?
790 ZMP=1
800 FOR F=1 TO N
810 YTR=X(1)+X(2)*T(F)+X(3)*T(F)
^2+X(4)*T(F)^3:YTR(F)=YTR
820 ? "YTR(";F;")=";YTR(F)
830 ZMP=ZMP+1:IF ZMP=16 THEN ZMP
=0:? :? "c.d. po nac. RET.":INPU
T Q$
840 NEXT F
850 GOTO 1480
860 REM funkcja wykladnicza
870 LR=2
880 B(1)=S12:B(2)=S13:A(1,1)=N:A
(1,2)=S5:A(2,1)=S5:A(2,2)=S6
890 GOSUB 1180
900 ZMW1=10^X(1):ZMW2=10^X(2)
910 ? CHR$(125)
920 POSITION 8,9:? "WYKLADNICZA
FUNKCJA TRENDU"
930 POSITION 8,12:? "y=";"(";ZMW
1;")*(";ZMW2;")^t"
940 ? :? :? :? "c.d. po nac. RET
.":INPUT Q$
950 ? CHR$(125):? "Wartosci funk
cji trendu":?
960 ZMP=1
970 FOR F=1 TO N
980 YTR=ZMW1*(ZMW2)^T(F):YTR(F)=
YTR
990 ? "YTR(";F;")=";YTR
1000 ZMP=ZMP+1:IF ZMP=16 THEN ZM
P=0:? :? "c.d. po nac. RET.":INP
UT Q$
1010 NEXT F
1020 GOTO 1480
1030 REM funkcja hiperboliczna
1040 LR=2
1050 B(1)=S1:B(2)=S14:A(1,1)=N:A
(1,2)=S15:A(2,1)=S15:A(2,2)=S16
1060 GOSUB 1180
1070 ? CHR$(125):POSITION 7,6:?
"HIPERBOLICZNA FUNKCJA TRENDU"
1080 POSITION 7,9:? "y=";"(";X(1
);")+";"(1/t)*(";X(2);")"
1090 ? :? :? :? "c.d. po nac. RE
T.":INPUT Q$
```

```
1100 ? CHR$(125):? "Wartosci fun
kcji trendu":?
1110 ZMP=1
1120 FOR F=1 TO N
1130 YTR=X(1)+(1/T(F))*X(2):YTR(
F)=YTR
1140 ? "YTR(";F;")=";YTR(F)
1150 ZMP=ZMP+1 IF ZMP=16 THEN ZM
P=0:? :? "c.d. po nac.RET.":INPU
T Q$
1160 NEXT F
1170 GOTO 1480
1180 REM Obliczanie ukladu rowna
n
1190 EPS=1.0E-07
1200 FOR K=1 TO LR
1210 MAX=EPS
1220 FOR F=K TO LR
1230 T=A(F,K)
1240 IF ABS(T)>ABS(MAX) THEN MAX
=T G=F
1250 NEXT F
1260 IF ABS(MAX)<=EPS THEN ? "Uk
lad nie ma rozwiazania":FOR ZMP=
1 TO 500:NEXT ZMP:GOTO 400
1270 FOR F=K TO LR:T=A(G,F):A(G,
F)=A(K,F) A(K,F)=T NEXT F:T=B(G)
 B(G)=B(K):B(K)=T
1280 IF K=LR THEN GOTO 1370
1290 FOR F=K+1 TO LR
1300 T=A(F,K)/MAX
1310 FOR G=K+1 TO LR
1320 A(F,G)=A(F,G)-T*A(K,G)
1330 NEXT G
1340 B(F)=B(F)-T*B(K)
1350 NEXT F
1360 NEXT K
1370 X(LR)=B(LR)/A(LR,LR)
1380 FOR K=LR-1 TO 1 STEP -1
1390 X(K)=B(K)/A(K,K)
1400 FOR G=K+1 TO LR
1410 X(K)=X(K)-A(K,G)/A(K,K)*X(G
)
1420 NEXT G
1430 NEXT K
1440 FOR F=1 TO LR
1450 X(F)=INT(X(F)*10000+0.5)/10
000
1460 NEXT F
1470 RETURN
1480 REM Obliczanie bledu
1490 S11=0
1500 FOR F=1 TO N
1510 S11=S11+(Y(F)-YTR(F))^2
1520 NEXT F
1530 SY=SQR((1/(N-LR))*S11)
1540 ? :? "Sredni blad szacowani
a=",SY
1550 ? :? "c.d. po nac. RET.":IN
PUT Q$
1560 GOTO 350
```

O tym już się nie tylko mówi...



Programy narzędziowe na CPC — AMSTRAD 6128

Program KATALOG dostarcza informacji o plikach (zbiorach) umieszczonych w pamięci dyskowej, pozwala na zmianę stanu pliku (zbioru), umożliwia edycję na poziomie pliku (zbioru).

Podaje następujące informacje o 64 plikach (tyle plików można maksymalnie zapisać na dyskietce):

- — nazwa pliku (zbioru),
- — typ pliku (zbioru),
- — numer logiczny użytkownika systemu (CP/M),
- — stan pliku (zbioru).

Numer logiczny użytkownika systemu (CP/M) jest liczbą całkowitą z przedziału 0—15 i określa wydzielony obszar pamięci dyskowej dla poszczególnych użytkowników systemu (maksymalna liczba różnych użytkowników wynosi 16). W ten sposób pliki (zbiory) utworzone przez jednego użytkownika nie są bezpośrednio dostępne dla drugiego użytkownika. Na wstępie system CP/M przyjmuje, że użytkownikiem systemu jest użytkownik o numerze 0.

Każdy plik (zbiór) może znajdować się w jednym z czterech stanów:

1) R/O (READ/ONLY) — plik (zbiór) z dozwolonym tylko odczytem informacji (zapis jest niemożliwy), próba zapisu do zbioru powoduje komunikat błędu,
2) R/W (READ/WRITE) — plik (zbiór) z możliwością odczytu i zapisu informacji,
3) SYS — oznacza plik (zbiór) systemowy, nie wyprowadzony w zleceniu DIR,
4) DIR — usuwa atrybut pliku (zbioru) „systemowy".

Poprzez wyszczególnione opcje programu KATALOG możemy:

- — zmienić nazwę pliku (zbioru),
- — skasować plik (zbiór),
- — odzyskać skasowany poprzednio plik (zbiór),
- — zmienić stan pliku (zbioru),
- — zmienić numer logiczny użytkownika systemu,
- — zmienić stronę przeglądanych plików (zbiorów),
- — przejrzeć spis plików (zbiorów) innej dyskietki,
- — „wyjść" z programu KATALOG pod BASIC AMSTRADA.

OPIS DZIAŁANIA PROGRAMU

Po uruchomieniu programu KATALOG dyskietkę z interesującymi nas plikami (zbiorami) umieszczamy w napędzie dyskowym i wciskamy dowolny klawisz.

W dolnej części ekranu monitora wyświetlony zostanie opis wszystkich opcji programu. Powyżej opisu opcji programowych ukaże się opis pierwszych trzydziestu dwóch plików (zbiorów) zapisanych na dyskietce.

W celu przejrzenia opisu drugich trzydziestu dwóch plików (zbiorów) należy wciskać klawisz P (opcja zmiany strony). Po przejrzeniu opisu plików (zbiorów) dyskietki i dokonaniu niezbędnych poprawek należy nagrać wprowadzone zmiany, wciskając w tym celu klawisz N (opcja — nagranie zmian).

Chcąc przejrzeć inną dyskietkę wciskamy klawisz I (opcja — inna dyskietka), a powrót do BASIC-a umożliwi nam wciśnięcie klawisza X (opcja — ucieczka).

Program DYSKCOPY służy do kopiowania dyskietek.

OPIS DZIAŁANIA PROGRAMU

Po uruchomieniu programu na ekranie monitora pojawi się komunikat:

PODAJ ILOŚĆ STACJI DYSKÓW

Jeśli korzystamy z jednego napędu dyskowego (znajdującego się w komputerze) wciskamy klawisz z cyfrą 1, a jeśli korzystamy dodatkowo ze stacji dysków, to wciskamy klawisz z cyfrą 2. Kolejne czynności, jakie należy wykonać przy kopiowaniu dyskietek będą wyświetlane na ekranie monitora. Poprzez jedno uruchomienie programu DYSKCOPY możemy przekopiować dowolną ilość dyskietek.

Życzymy powodzenia w korzystaniu z prezentowanego oprogramowania użytkowego dla CPC AMSTRAD 6128.

oprac. J. SZANIAWSKI

```
10 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
20 REM $                                            $
30 REM $              PROGRAM  KATALOG              $
40 REM $                                            $
50 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
60 POKE &BDEE,&C9:KEY DEF 66,0,0,0,0:ON ERROR GOTO 710
70 IF HIMEM<>&FFF THEN OPENOUT"d":MEMORY &FFF:CLOSEOUT
:LOAD "progbin.bin",&1000
80 MODE 2:INK 0,13:INK 1,1:BORDER 13:ZONE 21:PRINT "KA
TALOG (C) - PROGRAM DOKONUJE ZMIAN NA POZIOMIE ZBIOROW
 NA DYSKIETCE":WINDOW 1,80,3,25:WINDOW #1,1,80,22,25
90 PRINT"WLOZ DYSK DO STACJI A: I NACISNIJ DOWOLNY KLA
WISZ":WHILE INKEY$="":WEND
100 CLS:CALL &1000,0:IF PEEK(&100C)<>0 THEN 90
110 sect=PEEK(&100D):IF sect<9 THEN PRINT" IBM ";:sect
=1:track=1:GOTO 140
120 IF sect<&4A THEN PRINT" SYSTEM ";:sect=&41:track=2
:GOTO 140
130 PRINT" DATA ";:sect=&C1:track=0
140 PRINT"FORMAT":PRINT
150 buff=&1100:FOR x=0 TO 3:CALL &1003,buff,sect,track
:buff=buff+512:sect=sect+1:NEXT:sect=sect-4
160 buff=&1101:l=1:lst=33
170 WHILE l<lst:PRINT l" ";:IF l<10 THEN PRINT" ";
180 GOSUB 190:buff=buff+24:l=l+1:WEND:l=l-32:GOTO 290
190 u=PEEK(buff-1):s=PEEK(buff)
200 FOR x=0 TO 7:PRINT CHR$(1)CHR$(PEEK(buff));:buff=b
uff+1:NEXT:
210 PRINT".";:r=PEEK(buff):d=PEEK(buff+1):f=PEEK(buff+
2)
220 r$="    ":d$="    "
230 IF s=&E5 THEN u$="PUSTY     ":GOTO 280 ELSE IF u=&E
5 THEN u$="SKASOWANY":GOTO 260 ELSE u$="USER"+STR$(u)+
" ":IF LEN(u$)<8 THEN u$=u$+" "
240 IF r>127 THEN r$="R/O " ELSE r$="R/W "
250 IF d>127 THEN d$="SYS " ELSE d$="DIR "
260 IF d>127 THEN d=d-128
270 IF r>127 THEN r=r-128
280 PRINT CHR$(1)CHR$(r)CHR$(1)CHR$(d)CHR$(1)CHR$(f)"
 "u$r$d$,:RETURN
290 WINDOW SWAP 0,1:CLS
300 PRINT" Z...Zmiana nazwy   K...Kasowanie        O...
Odzyskiwanie    S...Zmiana na SYS    D...Zmiana na DIR
  R...Zmiana na R/O   W...Zmiana na R/W   U...Zmiana U
SER'a    P...Zmiana strony  N...Nagranie zmian  I...In
na dyskietka  X...Ucieczka"
310 PRINT"";
320 i$="":WHILE i$="":i$=INKEY$:i$=UPPER$(i$):WEND
330 RESTORE 700:opt=1:FOR o=2 TO 13:READ o$:IF o$=i$ T
HEN opt=o
340 NEXT:ON opt GOTO 320,350,440,490,500,540,580,590,6
00,610,640,670,680
350 CLS:LINE INPUT ;"Nowa nazwa zbioru...?  ",nn$:lnn=
LEN(nn$):IF lnn=0 OR lnn>12 THEN 350
360 dot=INSTR(1,nn$,"."):IF INSTR(dot+1,nn$,".")<>0 TH
EN GOTO 350
370 IF dot=0 THEN dot=lnn+1
380 name$=LEFT$(nn$,dot-1):ext$=MID$(nn$,dot+1,lnn):IF
 LEN(name$)>8 OR LEN(ext$)>3 THEN 350
390 IF LEN(name$)<8 THEN name$=name$+" ":GOTO 390
400 IF LEN(ext$)<3 THEN ext$=ext$+" ":GOTO 400
410 GOSUB 690
420 nn$=UPPER$(name$+ext$):FOR x=0 TO 10:POKE buff+x,A
SC(MID$(nn$,x+1)):NEXT
430 GOTO 470
440 np=&E5
450 GOSUB 690
460 POKE buff-1,np
470 IF f>32 THEN f=f-32

480 WINDOW SWAP 0,1:LOCATE 48-(B40(f/2-INT(f/2))),2+CI
NT(f/2):GOSUB 190:GOTO 290
490 np=0:GOTO 450
500 off=9
510 GOSUB 690:np=PEEK (buff+off)
520 IF np<128 THEN POKE (buff+off),np+128
530 GOTO 470
540 off=9
550 GOSUB 690:np=PEEK (buff+off)
560 IF np>127 THEN POKE (buff+off),np-128
570 GOTO 470
580 off=8:GOTO 510
590 off=8:GOTO 550
```

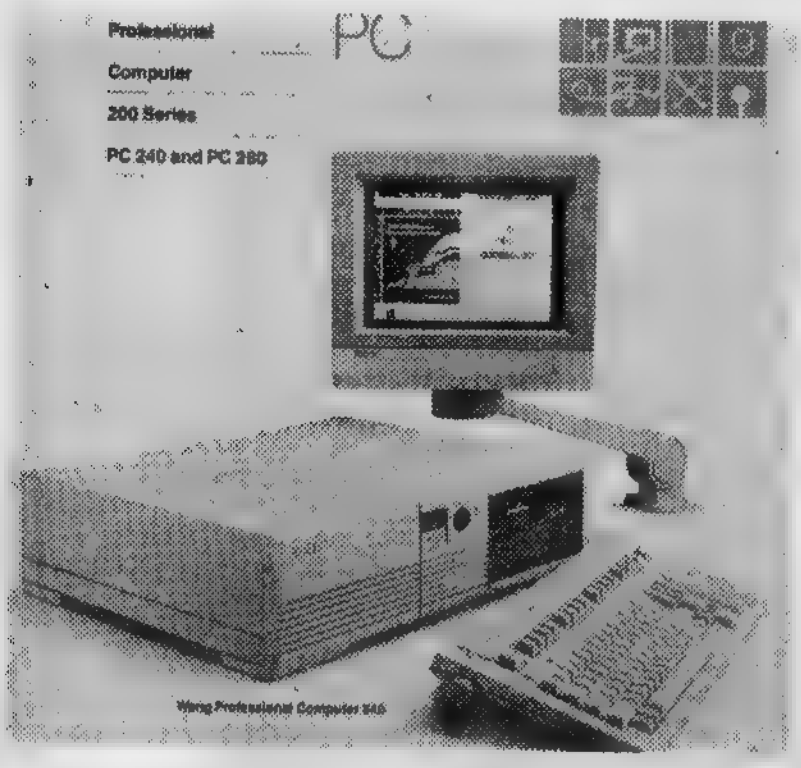
```
600 CLS:INPUT "Podaj numer nowego USER'a - ",np:IF np<
0 OR n>15 THEN 600 ELSE 450
610 WINDOW SWAP 0,1:LOCATE 1,3:PRINT"";
620 IF l=1 THEN l=33:lst=65:buff=&1501 ELSE l=1:lst=33
:buff=&1101
630 GOTO 170
640 CLS:PRINT TAB(33)"Nagranie zmian":PRINT TAB(34)" O
K. ? (T/N) ";:i$="":WHILE i$<>"N" AND i$<>"T":i$=INKEY
$:i$=UPPER$(i$):WEND:IF i$="N" THEN 660
650 buff=&1100:FOR x=0 TO 3:CALL &1009,buff,sect,0,tra
ck:buff=buff+512:sect=sect+1:NEXT:sect=sect-4
660 WINDOW SWAP 0,1:GOTO 290
670 RUN
680 CALL 0
690 CLS:INPUT " Podaj numer zbioru - ",f:IF f<1 OR f>l
+31 THEN 690 ELSE buff=(f-1)*32+&1101:CLS:RETURN
700 DATA Z,K,Q,S,D,R,W,U,P,N,I,X
710 RUN
```

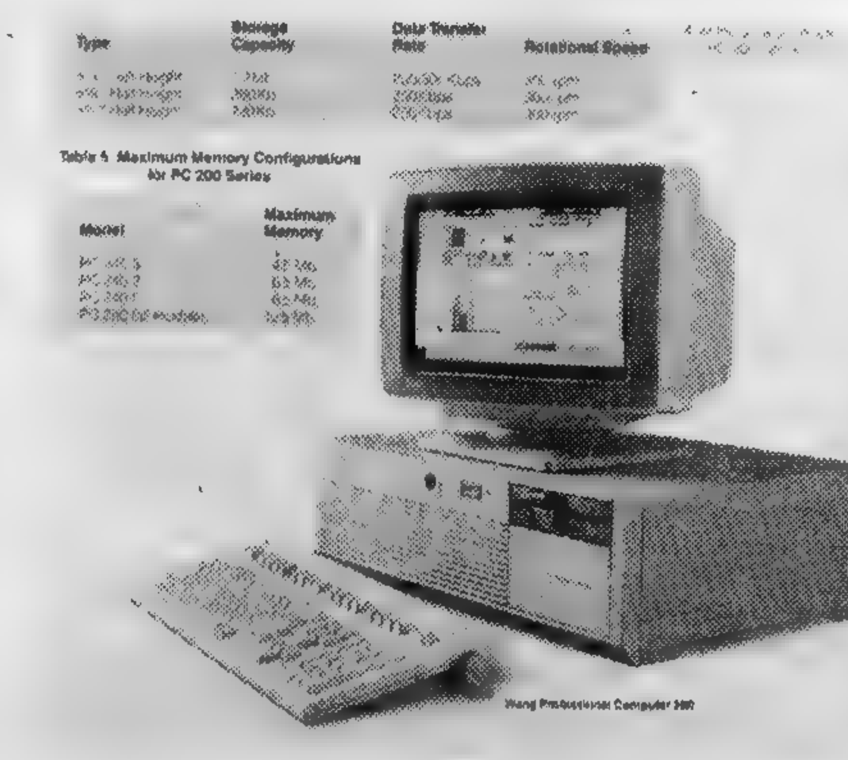
```
1 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2 REM $                                              $
3 REM $  UWAGA!                                      $
4 REM $  W liniach 50, 160, 240, w miejsce znaku "+" $

5 REM $ nalezy wpisac znak, wciskajac jednoczesnie   $
6 REM $ klawisz CONTROL oraz klawisz z litera "t"    $
7 REM $                                              $
8 REM $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
10 POKE &BDEE,&C9:KEY DEF 66,0,0,0,0:ON ERROR GOTO 320
20 IF HIMEM<>&FEE THEN OPENOUT"d":MEMORY &FEE:CLOSEOUT
:LOAD"progbin.bin",&1000:FOR x=&FEF TO &FFF:READ a:POK
E x,a:NEXT:CALL &FEF:DATA &3E,&FF,&DF,&FA,&0F,&3E,&01,
&DF,&FD,&0F,&C9,&72,&CA,&07,&03,&C6,&07
30 MODE 2:INK 0,13:INK 1,0:BORDER 13:ZONE 40:PRINT"DYS
KCOPY - (C) PROGRAM KOPIUJACY DYSKIETKI":WINDOW 1,80,3
,80
40 PRINT"PODAJ ILOSC STACJI DYSKOW  ?";:i$="":WHILE i$
<>"1" AND i$<>"2":i$=INKEY$:WEND:PRINT CHR$(8)i$:d=VAL
(i$)-1:PRINT:xp=POS(#0):yp=VPOS(#0)
45 DIM f(7,8),n(7)
50 LOCATE xp,yp:PRINT"+WLOZ DYSK ZRODLOWY DO A: ";:IF
d=1 THEN PRINT"a DYSK NA ZAPIS DO B: ";
60 PRINT"WCISNIJ DOWOLNY KLAWISZ"CHR$(13);:WHILE INKEY
$="":WEND
80 FOR g=0 TO 4:buff=&1100
90 FOR t=0 TO 7:track=g*8+t:go=1
100 CALL &1000,track:IF PEEK(&100C)<>0 THEN go=go+1:IF
 go<>101 THEN 100 ELSE n(t)=0:GOTO 160
110 mi=256:y=&100D:yy=y
120 p=PEEK(y):IF p=mi THEN 150
130 IF p<mi THEN mi=p:yy=y
140 y=y+1:GOTO 120
150 n(t)=y-yy:IF n(t)>9 OR (n(t)<8 AND n(t)<>0) THEN n
(t)=9
160 LOCATE xp,yp:PRINT"+CZYTANIE SCIEZKI"track" ";:IF
track<10 THEN PRINT" ";
170 PRINT"SEKTORY:  ";:IF n(t)=0 THEN PRINT"BLEDNE"
180 FOR s=0 TO n(t)-1:f(t,s)=PEEK(s+yy):PRINT HEX$(f(t
,s),2)" ";
190 CALL &FEF:CALL &1003,buff,f(t,s),track:buff=buff+5
12:NEXT
200 NEXT:buff=&1100:CALL &1000,0
210 IF d=0 THEN PRINT:PRINT:PRINT"WLOZ DYSK NA ZAPIS D
O A: - WCISNIJ DOWOLNY KLAWISZ":WHILE INKEY$="":WEND
220 FOR t=0 TO 7:track=g*8+t:IF n(t)=0 THEN 240
230 CALL &1006,f(t,0),f(t,1),f(t,2),f(t,3),f(t,4),f(t,
5),f(t,6),f(t,7),f(t,8),n(t),d,track
240 LOCATE xp,yp:PRINT"+ZAPISUJE SCIEZKE"track" ";:IF
track<10 THEN PRINT" ";
250 PRINT"SEKTORY:  ";:IF n(t)=0 THEN PRINT"BLEDNE"
260 FOR s=0 TO n(t)-1:CALL &1009,buff,f(t,s),d,track:P
RINT HEX$(f(t,s),2)" ";:buff=buff+512:NEXT:PRINT CHR$(
13);
270 NEXT
280 IF d=0 AND track <39 THEN PRINT:PRINT:PRINT"WLOZ D
YSK ZRODLOWY DO A: - WCISNIJ DOWOLNY KLAWISZ"CHR$(13);
:WHILE INKEY$="":WEND
290 NEXT
300 PRINT:PRINT:PRINT"CZY KOPIUJESZ NASTEPNY DYSK (T/N
) ?                        ";:WHILE i$<>"T" AND i$<>"N":
i$=INKEY$:i$=UPPER$(i$):WEND:PRINT i$
310 IF i$="T" THEN 50 ELSE CALL 0
320 RUN
```

```
10 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
20 $                                           $
30 $              PROGRAM JOB                  $
40 $                                           $
50 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
60 MODE 1
70 PRINT "        PROGRAMY NARZEDZIOWE "
80 PRINT "       ======================= "
90 PRINT:PRINT:PRINT
100 PRINT "      DYSKMAPA...............1"
110 PRINT
120 PRINT "      SEKTORED...............2"
130 PRINT
140 PRINT "      KATALOG................3"
150 PRINT
160 PRINT "      DYSKCOPY...............4"
170 LOCATE 1,22:PRINT "WCISNIJ KLAWISZ Z NUM
EREM PROGRAMU (1-4)"
180 I$=INKEY$: IF I$="" THEN 180
190 IF I$="1" THEN 240
200 IF I$="2" THEN 250
210 IF I$="3" THEN 260
220 IF I$="4" THEN 270
230 GOTO 180
240 RUN"!DYSKMAPA
250 RUN"!SEKTORED
260 RUN"!KATALOG
270 RUN"!DYSKCOPY
```

```
2 REM *****************
3 REM *               *
4 REM * SPADAJACA CEGLA *
5 REM *     ZX-81     *
6 REM *               *
7 REM *****************
8 LET A=0
10 LET B=10
15 LET C=0
20 LET A$="
                    "
25 LET D=0
30 LET E=INT(RND*15)
35 LET C=C+1
40 IF C=20 THEN GOTO 130
45 PRINT AT D,E;" "
50 PRINT AT 11,B;"W"
55 PRINT AT D+1,E;"█"
60 LET D=D+1
65 IF D=12 THEN GOTO 75
70 GOTO 85
75 LET A$(B+1)="█"
80 GOTO 25
85 PRINT AT 11,B;" "
90 PRINT AT 12,0;A$
95 IF INKEY$="5" THEN LET
B=B-1
100 IF INKEY$="8" THEN LE
T B=B+1
105 IF E=B AND D=11 THEN
GOTO 115
110 GOTO 45
115 LET A=A+1
120 LET A$(B+1)=" "
125 GOTO 25
130 PRINT " TRAFIONO: ";A
;" RAZY"
```





TAJEMNICA FORTU WRIGHT-PETTERSON

W niewielkim hallu było ciasno i duszno. Tę duchotę potęgowało jeszcze podniecenie emanujące z twarzy oczekujących, ich zachowanie i rozmowy. Co rusz ktoś spoglądał na obite dekoracyjnie czarną skórą drzwi mieszczącego się obok gabinetu. Każda ze znajdujących się tu osób dałaby wiele, żeby znaleźć się po ich drugiej stronie.

Zza dźwiękoszczelnych drzwi nie docierał jednak żaden odgłos; ani strzępek rozmowy, ani nawet stukot maszyny do pisania. Ponure i ciemne, skutecznie strzegły tajemnicy wnętrza.

— Do licha, sterczymy tu już dwie godziny i ani o krok nie posunęliśmy się do przodu. — Arian Robertson, sława miejscowej popołudniówki „Dayle News", zmiął niedopałek kolejnego papierosa. Czuł jak z każdą chwilą ulatniało się jego, zawodowe przecież, opanowanie. Kolejny dzień i kolejna porażka. Przychodził tu już czwarty raz w tym tygodniu i tak jak wszyscy, chciał wreszcie dowiedzieć się czegoś więcej. Czegoś sensacyjnego, odkrywczego. Rozwikłać tę pasjonującą tajemnicę, o której wszyscy wiedzieli coś niecoś. Zdawali sobie sprawę, że ujawnienie faktów może wywołać rewolucję w świadomości społeczeństw.

Czwarty raz i kolejna klapa. Żadnych informacji. Co wymyśleć, co napisać, żeby uspokoić i zadowolić opinię publiczną. Jak wywiązać się z obowiązków, skoro nie wie nic, nie ma żadnych danych. Tych kilku sloganów, które powtarzają od poniedziałku wszystkie dzienniki, ani nikt nie potwierdził, ani nie zdementował. Co za idiotyczna sytuacja. Minął już prawie tydzień od wydarzeń, a oni wciąż nie znają prawdy. Zresztą... Cóż to takiego jest, prawda? Co jest prawdziwe? Ciągnie każdy tę prawdę na swoją stronę, okrawa, spłaszcza, przecedza przez sito własnych imaginacji, zapamiętuje to, co mu odpowiada, co pasuje do jego układanki obrazka prawdy i świata.

Poirytowany i zniechęcony zaczął przeciskać się w kierunku wyjścia.

— Ari, spotkajmy się wieczorem w klubie — Peggy Norton filuternie zastąpiła mu przejście.

— Daj spokój, innym razem — odburknął. Nie miał dziś ochoty ani na bar, ani na jej towarzystwo. Wychodził już z sali, kiedy zaskoczyła go cisza za plecami. Rozgadane, nerwowe towarzystwo nagle zamarło w milczeniu. Podejrzliwie odwrócił się i przez ramię, kątem oka, spojrzał w kierunku drzwi. Były uchylone... Nareszcie...! Otworzyły się w chwili, kiedy zdecydowany był już odejść. Ech, gdzie jego dziennikarski instynkt...? Przecież gdyby wyszedł minutę wcześniej...

Snop ostrego światła z gabinetu oświetlił stojącą w nich sylwetkę niewysokiego mężczyzny z rudawą bródką i okularach w złoconej oprawce. Zza szkieł wodziły po zebranych bystre oczy. Wydawało się, że żaden szczegół nie jest w stanie ujść ich uwagi.

Arian bezwiednie zbliżył się do wchodzącego. Nie chciał uronić ni słowa. Profesor Stringfield, bo to był właśnie ów słynny Stringfield, którego nazwisko od poniedziałku było na ustach niemal całego świata, raz jeszcze obrzucił zebranych badawczym spojrzeniem. Przez krótką chwilę, jakby zastanawiał się, czy rzeczywiście powinien podzielić się z tymi ludźmi swoimi informacjami. Odetchnął w końcu i stłumionym nieco głosem rozpoczął.

— Drodzy państwo, raz jeszcze pragnąłbym potwierdzić moje poniedziałkowe wystąpienie na sympozjum naukowym. Oświadczam państwu, że jestem w posiadaniu zdjęć, które mogą udokumentować wszystkie moje słowa, nie ma w tym nic z szalbierstwa, przyrzekam. Fotografie te ukazują obcych humanoidów. Tak, nie jest to nieporozumienie.

— Ależ dowody, profesorze, dowody!

— Prosimy o zdjęcia!

— Proszę nam przedstawić dokumenty!

Gwar, jaki zapanował w hallu, był reakcją nie tyle na sensacyjne wiadomości, co na fakt, że ktoś wreszcie przerwał ten zaklęty krąg milczenia. Zebrani dziennikarze chcieli dowiedzieć się wszystkiego. I to tu, teraz. Natychmiast. Przekrzykiwali się wzajemnie, rzucali pytanie po pytaniu. Leonard Stringfield, z niezmąconym spokojem, podniósł obie ręce do góry i gdy się nieco uspokoili, powtórzył:

— Tak, tak. Mówię o forcie Wright-Patterson. Bo tam właśnie znajdują się ciała przybyszów z kosmosu. Są dobrze zabezpieczone w szklanych pojemnikach wypełnionych ciekłym azotem. Nie grozi im zniszczenie. O wszystkim się zresztą państwo dowiedzą osobiście dziś wieczorem, na konferencji prasowej zorganizowanej przez pełnomocnika ministerstwa.

— Profesorze, więcej szczegółów! — rozległy się zewsząd głosy.

— Panowie, to wszystko co mogę wam w tej chwili zakomunikować. Wieczorem, reszta wieczorem. — Profesor spojrzał jeszcze w głąb sali i prawie nieznacznie skinął głową. Arianowi gest ten skojarzył się natychmiast z dawaniem komuś tajemnego znaku. Odwrócił się. Za nim stała Peggy. Z jej twarzy znikł porozumiewawczy uśmiech.

* * *

W samochodzie Robertson raz jeszcze wracał pamięcią do ostatnich wydarzeń. I to dziwne zachowanie się Peggy... Sama proponowała mu spotkanie, a w chwilę potem, kiedy zbliżył się do niej i objął ramieniem, rozdrażniona odepchnęła go i jakby z wymówką, wymyśloną na poczekaniu, dlatego mało przekonującą, podbiegła w kierunku szybkobieżnej windy.

Dogonił ją, czuł, że trafił na jakiś ślad, nie chciał, żeby mu się to jeszcze bliżej niesprecyzowane „coś" wymknęło. Peggy mogła być kluczem do tej zagadki. Tuż przed drzwiami złapał ją raz jeszcze za ramię. Nie mógł jej stracić z oczu.

— Daj spokój, innym razem — powtórzyła z ironią w głosie jego niedawne słowa. Zdjęła dłoń Ariana ze swojego ramienia i zatrzasnęła drzwi. Na które piętro pojedzie? Do kogo? Wpatrywał się w napięciu w przeskakujące nad drzwiami dźwigu świetlne cyferki. Co się jej stało? Nigdy przecież nie zachowywała się tak. Od dwóch lat byli dobrymi kumplami i takie tam drobiazgi, jak dotąd, nie miały znaczenia. Dlaczego więc dziś nagle zawzięła się na niego? Urażona ambicja?

Gwałtownie skręcił przed jakąś ciężarówką i wjechał w boczną ulicę, przy której mieściła się redakcja gazety. Obiecał coś ekstra, a wraca z niczym. Jak się z tego wywinąć? Już wie. Powie szefowi, że ma murowany hit do jutrzejszego numeru. Po wieczornej konferencji nie da już jej się tak łatwo spławić. Może nawet lepiej, że odjechała sama. Bez osób trzecich, w nieskrępowanej atmosferze, łatwiej nawiązać kontakt z naukowcem. Był bowiem pewien, że Peggy udała się na spotkanie z profesorem Stringfieldem.

— Cześć Ari, no i co z twoimi sensacjami? — Ironiczny głos bossa nie wróżył nic dobrego. — Już prawie tydzień obiecujesz mi historię nie z tej ziemi, a ja wciąż nie mam nic rewelacyjnego na pierwszą stronę — Charlie częstował go niezmiennie owym przywitaniem, od kiedy rozpoczął pracę w tym nędznym szmatławcu. Wiele razy chciał już rzucić to nie dające mu satysfakcji zajęcie, ale w momentach determinacji trafiała się niespodziewanie „prawdziwa bomba"... Tym sposobem trwał już tutaj prawie siódmy rok.

— I będzie. Na pewno. Umówmy się na rano. Wiesz, że zawsze dotrzymuję słowa. Rano dostaniesz materiał z pierwszej ręki, na sto procent. — Grał va banque. W notesie nie miał nic, w głowie zupełną pustkę. Wierzył jednak, że dzisiejsza konferencja rozstrzygnie o wszystkim. W końcu muszą przecież coś ujawnić.

— Charlie, stary, daj jeszcze setkę i odlicz ją z mojego honorarium. — Charlie, zadziwiająco dzisiaj hojny, bez sprzeciwu sięgnął do kieszeni marynarki. Z podniszczonego portfela wyjął dwie pięćdziesiątki.

— Wiesz, że jeśli dobrze pójdzie, to to będzie tylko zaliczka — szef uśmiechnął się porozumiewawczo i jakby w jednej chwili zapomniał o ich rozmowie. Zniknął za pulpitem komputera zastępującego niegdysiejsze segregatory.

Wieczór zapowiadał się nieźle. Mógł wreszcie wrócić do domu. Miał jeszcze dość czasu na kolację, gorącą kąpiel i zmianę koszuli. Z trudem zdecydował się na kolejność tych czynności. Był potwornie głodny i piekielnie zmęczony.

* * *

— Panowie, mam do zakomunikowania przykrą wiadomość — prezes Stowarzyszenia Futurologicznego wyraźnie zwracał się do męskiej części zebranych, choć wśród nich znajdowało się również kilka kobiet.

— Z przykrością muszę państwu oznajmić — poprawił się po chwili — że nasz znakomity kolega, wieloletni członek

stowarzyszenia, wybitny naukowiec, profesor Leonard Stringfield, nie żyje. — W głosie profesora Toubala nie było żadnej emocji. Jakby informował zebranych, że właśnie na zewnątrz zaczął padać deszcz.

— Uległ wypadkowi samochodowemu w drodze na naszą konferencję. Właśnie dostałem telefon z kliniki pourazowej. Osiem minut temu, nasz nieodżałowanej pamięci kolega, nie odzyskawszy przytomności, zmarł. W tej sytuacji sami państwo rozumieją... konferencja nie odbędzie się.

— Sala wypełniła się groźnym hałasem.

— A zresztą, to był tylko propagandowy chwyt naszego szanownego kolegi — próbował przekrzyczeć zebranych — który w sposób, powiedziałbym mało elegancki, chciał zdobyć sobie popularność w świecie nauki i przejść do historii jako ojciec niekonwencjonalnych odkryć. Nie chciałbym, w takiej chwili, pomniejszać naukowych osiągnięć mojego kolegi, ale sami państwo rozumieją, to było zwykłe nieporozumienie. Niesmaczny żart.

Wypowiedź wstrząsnęła wszystkimi. Zaczęto snuć różnorodne spekulacje.

* * *

Arian siedział teraz w fotelu ze szklanką ginu. Mógł przecież to wszystko przewidzieć. Dlaczego takiego obrotu sprawy nie założył? Nie trzeba by było w ogóle ruszać się z domu. Z tego wygodnego fotela. Takie niewiarygodne sprawy muszą mieć przecież tak idiotyczne zakończenie. Ale zaraz, zaraz...? Że też nie pomyślał o tym wcześniej... Peggy... To chyba przez ten otępiający alkohol. Peggy... Co się dzieje z dziewczyną? Nie było jej wśród oczekujących na konferencję. A czekali przecież dość długo na jej rozpoczęcie. Może nie zauważył jej, podekscytowany czekającymi go wydarzeniami?

Nie, to niemożliwe. Dlaczego więc nie przyszła? Jeśli była z profesorem...? Wykręcał już kolejny raz numer jej telefonu. Przeciągły sygnał monotonnie dźwięczał w słuchawce. Cholera, gdzie też ona się podziewa o tej porze? Na pewno wcześniej dowiedziała się o wypadku i szuka jakiegoś zaczepienia. Zbyt dziwny to przypadek.

A może...? Nie, to przecież niemożliwe. Chociaż...? Jeśli jechała razem z profesorem na tę konferencję...? W radiu podawali, że w wypadku zginęły trzy osoby, ale do diabła...! Nie, czyżby mogła być wśród nich Peggy...?

Zadzwonił do szpitala, do którego przywieźli ofiary wypadku. W rejestrze rannych, ani zabitych w dzisiejszym dniu, nie figurowało nazwisko Peggy. Odetchnął z ulgą. Ale gdzie ona w takim razie może teraz być? Znowu zaczął się nad tym zastanawiać. Rozmyślania przerwał mu jednak dzwonek do drzwi.

— Pan Arni? — zupełnie obcy mężczyzna wyciągnął do niego dłoń. — Nazwiska niestety nie znam. Jestem Horst. Horst Wermer. Wracam prosto z lotniska. Jestem trochę zaszokowany. Wszystko stało się tak nagle. Peggy nawet nie zabrała ze sobą walizek. Wzięła tylko do podręcznej torby najpotrzebniejsze rzeczy. Myślę, że to kolejny jej żart, czy też wariacki wybryk. Nigdy nie znała w nich umiaru, prawda?

Potok słów nieznajomego, jakby z opóźnieniem docierał do niego. Nim się odezwał, mężczyzna wcisnął mu jeszcze do ręki dużą, szarą kopertę. Raz jeszcze na pożegnanie potrząsnął jego dłonią...

— To od Peggy. Prosiła, abym dziś jeszcze dostarczył to panu. Nie wydawało mi się to takie pilne, ale dałem słowo. Acha, no i najważniejsze. Wyjechała na rok do Afryki. Tak przynajmniej powiedziała. Rozumie pan coś z tego? Z podręczną torbą na drugi koniec świata. Nigdy nie rozumiałem tej kobiety — głos jego dobiegał za wciąż jeszcze otwartymi drzwiami. Nieznajomy odszedł równie szybko, jak się pojawił. Jedynym świadectwem jego tu obecności była szara, zaklejona koperta. Bez adresu, bez nazwiska, bez podpisu. Cała Peggy...

* * *

Wyjął z koperty kasetę. Pełen wątpliwości i sprzecznych myśli włączył magnetowid. Na ekranie pojawiły się kolorowe pasy, szumy... Po chwili zaś... Tak, to wnętrze gabinetu profesora Stringfielda. Poznał je od razu. Był w nim kiedyś, podczas reportażu o UFO... Na skórzanej kanapie Peggy, obok w fotelu, Stringfield. Między nimi, na niskim, szerokim stoliku stos zdjęć, jakieś dokumenty. Obok kaseta wideo, odtwarzacz.

— A więc profesorze, to nie fantazja? — głos Peggy zabrzmiał ze szklanego ekranu.

— Kochane dziecko, kiedy pani przeżyje już tyle lat co ja, niemożliwe wyda się pani możliwym. I odwrotnie. — Profesor mówił z lekkim uśmiechem na twarzy. — Mówiłem już, że mam dowody. Są to zmaterializowane, namacalne. Jak pani wiadomo opowiadałem o wypadku rozbicia się nieznanego pojazdu na Ziemi. Ostatni z nich miał miejsce właśnie w Nowym Meksyku. Z miejsca tej katastrofy przewieziono ciała rozbitych humanoidów samochodami wojskowymi do fortu Wright-Patterson.

— A teraz proszę spojrzeć, dziecko — profesor wcisnął klawisz odtwarzacza.

Na ekranie widać było wyraźnie jakieś ciała, zamknięte w szklanych pojemnikach. Ciała „obcych" były nagie. Muskularne istoty miały duże głowy, bez nosów i uszu. Były całkowicie bezpłciowe, pozbawione owłosienia... Umięśnione ręce sięgały poniżej kolan...

— Widzi pani, oglądałem tego osobnika — profesor wskazał na zbliżenie jednej z „trumien". — Proszę sobie wyobrazić, że w chwili katastrofy był on przygnieciony szczątkami pojazdu. Wielesetkilogramowym blokiem żelastwa. A niech pani zwróci uwagę — żadnych ran, żadnych obrażeń...

Peggy oczy miała szerokie ze zdziwienia.

— Ależ profesorze, jak to możliwe?

— Sam się dziwiłem. Ich delikatna skóra, jak wykazały moje późniejsze badania, okazała się warstewką niezwykłego plastiku, nałożonego bezpośrednio na warstwę mięśniową. Jest on odporny na wszelkiego rodzaju uszkodzenia mechaniczne, termiczne, chemiczne i jakie pani jest w stanie sobie jeszcze wyobrazić.

Podczas badań stwierdziliśmy niepodważalnie, że mózgi owych istot sprawiają wrażenie, jakby należały do osobników w bardzo zaawansowanym wieku. Pięćset, może nawet siedemsetletnich. Ciała ich natomiast mają wygląd istot zupełnie młodych, zresztą sama pani widzi. A mimo to, nie stwierdziliśmy żadnych śladów przeszczepów mózgów.

Mój kolega, profesor Henderson, którego poważam i skłonny jestem przyjąć jego koncepcję, wysunął hipotezę nieprawdopodobną, acz możliwą. Otóż twierdzi on, że owe humanoidy, to nic innego jak... — profesor zawiesił na chwilę głos — jak roboty.

Proszę sobie wyobrazić, że istoty te, ponad wszelką wątpliwość, zachowują podstawowe czynności biologiczne, a przede wszystkim zdolności myślenia, kojarzenia, podejmowania decyzji. Nie są jednak istotami, w ścisłym znaczeniu, rozumnymi.

Dziewczyna raz po raz spoglądała to na profesora, to na ekran.

— To znaczy, że wykonują tylko czynności, jakby to powiedzieć naszym językiem...

— Zlecone im. Czy to chciał mi pan zasugerować? — zapytała Peggy.

— Nie zasugerować, droga pani. To przecież oczywiste, że istoty te mają doskonałe warunki do wykonywania bardzo dalekich, długotrwałych podróży kosmicznych. Trwających nawet tysiące lat...

— Tysiące, to niewiarygodne...?

— Dla nas tak, ale ich inna przemiana materii, brak absorbcji tlenu zastąpiony absorbcją azotu świadczą, że właśnie do takich misji je stworzono. Zamrożone do temperatury 179 stopni Celsjusza, praktycznie są nieśmiertelne..

Profesor mówił coś jeszcze. Peggy nerwowo gestykulowała, ale nie słychać już było głosu. Po chwili na ekranie pojawiły się pasy i obraz znikł.

Arian siedział jak osłupiały. Bomba, bomba, bomba. Tak, to jest sensacja. W forcie Wright-Patterson ukrywają dowody na istnienie pozaziemskich cywilizacji. Kto wie, ilu takich kosmicznych rozbitków tam się znajduje. Zaraz, zaraz... Jak to on powiedział? Zamrożone, są praktycznie nieśmiertelne? Cofnął taśmę. Włączył magnetowid. Nic... Przewinął raz jeszcze. Ponownie nic...

Nie, to niemożliwe... Skasowany cały zapis! Cały zapis tej jedynej, unikatowej rozmowy, został skasowany. Niewiarygodne... Jak to się mogło stać?

Ogarnęła go niepohamowana wściekłość. Krew napłynęła mu do twarzy. Podszedł do telefonu, wykręcił numer.

— Charlie...? Chciałem ci powiedzieć... Oszukali nas dranie... Ale to jest prawda... Mają tych rozbitków, widziałem ich... Nie, nie można tego udowodnić... Ale mają ich, mówię ci...

Odłożył słuchawkę. Nalał sobie do pełna ginu i ciężko opadł na fotel.

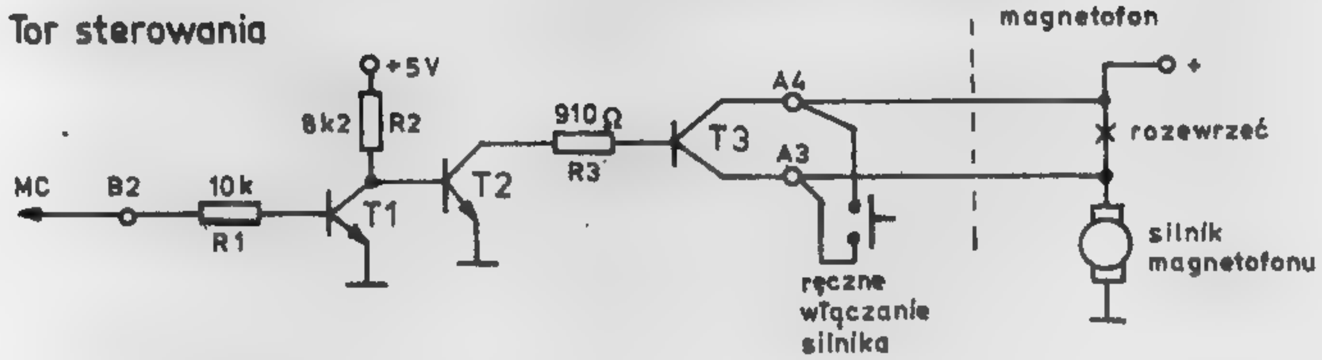
Anna KALETA

KASETOWY SYSTEM OPERACYJNY 2T06 TURBO cd.

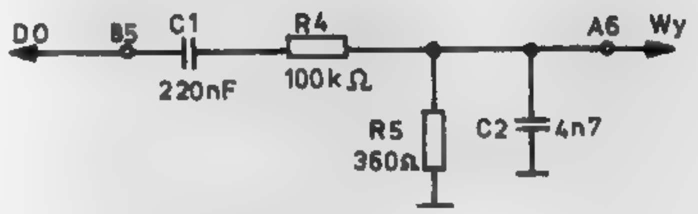
WOJCIECH ZABOŁOTNY

SCHEMAT INTERFEJSU

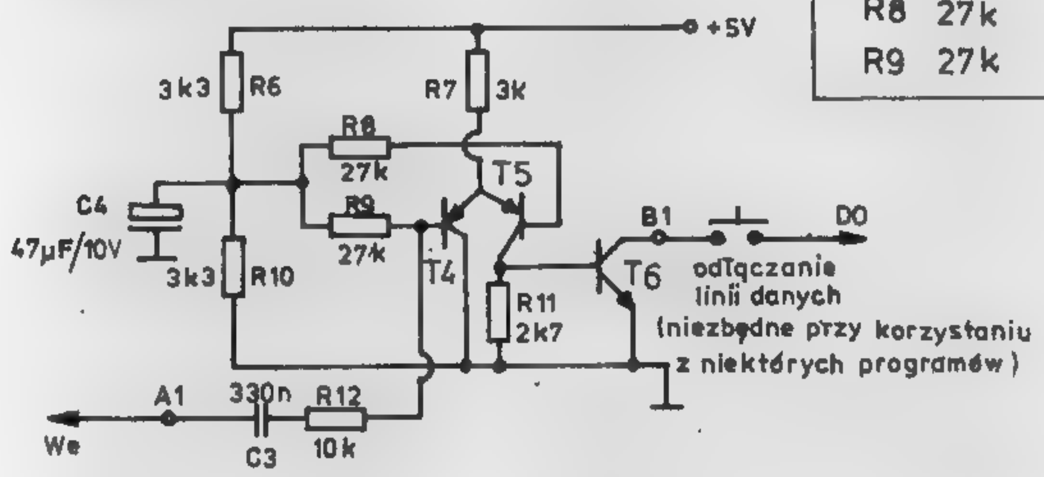
Tor sterowania



Tor zapisu

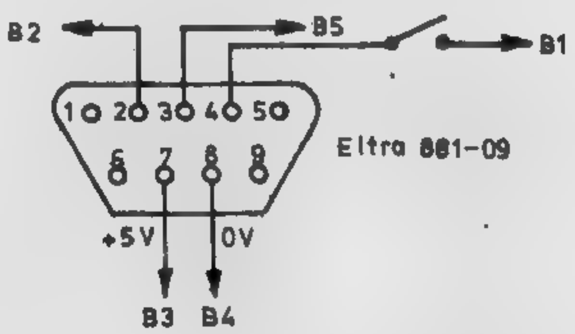


Tor odczytu

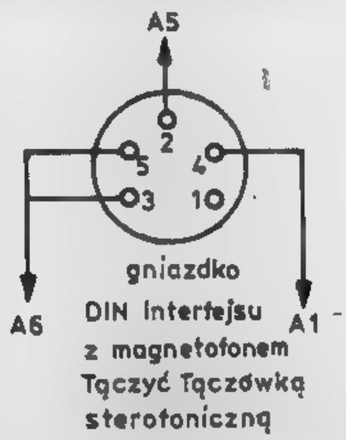


T_1, T_2 BC 147 148		
T_3 BD 136		
T_4, T_5 BC 157C, 158C lub podobne		
T_6 BC147, 148		
R1 10k	R10 3k3	C4 47μF/10V
R2 8k2	R11 2k7	
R3 910Ω	R12 10k	
R4 100kΩ		
R5 360Ω		
R6 3k3	C1 220nF	
R7 3k	C2 4n7	
R8 27k	C3 330nF	
R9 27k		

Podłączenie do komputera



Podłączenie do magnetofonu



Rys. 1

Ciąg dalszy na stronie 31

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

I

INDIVIDUAL COMPONENT ERROR - błąd cząstkowy,
INDIVISIBLE - niepodzielny,
INDUSTRIAL DATA ACQUISITION SYSTEM - przemysłowy system centralnej rejestracji danych,
INEFFECTIVE TIME - czas strat, czas przestoju, czas nieefektywny,
INEFFICIENT - niesprawny, niewydajny, nieefektywny,
INEQUALITY - nierówność,
INEQUIVALENCE - niejednoznaczność,
INESCAPABLE - nie do uniknięcia,
INEVITABLE ERROR - błąd nieunikniony,
INEXACT - niedokładny,
IN EXCESS - w nadmiarze,
INFER - wnioskować,
INFERENCE - 1. wniosek, konkluzja, 2. wnioskowanie,
INFERENCE CHAIN - ciąg wniosków, łańcuch wniosków,
INFERENCE ENGINE - mechanizm wnioskowania,
INFERENCE METHOD - metoda wnioskowania,
INFERENCE RULE - reguła wnioskowania, zasada wnioskowania,
INFERIOR - podwładny,
INFERIOR QUALITY - gorszy gatunek,
IN FIG. - na rysunku ...,
INFINITE - nieskończony,
INFINITELY SMALL - nieskończenie mały, elementarny,
INFINITE MEMORY FILTER - filtr z nieograniczoną pamięcią,
INFINITE POINT - punkt niewłaściwy,
INFINITESIMAL - nieskończenie mały, elementarny,
INFINITY - 1. nieskończoność, 2. liczba poza zakresem,
INFIX NOTATION - sposób zapisu wyrażeń arytmetycznych gdzie znak operacji binarnej umieszcza się pomiędzy operandami,
INFIX OPERATOR - operacja binarna, której znak zapisywany jest pomiędzy operandami,
INFLUENCE - wpływ, oddziaływanie,
INFO - patrz: INFORMATION FIELD,
INFORM - informować,
INFORMATICS - informatyka,
INFORMATION - informacja,
INFORMATION BIT - dwójkowa jednostka informacyjna, bit informacyjny,
INFORMATION BLOCK - blok informacyjny,
INFORMATION CAPACITY - pojemność informacyjna,
INFORMATION-CARRYING MEDIUM - nośnik informacji,
INFORMATION CENTRE - ośrodek informacji,
INFORMATION CHANNEL - kanał informacyjny,
INFORMATION CHARACTER - znak informacyjny, symbol informacyjny, znak tekstowy,
INFORMATION CONTENT - zawartość informacji, treść informacji,
INFORMATION FIELD - pole informacyjne,
INFORMATION FLOW - strumień informacji, przepływ informacji,
INFORMATION FLOW RATE - szybkość przesyłania informacji,
INFORMATION FORMAT - postać informacji,
INFORMATION HANDLING - przetwarzanie informacji,
INFORMATION HIDING - ukrycie informacji, zakrycie informacji,
INFORMATION INTERCHANGE - wymiana zakodowanej informacji między maszynami,
INFORMATION MANAGEMENT SYSTEM - hierarchiczny system zarządzania bazami danych opracowany przez firmę IBM,
INFORMATION MESSAGE - wiadomość informacyjna,
INFORMATION NOISE - szum informacyjny,
INFORMATION PATH - droga przesyłania informacji,
INFORMATION PROCESSING - przetwarzanie informacji,
INFORMATION-PROCESSING MACHINE - maszyna informacyjna,
INFORMATION RECEIVER - odbiorca (odbiornik) informacji,
INFORMATION REPRESENTATION - reprezentacja informacji, sposób przedstawienia informacji,
INFORMATION RETRIEVAL - wyszukiwanie informacji,
INFORMATION RETRIEVAL LANGUAGE - język informacyjny, język informacyjno-wyszukiwawczy,
INFORMATION RETRIEVAL SYSTEM - system informacyjno-wyszukiwawczy,
INFORMATION SCIENCE - informatyka,
INFORMATION SCIENTIST - informatyk,
INFORMATION SEPERATOR - znak rozdzielający, separator,
INFORMATION SOURCE - źródło informacji,
INFORMATION STORAGE - zapamiętywanie informacji,
INFORMATION SYSTEM - system informacyjny, system informowania,
INFORMATION TECHNOLOGY - technika informatyczna,
INFORMATION THEORY - teoria informacji,
INFORMATION TRACK - ścieżka informacyjna,
INFORMATION UNIT - jednostka (ilości) informacji,
IN FRONT - z przodu,
IN GENERAL - na ogół,
INHERENT REGULATION - samoregulacja, samowyrównywanie,
INHERENT STORE - pamięć samoczynna,
INHERITED ERROR - błąd propagowany,
INHIBIT - hamować, wstrzymywać, przeszkadzać, uniemożliwiać, blokować,
INHIBITING CIRCUIT - układ blokujący,
INHIBITING SIGNAL - sygnał zakazujący,
INHIBITORY GATE - bramka blokująca,
INHIBITORY INPUT - wejście blokujące,
IN-HOUSE - własny, wewnętrzny,
IN-HOUSE LINE - cząstkowa linia komunikacyjna, dołączona do sieci komputerowej ogólnego dostępu,
IN-HOUSE TRAINING - przygotowanie specjalistów z wykorzystaniem własnych sił i środków,
INITIAL - 1. początkowy, 2. inicjał,
INITIAL ADDRESS - adres początkowy,
INITIAL CONDITIONS - warunki początkowe,
INITIAL DATA - dane pierwotne, dane źródłowe,
INITIAL INSTANT - chwila początkowa,
INITIALISM - skrót literowy,
INITIALIZATION - inicjalizacja,
INITIALIZATION COMMAND WORD - słowo inicjujące,
INITIALIZE - 1. inicjować, rozpoczynać, zapoczątkować, 2. przygotować
INITIALIZER - program wprowadzający,
INITIALLY - początkowo,
INITIAL PROGRAM LOAD - ładowanie początkowe, program ładowania początkowego,
INITIAL PROGRAM LOADER - program ładowania początkowego, praprogram ładujący,
INITIAL VALUE - wartość początkowa,
INITIATE - inicjować, rozpoczynać, zapoczątkować, powstawać, wystawiać,
INK - atrament, tusz, farba drukarska,
INK BEAM RECORDER - pisak strumieniowy,
INK DENSITY - intensywność atramentu,
INKED RIBBON - taśma nasycona tuszem,
INKING - rysowanie, pozostawianie śladu (dot. grafiki),
INK-JET PRINTER - drukarka strumieniowa,
INK-JET RECORDING - metoda druku strumieniem rozpylonej farby,
IN-LINE - 1. wbudowany, wmontowany, 2. przyłączony, dołączony,
IN-LINE CHECK - sprawdzanie maszynowe, kontrola maszynowa,
IN-LINE CODE - rozkazy maszynowe,
IN-LINE PROCESSING - przetwarzanie (danych) swobodne,
IN-LINE SUBROUTINE - podprogram podstawialny, podprogram otwarty, (w czasie kompilacji jest wstawiany w miejsce swojego wywołania),
IN-LINE SUBROUTINE EXPANSION - wstawienie podprogramu w miejscu jego wywołania,
INMOS - firma angielska, zajmująca się opracowywaniem nowych komputerów (w firmie tej powstał między innymi transputer i język równoległego programowania Occam),
INMUTABLE FILE - plik (zbiór) stały, plik (zbiór) tylko do odczytu,
INNER FACE - wewnętrzna strona (np. taśmy dziurkowanej),
INNER LOOP - cykl wewnętrzny,
IN OPERATION - w eksploatacji,

INOPERATIVE - nieczynny, nie działający, niezdolny do wykonania czynności,
IN ORDER OF ... - w kolejności..., w celu...,
IN-OUT BLOCK - blok wejścia/wyjścia,
IN OUTLINE - w ogólnym zarysie,
IN PARALLEL - równolegle,
IN PARAMETER - parametr wejściowy,
IN PARTICULAR - w szczególności,
IN POSITION - na swoim miejscu, we właściwym położeniu,
IN PRACTICE - w praktyce,
IN PRESS - w druku,
INPUT - 1. wejście, 2. dane wejściowe, 3. urządzenie wejściowe, 4. sygnał wejściowy, 5. wprowadzać (np. dane),
INPUT AREA - obszar wejścia,
INPUT BLOCK - blok wejściowy,

INPUT BLOCK COUNT - liczenie bloków na wejściu,
INPUT BUFFER FULL - sygnał zapełnienia rejestru,
INPUT CHANNEL - kanał wejściowy,
INPUT DATA - dane wejściowe,
INPUT DEVICE - urządzenie wejściowe,
INPUT FILE - plik (zbiór) wejściowy,
INPUT INSTRUCTION - rozkaz wejścia,
INPUT LANGUAGE - język wejściowy, język zewnętrzny,
INPUT MODE - tryb wprowadzania,
INPUT-OUTPUT - wejście/wyjście (operacje przesyłania danych pomiędzy pamięcią operacyjną a urządzeniami zewnętrznymi),
INPUT/OUTPUT AREA - obszar wejścia/wyjścia,
INPUT-OUTPUT BLOCK - blok wejścia/wyjścia,
INPUT/OUTPUT CALL - wywołanie wejścia/wyjścia,
INPUT/OUTPUT CHANNEL - kanał wejściowo/wyjściowy,
INPUT/OUTPUT CHARACTER - znak wejścia/wyjścia,
INPUT/OUTPUT CONTROLLER - jednostka sterująca wejścia/wyjścia,
INPUT/OUTPUT CONTROL PROGRAM - patrz: INPUT/OUTPUT SYSTEM,
INPUT/OUTPUT CONTROL SYSTEM - patrz: INPUT/OUTPUT SYSTEM,
INPUT/OUTPUT OPERATION - operacja wejścia/wyjścia,
INPUT-OUTPUT SPECIFICATION - opis wejściowych i wyjściowych parametrów,
INPUT/OUTPUT SYSTEM - system wejścia/wyjścia, system sterowania wejścia/wyjścia, program sterowania wejścia/wyjścia,
INPUT/OUTPUT UNIT - urządzenie wejściowo/wyjściowe, jednostka obsługi wejścia/wyjścia,
INPUT PROCESS - wprowadzanie danych,
INPUT PROGRAM - program wprowadzania,
INPUT QUANTITY - wielkość wejściowa,
INPUT REGISTER - rejestr wejściowy,
INPUT SPEED - szybkość wprowadzania (informacji),
INPUT STREAM - strumień wejściowy,
INPUT UNIT - patrz: INPUT DEVICE,
INQUIRY - 1. żądanie informacji, zapytanie, 2. dochodzenie, badanie,
INQUIRY UNIT - urządzenie końcowe za pośrednictwem którego można żądać informacji od komputera i otrzymać odpowiedź,
INSCRIBE - wpisać, wypełniony dokument uzupełnić pismem czytelnym dla komputera,
INSCRIPTION - napis,
INSECURE TAPE - taśma nieskatalogowana, taśma niezabezpieczona,
IN SEQUENCE - kolejno, po kolei.

IN SERIES - szeregowo,
INSERT - 1. wkładać, wstawiać, wprowadzać, zakładać, 2. wstawka,
INSERTION - 1. wstawka, 2. wprowadzanie kart do maszyny,
INSERT MODE - tryb wstawiania (w edytorach ekranowych, znak wprowadzany z klawiatury, wstawiany jest przed znakiem aktualnie podświetlanym przez kursor),
IN SERVICE - w eksploatacji,
IN SERVICE REGISTER - rejestr obsługiwanego przerwania,
INSIDE - strona wewnętrzna, wnętrze,
IN SITU - na miejscu,
INSPECTION LEVEL - poziom kontroli,
INSTABILITY - niestabilność,
INSTALL - urządzić, instalować,
INSTALLATED TASK - zadanie zainstalowane, zadanie włączone, zadanie uruchomione,
INSTALLATION - urządzenie, instalacja,
INSTALLATION PARAMETER - parametr instalacji,
INSTALLATIONS DATE - data zainstalowania (uruchomienia),
INSTANCE - 1. egzemplarz, 2. przypadek,
INSTANT - chwila, moment,
INSTANTANEOUS - chwilowy, natychmiastowy, bezzwłoczny, momentalny,
INSTANTANEOUS VALUE - wartość chwilowa,
IN STEP - w takt, synchronicznie,
INSTITUTE OF ELECTRICAL AND ELECTRONICAL ENGINEERS - Instytut Inżynierów o Specjalności Elektrotechniki i Elektroniki,
INSTRUCT - instruować, szkolić,
INSTRUCTION - 1. instrukcja, rozkaz, 2. szkolenie, instruktaż,
INSTRUCTION ADDRESS - adres rozkazu,
INSTRUCTION CODE - kod rozkazowy,
INSTRUCTION COMPLEMENT - uzupełnienie rozkazu,
INSTRUCTION COUNTER - licznik rozkazów,
INSTRUCTION COUNTING REGISTER - rejestr licznika rozkazów,
INSTRUCTION CYCLE - cykl rozkazu,
INSTRUCTION DECODER - dekoder rozkazów, deszyfrator rozkazów,
INSTRUCTION EXECUTION - wykonanie rozkazu,
INSTRUCTION FETCH PHASE - faza wywoływania rozkazu,
INSTRUCTION FIELD - pole rozkazu,
INSTRUCTION FLOWCHART - schemat blokowy wykonywania instrukcji (rozkazów),
INSTRUCTION FORMAT - struktura rozkazu, budowa rozkazu,
INSTRUCTION FOR USE - instrukcja użytkowania,
INSTRUCTION INDEX - indeks rozkazu,
INSTRUCTION LENGTH - długość rozkazu,
INSTRUCTION LIST - patrz: INSTRUCTION SET,
INSTRUCTION LOOP - pętla rozkazowa, pętla (programowa),
INSTRUCTION MIX - mieszanka rozkazów (program zbudowany z rozkazów różnych typów, wykorzystywany do określenia szybkości działania procesora lub systemu komputerowego),
INSTRUCTION PART - część rozkazowa,
INSTRUCTION POINTER - wskaźnik rozkazów,
INSTRUCTION REGISTER - rejestr rozkazów,
INSTRUCTIONS DECODING - dekodowanie rozkazów,
INSTRUCTION SEQUENCE - sekwencja rozkazów,
INSTRUCTION SEQUENCE REGISTER - rejestr kolejności rozkazów,
INSTRUCTION SET - repertuar rozkazów, lista rozkazów,
INSTRUCTION STREAM - łańcuch rozkazów (ciąg rozkazów przenoszonych z pamięci do procesora),
INSTRUCTION SYSTEM - system rozkazowy,
INSTRUCTION TIME - czas wykonania rozkazu (maszynowego),
INSTRUCTION TYPE - typ rozkazu,
INSTRUCTION WORD - słowo rozkazowe,
INSTRUMENT - przyrząd,
INSTRUMENTATION - oprzyrządowanie,
IN SUCCESSION - kolejno,
INSULATE - izolować,
INSURE - upewniać się,
INSWAP - ładować, załadowywać,
IN-SYSTEM PERFORMANCE - funkcjonowanie części w danym systemie,
INT - patrz: 1. INTEGER, 2. INTERRUPT,
INTANGIBLE - niewymierny, nieuchwytny,
INTEGER (NUMBER) - liczba całkowita,
INTEGER PROGRAMMING - programowanie całkowitoliczbowe,
INTEGER-VALUED - całkowitoliczbowy,
INTEGER VARIABLE - zmienna liczba całkowita,
INTEGRAL - 1. całkowitoliczbowy, 2. wbudowany, wmontowany, 3. integralny, nierozdzielny, 4. całka,
INTEGRAL TYPE - typ całkowitoliczbowy (typ danych przedstawiający liczby całkowite),
INTEGRATE - łączyć w całość, scalać, komasować, integrować,
INTEGRATED CIRCUIT - układ scalony, obwód scalony,
INTEGRATED COMPUTING SYSTEM - zintegrowany system przetwarzania danych,
INTEGRATED DATA BASE - zintegrowana baza danych,
INTEGRATED DATA PROCESSING - zintegrowane przetwarzanie danych,
INTEGRATED DATA STORE - kompleksowe magazynowanie danych,
INTEGRATED ENVIRONMENT - środowisko zintegrowane, otoczenie zintegrowane,
INTEGRATED MANAGEMENT INFORMATION SYSTEM - zintegrowany system informacyjny kierownictwa,
INTEGRATED MODEM - modem wbudowany,
INTEGRATED PACKAGE - pakiet zintegrowany,
INTEGRATED SOFTWARE - patrz: INTEGRATED SYSTEM,
INTEGRATED SYSTEM - pakiet zintegrowany, system zintegrowany,
INTEGRATION - 1. łączenie w całość, 2. całkowanie,
INTEGRITY - całkowitość,
INTEL CORPORATION - firma amerykańska, opracowująca i produkująca między innymi elementy półprzewodnikowe (w firmie Intel wyprodukowano pierwsze mikroprocesory Intel 4004 i Intel 8008),
INTELLIGENCE - 1. inteligencja, 2. wiadomość, informacja (przekazywana), wywiad,
INTELLIGENCE QUOTIENT - iloraz inteligencji,
INTELLIGENT COMPUTER-ASSISTED INSTRUCTION - inteligentny system nauczania komputerowego,
INTELLIGENT DATA BASE - inteligentna baza danych,
INTELLIGENT TERMINAL - terminal inteligentny (programowany), urządzenie końcowe inteligentne (z pamięcią i mikroprocesorem),
INTELLIGIBLE - zrozumiały,
INTENSE - intensywny,
INTENSIFICATION - wzmacnianie, intensyfikacja,
INTENSIFIER - wzmacniacz,
INTENSIFY - intensyfikować, wzmacniać,
INTENSION OF A CONCEPT - treść pojęcia,
INTENSITY - natężenie, intensywność, nasilenie,
INTENSITY LEVEL - poziom intensywności świecenia,

LIGA MYŚLĄCYCH

ZADANIE 1

Turysta wsiadł w Warszawie do samolotu, który poleci po najkrótszej trasie do pewnego lotniska położonego na równiku. Osoby odprowadzające turystę zauważyły, że samolot wystartował w kierunku zachodnim. Należy podać ile kilometrów (w przybliżeniu) wynosi długość trasy samolotu?

ZADANIE 2

Dziadek i babcia mają w sumie 154 lata. Dziadek ma dwa razy tyle lat, ile babcia miała wtedy, kiedy dziadek miał tyle, ile babcia ma obecnie. Proszę podać ile lat ma dziadek, a ile lat ma babcia?

ZADANIE 3

Na szpulę kasety magnetofonowej o promieniu zewnętrznym r nawijana jest ze stałą prędkością liniową v taśma magnetofonowa o grubości d. Należy obliczyć ile zwojów n taśmy nawinie się w ciągu czasu t?

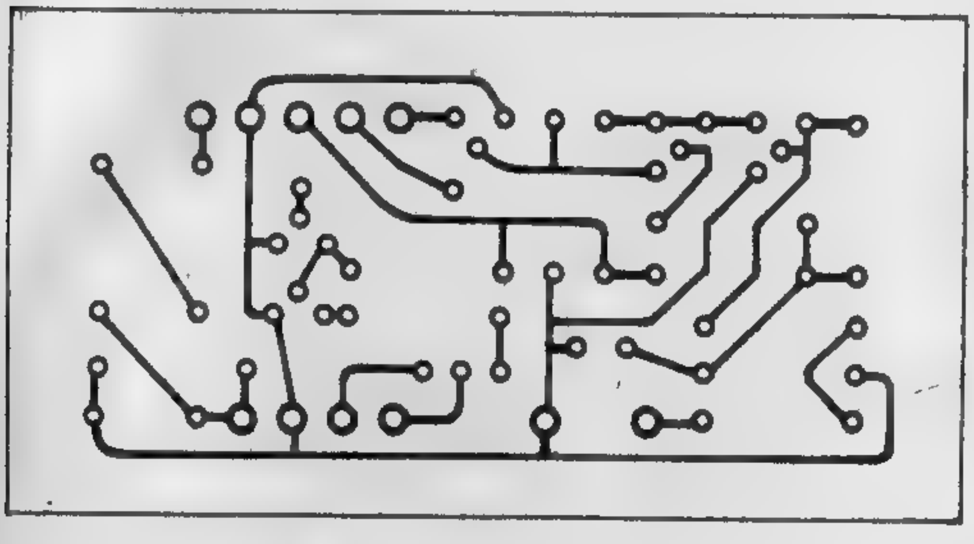
ZADANIE 4

W pewnym kraju afrykańskim, w wyniku nieurodzaju i klęsk żywiołowych podwyższono ceny na wszystkie towary, najpierw o 20 proc., a później jeszcze o 25 proc. O ile zmalała siła nabywcza pieniądza w tym kraju?

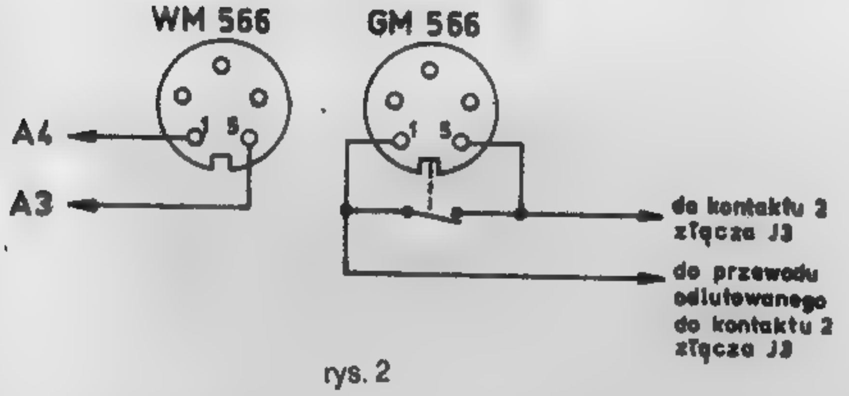
ZADANIE 5

Wiedząc, że liczba $a = x\,y\,2\,3\,4\,z$ jest podzielna przez 396 znaleźć cyfry x, y, z.

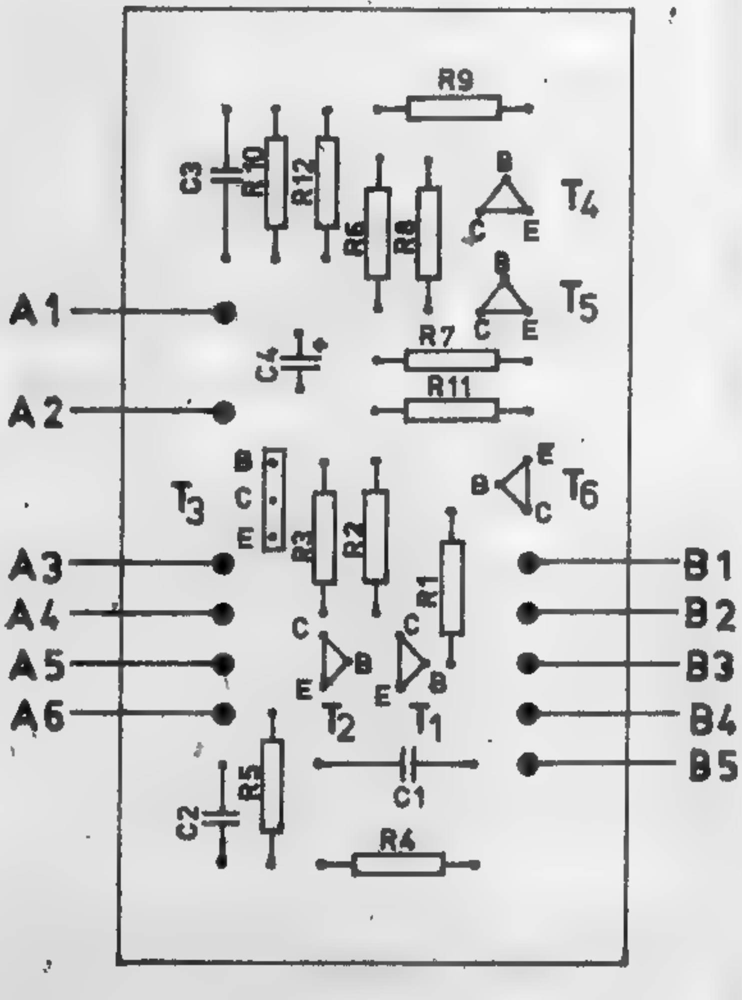
Rozwiązania zadań prosimy przysyłać do redakcji do końca lutego br., z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzcę „Ligi" czekają dodatkowe nagrody.



rys. 1

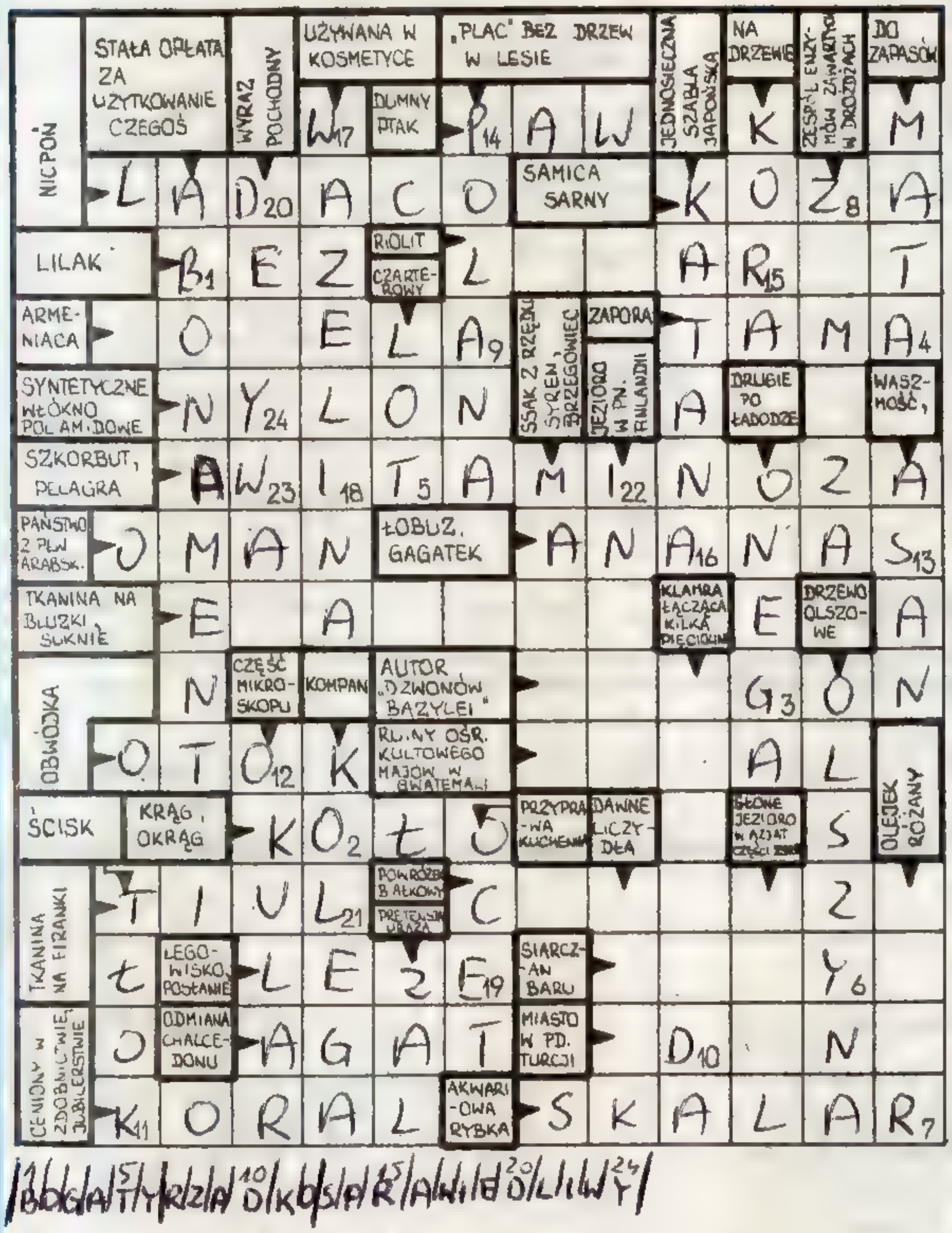


rys. 2



rys. 3

KRZYŻÓWKA NR 1



Litery z kratek ponumerowanych od 1 do 24 dadzą hasło, które wystarczy nadesłać jako rozwiązanie zadania pod adresem redakcji na kartach pocztowych w terminie do końca lutego, naklejając kupon „IKS-a". Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.



Drukarki

Najbardziej znaną na rynku drukarek komputerowych w Polsce jest firma Star, której kolejne modele SG-10, NL-10, czy LC-10 podbiły nasz rynek i mocno się na nim ulokowały.

Jedną z ostatnich nowości firmy Star jest drukarka 24-igłowa Multifont LC 24-10. Można w niej korzystać aż z czterech wbudowanych krojów pisma: Courier, Prestige, Script, Orator, z tym, że każda czcionka może być pisana normalnie lub pochyło. Prędkość druku nie jest imponująca, lecz zastosowanie tej ilości igieł, zamiast jak zwykle 9, zdecydowanie poprawia jakość druku. Jest on zbliżony do tego z drukarek laserowych.

Te ostatnie drukarki produkowane przez firmę Hewlett-Packard pojawiły się w 1984 roku. Bardzo duża prędkość i znakomita jakość druku spowodowały, że błyskawicznie znalazły zastosowanie w komputerach. W rok później powstał standard opisu tekstu i grafiki w postaci języka opisu strony — PDL (Page Description Language). Dzisiaj z tego języka korzystają wszyscy producenci drukarek.

Drukarki kodem w języku Post Script wydrukują każdą stronę. Mają własny komputer, własną pamięć, a często też dysk sztywny. Niedawno pojawiła się już pierwsza laserowa drukarka kolorowa. Pod nazwą QMS Color Script 100 wyprodukowała ją firma QMS Inc. Jest ona wyposażona w procesor 68020 z zegarem 16,7 MHz, pamięć ROM-1MB, RAM-8MB i dysk sztywny — 20MB. Uzyskano bardzo dobry kolorowy obraz o rozdzielczości 300 cpi, jaki daje normalna drukarka laserowa. Lecz cena za to kolorowo drukujące cudo jest „nienormalna" i wynosi 25 tys. dol., co jest sumą kilkakrotnie wyższą od cen normalnych drukarek laserowych.

Badania prowadzone są jednak w błyskawicznym tempie. Okazało się, że parametry drukarek laserowych można zachować usuwając z nich najdroższe elementy: laser i zwierciadła. Dokonano tego w drukarce Crystal Print VIII firmy Data Technology Corp. Bęben jest tam naświetlany za pomocą 2400 oddzielnie kontrolowanych wyświetlaczy ciekłokrystalicznych. Uzyskano rozdzielczość podobną jak w urządzeniach laserowych, a i szybkość druku jest bardzo zbliżona — 8 stron na minutę. Drukarki działające na tej zasadzie są całkowitą nowością i nazywane są drukarkami LCS (liquid-crystal-shutter).

Podobną nowością jest wprowadzenie drukarek systemem Braille'a. Holenderska firma Resus wypuściła takie urządzenia oznaczone symbolem RS214 charakteryzujące się wydajnością 200 linii czterdziestoznakowych na minutę. Drukarka może pełnić rolę terminala dla niewidomych, umożliwiając im czytanie danych wprowadzanych z klawiatury oraz wyników podawanych przez komputer. Dzięki niej można pisać listy, a także drukować czasopisma i książki. Cena ponad 10 tys. dol., dla Polaków trochę szokująca, dla zachodniego użytkownika nie jest wcale taka wygórowana.

J. RAJCH

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 2136. Nr ind. 361682. A-52

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 zł

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" NR 2 (1/89) ISSN 0866 – 2794

TAJEMNICE SUPERKOMPUTERÓW
strona 5

KONIEC „DŻUMY"?
strona 28

PIOTRUŚ PAN
strona 32







Od zakończenia drugiej edycji „Ligi Myślących" upłynęło ponad pół roku. Przypominamy: pierwsze miejsce zajął **Zdzisław Bedynek,** *pozostałymi laureatami byli* **Wiesław Rychlicki, Magdalena Rojek, Andrzej Barcewicz i Adam Domiński.** *Zwycięzcy odebrali nagrody z rąk zastępcy redaktora naczelnego „Żołnierza Wolności", płk. Marka Zdziecha.*

SPIS TREŚCI

Work Station — 3
Superkomputery — 5
Skojarzenie czterech par — 7
Komputerowa analiza układów elektronicznych — 9
Jak oszczędnie przechowywać ekrany na dyskietce — 15
Systemy operacyjne i inne gotowe programy — 16
Program „MIERNIK" — 19
Funkcja dyskryminująca 20
Niespodziewana wizyta — 22
CPC 6128 — programy narzędziowe — 26
Koniec „dżumy" — 28
Słownik — 29
Piotruś Pan — 32

Ten numer „IKS-a" oddajemy do rąk Czytelnika z opóźnieniem. Przepraszamy. Mamy nadzieję, że w obliczu „uzdrawiania gospodarki" i stawiania spraw oczywistych „na nogi" terminowość ukazywania się naszego pisma będzie ekonomiczną koniecznością, prawa rynku odgrywają również w naszej redakcyjnej pracy niemałą rolę.

Sądzimy jednak, że nasze pismo nadal towarzyszyć będzie Czytelnikom poszukującym ciągle nowej wiedzy o informatyce. Jaki będzie „IKS"? O tym decydują jak zwykle Czytelnicy. Uważnie czytamy Wasze listy, choć nie na wszystkie jesteśmy w stanie odpowiedzieć — ale każdą uwagę skierowaną pod naszym adresem starannie wykorzystujemy. Zatem czekamy na Wasze uwagi.

FIRMA	MODEL	CPU	FPU	MIPS	RAM (Mb)	max rozszerzenie RAM (Mb)	HARD DYSKI	MONITOR	Rozdzielczość	System operacji (OS)
APOLLO- -COMPUTERS	DN-3000 DN-4000 DN-10.000	68020-12 68030-33 RISC	68881-12 68881-33 —	1.3 4 12	4 8 64	[illegible] 32 lub 64 128 lub 256	80/155/330 155/330/696 [illegible]	15"mono 19"mono 19"color 19"color 19"color	1024×800 1024—1024 1024×1024 1024×1024 1280×1024	UNIX- -Berkeley w.4.3 Domain Aegis
SUN-Microsystems	Sun-3/50 Sun-3/160 Sun-4/260	68020-15 [illegible] RISC	68881-16 68881-25 weitek 1164/1165	1.5 [illegible] 10	[illegible] 8 32	[illegible] [illegible] 128	49 141 560	19"mono 16"color 19"color	1152×900 1152×900 1152×900	Sun (OS) (UNIX)
Televideo-systems	15 DL 17 C 19M 19C	80386-16 80386-16 80386-16 80386-16	80287-10 80387-16 80387-16 80387-16	2.2 2.2 2.2 2.2	4 4 4 4	16 16 16 16	— 40 71 71	15"mono 17"color 19"mono 19"color	1280×1024 1024×1024 1024×1024 1024×1024	Macroports DOS-Merge -386 (UNIX)
Hewlett-Packard	9000-350 [illegible]	68020-25 RISC	68881-25 —	3.7 8	8 8	32 56	80 80/131	19"color 19"color	1280×1024 1280×1024	UP-Ux (UNIX)
DEC	VAX-Station 2000	II procesor	— —	0.9	4	[illegible]	159	15"mono 15"color	1024×864 1024×864	ULTRIX (UNIX) VMS
XEROX	XEROX 228S	68020-16	68881-16	2.5	[illegible]	16	86	19"color 19"mono	1024×792 1024×792	Berkeley UNIX 4.3
MASSCOMP	MC 5350 MC 5450	68020-16 [illegible]	68881-16 68881-20	[illegible] 2.5	2 2	4 [illegible]	71 71	12"color 13"mono	[illegible] 1152×910	RTV (UNIX)

WORK STATION — STACJE GRAFICZNE do zastosowań inżynierskich

Już w roku 1980 użytkownicy komputerów; technicy, inżynierowie, naukowcy, architekci i różnego rodzaju projektanci — rozważali dwa ponure fakty: niewielką moc mikrokomputerów i bardzo wysokie ich ceny. Mikrokomputery posiadały niewystarczającą moc jednostek centralnych (CPU), pamięci wspólnych zasobów, jak również nie posiadały właściwej (odpowiedniej dla celów inżynierskich) rozdzielczości ekranów. Te cechy posiadały minikomputery, które z kolei nie umożliwiały interaktywnej pracy, ze względu na swoiste systemy operacyjne. Problemy te rozwiązała najpierw firma APOLLO COMPUTERS w 1981 roku komputerem DOMAIN DN 100, później firma SUN MICROSYSTEMS w 1982 roku komputerem SUN-1 zapełniając lukę Stacjami Roboczymi ogólnego przeznaczenia. Systemy te zostały stworzone dla potrzeb inżynierskich i posiadały szybkie jednostki centralne (CPU), duże pamięci RAM, ekrany o wysokiej rozdzielczości, duże pamięci dyskowe oraz możliwości pracy w sieciach przy użyciu *file serverów,* jak również pozwalały połączyć ze sobą różne stacje z różnymi urządzeniami peryferyjnymi, takimi jak plottery, digitizery, scanery, drukarki itp.

W szczególności APOLLO i kroczący tuż za nim SUN rozrastały się gwałtownie tworząc nowe urządzenia dla elektroniki, projektowania kosmicznego i lotniczego, CAD-u, prac wydawniczych, sztucznej inteligencji, wspomagania komputerowego prac inżynierskich, przetwarzania obrazów, trójwymiarowego modelowania struktur, jak również obsługi służb finansowych i księgowych. Tym samym wzrosło zainteresowanie innych firm komputerowych tego typu produktami.

STANDARD STACJI ROBOCZEJ (GRAFICZNEJ)

Najprostsze stacje robocze są zadziwiająco podobne do siebie, a to za sprawą standardu preferowanego od początku swej działalności przez firmę APOLLO.

Podstawowy typ Stacji Roboczej posiada jednostkę centralną (CPU) 32-bitową, 4 do 8 Mb RAM, 40 do 80 Mb Karol dysk, ekran megapixelowy, połączenia sieciowe o szybkości transmisji rzędu 10 Megaboadów, system operacyjny wielozadaniowy oparty na pamięci Virtualnej (tzw. wspomaganie herolwerowe grafiki). Większość z nich może pracować w sieci we współpracy z file serverem. Zarówno prekursor w dziedzinie produkcji stacji graficznych — firma APOLLO, jak również SUN, HEWLETT PACKARD i prawie wszystkie współczesne stacje graficzne oparte są na Motoroli serw 68000 i koprocesorze 68881 z paroma wyjątkami, jak DEC VAX Stadion 2000. Wszystkie firmy pragną maksymalnie zwiększyć częstotliwość zegara, jednostki centralnej, stosując następny w generacji mikroprocesor. Wreszcie wielu projektantów jednostek centralnych zwraca się do własnych chipów, często opartych na technologii RISC *(Reduced Instruction set Computer)* — bezwzględnym liderem jest APOLLO z modelem DN 10000, którego wartość porównywalna jest z dużymi superkomputerami CRAY-a.

Komputery PC często mierzą swoje osiągnięcia szybkością zegarów. Stacje graficzne natomiast opierają swoje osiągnięcia na milionach operacji na sekundę (MIPS), od 1 MIPS-a w pierwszych modelach do MIPS-ów np. w APOLLO serii 4000, którego cena w rozbudowanej konfiguracji 32 Mb RAM, 330 Mb nad dyskiem i kolorowym monitorem 19 o rozdzielczości 1024×1024 wynosi około 115 000 dolarów, co wydaje się w porównaniu z innymi minikomputerami dość niewygórowaną ceną.

Wszystkie stacje robocze zapewniają wielozadaniowość poprzez system operacyjny i dzięki odpowiedniej architekturze wewnętrznej gwarantują doskonałe osiągnięcia. Większość nowych systemów posiada na przykład pamięć typu Cache Memory (8 Kb), która stała się standardem. Wiele programów inżynierskich



jest bardzo rozbudowanych i wykorzystują często interaktywną grafikę. Inżynierowie często korzystają z wielozadaniowości dzięki czemu mogą przez cały czas pracować, podczas gdy długie i uciążliwe obliczenia dokonywane są w tle.

Cechą stacji, decydującą o sukcesie są duże ekrany z dobrą rozdzielczością mierzoną już w megapixelach, pozwalające praktycznie na stosowanie wielu okien jednocześnie, np. w modelu APOLLO serii 4000 rozdzielczość ekranu 19" wynosi 1280×1024, co jest obecnie absolutnym rekordem.

OPROGRAMOWANIE

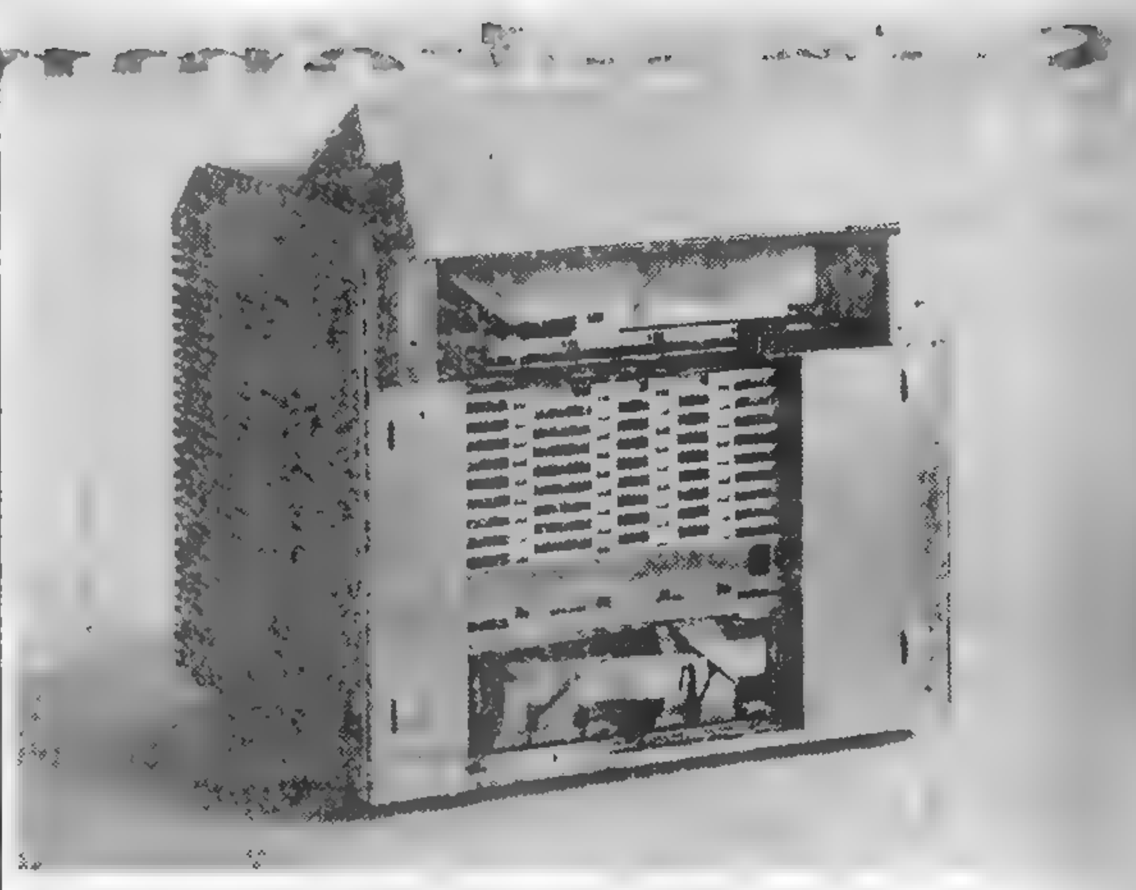
Wspólną cechą wszystkich niemalże Stacji Graficznych jest ich system operacyjny (OŚ). Prawie wszystkie wykorzystują adaptację UNIX — a łączącą osiągnięcia systemu V z elementami systemu BERUELEY 4.2.

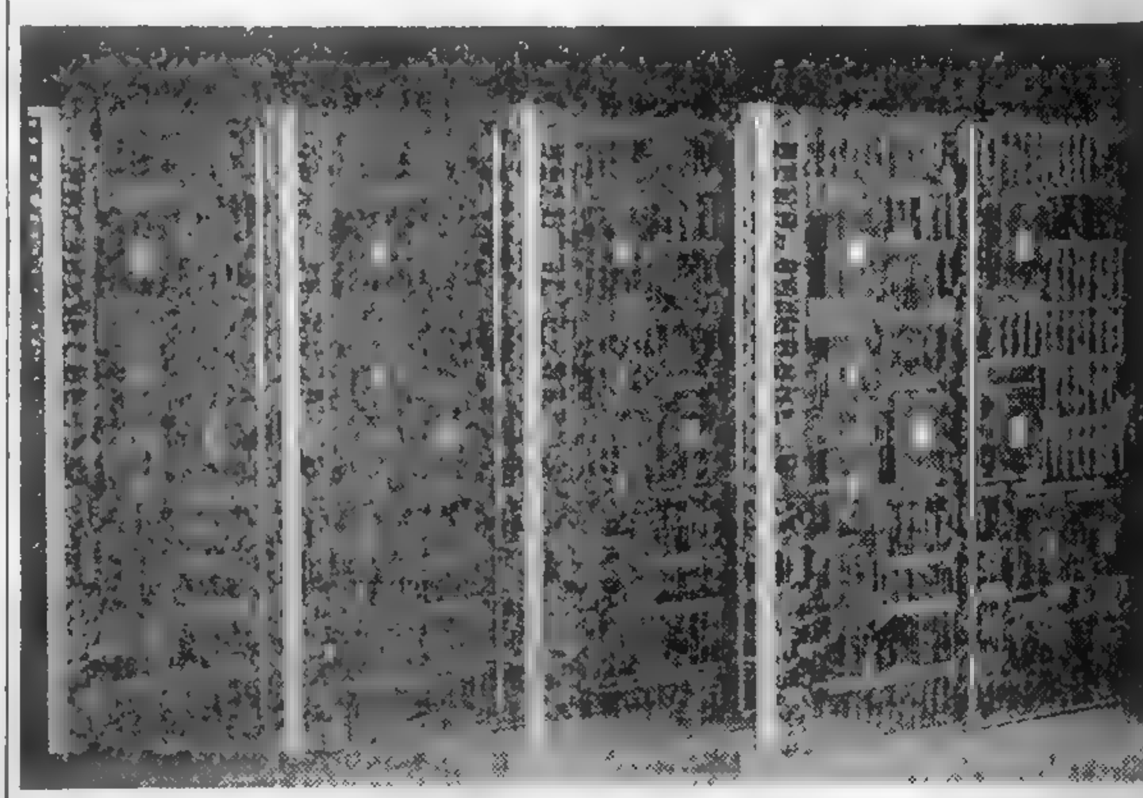
APOLLO rozpoczął pracę z własnym systemem o nazwie AEGIS, ale wkrótce przeszedł na UNIX-a pociągając za sobą prawie wszystkie firmy. SUN krocząc tuż za APOLLO musiał nawet zatrudnić bezpośredniego projektanta systemu UNIX 4.2., myśląc, że to właśnie pomoże mu w dotrzymaniu kroku APOLLO.

Firma DEC oferuje zarówno UNIX, jak i swój własny system operacyjny pod nazwą VMS, który z uwagi na długotrwałą obecność firmy na rynku prawie stał się standardem.

Nowe firmy w branży; jak APPLE, czy Televideo, zdając sobie sprawę z popularności UNIX-a oferują go równolegle z własnym systemem.

Większość użytkowników MACINTOSHY zaraz poczuje się jak w domu, pracując na stacjach roboczych, bo używają one okien do zarządzania systemem. Dwa systemy walczą tu o rangę standardu. APOLLO i jego naśladowcy DEC, HP i wielu innych popiera X-WINDOWS SYSTEM, pakiet stworzony na uniwersytecie MIT jako część projektu ATHENA oraz NEWS (Network-extensible windowing system) preferowany przez firmę SUN, który odstrasza potencjalnych klientów ze względu na zbyt trudny aparat





użycia procedur nie znajdujących miejsca przy pracy w sieci. Szybka zintegrowana sieć jest elementem vitalnym, jeżeli chce się wymieniać informacje graficzne pomiędzy użytkownikami, wspólnie korzystać z drogich peryferii, jak plottery, drukarki, duże dyski laserowe czy streamery. Najczęściej używaną siecią w świecie stacji graficznych jest Ethernet. Z uwagi na rolę sieci wielu producentów stacji wytwarza również serwery, które są wyspecjalizowane w obsłudze węzłów sieciowych.

Moim celem było przybliżenie problemu mało znanego i niedocenionego na polskim rynku komputerowym, a przecież trudno sobie wyobrazić w wysoko uprzemysłowionych krajach świata prace projektowo-graficzne bez WORUK STATION.

Można optymistycznie przewidzieć wielkość postępu naukowo-badawczego w dziedzinie projektowania i grafiki w Polsce przy zwiększonych wysiłkach popularyzujących ten kierunek informatyki.

BRYSKOW

Superkomputery

Komputer w Uniwersytecie Illinois symuluje coś czego nikt wcześniej nie widział. Pierwsze chwile po „Big Bang'u". Można oglądać wydarzenia, które miały miejsce miliardy lat temu — fantastyczne kształty kolorowych obłoków masy międzygalaktycznej. Owe kolory wskazują różnice gęstości wędrujących chmur materii. Natomiast w Instytucie Massachusetts inny komputer usiłuje nauczyć się tego, co umie trzyletnie dziecko. Na przykład zdolności oceny różnicy między kubkiem a spodkiem. To, co dzieci dostrzegają natychmiast, komputer musi krok po kroku przeanalizować. Najpierw rozpoznać klasę obiektu, fizyczne cechy różniące go od otoczenia, potem zwraca uwagę na atrybuty, itp.

Eksperymenty te wskazują na paradoks istoty współczesnej informatyki. Z jednej strony gigantyczne moce obliczeniowe, z drugiej natomiast intelektualne zdolności, które często są znacznie mniejsze od poziomu intelektualnego kilkuletniego dziecka. Wyraźnie dominują typowe (tradycyjne) zastosowania komputerów — szczególnie tam, gdzie mamy do czynienia ze specjalizowanymi zadaniami. Mimo ich ogromnej skali, maszyny wykonują te prace w czasie kilku godzin — przed erą komputerów wymagało by to zaangażowania wielu ludzi w czasie kilku pokoleń...

Od 40 lat naukowcy pracują nad tymi obiema drogami badań komputerowych. Jedna grupa zajmuje się szybkimi jak światło maszynami — i niezmiennie dąży do zwiększenia ich szybkości, mocy. Druga grupa pisze programy o znamionach początków sztucznej inteligencji. Obie te grupy wybitnych uczonych, dysponując miliardami dolarów działają indywidualnie, jak gdyby druga grupa nie istniała.

Ostatnio pojawiły się pewne oznaki, iż dwie szerokie drogi badań komputerowych mogą się połączyć. Być może wkrótce najlepsze komputery zostaną sprzężone w jeden elektroniczny mózg, który będzie nie tylko nadzwyczaj szybki, ale i mądry. Trudności owego „spotkania" należy głównie upatrywać w różnych poziomach zaawansowania tych prac. Sztuczna inteligencja właśnie rozpoczyna swoje życie. Pierwszy komercyjny projekt powstał zaledwie pięć lat temu. Mimo to ma on już dzisiaj wiele poważnych zastosowań. Jednak z drugiej strony fabryki produkujące sprzęt, o którym jest mowa, czyli superkomputery, zajmują się tym już od ćwierć wieku. Mają ogromne doświadczenia i sprawdzony rynek. Sprzedaż tych maszyn, a kosztują one od 5 do 25 milionów dolarów każda, rośnie o 25% każdego roku.

Około 300 superkomputerów wykorzystywane jest do badań nad problemami paliwowymi, do analizowania struktury mięśni i tworzenia specjalnych efektów dla hollywoodzkich filmów. Ogromna moc superkomputerów dostępna jest, via telefon i sputnik, dla każdego kto posiada personalny komputer. „Świat nigdy już nie będzie ten sam" — twierdzi jeden ze współtwórców superkomputerów. „Wkrótce każda firma, każdy naukowiec, a nawet każdy krok życia będzie w pewien sposób związany z superkomputerami".

Szybkość — podstawowa miara mocy — stanowi główną różnicę między superkomputerami a ich mniejszymi krewniakami. Szybkość początkowo była mierzona tysiącami FLOPS'ów (akronim od „floating-point operations per second"), innymi słowy — liczb zmiennoprzecinkowych. Owa miara w dzisiejszych maszynach wynosi giga-FLOPS'y lub inaczej — superkomputery wykonują miliardy operacji na sekundę. Przewiduje się, że już wkrótce pojedyncza maszyna będzie pracowała z szybkością tera-FLOPS'ów, co daje moc dziesięciu milionów personalnych komputerów pracujących „na pełen gaz".

Superkomputery o największej mocy są zaskakująco małe i estetyczne (lśniące). Wewnątrz bardzo upakowane — by nie tracić czasu na przesyłanie informacji zbyt długimi drutami. Jak już wspomniano, maszyna ta może być połączona kablami lub via satelita z setkami terminali, które zamieniają szeregi cyfr na przykład na piękną kolorową trzywymiarową grafikę. Sprzęt ten, jak każde intensywnie pracujące urządzenie, musi być chłodzony. Bardzo sprytnie poradzono sobie z tym problemem w Uniwersytecie Minnesota — uzyskane tą drogą ciepło wykorzystywane jest do... ogrzewania garaży.

Rynek superkomputerów zdominowany został przez jedną firmę: CRAY Research of Minneapolis, która wyprodukowała 178 komputerów o charakterystycznym kształcie litery C (jak Ciechocinek). Owe superkomputery zainstalowane są na całym świecie i stanowią około 60% potencjału tego typu sprzętu. Najpoważniejszym konkurentem CRAY'a jest Control Data Corp. CDC, firma, która w 1983 roku stworzyła superkomputer ETA. Maszyny te, to około 13% światowych superkomputerów. Natomiast 23% tego typu komputerów wyprodukowano w Kraju Kwitnącej Wiśni przez NEC'a, Hitachi i Fujitsu, począwszy od 1983 roku.

Niespodziankę (oczekiwaną chyba przez wszystkich) szykuje gigant komputerowy IBM (sprzedaż w 1987 roku 54,2 miliarda dolarów), który zatrudnił (podkupił?) mistrza od superkomputerów Steve'a Chen'a („odzysk" z CRAY'a). Propozycja IBM'a, to komputer z 64 pracującymi równolegle procesorami — źródło znacznego przyspieszenia pracy komputera. Uznawane jest to jako znacząca nowość.

Okazuje się jednak, że rozwiązanie takie nie jest oryginalnym pomysłem IBM'a, mówi się bowiem, że inni projektanci tworzą maszyny z setkami, a nawet tysiącami procesorów. Wiosną 1988 roku naukowcy z Snandia National Labolatories w Albuquerque ogłosili, że ich 1024-procesorowy komputer jest 1000 razy szybszy od każdego komputera z pojedynczym procesorem, sugerując jednocześnie, iż opracowali metodę proporcjonalnego zwiększania szybkości komputera w zależności od liczby zainstalowanych w nim procesorów.

Większość badań nad superkomputerami w USA finansowana jest przez władze. Oczywiście najbardziej zainteresowane wzrostem mocy komputerów jest wojsko i wywiad. W 1987 roku Pentagon poświęcił setki milionów dolarów na doskonalenie superkomputerów. Ponadto sprzęt ten stanowi znaczne uwarunkowanie postępów w podboju przestrzeni kosmicznej. Jednym ze spektakularnych tego typu zastosowań jest wspomaganie prac projektowych nad samolotem „fruwającym" z szybkością 25 razy więk-

szą od głosu. Potrzeba taka wynikła z braku dostępu do tunelu aerodynamicznego, w którym można by było symulować eksploatację takiego wehikułu. Tradycyjny sposób testowania samolotów zastąpiony został w tym wypadku przez wykorzystanie komputera. Ponad 1,7 miliarda dolarów ma kosztować superkomputer w 1990 roku, który przekroczy szybkość mierzoną tera-FLOPS'ami.

Już od 1976 roku pierwszy CRAY pracował nad bombą wodorową. Inny „wczesny" CRAY przeznaczono do prac w National Security Agency, gdzie do dzisiaj rozwiązuje się „wojskowe zagadki", przegląda wywiadowcze dane, które każdego dnia napływają do agencji.

A inne zastosowania? Jest ich bardzo dużo. Na przykład przy projektowaniu sprzętu elektronicznego, finansiści radzą się superkomputerów, gdzie można opłacalnie inwestować, biomedycy zaprzęgają maszyny do analizy molekuł, by stwierdzić, które z nich mogą się stać nowymi lekami. Natomiast inżynierom CRAY pomaga projektować samochody, silniki, żaglówki, lodówki i nowe rośliny. Jednak najwięcej czasu z superkomputerami spędzają ludzie nauki. Stało się to osiągalne dzięki wspomnianym wcześniej połączeniom (via telefon). Na przykład superkomputer Narodowej Fundacji Nauki (koszt 200 mln dol.) „obsługuje" 6000 naukowców oraz 200 innych instytucji. Wspomniane przedsięwzięcia najczęściej wspomagane są przez różne wersje CRAY'a (CRAY-2, CRAY X-MP).

Mimo ogromnego postępu, użytkownicy superkomputerów nie są w pełni usatysfakcjonowani parametrami tego sprzętu. Potrzebne są superkomputery 100 razy szybsze. Wyścig w budowaniu szybszych superkomputerów jest na dobrej drodze. W dziesiątkach laboratoriów USA, Europy i Japonii prowadzone są bardzo intensywne prace nad doskonaleniem superkomputerów. Podstawowym celem jest zbudowanie najszybszej maszyny.

Podobnie jak Edison żarówkę, Bell telefon — Seymour CRAY stworzył superkomputer. Ten genialny inżynier zaprojektował: CDC 1604 (1960 rok), CDC (1964), CDC 7600 (1969), CRAY-1 (1976) i CRAY-2 (1985). Każdy z tych komputerów zawsze był uznawany w danym okresie za najlepszy. Obecnie ten wybitny konstruktor zajmuje się budową komputera CRAY-3, który ma być gotowy w 1988 roku i zaraz potem należy spodziewać się kolejnej maszyny — CRAY-4. CRAY-3 będzie pierwszym komercyjnym komputerem, który zbudowano korzystając z układów scalonych produkowanych na bazie arsenku galu (zamiast krzemu). Podłoże to jest 10-krotnie szybsze dla wędrujących elektronów. W czwartym modelu CRAY'a zostaną zainstalowane 64 procesory (numer 2 miał tylko cztery, w CRAY-3 zainstalowano 16 procesorów).

Zapewne usłyszymy jeszcze nieraz o wybitnym uczniu Cray'a 44-letnim Chińczyku Chen. Udoskonalił on konstrukcję swego mistrza i w efekcie z CRAY-1 powstał CRAY X-MP — dwuprocesorowa maszyna, jak dotychczas jest ona najlepiej sprzedawana (aż 120 instalacji). Kolejny wytwór Chena'a, to CRAY Y-MP. Jednak różnica pokoleń, nawet tam za oceanem, dała znać o sobie i Chen, którego praca (zdaniem Cray'a) charakteryzowała się zbyt dużym rozmachem musiał pożegnać się z nauczycielem. Wizjonerstwo Chen'a nie przestraszyło IBM'a, gdzie może realizować swoje plany, a sięgają one 100 miliardów giga-FLOPS'ów!

IBM poza projektem Chen'a ma „na warsztacie" sześć innych przedsięwzięć. Jedno z nich to GF-11, który zajmuje około 50 m², inny komputer to RP-3, ale najbardziej ambitnym wytworem IBM'a jest TF-1 zawierający 33000 dużej szybkości procesorów. Szybkość tej maszyny jest 2000 razy większa od dzisiejszych superkomputerów. IBM nie obawia się Cray'a, ale... konkurencji japońskiej — Hitachi, Fujitsu, i NEC'a. Superkomputery z Japonii (pierwsza generacja) świadczą o tym, iż amerykański przemysł komputerowy zostanie „zniszczony" w ciągu 25 lat. Współczesne maszyny made in Japan są porównywalne z najszybszymi made in USA, a w pewnych zastosowaniach są nawet lepsze. Porównując najnowszy pojedynczy procesor Hitachi S-820/80 z dwuprocesorowym Cray'a X-MP, Hitachi jest 10 razy lepszy. Japończycy skupili się na szybkich układach procesorowych. Przed nimi są jeszcze ogromne rezerwy w budowie komputerów z wieloma procesorami.

W komputerach równoległych (gdzie pracuje wiele procesorów jednocześnie), głównym problemem okazuje się oprogramowanie, całkowicie różne od oprogramowania komputerów jednoprocesorowych. To nowe oprogramowanie zaczyna dopiero „raczkować", mimo to efekty są widoczne — komputery równoległe są bezkonkurencyjne. Oprogramowanie na równi ze sprzętem decyduje o szybkości superkomputerów (równoległych). Powstały zatem warunki, wręcz konieczność połączenia wysiłków twórców techniki i specjalistów od oprogramowania, którzy dotychczas zajmowali się sztuczną inteligencją. Efekt owej współpracy najlepiej widać na przykładzie, obecnie najdoskonalszego równoległego komputera — Connection Machine. Urządzenie to zawiera 65 536 procesorów. Nie mniej zadziwiające jest to, że maszynę tę zaprojektował 31-letni Daniel Hillis. Na zakończenie warto zacytować jego wypowiedź: *„Tak jak nie można zbudować samolotu bez wcześniej skonstruowanego wystarczająco mocnego silnika, nie można tworzyć sztucznej inteligencji bez szybkich komputerów"*. Warto zatem nieco więcej uwagi poświęcić owej sztucznej inteligencji, która odradza się ostatnio, ale o tym już innym razem.

WŁODZIMIERZ GOGOŁEK

Opracowano na podstawie: TIME, March 28, 1988



SKOJARZENIE CZTERECH PAR

DANUTA KWASIŻUR, MIECZYSŁAW SKONIECZNY

Z szóstego rozdziału „LILAVATI" S. Jeleńskiego zatytułowanego „Z tajników szachownicy, kart i domina" wybraliśmy grę nr 2, którą chcemy zaprezentować Czytelnikom w dzisiejszym odcinku mikrokomputerowych anegdot matematycznych.

Zasady gry pojawiają się po uruchomieniu programu.

```
Na planszy gry zostaly rozlozone w rzad
cztery karty czerwonej i cztery czarnej
masci tak, ze barwy sie przeplataja:
czerwona-czarna-czerwona-czarna itd.
Po lewej stronie szeregu sa dwa miejsca
wolne. Na wolne miejsca mozna przeniesc
tylko dwie karty obok siebie lezace, nie
zmieniajac porzadku w jakim lezaly;i tak
samo na miejsce oproznione mozna znowu
przeniesc dwie karty sasiadujace.
Zadanie polega na tym, zeby w kilku
poruszeniach przesuwajac karty parami
ulozyc je w ten sposob, by lezaly kolo
siebie pod rzad cztery czarne, a za nimi
cztery czerwone karty. /Karty musza byc
ulozone parami/. Jezeli w czasie gry
rezygnujesz z rozwiazywania tego wa-
tu wprowadz zamiast numeru karty 11.
```

Po naciśnięciu dowolnego klawisza ukazuje się plansza gry, na której karty ułożone są w sposób losowy. Program oczekuje wprowadzenia dwóch numerów kart, które należy przesunąć. Muszą one leżeć obok siebie.

Sytuację po pierwszym przesunięciu przedstawia poniższy rys.




```
1 PRZESUNIECIE
PODAJ NUMER PIERWSZEJ KARTY 5
PODAJ NUMER DRUGIEJ KARTY 6
```

Gra prowadzona jest dalej w ten sam sposób do momentu poprawnego ułożenia kart. Wówczas pojawi się komunikat:

**„KARTY PRAWIDŁOWO UŁOŻONE
PO n PRZESUNIĘCIACH
SPRÓBUJESZ JESZCZE RAZ? (T/N)"**

i użytkownik może rozpocząć grę od początku z tym samym ułożeniem początkowym kart lub nowym, losowo wybranym przez program.

Wersja źródłowa programu jest następująca:

W segmencie sterującym wykorzystuje się 4 podprogramy, które realizują następujące funkcje:

1. Podprogramy z komunikatami o błędach: (instrukcje 900—960)
2. Podprogram badania sukcesu gry: (instrukcje 970—1200)
3. Podprogram rysowania dowolnej karty na ekranie: (instrukcje 1270—1410)
4. Podprogram losowego ustalenia początkowego ułożenia kart: (instrukcje 1420—1610)

```
10 '
20 ' Ekran informacyjny
30 '
40 MODE 1:INK 0,26:INK 1,0:INK 2,6:INK 3,9:PAPER 0:BORDER 26:CLS:PEN 1
50 SYMBOL AFTER 160
60 SYMBOL 160,3,15,31,31,63,63,63,63
70 SYMBOL 161,192,240,248,248,252,252,252,252
80 SYMBOL 162,63,63,63,63,63,31,31,15,3
90 SYMBOL 163,252,252,252,252,252,248,248,240,192
100 SYMBOL 164,63,63,63,63,63,63,63,63
110 SYMBOL 165,252,252,252,252,252,252,252,252
120 LOCATE 1,3
130 PRINT "Na planszy gry zostaly rozlozone w rzad"
140 PRINT "cztery karty czerwonej i cztery czarnej"
150 PRINT "masci tak, ze barwy sie przeplataja:  "
160 PRINT "czerwona-czarna-czerwona-czarna itd.  "
170 PRINT "Po lewej stronie szeregu sa dwa miejsca"
180 PRINT "wolne. Na wolne miejsca mozna przeniesc"
190 PRINT "tylko dwie karty obok siebie lezace, nie";
200 PRINT "zmieniajac porzadku w jakim lezaly;i tak";
210 PRINT "samo na miejsce oproznione mozna znowu "
220 PRINT "przeniesc dwie karty sasiadujace.     "
230 PRINT "Zadanie polega na tym, zeby w kilku    "
240 PRINT "poruszeniach przesuwajac karty parami  "
250 PRINT "ulozyc je w ten sposob, by lezaly kolo "
260 PRINT "siebie pod rzad cztery czarne, a za nimi";
270 PRINT "cztery czerwone karty. /Karty musza byc"
280 PRINT "ulozone parami/. Jezeli w czasie gry   "
290 PRINT "zrezygnujesz z rozwiazywania tego wa-  "
300 PRINT "tu wprowadz zamiast numery karty 11.   "
310 LOCATE 8,25:PRINT "NACISNIJ DOWOLNY KLAWISZ"
320 CLEAR INPUT
330 A$=INKEY$
340 IF A$="" THEN GOTO 330
350 '
360 ' Segment organizacyjny
370 '
380 MODE 1
390 DIM KARTY(10,2):DIM KARTY1(10,2)
400 GOSUB 1450
410 WINDOW £1,1,40,2,10:PAPER £1,1:CLS £1
420 NUM=1:NUM1=2
430 B$=CHR$(143)+CHR$(143):B1$=CHR$(164)+B$+CHR$(165)
440 FIG$(1)="D":FIG$(2)="K"
450 PAPER 1:PEN 0
460 FOR I=1 TO 10
470 LOCATE 2+(I-1)*4,3:PRINT USING "££";I
480 NEXT I
490 FOR K=1 TO 10
500 GOSUB 1300
510 NEXT K
520 LP=1
530 WINDOW £2,4,35,14,15:PAPER £2,3:PEN £2,0
540 WINDOW £3,4,35,16,18:PAPER £3,3:PEN £3,0
550 WINDOW £4,4,35,21,23:PAPER £4,2:PEN £4,0
560 WINDOW £5,4,35,12,13:PAPER £5,3:PEN £5,0:CLS £5
570 LOCATE £5,10,2:PRINT £5,"PRZESUNIECIE"
580 CLS £2:CLS £3
590 LOCATE £5,7,2:PRINT £5,USING "££";LP
600 CLS £2:LOCATE £2,2,2:INPUT £2,"PODAJ NUMER PIERWSZEJ KARTY ",NR
610 IF nr<1 OR nr>11 THEN GOTO 600
620 IF NR=11 THEN GOTO 1150
630 IF KARTY(NR,1)=0 THEN GOSUB 930:GOTO 600
640 IF nr=1 AND karty(2,1)=0 THEN GOSUB 950:GOTO 600
650 IF nr=10 AND karty(9,1)=0 THEN GOSUB 950:GOTO 600
660 IF karty(nr-1,1)=0 AND karty(nr+1,1)=0 THEN GOSUB 950:GOTO 600
```

```
670 CLS £3:LOCATE £3,2,2:INPUT £3,"PODAJ NUMER DRUGIEJ KARTY   ",NR1
680 IF nr1<1 OR nr1>11 THEN GOTO 670
690 IF NR1=11 THEN GOTO 1150
700 IF KARTY(NR1,1)=0 THEN GOSUB 930:GOTO 670
710 IF NR=NR1+1 OR NR=NR1-1 THEN GOTO 730
720 GOSUB 940:GOTO 670
730 IF NR<NR1 THEN GOTO 750
740 B=NR:NR=NR1:NR1=B
750 FOR L=1 TO 5
760 PEN 0:PAPER 1:LOCATE 1+(NR-1)*4,4+L:PRINT SPC(8);
770 NEXT L
780 KARTY(NUM,1)=KARTY(NR,1)
790 KARTY(NUM,2)=KARTY(NR,2)
800 KARTY(NUM1,1)=KARTY(NR1,1)
810 KARTY(NUM1,2)=KARTY(NR1,2)
820 KARTY(NR,1)=0:KARTY(NR,2)=0
830 KARTY(NR1,1)=0:KARTY(NR1,2)=0
840 K=NUM:GOSUB 1310
850 K=NUM1:GOSUB 1310
860 NUM=NR:NUM1=NR1
870 GOSUB 1020
880 LP=LP+1
890 GOTO 580

900 '
910 ' Podprogramy z komunikatami o bledach
920 '
930 PAPER £4,2:CLS £4:LOCATE £4,2,2:PRINT £4,"W TYM MIEJSCU NIE MA KARTY.":GOTO
960
940 PAPER £4,2:CLS £4:LOCATE £4,2,2:PRINT £4,"KARTY NIE LEZA OBOK SIEBIE.":GOTO
960
950 PAPER £4,2:CLS £4:LOCATE £4,2,2:PRINT £4,"PRZESUNIECIE NIEMOZLIWE":GOTO 960
960 SOUND 1,300:FOR T=1 TO 2000:NEXT T:PAPER £4,0:CLS £4:RETURN

970 '
980 ' Podprogram badania sukcesu gry
990 '
1000 IF karty(9,1)=0 AND karty(10,2)=0 THEN GOTO 1020
1010 RETURN
1020 FOR J=1 TO 10 STEP 2
1030 IF KARTY(J,2)=KARTY(J+1,2) THEN GOTO 1050
1040 RETURN
1050 NEXT J
1060 FOR J=1 TO 4
1070 IF KARTY(J,2)=2 OR KARTY(J,2)=3 THEN RETURN
1080 NEXT j
1090 CLS £5:CLS £2:CLS £3
1100 IF LP>10 THEN GOTO 1140
1110 LOCATE £5,12,2:PRINT £5,"BRAWO!!!"
1120 LOCATE £2,5,2:PRINT £2,"KARTY PRAWIDLOWO ULOZONE":PRINT £3,"    PO ";USING
"££";lp;:PRINT £3," PRZESUNIECIACH."
1130 GOTO 1150
1140 LOCATE £2,5,2:PRINT £2,"KARTY PRAWIDLOWO ULOZONE":PRINT £3,"    PO ";USING
"££";LP;:PRINT £3," PRZESUNIECIACH."
1150 PAPER £4,2:CLS £4:LOCATE £4,2,2:INPUT £4,"SPROBUJESZ JESZCZE RAZ(T/N)";T$
1160 IF T$="T" OR T$="t" THEN GOTO 1190
1170 IF T$="n" OR T$="N" THEN END
1180 GOTO 1150
1190 CLS £4:LOCATE £4,2,2:INPUT £4,"TEN SAM UKLAD KART (T/N)";T$
1200 IF T$="T" OR T$="t" THEN MODE 1:GOTO 1230
1210 IF T$="n" OR T$="N" THEN MODE 1:GOSUB 1430:GOTO 410
1220 GOTO 1190
1230 FOR K=1 TO 10
1240 KARTY(K,1)=KARTY1(K,1):KARTY(K,2)=KARTY1(K,2)
1250 NEXT K
1260 GOTO 410

1270 '
1280 ' Podprogram rysowania dowolnej karty
1290 '
1300 IF KARTY(K,1)=0 THEN RETURN
1310 PAPER 1:PEN 0
1320 LOCATE 1+(K-1)*4,5:PRINT CHR$(160);B$;CHR$(161);
1330 FOR I=6 TO 8
1340 LOCATE 1+(K-1)*4,I:PRINT B1$;
1350 NEXT I
1360 LOCATE 1+(K-1)*4,9:PRINT CHR$(162);B$;CHR$(163);
1370 PAPER 0
1380 IF KARTY(K,2)=1 OR KARTY(K,2)=4 THEN PEN 1:GOTO 1400
1390 PEN 2
1400 LOCATE 2+(K-1)*4,7:PRINT FIG$(KARTY(K,1));CHR$(225+KARTY(K,2))
1410 RETURN

1420 '
1430 ' Podprogram losowego ustalania poczatkowego ustawienia kart
1440 '
1450 KARTY(1,1)=0:KARTY(1,2)=0
1460 KARTY(2,1)=0:KARTY(2,2)=0
1470 karty(3,1)=1+INT(RND*2)
1480 karty(3,2)=1+INT(RND*4)
1490 FOR k=4 TO 10
1500 fig=1+INT(RND*2)
1510 kol=1+INT(RND*4)
1520 FOR i=3 TO k-1
1530 IF fig=karty(i,1) AND kol=karty(i,2) THEN GOTO 1500
1540 NEXT i
1550 karty(k,1)=fig
1560 karty(k,2)=kol
1570 NEXT k
1580 FOR K=1 TO 10
1590 KARTY1(K,1)=KARTY(K,1):KARTY1(K,2)=KARTY(K,2)
1600 NEXT K
1610 RETURN
```



KOMPUTEROWA ANALIZA UKŁADÓW ELEKTRONICZNYCH

ANALIZA STAŁOPRĄDOWA — DC

Chciałbym przybliżyć Czytelnikom metody wykorzystywane w komputerowej analizie układów elektronicznych. Ponieważ istnieje wiele profesjonalnych programów tego typu, nie silę się na stworzenie jeszcze jednej ich wersji. Postaram się w możliwie czytelny i ogólny sposób przedstawić algorytm działania tych programów.

Użytkownik wprowadza do komputera strukturę układu, parametry elementów elektronicznych (ewentualnie temperaturę otoczenia, tolerancje itp.). Na podstawie tych danych program generuje układ równań (w ogólnym wypadku są to nieliniowe równania różnicowe). Rezultatem obliczeń są najczęściej napięcia i prądy występujące w układzie.

Zagadnienie podzieliłem na kilka części tematycznych. W pierwszym odcinku proponuję zająć się **analizą stałoprądową** zwaną inaczej **analizą DC**.

Rozważmy prosty układ elektroniczny przedstawiony na **rys. 1**. Przypominam, że **źródło prądowe** wymusza przepływ stałego prądu między swoimi węzłami. **Węzłem** nazywamy punkt połączenia elementów (na schemacie punkty te oznaczone są kropkami i ponumerowane). Węzeł o numerze 0 *(zero)* jest węzłem odniesienia i stanowi tzw. **masę** układu elektronicznego. Względem niego są obliczane napięcia występujące na pozostałych węzłach.

Proponuję obliczyć napięcia występujące w układzie (oczywiście względem węzła 0) wykorzystując **I i II prawo Kirchhoffa**. Na schemacie zaznaczyłem rozpływ wszystkich prądów. Zgodnie z II prawem Kirchhoffa **suma prądów wpływających i wypływających z węzła wynosi zero**. Umówmy się: *prąd wypływający z węzła jest dodatni, a prąd do niego wpływający jest ujemny.* Analizując poszczególne węzły otrzymujemy:

węzeł 1: $I_1 + I_2 - J = 0$

węzeł 2: $-I_1 + I_3 - I_4 = 0$

węzeł 3: $-I_2 + I_3 - I_5 = 0$

Uwaga: Ponieważ suma prądów płynących w układzie wynosi 0 *(zero)*, to analiza węzła odniesienia nie ma sensu, gdy prądy wpływające i wypływające z tego węzła są w sposób jednoznaczny określone przez rozpływ prądów w pozostałych węzłach.

Chcemy, aby wynikiem naszych obliczeń były napięcia, stąd podstawmy zamiast prądów wyrażenia zawierające napięcia. Np. prąd I_1 na podstawie **prawa Ohma** można zapisać następująco:

$I_1 = \frac{U_{1-2}}{R_1}$, gdzie U_{1-2} jest napięciem występującym pomiędzy węzłami 1 i 2.

Ponieważ interesują nas napięcia względem masy stąd:

$$U_{1-2} = U_1 - U_2$$

Na tej podstawie prąd I_1 możemy zapisać jako:

$$I_1 = \frac{U_1 - U_2}{R_1} = (U_1 - U_2) * Y_1,$$

gdzie

$Y_1 = \frac{1}{R_1}$ — **admitancja** (stosujemy admitancję aby uprościć zapis)

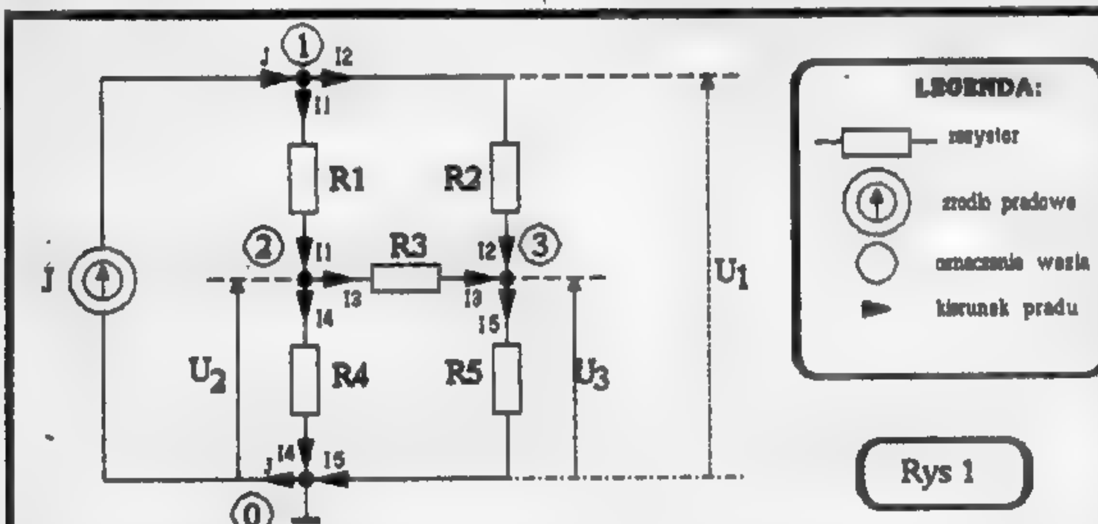
Analogicznie: $I_2 = (U_1 - U_3) * Y_2$

$I_3 = (U_2 - U_3) * Y_3$

$I_4 = U_2 * Y_4$

$I_5 = U_3 * Y_5$

Wyliczone prądy należy podstawić do bilansu prądów w węzłach. Po wymnożeniu i uporządkowaniu wyrażeń (proponuję czytelnikom samodzielnie wykonać te przekształcenia) otrzymu-



Rys 1

napięcie / węzeł	U1	U2	U3	b
1	Y1+Y2	-Y1+Y3	-Y2	J
2	-Y1	Y1+Y3+Y4	-Y3	0
3	-Y2	-Y3	Y2+Y3+Y5	0

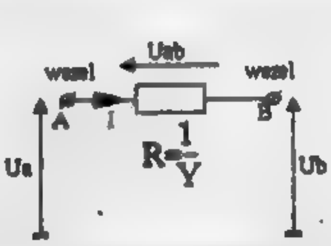
UKLAD ROWNAN dla schematu z rysunku 1

U1, U2, U3 - zmienne niewiadome

b - kolumna wyrazow stalych

1, 2, 3 - numery wezlow

Rys 2



Szablon dla rezystora

Uab-napiecie na rezystorze

Ua-napiecie na wezle A wzgl. masy

Ub-napiecie na wezle B wzgl. masy

napięcie / węzeł	Ua	Ub
A	Y	-Y
B	-Y	Y

Rys 3

jemy następujący układ równań:

węzeł 1: $(Y_1 + Y_2) \cdot \mathbf{U_1} + (-Y_1 + Y_3) \cdot U_2 + (-Y_2) \cdot \mathbf{U_3} = J$

węzeł 2: $(-Y_1) \cdot U_2 + (Y_1 + Y_3 + Y_4) \cdot \mathbf{U_2} + (-Y_3) \cdot \mathbf{U_3} = 0$

$\cdot U_2 + (-Y_2) \cdot \mathbf{U_3} = J$

węzeł 3: $(-Y_2) \cdot + U_1 + (-Y_3) \cdot \mathbf{U_2} + (Y_2 + Y_3 + Y_5) \cdot \mathbf{U_3} = 0$

Otrzymaliśmy w ten sposób układ trzech równań z trzema niewiadomymi: $\mathbf{U_1}, \mathbf{U_2}, \mathbf{U_3}$. Prąd źródła stałego (ponieważ nie zależy od żadnego napięcia) został przeniesiony na prawą stronę równania jako wartość stała

Osiągnęliśmy pierwszy sukces: wiemy jak na podstawie schematu, z praw Kirchhoffa, stworzyć układ równań matematycznych obliczających napięcia we wszystkich węzłach układów względem węzła odniesienia (masy).

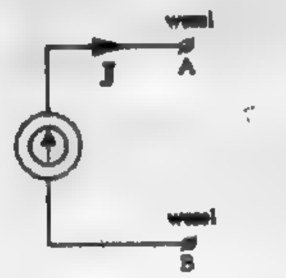
Na rys. 2 przedstawiłem tablicowy sposób zapisu tego równania. Będzie on bardzo pomocny w późniejszych rozważaniach. Wiersze tablicy odpowiadają kolejnym węzłom, a kolumny kolejnym napięciom. W ostatniej kolumnie (nazwanej **„b"**, tzw. kolumnie wyrazów wolnych) umieszcza się prądy stałe, niezależne od napięć w układzie. Przedstawiona tablica dokładnie odwzorowuje poprzednio napisane równania.

Zadaniem naszym jest stworzenie programu, który w sposób automatyczny będzie generował układ równań i obliczał napięcia w węzłach układu. Przyjrzyjmy się np. rezystorowi R_2. Jest on połączony z węzłami 1 i 3. W tablicy (na rys. 2) występuje w postaci admitancji Y_2. Zaznaczyłem ją dla wyróżnienia grubszą czcionką. Okazuje się, że występowanie admitancji Y_2 w tablicy jest określone przez węzły, do których ten rezystor jest dołączony: znajdują się na przecięciu wierszy 1 i 3 oraz kolumn U_1 i U_3. Na rys. 3 przedstawiłem uogólniony rezystor o rezystancji **R** (admitancji Y) podłączony do węzłów **A** i **B.** Uproszczona tablica obok przedstawia miejsca występowania admitancji Y w rzeczywistej tablicy układu. Niedowiarkom proponuję proste sprawdzenie: rozważmy występowanie admitancji dla naszego rezystora R_2: wtedy węzłowi A będzie odpowiadać węzeł 1, węzłowi B węzeł 3, napięciu Ua napięcie U_1, a napięciu Ub napięcie U_2. Okazuje się, że admitancje Y pokrywają się z admitancjami Y_2. Proponuję czytelnikom sprawdzenie zgodności tej metody dla rezystorów R_1 i R_3. Uproszczona tablica z rys. 3 nazywana jest **szablonem.** W omawianym wypadku jest to szablon dla rezystora. Każdy element występujący w układzie posiada swój szablon. Wtedy tworzenie tablicy dla dowolnego układu staje się bardzo proste: zgodnie z szablonem danego elementu dołączamy do tablicy układu odpowiednie współczynniki tego elementu (dla rezystora będzie to jego admitancja). Znaleźliśmy w ten sposób algorytm tworzenia tablicy dla dowolnego układu. Komputer musi posiadać następujące informacje: do których węzłów dany element jest podłączony i jaką ma wartość (wartością dla rezystora będzie rezystancja R, dla źródła prądowego wydajność prądowa J itp.) Na podstawie szablonu danego elementu i wyżej wymienionych informacji do odpowiednich miejsc tablicy układu dołączy współczynniki tego elementu.

Dociekliwi czytelnicy na pewno spostrzegli rażące odstępstwo od tych zasad dla rezystorów R_4 i R_5. Jednak tak nie jest: elementy te podłączone są do węzła zerowego, czyli węzła masy; a wszystko to, co dotyczy tego węzła jest odrzucane. Przypatrzmy się jeszcze raz szablonowi z rys. 3. Jeżeli np. węzeł B będzie węzłem masy, to wiersz szablonu opisujący węzeł B i kolumnę napięcia Ub należy odrzucić — zostanie wtedy jeden współczynnik Y na przecięciu wiersza A i kolumny Ua (współczynnik ten zaznaczony jest dla wyróżnienia grubszą czcionką). Taka właśnie sytuacja zachodzi dla rezystorów R_4 i R_5.

Reguła ta dotyczy wszystkich elementów!

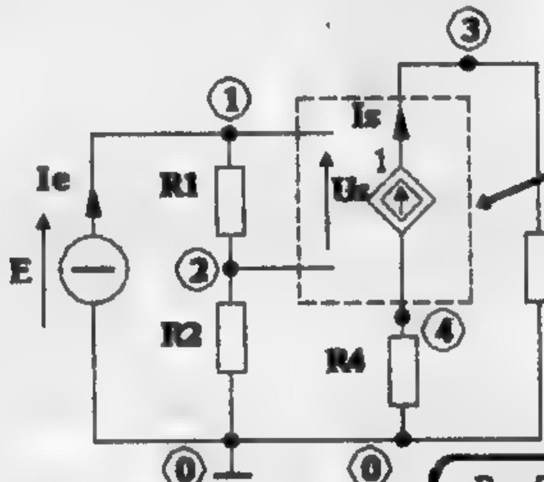
Podobna sytuacja zachodzi dla źród-



węzeł		b
A		J
B		-J

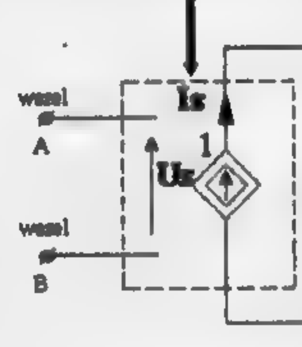
Rys 4

Szablon dla źródła prądowego o wydajności prądowej J



Rys 5a

źródło prądowe sterowane napięciowo Is=ys*Us



Rys 6a

węzeł / gałąź \ napięcie	U1	U2	U3	U4	Ie	b
A	Y1	-Y1			-1	
B	-Y1	Y1+Y2				
C	-ys	ys	Y3			
D	ys	-ys		Y4		
gałąź źródła napięc.	-1					-E

Rys 5b

	Ua	Ub
C	-ys	ys
D	ys	-ys

Rys 6b

ła prądowego, szablon którego przedstawiłem na rys. 4

Dużą grupę elementów stanowią tzw **źródła sterowane.** Rozważmy prosty schemat (rys. 5a) zawierający *źródło prądowe sterowane napięciem.* Źródło takie wymusza prąd Ju zależny od napięcia Us. W najprostszej wersji, którą będziemy stosować, **Ju = ys*Us (ys** jest współczynnikiem podawanym przez użytkownika). Do zasilania układu zastosowałem źródło napięciowe o napięciu E. Wprowadzi ono trochę zamieszania, bowiem bilans prądów w węzłach będzie wymagał przyjęcia nowej zmiennej prądu źródła **Ie**:

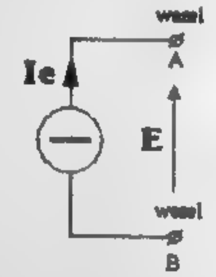
węzeł 1: $-Ie + I_1 = 0$
węzeł 2: $-I_1 + I_2 = 0$
węzeł 3: $-Is + I_3 = 0$
węzeł 4: $-I_4 + Is = 0$

równanie gałęziowe źródła E: $E = U_1$
gdzie: $I_1 = (U_1 - U_1) * Y_1$, $I_2 = U^2 * Y_2$, $I_3 = U_3 * Y_3$, $I_4 = U_4 * Y_4$, $Is = Us * ys = (U_1 - U_2) * ys$

Na rys. 5b przedstawiłem tablicę układu. Proponuję porównać z nią szablon modelu źródła prądowego (rys. 6a, rys. 6b). Na rys. 7 przedstawiłem szablon źródła napięciowego.

Uwaga: Wszystkie źródła o charakterze napięciowym wymagają wprowadzenia dodatkowej zmiennej, jaką jest prąd przez nie przepływający. Ta dodatkowa zmienna wymaga dołączenia jeszcze jednego równania, tzw. **równania gałęziowego** opisującego napięcie na węzłach, do których podłączone jest źródło.

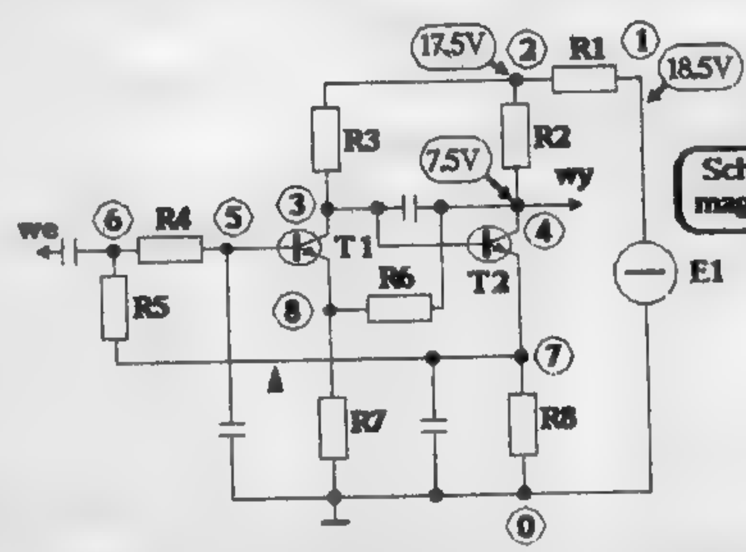
Na rys. 8a przedstawiłem schemat wzmacniacza wstępnego magnetofonu szpulowego M 2405 S. Ten układ poddałem stałoprądowej analizie numerycznej. Na rys. 8b przedstawiłem model tranzystora NPN, jaki stosuję w swoim programie: składa się on ze stałego źródła napięciowego i źródła prądowego sterowanego prądem przepływającym przez źródło napięciowe. Okazuje się, że nawet tak proste modele zapewniają dosyć dobrą dokładność.



napięcia / węzeł / gałąź	Ua	Ub	Ie	b
A			-1	
B			1	
gałąź źr.napięc.	-1	1		-E

Rys 7

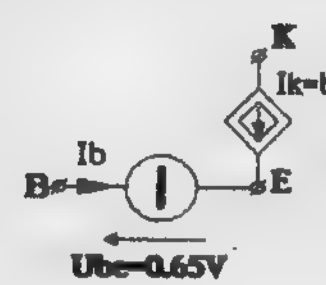
Szablon stałego źródła napięciowego



Rys 8a

Schemat wzmacniacza magnetofonu M 2405 S

E1-18.5V
T1-BC 413 B/C
T2-BC 149 B/C
R1-470
R2-5k1
R3-390k
R4-270
R5-27k
R6-100k
R7-620
R8-390



Rys 8b

Stałoprądowy model tranzystora NPN w stanie aktywnym
'b='beta'-stałoprądowe wzmocnienie
Ube-napięcie: baza-emiter

```
napiecia w wezlach ukladu:
wezel 1 = 18.5 V
wezel 2 = 17.575459 V

wezel 3 = 1.3730188 V
wezel 4 = 7.7550787 V
wezel 5 = 0.72019338 V
wezel 6 = 0.72022136 V
wezel 7 = 0.72301877 V
wezel 8 = .070193382 V

SZN1   U+=1   U-=0

       U = 18.5 V
       I = .0019671094 A
       P = .036391524 W

R1     U+=1   U-=2

       R = 470 Om
       U = 0.92454142 V
       I = .0019671094 A
       P = .0018186741 W

R2     U+=2   U-=4

       R = 5100 Om
       U = 9.8203799 V
       I = .0019255647 A
       P = .018909777 W

R3     U+=2   U-=3

       R = 390000 Om
       U = 16.20244 V
       I = .000041544717 A
       P = .00067312578 W

R4     U+=6   U-=5

       R = 270 Om
       U = .000027973903 V
       I = 1.0360705E-7 A
       P = 2.8982936E-12 W

R5     U+=7   U-=6

       R = 27000 Om
       U = .002797409 V
       I = 1.0360774E-7 A
       P = 2.8983322E-10 W

R6     U+=4   U-=8

       R = 100000 Om
```

```
U = 7.6848853 V
I = .000076848853 A
P = .00059057462 W

R7      U+=8   U-=0

        R = 620 Om
        U = .070193382 V
        I = .00011321513 A
        P = 7.9469531E-6 W

R8      U+=7   U-=0

        R = 390 Om
        U = 0.72301877 V
        I = .0018538943 A
        P = .0013404003 W

NPN1    B=5    K=3    E=8

        b = 350 [V/V]
      Ube= 0.65 V
      Uce= 1.3028254 V
       Ib= 1.0360764E-7 A
       Ic= .000036262672 A
       Ie= .00003636628
       P = .000047311275 W

NPN2    B=3    K=4    E=7

        b = 350 [V/V]
      Ube= 0.65 V
      Uce= 7.0320599 V
       Ib= 5.2820452E-6 A
       Ic= .0018487158 A
       Ie= .0018539979
       P = .013003714 W
```

Program dopuszcza posługiwanie się 9 elementami.
SZP — stałe źródło prądowe
SZN — stałe źródło napięciowe
ZPSN — źródło prądowe sterowane napięciem
ZPSP — źródło prądowe sterowane prądem
ZNSN — źródło napięciowe sterowane napięciem
ZNSP — źródło napięciowe sterowane prądem
R — rezystor
NPN — tranzystor bipolarny NPN w stanie aktywnym
PNP — tranzystor bipolarny PNP w stanie aktywnym
Dodatkowo można zasymulować:
rozwarcie: źródło prądowe o zerowej wydajności prądowej
zwarcie: źródło napięciowe o napięciu E = 0
wzmacniacz operacyjny: źródło napięciowe sterowane napięciem o współczynniku i równym wzmocnieniu wzmacniacza operacyjnego

Użytkownik za pomocą edytora wprowadza strukturę układu do komputera: podaje węzły, do których elementy są podłączone i ich współczynniki. Program wykonuje analizę dwuetapowo:

etap 1: następuje stworzenie macierzy (odpowiednik naszej tablicy) danego układu. Jednocześnie następuje kontrola poprawności struktury układu. Każdą niezgodność komputer wypisuje na ekranie. Po stwierdzeniu błędu program nie przechodzi do drugiego etapu.

etap 2: komputer rozwiązuje układ równań na podstawie macierzy stworzonej w etapie I wykorzystując tzw. **metodę Gaussa** z wyborem elementu głównego. Metoda ta, choć niezbyt szybka, jest bardzo dokładna numerycznie. Efektem końcowym jest wyświetlenie wyników na ekranie

Krzysztof POŻNIAK

```
   1 REM
*********************************
*                               *
*                        ©1988  *
*                               *
*      Krzysztof Poźniak        *
*                               *
* Analiza DC ukladow elektron.  *
*                               *
*********************************
  10 DEF FN z(p,k,z)=(z<p) OR (z
>k) OR (z<>INT z)
  11 DEF FN u(k)=t(2,e(x,l,k))
  12 DEF FN p(k)=t(2,k+9)
 100 REM
 105 PAPER 0. INK 7. BORDER 0. C
LS
 110 LET maxw=10: LET maxn=20
 120 GO SUB 7000
 130 GO SUB 7500
 140 GO TO 9000

1000 REM
1010 LET n=lz-1: FOR k=1 TO n: L
ET x(k)=k: NEXT k: FOR k=1 TO n:
 LET m=0: FOR y=k TO n: FOR x=k
TO n: IF ABS n(y,x)>m THEN LET m
=ABS n(y,x). LET my=y: LET mx=x
1020 NEXT x: NEXT y: IF m=0 THEN
 PRINT '"uklad nie ma rozwiazani
a ": RETURN
1030 FOR l=1 TO n: LET m=n(k,l):
 LET n(k,l)=n(my,l): LET n(my,l)
=m: NEXT l: LET m=b(k): LET b(k)
=b(my): LET b(my)=m: FOR l=1 TO
n: LET m=n(l,k): LET n(l,k)=n(l,
mx): LET n(l,mx)=m: NEXT l: LET
m=x(k): LET x(k)=x(mx): LET x(mx
)=m
1040 FOR l=k+1 TO n: LET m=n(l,k
)/n(k,k): FOR j=k+1 TO n: LET n(
l,j)=n(l,j)-n(k,j)*m: NEXT j: LE
T b(l)=b(l)-b(k)*m: NEXT l: NEXT
 k: FOR k=n TO 1 STEP -1: LET m=

0: FOR l=k+1 TO n: LET m=m+t(1,x
(l))*n(k,l): NEXT l: LET t(1,x(k
))=(b(k)-m)/n(k,k): NEXT k: LET
m=1: PRINT ''analiza numeryczna
zakonczona". RETURN
5000 REM
5001 REM uwagi i bledy
5005 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte zrodlo SZP";e
(x,l,1)
5006 IF e(x,l,4)=0 THEN PRINT "u
waga-zerowa wyd. pradowa SZP";e(
x,l,1)
5010 REM tworzenie macierzy
5015 LET k=3: GO SUB 6100: IF st
op=1 THEN RETURN
5020 IF e(x,l,2)>0 THEN LET b(FN
 u(2))=b(FN u(2))+e(x,l,4)
5021 IF e(x,l,3)>0 THEN LET b(FN
 u(3))=b(FN u(3))-e(x,l,4)
5022 RETURN
5030 REM wizualizacja wynikow

5035 PRINT ' INVERSE 1;"SZP";e(x
,l,1); INVERSE 0;TAB 6;"U+=";e(x
,l,2);TAB 12;"U-=";e(x,l,3)
5036 LET m1=2: LET m2=3: GO SUB
6400: PRINT 'TAB 6;"U = ";m;" V"
,TAB 6;"I = ";e(x,l,4);" A": PRI
NT TAB 6;"P = ";m*e(x,l,4);" W"
5040 RETURN
5100 REM
5105 IF e(x,l,4)=0 THEN PRINT "u
waga-zerowa wartosc nap. SZN";e(
x,l,1)
5106 IF e(x,l,2)=e(x,l,3) THEN P
RINT INVERSE 1;"blad-zwarte zrod
lo SZN";e(x,l,1): LET blad=1: RE
TURN
5110 REM tworzenie macierzy
5115 LET k=3: GO SUB 6100: LET k
=1: GO SUB 6200: IF stop=1 THEN
RETURN
5120 IF e(x,l,2)>0 THEN LET n(FN
 u(2),FN p(lp))=1: LET n(FN p(lp
),FN u(2))=-1
5121 IF e(x,l,3)>0 THEN LET n(FN
 u(3),FN p(lp))=-1: LET n(FN p(l
p),FN u(3))=1
5122 LET b(FN p(lp))=-e(x,l,4)
5125 RETURN
5130 REM wizualizacja wynikow
5135 PRINT ' INVERSE 1;"SZN";e(x
,l,1); INVERSE 0;TAB 6;"U+=";e(x
,l,2);TAB 12;"U-=";e(x,l,3)
5136 LET m1=2. LET m2=3: GO SUB
6400: PRINT 'TAB 6;"U = ";m;" V"
;TAB 6;"I = ";-t(1,FN p(lp));" A
": PRINT TAB 6;"P = ";-m*t(1,FN
p(lp)),"  W"
5140 LET lp=lp+1. RETURN
5200 REM
5201 REM uwagi i bledy
5205 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
PSN";e(x,l,1)
5206 IF e(x,l,4)=e(x,l,5) THEN P

RINT "uwaga-zwarte wezly prad. Z
PSN";e(x,l,1)
5207 IF e(x,l,6)=0 THEN PRINT IN
VERSE 1;"Blad-zerowy wspolczynni
k ZPSN";e(x,l,1): LET stop=1: RE
TURN
5210 REM tworzenie macierzy
5215 LET k=5: GO SUB 6100· IF st
op=1 THEN RETURN
5220 IF e(x,l,2)>0 AND e(x,l,4)>
0 THEN LET n(FN u(4),FN u(2))=n(
FN u(4),FN u(2))-e(x,l,6)
5221 IF e(x,l,2)>0 AND e(x,l,5)>
0 THEN LET n(FN u(5),FN u(2))=n(
FN u(5),FN u(2))+e(x,l,6)
5222 IF e(x,l,3)>0 AND e(x,l,4)>
0 THEN LET n(FN u(4),FN u(3))=n(
FN u(4),FN u(3))+e(x,l,6)
5223 IF e(x,l,3)>0 AND e(x,l,5)>
0 THEN LET n(FN u(5),FN u(3))=n(
FN u(5),FN u(3))-e(x,l,6)
5225 RETURN

5230 REM wizualizacja wynikow
5235 PRINT ' INVERSE 1;"ZPSN";e(
x,l,1), INVERSE 0;TAB 6;"S+=";e(
x,l,2);TAB 12;"S-=";e(x,l,3);TAB
 18;"I+=";e(x,l,4);TAB 24;"I-=";
e(x,l,5)
5236 LET m1=2: LET m2=3: GO SUB
6400: LET u1=m: LET m1=4: LET m2
=5· GO SUB 6400: PRINT 'TAB 6;"U
r=";e(x,l,6);TAB 6;"Us= ";u1;" V
";TAB 6;"U = ";m;" V";TAB 6;"I =
 ";u1*e(x,l,6);" A": PRINT TAB 6
;"P = ";m*u1*e(x,l,6);" W"
5240 RETURN
5300 REM
5301 REM uwagi i bledy
5305 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
PSP";e(x,l,1)
5306 IF e(x,l,4)=e(x,l,5) THEN P
RINT "uwaga-zwarte wezly prad. Z
PSP";e(x,l,1)

5307 IF e(x,l,6)=0 THEN PRINT IN
VERSE 1;"Blad-zerowy wspolczynni
k ZPSP";e(x,l,1): LET stop=1: RE
TURN
5310 REM tworzenie macierzy
5315 LET k=5: GO SUB 6100: LET k
=1. GO SUB 6200. IF stop=1 THEN
RETURN
5320 IF e(x,l,2)>0 THEN LET n(FN
 u(2),FN p(lp))=1: LET n(FN p(lp
),FN u(2))=-1
5321 IF e(x,l,3)>0 THEN LET n(FN
 u(3),FN p(lp))=-1: LET n(FN p(l
p),FN u(3))=1
5322 IF e(x,l,4)>0 THEN LET n(FN
 u(4),FN p(lp))=-e(x,l,6)
5323 IF e(x,l,5)>0 THEN LET n(FN
 u(5),FN p(lp))=e(x,l,6)
5325 RETURN
5330 REM wizualizacja wynikow
5335 PRINT ' INVERSE 1;"ZPSP";e(
x,l,1); INVERSE 0;TAB 6;"S+=";e(

x,l,2),TAB 12;"S-=";e(x,l,3);TAB
 18;"I+=";e(x,l,4);TAB 24;"I-=";
e(x,l,5)
5336 LET m1=4: LET m2=5: GO SUB
6400  PRINT 'TAB 6;"Uk=";e(x,l,6
),TAB 6;"Is= ";t(1,FN p(lp));" A
",TAB 6;"U = ";m;" V";TAB 6;"I =
 ";t(1,FN p(lp))*e(x,l,6);" A":
PRINT TAB 6;"P = ";m*t(1,FN p(lp
))*e(x,l,6);" W"
5340 LET lp=lp+1: RETURN
5400 REM
5401 REM uwagi i bledy
5405 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
NSN";e(x,l,1)
5406 IF e(x,l,6)=0 THEN PRINT "u
waga-zerowy wspolcz. ZNSN";e(x,l
,1)
```

```
5407 IF e(x,l,4)=e(x,l,5) THEN PRINT INVERSE 1;"blad-zwarte wezly nap. ZNSN";e(x,l,1): LET stop=1: RETURN
5410 REM tworzenie macierzy
5415 LET k=5: GO SUB 6100: LET k=1: GO SUB 6200: IF stop=1 THEN RETURN
5420 IF e(x,l,2)>0 THEN LET n(FN p(lp),FN u(2))=-e(x,l,6)
5421 IF e(x,l,3)>0 THEN LET n(FN p(lp),FN u(3))=e(x,l,6)
5422 IF e(x,l,4)>0 THEN LET n(FN u(4),FN p(lp))=-1: LET n(FN p(lp),FN u(4))=1
5423 IF e(x,l,5)>0 THEN LET n(FN u(5),FN p(lp))=1: LET n(FN p(lp),FN u(5))=-1
5425 RETURN
5430 REM wizualizacja wynikow
5435 PRINT ' INVERSE 1;"ZPSP";e(x,l,1); INVERSE 0;TAB 6;"S+=";e(x,l,2);TAB 12;"S-=";e(x,l,3);TAB 18;"I+=";e(x,l,4);TAB 24;"I-=";e(x,l,5)
5436 LET m1=2: LET m2=3: GO SUB 6400: PRINT 'TAB 6;"Wr=";e(x,l,6);TAB 6;"Us= ";m;" V";TAB 6;"U = ";m*e(x,l,6);" V";TAB 6;"I = ";t(1,FN u(lp));" A": PRINT TAB 6;"P = ",m*e(x,l,6)*t(1,FN u(lp)),"W"
5440 LET lp=lp+1: RETURN
5500 REM
5501 REM uwagi i bledy
5505 IF e(x,l,2)=e(x,l,3) THEN PRINT "uwaga-zwarte wezly ster. ZNSP";e(x,l,1)
5506 IF e(x,l,6)=0 THEN PRINT "uwaga-zerowy wspolcz. ZNSP";e(x,l,1)
5507 IF e(x,l,4)=e(x,l,5) THEN PRINT INVERSE 1;"blad-zwarte wezly nap. ZNSP";e(x,l,1): LET stop=1: RETURN
5510 REM tworzenie macierzy
5515 LET k=5: GO SUB 6100: LET k=2: GO SUB 6200: IF stop=1 THEN RETURN
5520 IF e(x,l,2)>0 THEN LET n(FN u(2),FN p(lp-1))=1: LET n(FN p(lp-1),FN u(2))=-1
5521 IF e(x,l,3)>0 THEN LET n(FN u(3),FN p(lp-1))=-1: LET n(FN p(lp-1),FN u(3))=1
5522 IF e(x,l,4)>0 THEN LET n(FN u(4),FN p(lp))=1: LET n(FN p(lp),FN u(4))=-1
5523 IF e(x,l,5)>0 THEN LET n(FN u(5),FN p(lp))=-1: LET n(FN p(lp),FN u(5))=1
5524 LET n(FN p(lp),FN p(lp-1))=e(x,l,6)
5525 RETURN
5530 REM wizualizacja wynikow
5535 PRINT ' INVERSE 1;"ZNSP";e(x,l,1); INVERSE 0;TAB 6;"S+=";e(x,l,2);TAB 12;"S-=";e(x,l,3);TAB 18;"U+=";e(x,l,4);TAB 24;"U-=";e(x,l,5)
5536 PRINT 'TAB 6;"Wr=";e(x,l,6);TAB 6;"Is= ";t(1,FN p(lp));" A";TAB 6;"U = ";t(1,FN p(lp))*e(x,l,6);" V";TAB 6;"I = ";t(1,FN p(lp+1));" A": PRINT TAB 6;"P = ";e(x,l,6)*t(1,FN p(lp))*t(1,FN p(lp+1));" W"
5540 LET lp=lp+2: RETURN
5600 REM
5601 REM uwagi i bledy
5605 IF e(x,l,2)=e(x,l,3) THEN PRINT "uwaga-zwarty rezystor R";e(x,l,1)
5606 IF e(x,l,4)=0 THEN PRINT INVERSE 1;"blad-zerowa rezystancja R";e(x,l,1): LET stop=1: RETURN
5610 REM tworzenie macierzy
5615 LET k=3: GO SUB 6100
5620 IF e(x,l,2)>0 THEN LET n(FN u(2),FN u(2))=n(FN u(2),FN u(2))+1/e(x,l,4): IF e(x,l,3)>0 THEN LET n(FN u(2),FN u(3))=n(FN u(2),FN u(3))-1/e(x,l,4)
5621 IF e(x,l,3)>0 THEN LET n(FN u(3),FN u(3))=n(FN u(3),FN u(3))+1/e(x,l,4): IF e(x,l,2)>0 THEN LET n(FN u(3),FN u(2))=n(FN u(3),FN u(2))-1/e(x,l,4)
5625 RETURN
5630 REM wizualizacja wynikow
5635 PRINT ' INVERSE 1;"R";e(x,l,1); INVERSE 0;TAB 6;"U+=";e(x,l,2);TAB 12;"U-=";e(x,l,3)
5636 LET m1=2: LET m2=3: GO SUB 6400: PRINT 'TAB 6;"R = ";e(x,l,4);" Om";TAB 6;"U = ";m;" V";TAB 6;"I = ";m/e(x,l,4);" A": PRINT TAB 6;"P = ";m*m/e(x,l,4);" W"
5640 RETURN
5700 REM
5701 REM uwagi i bledy
5705 IF e(x,l,2)=e(x,l,3) THEN PRINT "uwaga-zwarte: B i K NPN",e(x,l,1)
5706 IF e(x,l,3)=e(x,l,4) THEN PRINT "uwaga-zwarte: K i E NPN";e(x,l,1)
5707 IF e(x,l,5)=0 THEN PRINT "uwaga-zerowe wzmocnienie NPN";e(x,l,1)
5708 IF e(x,l,2)=e(x,l,4) THEN PRINT INVERSE 1;"blad-zwarte: B i E NPN";e(x,l,1): LET stop=1: RETURN
5710 REM tworzenie macierzy
5715 LET k=4: GO SUB 6100: LET k=1: GO SUB 6200: IF stop=1 THEN RETURN
5720 IF e(x,l,2)>0 THEN LET n(FN u(2),FN p(lp))=1: LET n(FN p(lp),FN u(2))=-1
5721 IF e(x,l,3)>0 THEN LET n(FN u(3),FN p(lp))=e(x,k,5)
5722 IF e(x,l,4)>0 THEN LET n(FN u(4),FN p(lp))=-1-e(x,l,5): LET n(FN p(lp),FN u(4))=1
5723 LET b(FN p(lp))=-.65
5725 RETURN
5730 REM wizualizacja wynikow
5735 PRINT ' INVERSE 1;"NPN";e(x,l,1), INVERSE 0,TAB 6;" B=";e(x,l,2);TAB 12;" K=";e(x,l,3);TAB 18;" E=";e(x,l,4)
5736 LET m1=3: LET m2=4: GO SUB 6400: PRINT 'TAB 6;"b = ";e(x,l,5);" [V/V]";TAB 5;"Ube= 0.65 V";TAB 5;"Uce= ";m;" V";TAB 6;"Ib= ";t(1,FN p(lp));" A";TAB 6;"Ic= ";e(x,l,5)*t(1,FN p(lp));" A";TAB 6;"Ie= ";t(1,FN p(lp))*(1+e(x,l,5));TAB 6;"P = ";(.65+m*e(x,l,5))*t(1,FN p(lp));" W"
5740 LET lp=lp+1: RETURN
5800 REM
5801 REM uwagi i bledy
5805 IF e(x,l,2)=e(x,l,3) THEN PRINT "uwaga-zwarte: B i K PNP";e(x,l,1)
5806 IF e(x,l,3)=e(x,l,4) THEN PRINT "uwaga-zwarte: K i E PNP";e(x,l,1)
5807 IF e(x,l,5)=0 THEN PRINT "uwaga-zerowe wzmocnienie PNP";e(x,l,1)
5808 IF e(x,l,2)=e(x,l,4) THEN PRINT INVERSE 1;"blad-zwarte: B i E PNP";e(x,l,1): LET stop=1: RETURN
5810 REM tworzenie macierzy
5815 LET k=4: GO SUB 6100: LET k=1: GO SUB 6200: IF stop=1 THEN RETURN
5820 IF e(x,l,2)>0 THEN LET n(FN u(2),FN p(lp))=-1: LET n(FN p(lp),FN u(2))=1
5821 IF e(x,l,3)>0 THEN LET n(FN u(3),FN p(lp))=-e(x,k,5)
5822 IF e(x,l,4)>0 THEN LET n(FN u(4),FN p(lp))=1+e(x,l,5): LET n(FN p(lp),FN u(4))=-1
5823 LET b(FN p(lp))=-.65
5825 RETURN
5830 REM wizualizacja wynikow
5835 PRINT ' INVERSE 1;"PNP";e(x,l,1); INVERSE 0;TAB 6;" B=";e(x,l,2);TAB 12;" K=";e(x,l,3);TAB 18;" E=";e(x,l,4)
5836 LET m1=3: LET m2=4: GO SUB 6400: PRINT 'TAB 6;"b = ";e(x,l,5);" [V/V]";TAB 5;"Ube= -0.65 V";TAB 5;"Uce= ";m;" V";TAB 6;"Ib= ";t(1,FN p(lp));" A";TAB 6;"Ic= ";e(x,l,5)*t(1,FN p(lp));" A";TAB 6;"Ie= ";t(1,FN p(lp))*(1+e(x,l,5));TAB 6;"P = ";-(.65+m*e(x,l,5))*t(1,FN p(lp));" W"
5840 LET lp=lp+1: RETURN
6000 REM
6010 CLS
6020 GO SUB 8000+100*(x-1)
6030 GO SUB 9800
6040 GO SUB 9990
6050 IF x$="p" AND i(x)>0 THEN LET k=0: GO SUB 9600: IF e>0 THEN GO TO 6070
6051 IF x$="d" AND i(x)<10 THEN LET i(x)=i(x)+1: LET e=i(x): GO TO 6070
6052 IF x$="s" AND i(x)>0 THEN GO SUB 9600: IF e>0 THEN GO SUB 8050+100*(x-1): GO TO 6000
6053 IF x$="k" THEN RETURN
6060 BEEP .1,20: GO TO 6040
6070 GO SUB 8030+100*(x-1): GO TO 6000
6100 REM
6105 IF lz+k>maxn THEN LET stop=1: PRINT "blad-przekroczono rozmiar macierzy": RETURN
6110 FOR j=2 TO k: IF e(x,l,j)>0 THEN IF t(2,e(x,l,j))=0 THEN LET t(2,e(x,l,j))=lz: LET lz=lz+1
6115 NEXT j: RETURN
6200 REM
6205 IF lp+k>maxn/2 THEN LET stop=1: PRINT "blad-przekroczono ilosc pradow": RETURN
6210 FOR j=1 TO k: LET t(2,lp+10)=lz: LET lp=lp+1: LET lz=lz+1: NEXT j: RETURN
6300 REM
6310 CLS : GO SUB 7500: LET stop=0: LET lz=1: LET lp=1: PRINT "analiza ukladu rozpoczeta"''
6350 FOR x=1 TO 9: FOR l=1 TO i(x): GO SUB 5000+100*(x-1): NEXT l: NEXT x: IF stop=1 THEN PRINT '"bledy w ukladzie-koniec analizy": GO TO 6395
6351 IF lz=1 THEN PRINT "brak ukladu - koniec analizy": GO TO 6395
6355 PRINT '"analiza ukladu zakonczona"'''"analiza numeryczna rozpoczeta"
6360 GO SUB 1000: IF m=0 THEN GO TO 6395
6362 PRINT '"napiecia w wezlach ukladu:": FOR x=1 TO maxn/2: IF t(2,x)=0 THEN GO TO 6364
6363 PRINT "wezel ";x;" = ";t(1,t(2,x));" V"
6364 NEXT x
6365 LET lp=2: FOR x=1 TO 9: FOR l=1 TO i(x): GO SUB 5030+100*(x-1): NEXT l: NEXT x
6370 PRINT #0;"czy powtorzyc wyniki? (t/n)": PAUSE 1: PAUSE 0: INPUT "": IF INKEY$="t" THEN GO TO 6362
6375 RETURN
6395 PRINT #0;"nacisnij dowolny klawisz": PAUSE 1: PAUSE 0: INPUT "": RETURN
6400 REM
6410 LET x1=0: IF e(x,l,m1)>0 THEN LET x1=t(1,FN u(m1))
6420 LET x2=0: IF e(x,l,m2)>0 THEN LET x2=t(1,FN u(m2))
6430 LET m=x1-x2: RETURN
7000 REM
7010 DIM e(9,10,7): DIM i(10)
7020 RETURN
7500 REM
7510 DIM n(maxn,maxn): DIM b(maxn): DIM t(2,maxn): DIM x(maxn)
7520 RETURN
8000 REM
8010 REM lista
8015 PRINT "Stale Zrodla Pradowe": PRINT
8020 PRINT "nr element I+ I- I[A]"
8025 LET x$="NZP": GO SUB 9110: RETURN
8030 REM edycja elementu
8040 LET k=0: GO SUB 9600
8041 LET x$="nr wezla I+": GO SUB 9500
8042 LET x$="nr wezla I-": GO SUB 9500
8043 LET x$="prad zrodla w [A]": GO SUB 9550
8045 RETURN
8050 REM kasowanie elementu
8055 LET l=4
8060 GO SUB 9400
8065 RETURN
8100 REM
8110 REM lista
8115 PRINT "Stale Zrodla Napieciowe": PRINT
8120 PRINT "nr element U+ U- U[V]"
8125 LET x$="NZN": GO SUB 9110: RETURN
8130 REM edycja elementu
8140 LET k=0: GO SUB 9600
8141 LET x$="nr wezla U+": GO SUB 9500
8142 LET x$="nr wezla U-": GO SUB 9500
8143 LET x$="napiecie zrodla w [V]": GO SUB 9550
8145 RETURN
8150 REM kasowanie elementu
8155 LET l=4
8160 GO SUB 9400
8165 RETURN
8200 REM
8210 REM lista
8215 PRINT "Zrodla Pradowe Ster. Napieciowo": PRINT
8220 PRINT "nr element S+ S- U+ U- Uy[S]"
8225 LET x$="ZPSN": GO SUB 9130: RETURN
8230 REM edycja elementu
8240 LET k=0: GO SUB 9600
8241 LET x$="nr wezla S+": GO SUB 9500
8242 LET x$="nr wezla S-": GO SUB 9500
8243 LET x$="nr wezla U+": GO SUB 9500
8244 LET x$="nr wezla U-": GO SUB 9500
8245 LET x$="wspolczynnik R[Om]": GO SUB 9550
8246 RETURN
8250 REM kasowanie elementu
8255 LET l=6
8260 GO SUB 9400
8265 RETURN
8300 REM
8310 REM lista
8315 PRINT "Zrodla Pradowe Ster. Pradowo": PRINT
8320 PRINT "nr element S+ S- I+ I- Uk[]"
8325 LET x$="ZPSP": GO SUB 9130: RETURN
8330 REM edycja elementu
8340 LET k=0: GO SUB 9600
8341 LET x$="nr wezla S+": GO SUB 9500
8342 LET x$="nr wezla S-": GO SUB 9500
8343 LET x$="nr wezla I+": GO SUB 9500
8344 LET x$="nr wezla I-": GO SUB 9500
```

```
8345 LET x$="wspolczynnik k[]":
GO SUB 9550
8346 RETURN
8350 REM kasowanie elementu
8355 LET l=6
8360 GO SUB 9400
8365 RETURN
8400 REM
8410 REM lista
8415 PRINT "Zr. Napieciowe Ster.
 Napieciowo": PRINT
8420 PRINT "nr element S+ S- U+
U- Ut[]"
8425 LET x$="ZNSN": GO SUB 9130:
 RETURN
8430 REM edycja elementu
8440 LET k=0: GO SUB 9500
8441 LET x$="nr wezla S+": GO SU

8442 LET x$="nr wezla S-": GO SU
B 9500
8443 LET x$="nr wezla U+": GO SU
B 9500
8444 LET x$="nr wezla U-": GO SU
B 9500
8445 LET x$="wspolczynnik t[]":
GO SUB 9550
8446 RETURN
8450 REM kasowanie elementu
8455 LET l=6
8460 GO SUB 9400
8465 RETURN
8500 REM
8510 REM lista
8515 PRINT "Zrodla Napieciowe St
er. Pradowo": PRINT
8520 PRINT "nr element S+ S- I+
I- Wr[Om]"
8525 LET x$="ZNSP": GO SUB 9130:
 RETURN
8530 REM edycja elementu
8540 LET k=0: GO SUB 9500
8541 LET x$="nr wezla S+": GO SU
B 9500
8542 LET x$="nr wezla S-": GO SU
B 9500
8543 LET x$="nr wezla I+": GO SU
B 9500
8544 LET x$="nr wezla I-": GO SU

8545 LET x$="wspolczynnik Y[S]":
 GO SUB 9550
8546 RETURN
8550 REM kasowanie elementu
8555 LET l=6
8560 GO SUB 9400
8565 RETURN
8600 REM
8610 REM lista
8615 PRINT "Rezystory": PRINT
8620 PRINT "nr element W+ W- R[O
m]"
8625 LET x$="R": GO SUB 9110: RE
TURN
8630 REM edycja elementu
8640 LET k=0: GO SUB 9500
8641 LET x$="nr wezla W+": GO SU

8642 LET x$="nr wezla W-": GO SU
B 9500
8643 LET x$="rezystancja R[Om]":
 GO SUB 9550
8644 RETURN
8650 REM kasowanie elementu
8655 LET l=4
8660 GO SUB 9400
8665 RETURN
8700 REM
8710 REM lista
8715 PRINT "Tranzystory NPN": PR
INT
8720 PRINT "nr element B  K  E
b[A/A]"
8725 LET x$="NPN": GO SUB 9120:
RETURN
8730 REM edycja elementu
8740 LET k=0: GO SUB 9500
8741 LET x$="nr wezla B": GO SUB
 9500
8742 LET x$="nr wezla K": GO SUB
 9500
8743 LET x$="nr wezla E": GO SUB

8744 LET x$="wzmocnienie b[A/A]"
: GO SUB 9550
8745 RETURN
8750 REM kasowanie elementu
8755 LET l=5
8760 GO SUB 9400
8765 RETURN
8800 REM
8810 REM lista
8815 PRINT "Tranzystory PNP": PR
INT
8820 PRINT "nr element B  K  E
b[A/A]"
8825 LET x$="PNP": GO SUB 9120:
RETURN
8830 REM edycja elementu
8840 LET k=0: GO SUB 9500
8841 LET x$="nr wezla B": GO SUB
 9500
8842 LET x$="nr wezla K": GO SUB
 9500
8843 LET x$="nr wezla E": GO SUB

8844 LET x$="wzmocnienie b[A/A]"
  GO SUB 9550
8845 RETURN
8850 REM kasowanie elementu
8855 LET l=5
8860 GO SUB 9400
8865 RETURN
9000 REM
9010 GO SUB 9900
9020 GO SUB 9990
9030 IF x$>="0" AND x$<="8" THEN
 LET x=VAL x$+1: CLS : GO SUB 80
00: GO TO 9000
9040 IF x$="w" THEN GO SUB 6300
 GO TO 9000
9041 IF x$="n" THEN PAUSE 1: PRI
NT #0;"nowy uklad? (t/n)": PAUSE
 0: INPUT "": IF INKEY$="t" THEN
 GO SUB 7000: GO TO 9000
9042 IF x$="l" THEN CLS : PRINT
"lista elementow ukladu": FOR x=
1 TO 9: GO SUB 8025+100*(x-1): N
EXT x: PRINT : PRINT #0;"Nacisni
j dowolny klawisz": PAUSE 1: PAU
SE 0: INPUT "": GO TO 9000
9045 BEEP .1,20
9050 GO TO 9020
9100 REM
9110 REM 2 wezly,1 parametr
9115 FOR k=1 TO i(x): PRINT k;TA
B 3;x$;e(x,k,1);TAB 11;e(x,k,2),
TAB 14;e(x,k,3);TAB 17;e(x,k,4)
 NEXT k: RETURN
9120 REM 3 wezly,1 parametr
9125 FOR k=1 TO i(x): PRINT k;TA
B 3;x$;e(x,k,1);TAB 11;e(x,k,2);
TAB 14;e(x,k,3);TAB 17;e(x,k,4);
TAB 20;e(x,k,5): NEXT k: RETURN
9130 REM 4 wezly,1 parametr
9135 FOR k=1 TO i(x): PRINT k;TA
B 3;x$;e(x,k,1);TAB 11;e(x,k,2);
TAB 14;e(x,k,3);TAB 17;e(x,k,4);
TAB 20;e(x,k,5),TAB 23;e(x,k,6):
 NEXT k: RETURN
9400 REM
9410 LET i(x)=i(x)-1: IF e=10 TH
EN RETURN
9420 FOR k=e TO i(x): FOR j=1 TO
 l
9430 LET e(x,k,j)=e(x,k+1,j)
9440 NEXT j: NEXT k: RETURN
9500 REM
9505 IF k=0 THEN LET x$="numer u
kladowy elementu"
9510 LET k=k+1
9515 INPUT (x$);": ";e(x,e,k)
9520 IF FN z(k=1,maxw,e(x,e,k))
THEN GO TO 9515
9530 RETURN
9550 REM
9560 LET k=k+1: INPUT (x$);": ";
e(x,e,k)
9565 RETURN
9600 REM
9610 INPUT "podaj nr elementu: "
,e: IF FN z(0,i(x),e) THEN GO TO
 9610
9620 RETURN
9800 REM
9810 PRINT AT 15,11; INVERSE 1;"
FUNKCJE."
9820 PRINT
9830 IF i(x)>0 THEN PRINT TAB 5;
"p-poprawienie elementu"
9831 IF i(x)<10 THEN PRINT TAB 5
;"d-dolaczenie elementu"
9832 IF i(x)>0 THEN PRINT TAB 5;
"s-skasowanie elementu"
9833 PRINT TAB 5;"k-koniec edycj
i zbioru"
9840 RETURN
9900 REM
9910 CLS : PRINT TAB 7; INVERSE
1;"FUNKCJE EDYTORA:"
9915 PRINT
9920 PRINT "0-Stale Zrodlo Prado
we"
9921 PRINT "1-Stale Zrodlo Napie
ciowe"
9922 PRINT "2-Zr. Pradowe Ster.
Napieciowo"
9923 PRINT "3-Zr. Pradowe Ster.
Pradowo"
9924 PRINT "4-Zr. Napiecowe Ster
. Napieciowo"
9925 PRINT "5-Zr. Napieciowe Ste
r. Pradowo"
9926 PRINT "6-Rezystor"
9927 PRINT "7-Tranzystor NPN"
9928 PRINT "8-Tranzystor PNP"
9930 PRINT : PRINT "w-wykonanie
obliczen"
9931 PRINT "n-nowy uklad"
9932 PRINT "l-lista elementow uk
ladu"
9940 RETURN
9990 REM
9991 PRINT #0;AT 0,5;"wybierz fu
nkcje"
9992 PAUSE 0: LET x$=INKEY$: IF
x$<>"" THEN INPUT "": RETURN
9993 GO TO 9991
9996 REM
9999 CLEAR : SAVE "analiza DC" L
INE 1
```

Zgadywanka — gra

```
#include <stdio.h>

void main()
{
int wzorzec,liczba,licznik;
printf ("liczbe wzorcowa podaje pierwsza ");
printf (" osoba, druga ja zgaduje\n\n");
do
{
   printf("podaj wzorzec <0-999>: ");
   wzorzec=dana ();
   system ( cls );
   for (licznik=1;licznik<=20;licznik++)
   {
      printf("trafiasz %d raz (wpisz liczbe <0-999>): ",
      licznik);
      liczba=dana ();
      if (liczba==wzorzec)
       {
       printf(" - trafiles!!!\n");
       goto end;
       }
       else
       {
       if (liczba>wzorzec)
        printf(" - liczba za duza\n");
        else
           printf(" - liczba za mala\n");
       }
    }
    printf("\ntrafiasz i trafiasz, az do znudzenia!.");
    printf(" Przerywam program!");
    break;
    end:
    printf("\nczy grasz jeszcze raz? (t/n)\n\n");
    }
while (getch()=='t');
}

dana () /*wczytanie liczby 3-cyfrowej*/
{
   int licznik,liczba=0;
   char znak;
   #define maxchar 3
   #define enter 13
   for (licznik=1;licznik<=maxchar;licznik++)
   {
      do
      {
         znak=getch();
         if ((znak==enter)&&(licznik>1))
            goto end;
      }
      while((znak<'0')||(znak>'9'));
      putchar(znak);
      liczba=liczba*10+znak-'0';
   }
end:
return(liczba);
}
```

Jak oszczędnie przechowywać ekrany na dyskietce

MIECZYSŁAW SKONIECZNY, PIOTR RAKOWSKI

Obraz wyświetlany na ekranie mikrokomputera AMSTRAD CPC 6128 zajmuje 17 KB pamięci począwszy od adresu C000. Można go przechowywać na dyskietce, gdzie zajmuje około 10% zawartości jednej strony. Chcemy przedstawić programy do kompresji i dekompresji obrazów. Używając ich musimy ograniczyć wielkość pamięci przeznaczonej na BASIC do 16383 bajtów (instrukcja MEMORY 16383). Pierwszy z omawianych programów służy do kompresji ekranu. Skompresowany obraz jest umieszczony w pamięci od adresu 4002, a jego długość (potrzebna do nagrania na dyskietkę) można odczytać pod adresem 4000. Kompresja obrazu jest realizowana przez zastąpienie ciągu identycznych bajtów dwoma przedstawiającymi ich wartość i długość ciągu. Program ten jest relokowalny w pamięci mikrokomputera. Wywołuje się go instrukcją:

CALL adres

gdzie „adres" jest wartością, od której nagramy program do pamięci. Wersja źródłowa programu jest następująca:

```
10 'kod maszynowy  "kompres"
20 adres=38500:iloscl1nii=5
30 RESTORE 200
40 FOR i=1 TO iloscl1nii
50 s=0:READ a$,w$
60 FOR j=1 TO 29 STEP 2
70 a=VAL("&"+MID$(a$,j,2))
80 POKE adres,a
90 s=s+a:adres=adres+1
100 NEXT j
110 IF s<>VAL(w$) THEN PRINT"Blad w linii nr ";190+i*10:STOP
120 NEXT i
130 SAVE"kompres",b,38500,65
200 DATA 1102402100c00ec80650c5e50e007e,1174
210 DATA 0c2305280abe28f8127913121318ee,1037
220 DATA 1279131213e17cfef828023809d638,1423
230 DATA 6701500009180401000809c10d20ce,683
240 DATA eb220040c900000000000000000000,534
```

Drugi z programów służy do dekompresji obrazów utworzonych poprzednio programem „kompres". Aby otrzymać obraz na ekranie należy wgrać do pamięci mikrokomputera od adresu 4000 skompresowany obrazek i wywołać program „dekompres" instrukcją:

CALL adres

gdzie „adres„ jest wartością, od której nagrywamy program do pamięci. Wersja źródłowa tego programu jest następująca:

```
10 'kod maszynowy  "dekompres"
20 adres=38600:iloscl1nii=4
30 RESTORE 200
40 FOR i=1 TO iloscl1nii
50 s=0:READ a$,w$
60 FOR j=1 TO 29 BTEP 2
70 a=VAL("&"+MID$(a$,j,2))
80 POKE adres,a
90 s=s+a:adres=adres+1
100 NEXT j
110 IF s<>VAL(w$) THEN PRINT"Blad w linii nr ";190+i*10:STOP
120 NEXT i
130 SAVE"dekompre",b,38600,58
200 DATA 2102401100c00ec8d50650c57e234e,1257
210 DATA 231213050d20fa78b720f2c1d1c57a,1670
220 DATA fef82802380bd63857015000eb09eb,1528
230 DATA 1806010008eb09ebc10d20cfc90000,1164
```

Program demonstracyjny napisany w języku BASIC przedstawia, jak należy wykonać kompresję (linie 120—360) i dekompresję obrazu (linie 380—520). Kompresowane obrazy są nagrywane na dyskietkę z rozszerzeniem CMP. Dekompresja jest realizowana także w zbiorach z rozszerzeniem CMP. Wersja źródłowa tego programu jest następująca:

```
10 'program demonstracyjny
20 'kompresja i dekompresja ekranu
30 '
40 MEMORY 16383
50 MODE 1:WINDOW £3,12,28,13,18:PAPER £3,1:PEN £3,0:CLS £3
60 PRINT£3:PRINT£3," 1 - kompresja"
70 PRINT£3," 2 - dekompresja"
80 PRINT£3:PRINT£3,"      Wybierz"
90 a$=INKEY$:IF a$<>"1" AND a$<>"2" THEN 90
100 IF a$="2" THEN 380
110 '
120 'kompresja
130 MODE 2
140 LOAD"kompres
150 WINDOW £2,20,60,21,25:PAPER £2,1:PEN £2,0
160 CLS£2
170 PRINT£2:PRINT£2," Wloz dyskietke z obrazkiem do kompresji"
180 PRINT£2," i nacisnij dowolny klawisz."
190 CALL &BB06
200 CAT
210 CLS£2
220 PRINT£2:PRINT£2," Nazwa obrazka do kompresji :";:INPUT£2,"", a$
230 PRINT£2," Nazwa obrazka po kompresji :";:INPUT£2,"",b$
240  LOAD a$+".scr",&C000
250 CALL 38500
260 CLS:CLS£2
270 PRINT£2
280 PRINT£2,"   Zmien dyskietke i nacisnij dowolny"
290 PRINT£2,"   klawisz ,aby nagrac skompresowany "
300 PRINT£2,"   obrazek."
310 CALL &BB06
320 DL=PEEK(&4000)+256*PEEK(&4001)-16385
330 SAVE b$+".cmp",B,&4002,DL
340 CLS£2:PRINT£2:PRINT£2,"   Nastepny obrazek ? :";:INPUT£2,"",a$
350 IF a$="t" OR a$="T" THEN 160
360 RUN
370 '
380 'dekompresja
390 MODE 2:LOAD"dekompre
400 WINDOW £1,20,60,21,25:PAPER £1,1:PEN £1,0
410 CLS£1:PRINT£1
420 PRINT£1,"     Wloz dyskietke z obrazkiem do "
430 PRINT£1,"     dekompresji i nacisnij dowolny"
440 PRINT£1,"     klawisz."
450 CALL &BB06
460 CLS£1:CAT:PRINT£1:PRINT£1," Nazwa obrazka do dekompresji :";:INPUT£1,"",a$
470 PRINT£1:PRINT£1," W jakim trybie ?:";:INPUT£1,"",tryb
480 MODE tryb
490 LOAD a$+".cmp",&4002
500 CALL 38600
510 FOR i=1 TO 10000:NEXT
520 RUN
```

SYSTEMY OPERACYJNE I INNE GOTOWE PROGRAMY

O sprzęcie powiedzieliśmy sobie już bardzo dużo, w każdym razie wiele jak na Podstawy Informatyki. Niestety w tym przypadku ilość nie "przechodzi" w jakość. Urządzenia te, choćby wszystkie zgromadzone razem w jednym miejscu będą po prostu górą (zresztą pokaźną) złomu metali kolorowych, sztucznego tworzywa, szkła i innych składowych tego co potocznie nazywamy komputerem.

Po "ożywieniu" sprzęt ten, jak niegdyś, *Frankenstein*, natychmiast przystępuje do działania. Szczęśliwie komputery nie zachowują się jak dziecko wyobraźni pani Shelly, są bardzo "łagodne", cierpliwe - po prostu są *życzliwe*. Jest to możliwe dzięki specjalnej formie owego *ożywienia*. Polega ono na podaniu energii (w Europie najlepiej 220V i 50 Hz) oraz wprowadzeniu do pamięci maszyny odpowiedniego programu. Pamiętając o naszych doświadczeniach z programowaniem (IKS nr 11/88) nie trudno jest sobie wyobrazić ogrom pracy, jaki trzeba włożyć w pisanie programów, które zapewnią prawidłowe działanie tysięcy komputerów.

Wróćmy na chwilę do *monstrów*, które współcześnie nam "obrodziły", na przykład w filmach "Superglina" i "Elektroniczny morderca". Ożywienie tych postaci polegało na przekazaniu naszym bohaterom zdolności (czytaj programów) poruszania kończynami, widzenia i analizy obrazu, słyszenia i rozumienia mowy ludzkiej. Poza sposobem realizacji poleceń otrzymywanych przez owe istoty, gdy uwidacznia się ich indywidualność (na przykład czarny charakter) pozostałe zdolności są typowe, niezmienne. Czy był to Elektroniczny morderca, czy Superglina - istoty te wykonywały te same czynności. Jeśli okazało by się, że są to komputery, a było tak w istocie (prawie), wystarczyło powielać ten sam zakodowany w ich *sztucznym mózgu (komputerze?)* program, by standardowe zachowania owych postaci były realizowane. Nie trzeba zatem płacić ogromnych sum za *oprogramowanie* kolejnych egzemplarzy naszych bohaterów - przynajmniej w części dotyczącej ich podstawowych zachowań, jakimi w tym przypadku, poza chodzeniem, bieganiem, była nadzwyczajna celność strzelania z broni palnej. Można bowiem powielać program sterujący wykonywaniem typowych czynności... komputera w każdej podobnej maszynie.

Te *realistyczne* przykłady mają na celu zwrócenie uwagi na ideę konstrukcji tego typu programów:

a) wykonywanie często powtarzanych przez użytkownika czynności,

b) możliwość wykorzystania określonej wersji programu (systemu) w wielu komputerach.

Kupując zatem nowy komputer, nie musimy budować od nowa programu, który na przykład *przejrzy* nam dysk twardy i wyświetli jego zawartość, w czytelnej dla nas postaci. Podobnie przydałby się programik, który automatycznie sprawdzi dyskietkę, czy jest dobra (bez *dziur*) i co na niej zapisano. Mało tego, nawet ruch głowicy po powierzchni dysku też trzeba *oprogramować*. Oznacza to, że każdą najdrobniejszą czynność wykonywaną przez maszynę należało wcześniej przewidzieć i opisać (w języku wewnętrznym komputera). Zważywszy zatem na wspomnianą uniwersalność omawianych operacji napisano podprogramy (moduły), które złożone w całość stanowią o podstawowych *zdolnościach* komputera. Nie jest to oczywiście *chodzenie*, a na przykład czytanie (z dysku). Wspomniana całość to SYSTEM OPERACYJNY - program, który wykonuje wszystkie standardowe (typowe) funkcje powierzane maszynie. Oczywiście mamy bardzo wiele typów systemów operacyjnych - małych i dużych.

Przyjrzyjmy się najpopularniejszemu (na pewno u nas) - DOSowi - Dyskowy Operacyjny System dla mikrokomputerów PC. Jego nazwa wskazuje, że program ten powstał głównie po to, by usprawnić wykorzystywanie dysków (twardych i elastycznych), i tak było faktycznie. Ale *przy okazji* nie pominięto zainstalowania tam modułów wspomagających inne bardzo ważne zdolności mikrokomputera.

Zważywszy na to, że system operacyjny jest tylko programem, po każdym włączeniu komputera DOS musi być od nowa wczytywany z dysku (do RAM). *Ktoś* to musi zrobić, w związku z tym, że RAM jest pamięcią ulotną (zobacz IKS nr 12/88), w maszynie, po włączeniu sieci, nie ma żadnego programu. Stanowiło to o konieczności zainstalowania w komputerze takiej pamięci (ROM lub RAM zasilanej bateryjką), która nie *zgubi* (po wyłączeniu sieci) programu, *potrafiące-go* wczytać DOSa. Jest to BIOS - program automatycznie uruchamiany zawsze po włączeniu maszyny do sieci.

BIOS *zajmuje się* także obsługą operacji związanych z wprowadzaniem i wyprowadzaniem innych (poza DOSem) informacji, a także automatycznym testowaniem podstawowych układów komputera. Innymi słowy program ten (wspomagany przez programy MSDOS.SYS oraz IO.SYS) np.: automatycznie wpisuje *trudne* liczby (np.: 1010101) do pamięci RAM następnie odczytuje je by ostatecznie porównać to co zostało zapisane z rezultatem odczytania. Proces

owej kontroli jest sygnalizowany na ekranie monitora - ukazują się tam kolejne adresy (liczby) testowanych komórek pamięci. Wynik porównania różny od zera oznacza, że określona kostka pamięci pracuje błędnie i należy ją wymienić. Tak wygląda w skrócie testowanie pamięci, od którego rozpoczyna pracę każdy komputer PC. Testowanie stanowi przykład standardowych czynności maszyny. Stanowi to, dla nas użytkowników, ciągle odnawiany kredyt zaufania co do jakości usług oferowanych przez komputer. Czyli raz napisany test powtarzany jest codziennie w tysiącach mikrokomputerów określonego typu, np.: w IBM PC/XT.

Podobnie jest z procesem (a jest on dosyć złożony) obsługi klawiatury czy monitora. Wykorzystywany jest do tego celu specjalny program, który rozpozna naciskany przez nas klawisz, umieści jego kod w pamięci, pobierze odpowiedni wzorzec (mozaikę punktów) tworzący znak i ostatecznie wyśle tę mozaikę do monitora. Na szczęście wszystkie te czynności wykonywane są przez oprogramowanie systemowe, jesteśmy zwolnieni od troski o te podstawowe zdolności maszyny.

Oczywiście jest to drobna część uniwersalnych czynności komputera. DOS realizuje ich znacznie więcej. Na przykład dostałem (kupiłem, załatwiłem) dyskietkę z doskonałym opisem edytora testów pod nazwą IKSEDYT.CHI. Niestety dyskietkę muszę za kwadrans oddać. Nie ma problemu - wkładam dyskietkę do napędu oznaczonego literą A i na wszelki wypadek polecam DOSowi by sprawdził, czy aby na pewno jest tam zapisany zbiór (nasz opis edytora). Wystarczy napisać:

DIR

i na ekranie ukazują się nam wszystkie zapisane na dyskietce zbiory,np.:

```
IKSEDYT   CHI    30000 10/01/80   9:55
BAJBAJ    CHI    30000 10/01/80  10:11
KONKON    CHI   120000 10/12/88   9:03
      File(s)   180000 bytes free
```

Okazuje się, że mam ostatnią dyskietkę, na której może być trochę miejsca. Wkładam ją do napędu B i przekazuję DOSowi, aby teraz zaczął interesować się napędem B pisząc B: . Powtarzam znane już nam polecenie DIR. W odpowiedzi na ekranie ukazuje się:

```
TEST1   TXT  100000  10/12/88   11:12
TEST2   TXT  200000  10/10/87   12:11
TEST3   TXT   50000   8/08/88    4:38
NOWA    TXT  150000   7/11/88   10:23
    4 File(s)     310000 bytes free
```

Powyższy komunikat oznacza, że tam zapisane cztery zbiory (files), wolne jest aż 310 000 bajtów (bytes free). Wróćmy do naszego remanentu na dyskietce w napędzie A. Okazuje się, że interesujący nas zbiór IKSEDYT.CHI wymaga tylko 30 000 bajtów pamięci (druga kolumna). Zatem jest dostatecznie dużo pamięci na mojej dyskietce. Wobec tego kopiuję interesujący mnie zbiór z napędu A na B.

B:\) COPY A:IKSEDYT.CHI

w odpowiedzi maszyna pisze:

1 File(s) copied

Oznacza to, że kopiowanie zostało zakończone.Na wszelki wypadek sprawdzam, czy operacja została poprawnie wykonana.

DIR

Na ekranie monitora ukazuje się:

```
TEST1     TXT   100000  10/12/88  11:12
TEST2     TXT   200000  10/10/87  11:11
TEST3     TXT    50000   8/08/88   4:08
NOWA      TXT   150000   7/11/88  10:00
IKSEDYT   CHI    30000  10/01/80   9:55
    5 File(s)     280000 bytes free
```

W związku z tym, że była to jednak tylko demonstracja - szybko kasuję efekt dotychczasowej pracy i sprawdzam skuteczność mazania;

B:\) ERASE IKSEDYT.CHI

B:\) DIR

```
TEST1   TXT  100000  10/12/88  11:12
TEST2   TXT  200000  10/10/87  12:11
TEST3   TXT   50000   8/08/88   4:38
NOWA    TXT  150000   7/11/88  10:23
    4 File(s)     310000 bytes free
```

Kończąc prezentowane przykłady, przy okazji wyświetlania zawartości dyskietek, pragnę zwrócić uwagę na bardzo użyteczne informacje, które zawarte są w dwóch ostatnich kolumnach remanentu dyskietek, są to: data i czas założenia (zbudowania) danego zbioru.

Poza przedstawionymi przykładami DOS potrafi także między innymi zmieniać pamiętaną przez maszynę datę i czas zegarowy, kasować, drukować, wyświetlać i szukać wybrane informacje, układać po kolei czyli sortować np. alfabetycznie. Pełna lista zdolności DOSa z krótkim opisem, jak je wykorzystać, to dokumentacja w postaci pokaźnego tomu.

Jak już kilkakrotnie wspominałem System Operacyjny - w naszym przykładzie DOS - świadczy podstawowe, niezbędne każdemu użytkownikowi usługi. Wydaje się, że można go przyrównać do solidnego fundamentu. Konstrukcja wznoszonego na nim budynku może być różna - zależna od praktycznego przeznaczenia owego dzieła. Na przykład dla samochodziarzy będzie to garaż, dla handlowców - hala targowa, dla Dyrekcji PKS - dworzec autobusowy.

Wróćmy jednak do komputera. Nasz fundament to system operacyjny, budowla - narzędziowe i użytkowe programy. Istnieją ich tysiące. Oznacza to, że kupujemy komputer wraz z systemem operacyjnym, a dokupienie programu dostosowanego do naszych specyficznych potrzeb nie stanowi problemu. W wielu przypadkach (chyba w większości) użytkownicy sprzętu PC nie muszą tworzyć nowego, dostosowanego do ich potrzeb programu. Oznacza to, że niemal na drugi dzień po kupieniu komputera i odpowiedniego oprogramowania możemy wykorzystywać maszynę do wykonywania (wspomagania) określonych prac.

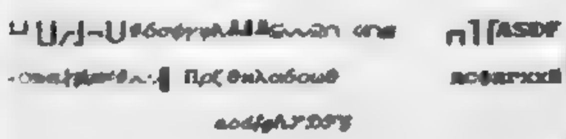
INFORMATYKA W SZKOLE

Bardzo użyteczne okazały się tak zwane programy narzędziowe. Jest ich bardzo wiele (NORTON, PC TOOLS, XTPEE i inne). Każdy z nich to kolejny krok w uproszczeniu obsługi komputera. Wykonują one bowiem wszystkie funkcje DOSa i jeszcze sporo innych. Wywołanie, uruchomienie owych funkcji zazwyczaj polega na przesuwaniu kursora po ekranie lub napisaniu jednej litery wyróżniającej określone polecenie. Na monitorze wyświetlane jest bowiem kilkadziesiąt, ładnie ułożonych w tabelkach, słów i zwrotów (niestety zazwyczaj w języku angielskim). Na przykład: DYSK, DRUKUJ, KASUJ, KOPIUJ. Wystarczy kursorem *dotknąć* słowo DYSK lub napisać literkę D. Gdy wybieram drukowanie piszę znak P, kasowanie - znak K i odpowiednio - O dla kopiowania) a na ekranie wyświetlą się nam literki - imiona napędów dyskowych: A, B, C. *Dotykamy* A i na ekranie wyświetlona zostaje lista zbiorów, które są zapisane na dyskietce (aktualnie włożonej do napędu A). Podobnie jak miało to miejsce przy wykorzystaniu instrukcji DIR w DOSie.

TEST1	TXT	100000	10/12/88	11:1.
TEST2	TXT	200000	10/10/87	11:11
TEST3	TXT	50000	8/08/88	4:0.
NOWA	TXT	150000	7/11/88	10:0.
IKSEDYT	CHI	30000	19/01/89	9:5.
5 File(s)		280000 bytes free		

Po co zatem ten program? Znakomicie usprawnia (przyspiesza) wykonywanie operacji *narzędziowych*, choćby tylko przez to, że wystarczy *dotknąć* dane polecenie (a nie napisać) i natychmiast jest ono realizowane. Wróćmy do naszego przykładu. Po wyświetleniu zawartości dyskietki, także tym razem *dotykam* wybrany zbiór i oznaczam go przyciskając klawisz F1 (klawisz funkcyjny). Na przykład . zbiór NOWA zostaje podświetlony (*napisany tłustym drukiem*). Teraz wybieram jedną z funkcji: KOPIUJ, KASUJ, DRUKUJ - w zależności od tego co zamierzam wykonać. Oczywiście lista dostępnych operacji jest znacznie większa, a opisany, przykładowy, program tylko w drobnej części ilustruje możliwości omawianych narzędzi. Ich umiejętne wykorzystanie znacznie ułatwia proces współpracy z maszyną - staje się ona (praca i maszyna) coraz sympatyczniejsza.

A cóż te inne użytkowe programy mogą wykonywać? Na pewno wszystko to co było realizowane na starszym sprzęcie komputerowym! Świadom jestem tego, że systemy finansowe, kadrowe, materiałowe (a jest ich wiele) są odległe od naszych zainteresowań. Stąd też pragnę zwrócić uwagę na kilka atrakcyjniejszych dla nas zastosowań. Przyjrzyjmy się na przykład edytorowi tekstów. Dzięki temu programowi, komputer zamienia się w uniwersalną maszynę do pisania. To co napiszemy na klawiaturze zostanie wydrukowane. Nie prościej i taniej (!) kupić sobie po prostu maszynę do pisania? Przede wszystkim edytor (typowy) dysponuje bogatym zestawem krojów czcionek (np.: 20 razy 2 - bo duże i małe litery) z nieograniczoną możliwością tworzenia nowych własnych znaków - może to być cyrylica, znaki matematyczne, symbole elktroniczne lub chińskie obrazki, np.:



Wybór dowolnego kroju liter (znaków) polega na naciśnięciu jednego lub dwóch klawiszy. Mimo tych zalet cena komputera - kilka milionów - budzi nadal duże wątpliwości co do celowości takiej inwestycji.

Gdy piszemy, maszyna dba o to by każdy wiersz miał zaplanowaną przez nas liczbę znaków. Polega to na tym, że piszemy ciurkiem - bez zastanawiania się gdzie skończyć wiersz - kontroluje to maszyna przenosząc słowo, które powoduje przepełnienie zaplanowanej przez nas liczby znaków w wierszu do następnej linii tekstu. Aby wiersz z którego zabrano wyraz był odpowiednio długi, komputer dodaje w nim, między poszczególne słowa, odpowiednią liczbę odstępów. Marginesy są idealnie równe. Przykładem tego jest niniejszy tekst, który był pisany właśnie przy pomocy omawianego edytora. Po ukończeniu całej strony, zauważam (oglądając ją na ekranie monitora), że *zjadłem* słowo i parę liter. Przesuwając kursor w odpowiednie miejsce *wciskam* tam brakujące słowa i znaki. Itp., itp.

Jeszcze o jednej własności niektórych tekstowych edytorów należy powiedzieć parę słów - wykluczają one błędy ortograficzne! Porównują każde pisane przez nas słowo z pamiętanym słownikiem ortograficznym - w efekcie komputer czasami buczy, *oj buczy!*

Opisałem tutaj zaledwie drobną część dostępnych możliwości edytora tekstów. Mimo ich niewątpliwej atrakcyjności (także użyteczności) - proponowane zastosowanie nie powinno stanowić koronnego argumentu podjęcia decyzji o zakupie komputera. Dodajmy do tego zatem kolejne argumenty - propozycje innych zastosowań maszyn. Uczynię to podczas naszego następnego spotkania (to już chyba wiosną?). Przedmiotem VII części **PODSTAW INFORMATYKI** będzie nieco inny edytor - tym razem obrazów (szkoda że nasz IKS nie jest kolorowy), ciekawe programy obliczeń statystycznych oraz podzielę się wrażeniami z przebiegu łączności komputerowej, nie tylko w skali kraju.

Władzimierz Gogołek

Program „MIERNIK"

W pracowniach laboratoryjnych naszych szkół jest szeroko wykorzystywany do pomiarów miernik uniwersalny typu UM3 (lub podobny). Szybkie i prawidłowe odczytywanie wskazań mierników tego typu wymaga wprawy poprzedzonej długotrwałym treningiem. Zastąpienie w czasie treningu przyrządu pomiarowego programem mikrokomputerowym pozwala uniknąć ryzyka jego uszkodzenia i uatrakcyjnia proces nauki.

Przedstawiany program „MIERNIK" służy spełnieniu przedstawionych wyżej celów. Został opracowany w języku Basic na mikrokomputer typu ZX Spectrum (lub podobny). Po uruchomieniu na ekranie rysowana jest podziałka miernika UM3 z literowym oznaczeniem amperomierza i woltomierza. Obrazy te są zapamiętywane do późniejszego wykorzystania. Jest to część wstępna, przygotowawcza programu. W części głównej losowany jest rodzaj wykorzystania miernika (woltomierz, amperomierz), zakres pomiarowy oraz wartość wielkości mierzonej (pamiętana przez komputer). W efekcie na ekranie ukazuje się skala przyrządu ze wskazówką w położeniu odpowiadającym wylosowanej wcześniej wartości wielkości mierzonej.

Rodzaj wielkości mierzonej (rodzaj pracy miernika) jest obrazowany przez wyświetlenie symbolu „A" lub „U" poniżej podziałki. W oknie systemowym pojawia się komunikat o wylosowanym zakresie pomiarowym i polecenie wprowadzenia odczytanej przez trenującego wartości. Obraz ten jest przedstawiony na poniższym rysunku. Wprowadzona wartość jest porównywana z zapamiętaną. Program dopuszcza popełnienie błędu odczytu nie większego niż 0,5 działki (1.5%). Wystąpienie większego błędu powoduje wyświetlenie komunikatu o nim i polecenie dokonania powtórnego odczytu. Po trzecim błędnym odczycie na ekranie ukazuje się informacja na temat prawidłowego postępowania podczas odczytywania wskazań mierników wielozakresowych. Po jej przeczytaniu ćwiczący kieruje program (przez naciśnięcie dowolnego klawisza) do początku, celem dalszego ćwiczenia.

Podanie przez trenującego wartości z dopuszczalnym błędem powoduje automatyczne powtórzenie ćwiczenia dla nowych wartości.

Program działa szybko dzięki zapamiętaniu obrazów z narysowanej podziałki i przenoszeniu ich za pomocą wstawki w kodzie maszynowym.

Podstawowe uwagi dotyczące korzystania z załączonego listingu programu

1. Linie 20,30 — program w języku maszynowym do przenoszenia obrazów. Adres oryginału w komórkach 50001,50002; adres kopii w komórkach 50004,50005
2. Linie 40,50 i 2000—2000 — rysowanie i zapamiętanie obrazów z podziałkami amperomierza i woltomierza.
3. Linie 60,70 — wczytanie do obszaru UDG polskich liter
4. Linie 1000—1060 — podprogram wyświetlający instrukcję o prawidłowym postępowaniu podczas odczytywania wskazań miernika.

Ryszard ZIEMKOWSKI

```
 10 BORDER 0: INK 0: PAPER 7: CLEAR 49999
 20 FOR i=50000 TO 50011: READ b: POKE i,b: NEXT i
 30 DATA 33,0,64,17,56,199,1,0,27,237,176,201
 40 PLOT 119,81: DRAW 0,13: DRAW 4,3: DRAW 3,0: DRAW 4,-5: DRAW 0,-13: DRAW -1,
0: DRAW 0,13: DRAW -3,4: DRAW -3,0: DRAW -3,-4: DRAW 0,-13: DRAW 0,7: DRAW 8,0:
DRAW 0,1: DRAW -8,0: GO SUB 2000: RANDOMIZE USR 50000
 50 PRINT AT 9,14;"   ";AT 10,14;"   ";AT 11,14;"   ": PLOT 124,82: DRAW 1,0: D
RAW 7,18: DRAW -1,0: DRAW -6,-17: DRAW -1,0: DRAW -6,17: DRAW -1,0: DRAW 7,-18:
POKE 50004,144: POKE 50005,226: RANDOMIZE USR 50000: POKE 50004,0: POKE 50005,64
 60 FOR i=0 TO 167: LET a=USR "a"+i: READ b: POKE a,b: NEXT i
 70 DATA 0,0,56,4,60,68,60,6,0,60,66,66,126,66,70,8,4,8,28,32,32,32,28,0,8,60,8
2,64,64,66,60,0,0,0,56,68,120,64,60,8,0,126,64,124,64,64,126,12,16,0,124,8,16,32
,124,0,24,126,4,8,16,32,126,0,8,16,124,8,16,32,124,0,8,126,20,8,16,32,126,0,0,64
,72,80,96,64,126,0,0,16,20,24,48,16,12,0,8,74,98,82,74,70,66,0,8,16,120,68,68,68
,68,0,8,16,56,68,68,68,56,0,8,60,82,66,66,66,60,0,0,60,66,66,66,36,102,0,36,74,7
4,74,60,8,8,48,8,16,56,64,36,4,120,0,8,60,64,60,2,66,60,0,0,0,34,34,34,34,93,120
 80
 100 DIM t(16): FOR i=1 TO 16: READ t(i): NEXT i
 110 DATA .0015,.006,.015,.06,.15,.6,1.5,6,1.5,6,15,30,60,150,300,600
 120 LET KON=49.5: LET ZAKRES=KON-POCZ
 130 LET I=0: LET WSKAZ=RND*ZAKRES: LET MIER=INT (RND*16): CLS
 140 IF MIER<8 THEN  POKE 50001,56: POKE 50002,199: RANDOMIZE USR 50000: GO TO 1
60
 150 POKE 50001,144: POKE 50002,226: RANDOMIZE USR 50000
 160 LET KAT=(POCZ+WSKAZ)/180*PI: LET DL=149: LET X0=125: LET Y0=0: PLOT X0,Y0:
DRAW -1,0: LET XC=DL*(SIN (KAT)): LET YC=DL*(COS (KAT)): DRAW XC+1,YC: DRAW -XC+
2,-YC
 170 LET t$="V": IF MIER<8 THEN  LET t$="A"
 180 LET t$=STR$ t(MIER+1)+t$: IF MIER<4 THEN  LET t$="0"+t$
 190 INPUT AT 0,7;"ZAKRES  -  ";(t$)'"PODAJ WARTOSC MIERZONA : ";A: LET ODCZ=A*Z
AKRES/t(MIER+1)
 200 LET BLAD=ABS (ODCZ-WSKAZ)
 210 IF BLAD<=1.5 THEN  GO TO 130
 220 LET I=I+1
 230 IF I>2 THEN  GO SUB 1000: GO TO 130
 240 PRINT #1;"ZŁY ODCZYT - SPROBUJ JESZCZE RAZ": PAUSE 100
 250 GO TO 170
 260 STOP
 270
1000 CLS
1010 PRINT "             PAMIETAJ !!! ": PRINT : PRINT
1020 PRINT " Podczas korzystania z miernika ": PRINT "wielozkresowego należy pos
ługi- wać się stałą podziałki 'C'.": PRINT : PRINT
1030 PRINT "          zakres pomiarowy": PRINT "   .  C=": PLOT 82,100: DRAW 138
,0: PRINT "           liczba działek": PRINT : PRINT : PRINT
1040 PRINT "Aby otrzymać wartość wielkości ": PRINT "mierzonej należy wskazywaną
 li-": PRINT "czbę działek pomnożyć przez stu-": PRINT "łą podziałki(dla aktualn
ie usta-": PRINT "lonego zakresu pomiarowego)."
1050 PRINT #1;" PO PRZECZYTANIU TEKSTU WCISNIJ      DOWOLNY KLAWISZ"
1060 PAUSE 0: RETURN
1070
2000 LET POCZ=-48.5
2010 FOR i=0 TO 60 STEP 10: LET xr=125+150*SIN ((pocz+1.625*i)*PI/180): LET yr=1
50*COS ((pocz+1.625*i)*PI/180): PLOT xr,yr
2020 LET xrn=125+160*SIN ((pocz+1.675*i)*PI/180): LET yrn=160*COS ((pocz+1.625*i
)*PI/180): DRAW xrn-xr,yrn-yr: NEXT i
2030 PRINT AT 8,0;"0": PRINT AT 4,4;"5": PRINT AT 1,9;"10": PRINT AT 0,15;"15":
PRINT AT 1,21;"20": PRINT AT 4,26;"25": PRINT AT 8,30;"30"
2040 FOR i=2 TO 60 STEP 2: LET xr=125+150*SIN ((pocz+1.625*i)*PI/180): LET yr=15
0*COS ((pocz+1.625*i)*PI/180): PLOT xr,yr
2050 LET xrn=125+158*SIN ((pocz+1.625*i)*PI/180): LET yrn=158*COS ((pocz+1.625*i
)*PI/180): DRAW xrn-xr,yrn-yr: NEXT i
2060 FOR i=1 TO 60 STEP 2: LET xr=125+150*SIN ((pocz+1.625*i)*PI/180): LET yr=15
0*COS ((pocz+1.625*i)*PI/180): PLOT xr,yr
2070 LET xrn=125+154*SIN ((pocz+1.625*i)*PI/180): LET yrn=154*COS ((pocz+1.625*i
)*PI/180): DRAW xrn-xr,yrn-yr: NEXT i
2080 RETURN
```



FUNKCJA DYSKRYMINUJĄCA

Program „Funkcja dyskryminująca" stanowi narzędzie pomocne w trakcie prowadzenia analizy dyskryminacyjnej dla dwóch P-wymiarowych populacji o licznościach prób N1 i N2. Program oblicza podstawowe wielkości potrzebne w trakcie analizy, a mianowicie: współczynniki liniowej funkcji dyskryminującej Fishera, L(P), odległość Mahalanobisa D2, statystykę F, wartości średnie funkcji dyskryminującej Y1SR i Y2SR, wartość funkcji dyskryminującej w kolejnych próbach dla obu populacji Y1 (N1), Y2 (N2) oraz sumaryczną macierz sum i iloczynów A (P, P) z (N1 + N2 — 2) stopniami swobody. Powyższe obliczenia można również przeprowadzić dla zmiennych standardowych (o wariancji równej 1). Możliwa jest wówczas ocena wpływu poszczególnych zmiennych (cech) na wartość funkcji dyskryminującej. Informacje takie wykorzystywane są na przykład w tzw. złożonej metodzie oceny stanów techniki bazującej na analizie dyskryminacyjnej (Devendra Sahal: Foundations of Technometrics, Technological Forecasting and Social Change, Vol. 27, Number 1, February 1985).

Program testowany był w oparciu o dane z badań R. A. Fishera przytoczonych w monografii: M. G. Kendall, A. Stuart: The Advancend Theory of Statistics, Vol. 3, Second Edition (wyd. w jęz. rosyjskim z 1976 r., str. 442)

Dane, X1 (P, N1), X2 (P, N2), wprowadza się instrukcjami DATA poczynając od linii 1300. Kolejność wprowadzania danych (dla P = 2): X1(1, 1), X2 (2, 1),....X1 (1, N1), X1 (2, N1), X2 (1, 1), x2 (2, 1),..., X2 (1, N2), X2 (2, N2). Wprowadzanie danych można przerwać (bez uszczerbku w procesie wprowadzania) w dowolnym momencie, naciskając jednocześnie klawisze **Control i 1**, wznowienie procesu wczytywania następuje po kolejnym naciśnięciu klawiszy.

W celu przyspieszenia działania programu ekran zostaje wyłączony (instrukcja POKE 559, 0) w trakcie obliczeń sum kwadratów i iloczynów (linie 360—430) i wartości zmiennych standaryżowanych (linie 50—180), a te ostatnie obliczenia (szczególnie czasochłonne) umieszczone są w początkowych liniach programu.

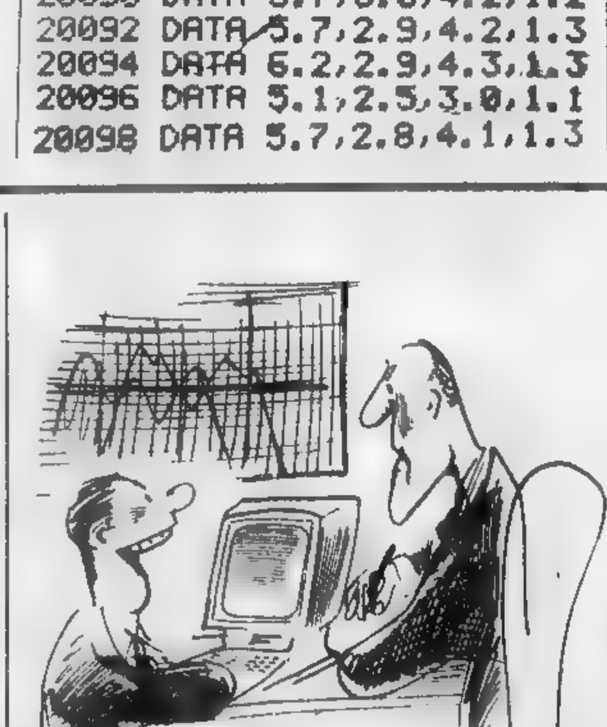
```
10 GOTO 1060
20 REM Podprogram:obliczanie zmiennych standaryzowanych
30 ? CHR$(125):? "OBLICZANIE F.DYSKRYMINUJACEJ DLA ZMIENNYCH STANDARYZOWANYCH":? :FOR G=1 TO 500:NEXT G
40 POKE 559,0
50 FOR G=1 TO P:ZMP=0
60 FOR F=1 TO N1:ZMP=ZMP+(X1(G,F)-X1SR(G))*(X1(G,F)-X1SR(G)):NEXT F
70 ZMP=ZMP/(N1-1):WAR1(G)=SQR(ZMP)
80 NEXT G
90 FOR G=1 TO P:FOR F=1 TO N1
100 ZMP=X1(G,F)/WAR1(G):X1(G,F)=ZMP
110 NEXT F:NEXT G
120 FOR G=1 TO P:ZMP=0
130 FOR F=1 TO N2:ZMP=ZMP+(X2(G,F)-X2SR(G))*(X2(G,F)-X2SR(G)):NEXT F
140 ZMP=ZMP/(N2-1):WAR2(G)=SQR(ZMP)
150 NEXT G
160 FOR G=1 TO P:FOR F=1 TO N2
170 ZMP=X2(G,F)/WAR2(G):X2(G,F)=ZMP
180 NEXT F:NEXT G
190 OFS=1:POKE 559,34:? CHR$(125)
200 REM Podprogram: obliczanie wartosci posrednich
210 REM Obliczanie wartosci X1SR(P),X2SR(P)
220 FOR G=1 TO P:ZMP=0
230 FOR F=1 TO N1:ZMP=ZMP+X1(G,F):NEXT F
240 ZMP=ZMP/N1:X1SR(G)=ZMP:? "X1SR(";G;")=";X1SR(G)
250 NEXT G
260 ? :?
270 FOR G=1 TO P:ZMP=0
280 FOR F=1 TO N2:ZMP=ZMP+X2(G,F):NEXT F
290 ZMP=ZMP/N2:X2SR(G)=ZMP:? "X2SR(";G;")=";X2SR(G)
300 NEXT G
310 ? :? "c.d. po nac. ";CHR$(210);CHR4(197);CHR$(212):INPUT Q$:? CHR$(125)
320 REM Obliczanie wektora D
330 FOR G=1 TO P:ZMP=0:ZMP=X1SR(G)-X2SR(G):D(G)=ZMP:? "D(";G;")=";D(G):NEXT G
340 ? :? "c.d. po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$:? CHR$(125):POKE 559,0
350 REM Obliczanie iloczynow mieszanych
360 FOR G=1 TO P:FOR J=G TO P:ZMP=0
370 FOR F=1 TO N1:ZMP=ZMP+(X1(G,F)-X1SR(G))*(X1(J,F)-X1SR(J)):NEXT F
380 SIX1(G,J)=ZMP
390 NEXT J:NEXT G
400 FOR G=1 TO P:FOR J=G TO P:ZMP=0
410 FOR F=1 TO N2:ZMP=ZMP+(X2(G,F)-X2SR(G))*(X2(J,F)-X2SR(J)):NEXT F
420 SIX2(G,J)=ZMP
430 NEXT J:NEXT G
440 REM Obliczanie iloczynow srednich
450 LI=0:POKE 559,34
460 FOR G=1 TO P:FOR F=G TO P:ZMP=0:LI=LI+1
470 ZMP=SIX1(G,F)+SIX2(G,F):ZMP=ZMP/(N1+N2-2)
480 A(G,F)=ZMP:? "A(";G;",";F;")=";A(G,F)
490 IF G<>F THEN A(F,G)=A(G,F)
500 IF LI=15 THEN LI=0:? "c.d. po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$
510 NEXT F:NEXT G
520 ? :? "c.d. po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$:? CHR$(125)
530 REM Podprogram: obliczanie wspolczynnikow funkcji dyskryminujacej (uklad rownan liniowych)
540 FOR G=1 TO P:ZMP=D(G):R(G)=ZMP:NEXT G
550 ZMP=1.0E-07
560 FOR K=1 TO P:ZMD=ZMP
570 FOR F=K TO P:T=A(F,K):IF ABS(T)>ABS(ZMD) THEN ZMD=T:G=F:NEXT F
580 IF ABS(ZMD)<=ZMP THEN ? "UKLAD ROWNAN NIE MA JEDNOZNACZNEGO ROZWIAZANIA":END
590 FOR F=K TO P:T=A(G,F):A(G,F)=A(K,F):A(K,F)=T:NEXT F:T=R(G):R(G)=R(K):R(K)=T
600 IF K=P THEN GOTO 630
610 FOR F=K+1 TO P:T=A(F,K)/ZMD:FOR G=K+1 TO P:A(F,G)=A(F,G)-T*A(K,G):NEXT G:R(F)=R(F)-T*R(K):NEXT F
620 NEXT K
630 Z(P)=R(P)/A(P,P)
640 FOR K=P-1 TO 1 STEP -1:Z(K)=R(K)/A(K,K):FOR G=K+1 TO P:Z(K)=Z(K)-A(K,G)/A(K,K)*Z(G):NEXT G:NEXT K
650 REM Wydruk
660 ? CHR$(125)
670 POSITION 2,2:? "WSPOLCZYNNIKI FUNKCJI DYSKRYMINUJACEJ":? :?
680 FOR F=1 TO P:? "  L";F;"=";Z(F):NEXT F
690 ? :FOR LI=1 TO 37:? CHR$(160);:NEXT LI
700 ? :? "c.d. po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$:? CHR$(125)
710 REM Podprogram: obliczanie wartosci funkcji dyskryminujacej
720 REM Obliczanie D2
730 ZMP=0
740 FOR G=1 TO P:ZMP=ZMP+Z(G)*D(G):NEXT G
750 D2=ZMP:? :? :? "D2=";D2
760 REM Obliczanie statystyki
770 ZMP=0:ZMP=(N1*N2*(N1+N2-P-1)*D2)/(P*(N1+N2)*(N1+N2-2))
780 ? :? "STATYSTYKA=";ZMP
790 REM Obliczanie wartosci srednich Y
800 ZMP=0
810 FOR G=1 TO P:ZMP=ZMP+Z(G)*X1SR(G):NEXT G
820 ? :? "Y1SR=";ZMP
830 ZMP=0
840 FOR G=1 TO P:ZMP=ZMP+Z(G)*X2SR(G):NEXT G
850 ? "Y2SR=";ZMP
860 ? :? "c.d.po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$:? CHR$(125)
870 REM Obliczanie wartosci funkcji dyskryminujacej
880 ? "OBLICZANIE WARTOSCI FUNKCJI DYSKRYMINU            JACEJ":? :FOR ZMP=1 TO 100:NEXT ZMP
890 LI=0
900 FOR F=1 TO N1:ZMP=0
910 FOR G=1 TO P:ZMP=ZMP+Z(G)*X1(G,F):NEXT G
920 ? "Y1(";F;")=";ZMP
930 LI=LI+1:IF LI=15 THEN LI=0:? "c.d. po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$
940 NEXT F
950 ? "c.d. po nac. ";CHR$(210);CHR$(197);CHR$(212):INPUT Q$:? C
```

```
HR$(125)
960 LI=0
970 FOR F=1 TO N2:ZMP=0
980 FOR G=1 TO P:ZMP=ZMP+Z(G)*X2
(G,F):NEXT G
990 ? "Y2(";F;")=";ZMP
1000 LI=LI+1:IF LI=15 THEN LI=0
?: "c.d. po nac. ";CHR$(210);CHR$
(197);CHR$(212):INPUT Q$
1010 NEXT F
1020 ? "c.d. po nac. ";CHR$(210)
;CHR$(197);CHR$(212):INPUT Q$
1030 IF OFS=1 THEN END
1040 GOTO 30
1050 REM Strona tytulowa
1060 ? CHR$(125)
1070 GRAPHICS 2:SETCOLOR 2,0,0:P
OSITION 7,2:? #6;"FUNKCJA":POSIT
ION 4,5:? #6;"DYSKRYMINUJACA"
1080 FOR ZMP=1 TO 200:NEXT ZMP
1090 ? "PODAJ LICZBE ZMIENNYCH P
=";:INPUT P:?
1100 ? "PODAJ LICZBE PROB N1=";:
INPUT N1:?
1110 ? "PODAJ LICZBE PROB N2=";:
INPUT N2
1120 DIM X1(P,N1),X2(P,N2),X1SR(
P),X2SR(P),Q$(3)
1130 DIM A(P,P),SIX1(P,P),SIX2(P
,P),D(P),Z(P),R(P),WAR1(P),WAR2(
P)
1140 GRAPHICS 0:TRAP 1200
1150 REM Wprowadzanie danych
1160 FOR F=1 TO N1:FOR G=1 TO P:
READ L:X1(G,F)=L:? "X1(";G;",";F
;")=";X1(G,F):NEXT G:NEXT F
1170 FOR F=1 TO N2:FOR G=1 TO P:
READ L:X2(G,F)=L:? "X2(";G;",";F
;")=";X2(G,F):NEXT G:NEXT F
1180 ? :? "Koniec wczytywania da
nych":? "c.d. po nac. ";CHR$(210
);CHR$(197);CHR$(212):INPUT Q$:?
 CHR$(125)
1190 OFS=0:GOTO 220
1200 ? "Dane wprowadza sie instr
ukcjami DATA, poczynajac od lini
 1300.";
1210 ? "Kolejnosc wprowadzania d
anych (dla P=2):":? :? "X1(1,1),
X1(2,1),X1(1,2),X1(2,2).......";
1220 ? "X1(1,N1),X1(2,N1); potem
 analogicznie X2(P,N2),gdzie: "
1230 ? :? "X1,X2 - pierwsza i dr
uga grupa danych":? "P - liczba
zmiennych":? "N1,N2 - licznosc p
rob"
1240 ? :? :? "Wprowadz dane!"
1250 STOP
```

```
20000 DATA 7.0,3.2,4.7,1.4
20002 DATA 6.4,3.2,4.5,1.5
20004 DATA 6.9,3.1,4.9,1.5
20006 DATA 5.5,2.3,4.0,1.3
20008 DATA 6.5,2.8,4.6,1.5
20010 DATA 5.7,2.8,4.5,1.3
20012 DATA 6.3,3.3,4.7,1.6
20014 DATA 4.9,2.4,3.3,1.0
20016 DATA 6.6,2.9,4.6,1.3
20018 DATA 5.2,2.7,3.9,1.4
20020 DATA 5.0,2.0,3.5,1.0
20022 DATA 5.9,3.0,4.2,1.5
20024 DATA 6.0,2.2,4.0,1.0
20026 DATA 6.1,2.9,4.7,1.4
20028 DATA 5.6,2.9,3.6,1.3
20030 DATA 6.7,3.1,4.4,1.4
20032 DATA 5.6,3.0,4.5,1.5
20034 DATA 5.8,2.7,4.1,1.0
20036 DATA 6.2,2.2,4.5,1.5
20038 DATA 5.6,2.5,3.9,1.1
20040 DATA 5.9,3.2,4.8,1.8
20042 DATA 6.1,2.8,4.0,1.3
20044 DATA 6.3,2.5,4.9,1.5
20046 DATA 6.1,2.8,4.7,1.2
20048 DATA 6.4,2.9,4.3,1.3
20050 DATA 6.6,3.0,4.4,1.4
20052 DATA 6.8,2.8,4.8,1.4
20054 DATA 6.7,3.0,5.0,1.7
20056 DATA 6.0,2.9,4.5,1.5
20058 DATA 5.7,2.6,3.5,1.0
20060 DATA 5.5,2.4,3.8,1.1
20062 DATA 5.5,2.4,3.7,1.0
20064 DATA 5.8,2.7,3.9,1.2
20066 DATA 6.0,2.7,5.1,1.6
20068 DATA 5.4,3.0,4.5,1.5
20070 DATA 6.0,3.4,4.5,1.6
20072 DATA 6.7,3.1,4.7,1.5
20074 DATA 6.3,2.3,4.4,1.3
20076 DATA 5.6,3.0,4.1,1.3
20078 DATA 5.5,2.5,4.0,1.3
20080 DATA 5.5,2.6,4.4,1.2
20082 DATA 6.1,3.0,4.6,1.4
20084 DATA 5.8,2.6,4.0,1.2
20086 DATA 5.0,2.3,3.3,1.0
20088 DATA 5.6,2.7,4.2,1.3
20090 DATA 5.7,3.0,4.2,1.2
20092 DATA 5.7,2.9,4.2,1.3
20094 DATA 6.2,2.9,4.3,1.3
20096 DATA 5.1,2.5,3.0,1.1
20098 DATA 5.7,2.8,4.1,1.3
```

```
20100 DATA 5.1,3.5,1.4,0.2
20102 DATA 4.9,3.0,1.4,0.2
20104 DATA 4.7,3.2,1.3,0.2
20106 DATA 4.6,3.1,1.5,0.2
20108 DATA 5.0,3.6,1.4,0.2
20110 DATA 5.4,3.9,1.7,0.4
20112 DATA 4.6,3.4,1.4,0.3
20114 DATA 5.0,3.4,1.5,0.2
20116 DATA 4.4,2.9,1.4,0.2
20118 DATA 4.9,3.1,1.5,0.1
20120 DATA 5.4,3.7,1.5,0.2
20122 DATA 4.8,3.4,1.6,0.2
20124 DATA 4.8,3.0,1.4,0.1
20126 DATA 4.3,3.0,1.1,0.1
20128 DATA 5.8,4.0,1.2,0.2
20130 DATA 5.7,4.4,1.5,0.4
20132 DATA 5.4,3.9,1.3,0.4
20134 DATA 5.1,3.5,1.4,0.3
20136 DATA 5.7,3.8,1.7,0.3
20138 DATA 5.1,3.8,1.5,0.3
20140 DATA 5.4,3.4,1.7,0.2
20142 DATA 5.1,3.7,1.5,0.4
20144 DATA 4.6,3.6,1.0,0.2
20146 DATA 5.1,3.3,1.7,0.5
20148 DATA 4.8,3.4,1.9,0.2
20150 DATA 5.0,3.0,1.6,0.2
20152 DATA 5.0,3.4,1.6,0.4
20154 DATA 5.2,3.5,1.5,0.2
20156 DATA 5.2,3.4,1.4,0.2
20158 DATA 4.7,3.2,1.6,0.2
20160 DATA 4.8,3.1,1.6,0.2
20162 DATA 5.4,3.4,1.5,0.4
20164 DATA 5.2,4.1,1.5,0.1
20166 DATA 5.5,4.2,1.4,0.2
20168 DATA 4.9,3.1,1.5,0.2
20170 DATA 5.0,3.2,1.2,0.2
20172 DATA 5.5,3.5,1.3,0.2
20174 DATA 4.9,3.6,1.4,0.1
20176 DATA 4.4,3.0,1.3,0.2
20178 DATA 5.1,3.4,1.5,0.2
20180 DATA 5.0,3.5,1.3,0.3
20182 DATA 4.5,2.3,1.3,0.3
20184 DATA 4.4,3.2,1.3,0.2
20186 DATA 5.0,3.5,1.6,0.6
20188 DATA 5.1,3.8,1.9,0.4
20190 DATA 4.8,3.0,1.4,0.3
20192 DATA 5.1,3.8,1.6,0.2
20194 DATA 4.6,3.2,1.4,0.2
20196 DATA 5.3,3.7,1.5,0.2
20197 DATA 5.0,3.3,1.4,0.2
```







Niespodziewana wizyta

Czytał właśnie gazetę. Gorąca aromatyczna kawa parzyła przyjemnie usta. Najbardziej lubił te chwile, przy porannej prasie. Nieśpieszne, dające jeszcze możliwość błogiego odpoczynku po krótkich nocach spędzonych na szukaniu tematu, rozleniwiające, sycące energią, której tak wiele trzeba do całodziennej pracy. Dziennikarstwo, to paskudnie wyczerpujący zawód.

Na pierwszej stronie, dużymi literami, krzyczał tytuł „Tajemnicze zaginięcie i co dalej? Czy policja wciąż będzie bezradna". Znowu tajemnicza afera. Wiedział, jak nikt inny, że często używa się takich chwytów celowo — dla zdobycia popularności, większej liczby czytelników, zwiększenia nakładu.... Szczególnie, kiedy nadchodził „sezon ogórkowy". A właśnie wakacje, lato w pełni, urlopy...

Dlatego bez większego zainteresowania i z pobłażliwym uśmiechem przebiegł oczyma tekst. Trójkąt bermudzki? I to w pobliżu Phoenix w Arizonie. „Bzdury" — pomyślał. Jeden z jego kolegów po fachu donosił, że wczoraj na autostradzie z Phoenix do Preskott znaleziono kolejny porzucony autobus rejsowy bez pasażerów. Intrygujący był fakt, że po trzydziestoośmioosobowej grupie podróżnych i dwuosobowej obsłudze nie pozostało ani śladu.

Nie znaleziono żadnych bagaży, ani śladów użycia przemocy, jakiejś walki czy paniki. Gdzie podziali się pasażerowie? Poszukiwania na szeroką skalę nie dały żadnych rezultatów. Na te i wiele innych pytań nie znaleziono, jak dotąd, odpowiedzi. Pikanterii dodaje fakt, iż to jest czwarty w tym miesiącu tego rodzaju wypadek na tej właśnie trasie.

Mark odrzucił gazetę na biurko i sięgnął po filiżankę, kiedy zadzwonił telefon. Podniósł słuchawkę. — *Redaktor Mark Henderson?* — Głos mówiącego zdradzał niepokój. — *Nareszcie... Chcieliśmy się z panem skontaktować już wcześniej, ale...*

— *Z kim mam przyjemność?*

— *O, przepraszam, zapomniałem się przedstawić. Doktor Martschewski z kliniki psychiatrycznej w Phoenikshill. Mamy do pana ogromną prośbę... Sprawa jest trochę nietypowa... Oto jeden z naszych pacjentów podaje się za pańskiego stryja...*

— *Co takiego? Mój stryj waszym pacjentem..?*

— *Nie niepokoilibyśmy pana, gdyby nie uporczywe żądania ze strony chorego... Domaga się konfrontacji i potwierdzenia swojej tożsamości przez pana. Proszę nas zrozumieć, przywieziono go bez dokumentów...*

— *Mam tylko jednego stryja... Doktora Percivalla Hendersona...*

— *Takie właśnie wymienił nazwisko...*

— *Ale cóż na miłość boską mógłby robić brat mego ojca w waszej klinice? Z tego co wiem, czuje się wspaniale, jest cenionym chirurgiem... Czyżby ubiegał się o stanowisko konsultanta?* — Sytuacja, że doktor Henderson może być w klinice psychiatrycznej, jako pacjent była tak nieprawdopodobna, że próbował żartować.

— *Sprawa jest bardziej skomplikowana, niż się panu wydaje... Czy mógłby pan przyjechać do nas? Jak najszybciej...*

Odłożył słuchawkę. Przez długą chwilę nie mógł zebrać myśli. Te dziwne niedomówienia doktora Martschewskiego... Klinika psychiatryczna, stryj Percivall, najinteligentniejsza osoba w rodzinie... Co to wszystko ma znaczyć? Wykręcił numer telefonu stryja. Nikt nie podnosił słuchawki. Tyle razy prosił, żeby nie mieszkał sam, w tym niewielkim domku na odludziu...

* * *

Asfaltową aleją podjechał niemal przed samo wejście kliniki. Po kilku zwyczajowych formalnościach przy tego rodzaju sytuacjach pokazano mu pacjenta. Nie mogło być żadnych wątpliwości. To był stryj Percivall. Wkrótce wpuszczono go do niewielkiego pokoju, w którym stało jedynie łóżko trochę dziwnej konstrukcji. Stryj, z rozwichrzonymi włosami, które jakby posiwiały od czasu, kiedy widział go po raz ostatni, przywitał go niecierpliwie, z kwaśnym uśmiechem.

— *A więc jesteś... Nareszcie jesteś... No siadaj, siadaj... Chłopcze, żebyś ty wiedział, co oni tu ze mną wyprawiają...* — To chłopcze brzmiało trochę groteskowo. Mówił przecież do prawie czterdziestoletniego mężczyzny. Było to jego ulubione słowo w rozmowie z Markiem.

— *Ale o co u licha chodzi? Co się stało? Skąd stryj tutaj?*

— *Spokojnie, spokojnie chłopcze. Zaraz ci wszystko opowiem. Tylko... Czekaj, może chcesz się upewnić, czy jestem normalny? No zadawaj pytania...* — Percivall zachowywał się zupełnie normalnie. Nic nie wskazywało, że nagle oszalał. Doktor uprzedził go jednak, że chory, na pierwszy rzut oka wygląda na normalnego i że pragnie w ten sposób uśpić czujność lekarzy i rodziny.

Mark przysiadł na krawędzi łóżka. Mimo wszystko był nieco sceptycznie nastawiony do dziwnych nastrojów stryja. Czasem nawet miewał wątpliwości... No cóż? Starość nie radość.

— *Wyobraź sobie* — doktor był wyraźnie podniecony — *że... A niech to... Nawet tobie boję się teraz to opowiedzieć... Przecież to właśnie przez to, ja... No wiesz, tutaj przez to trafiłem. To po prostu niesamowita historia... Słuchaj, nie ma chwili do stracenia... Tylko nie myśl, że zwariowałem...*

— *Ależ stryju...*

— *No dobrze już, dobrze... Tylko nie patrz tak na mnie i nie przerywaj, dopóki nie skończę.*

* * *

Za oknem zaszeleściły gałęzie leszczyny rosnącej niedaleko przy nim. Percivall dorzucił kolejną szczapę do kominka. Chciał odpocząć po nużącym dniu w klinice, przy herbacie, w fotelu. Nie miał już tyle sił, co w młodości. Zdrzemnął się przez chwilę. Widocznie zmarzł trochę, bo przebudził się nagle. Teraz, kiedy płomienie obejmowały suche bierwiono, przyjemne ciepło rozkwitło rumieńcem na jego policzkach.

Usłyszał delikatne stukanie w szybę. Nasłuchiwał przez moment pilnie. Czyżby się przesłyszał? Ale ten odgłos łamanych gałęzi... Nie, jest jak najbardziej realny. Podszedł do okna. Nie mylił się.

— *Niech mnie pan wpuści. Potrzebuję pomocy.* — Przez szybę, w gęstniejącym mroku, niewiele mógł dostrzec. Odniósł jednak wrażenie, że ten człowiek rzeczywiście potrzebuje natychmiastowej pomocy. Zdawało mu się, że w jego oczach, w wyrazie twarzy, wyczytał jakby błagalną prośbę.

— *Niech pan wejdzie, proszę, co się stało?*

W otwartych drzwiach stał mężczyzna średniego wzrostu, mocnej budowy ciała. Z całej jego postaci biła jakaś niezrozumiała groza. Opierał się na sękatej gałęzi, oderwanej niedawno z jakiegoś drzewa. Kikut jego prawej nogi owinięty był w poplamioną twedową marynarkę. Spod pachy drugiej ręki zwisała odcięta, czy

też odrąbana powyżej kolana noga. Mężczyzna wysunął ku niemu rękę z resztką nogi.

— *Niech pan to potrzyma... No, proszę się nie bać. Jest pan przecież chirurgiem, prawda?*

— *Tak, tak... Proszę wejść. Prędzej...* — Profesor usiłował usadzić nieznajomego w fotelu.

— *Już dzwonię do kliniki... Proszę się nie martwić. Uratujemy pana nogę.* — Mówił tak dla dodania otuchy mężczyźnie tak straszliwie doświadczonemu przez los.

— *Niech pan nie dotyka telefonu.* — Głos nieznajomego brzmiał ostro i twardo. Niezwyczajnie, jak na kogoś, kto uległ tragicznemu wypadkowi.

Percivall ani myślał tego słuchać. Był zbyt doświadczonym lekarzem, aby nie wiedzieć, jakie to dziwne rzeczy wygadują ludzie w szoku powypadkowym.

— *Proszę odłożyć słuchawkę. Nigdzie pan nie będzie dzwonił* — usłyszał rozkazujący ton.

„Rzeczywiście jest w szoku" — pomyślał wystukując numer na klawiaturze telefonu. Nagle sękaty kij wytrącił mu słuchawkę z ręki. Trzasnęła głucho o posadzkę. Kostur dusił teraz jego tchawicę.

— *Przyszedłem tutaj, żeby pan mi pomógł, a nie działał na moją szkodę* — mężczyzna akcentował dobitnie każde słowo. — *Szybko — narzędzia... Niech pan przygotuje wszystko do zabiegu.*

* * *

— *Czy ty sobie potrafisz to wyobrazić mój chłopcze?* — Siwa głowa doktora trzęsła się w niespokojnym zdumieniu, jakby raz jeszcze przyszło mu to przeżywać na nowo. — *Sam bym w to wszystko nie uwierzył, gdyby nie... Fakty są najbardziej okrutne. Ja do dziś nie mogę zrozumieć, jak to było możliwe...* — Odchrząknął, bo poczuł nagle suchość w gardle, która nie pozwalała mu mówić. Z oczu popłynęły łzy.

— *Wyjąłem oczywiście wszystkie narzędzia, jakie miałem w domu. On usiadł na ceratowej kozetce, wiesz tej co stoi w moim ambulatorium i... Drżącymi rękami pomagałem mu wiązać końcówki nerwów, ścięgien, wiązań... Łączyliśmy żyły, naczynia krwionośne, składaliśmy kości.*

Właściwie to on to wszystko robił. Ja mu tylko pomagałem. Rozumiesz... Człowiek bez nogi dokonuje sam operacji... I to jak precyzyjnie... Te jego zręczne palce... A przy tym twarz jego... Dziwnie nieludzka... Ja wiem, że ból zmienia oblicze każdego. Widziałem cierpienie i rozpacz na ludzkich twarzach... On nawet nie zapytał, czy mam coś na miejscowe znieczulenie.

Uwierz mi, taką twarz widziałem po raz pierwszy. Jakaś popielato-biała... Przez cały czas operacji nie drgnął na niej żaden mięsień, ani razu nie wykrzywiła się w grymasie bólu. Wyglądało to wszystko tak, jakby przyszywano nogę manekinowi. To znaczy on przyszywał, bo ja mu tylko asystowałem.

— *Stryju, ale przecież...*

— *A potem stała się rzecz najstraszniejsza.* — Percivall opadł na poduszki. Był tak wzburzony, że wydawało się, iż nie wykrztusi już z siebie ani słowa więcej. „Biedny stryjcio" — pomyślał Mark. Przez tyle lat był dobrym chirurgiem i nagle... Coś mu się w głowie widać jednak poprzestawiało...

— *To oni... Nie odchodź jeszcze...* — wykrzyknął widząc, że Mark podnosi się z łóżka. — *Nie opowiedziałem ci najważniejszego.*

* * *

Nieznajomy mężczyzna wstał z kanapy. Chwiejnym nieco krokiem przeszedł przez pokój.

— *Trzeba jeszcze tylko ukrwić mięśnie i po wszystkim... Dobra robota doktorze.*

— *To przecież niemożliwe!* — Percivall wrzucił narzędzia do sterylizatora. — *To wszystko nie może być realne... Chyba, że pan nie jest...*

— *No proszę, niech pan dokończy doktorze. Chyba, że co...? Że nie jestem człowiekiem, prawda? Tak pan chciał powiedzieć?* — Nieznajomy przywołał na usta coś w rodzaju uśmiechu. — *Ponieważ pomógł mi pan... No dobrze powiem panu prawdę, ale nie wiem, czy to panu wyjdzie na dobre. Czasem lepiej za dużo nie wiedzieć.*

No cóż, skoro pan chce... Na pewnym etapie rozwoju cywilizacji, rozum może nauczyć się kierowania wszystkimi czynnościami biologicznymi każdego organizmu. Przyszycie nogi, ręki, głowy nawet, to wtedy nic nadzwyczajnego.

Doktor patrzył ze zdziwieniem i niedowierzaniem na siedzącego obok człowieka. Człowieka? Czyżby był to... Nie, to jakiś koszmarny sen, z którego zaraz się zbudzi...

— *Nie, nie panie doktorze. Ma pan rację, to nie jest sen.* — Percivall zawisł wzrokiem na nieznajomym. On przecież czytał w jego myślach. — *Wy ludzie nie potraficie sobie wyobrazić ogromu wszechświata, nie chcecie uwierzyć, że mogą istnieć w nim inne, zupełnie odmienne od waszej cywilizacje.*

Czy byłby pan w stanie uwierzyć, że istnieje cywilizacja, która wybrała drogę rozwoju tylko rozumu. I to zupełnie w niewyobrażalnej dla człowieka postaci. Nic, tylko sam rozum.

— *To wszystko... To jest trudne do zrozumienia...* — Doktor odzyskiwał powoli zimną krew. — *Ale dlaczego pan... To znaczy czego wy od nas chcecie? Skąd wzięliście się na Ziemi i po co?*

— *Właśnie chciałem o tym powiedzieć.* — Mężczyzna wstał i zaczął wolno przechadzać się po pokoju. — *Już raz, przed tysiącami lat musieliśmy ingerować w wasz rozwój cywilizacyjny. Wasz potencjał umysłowy rozwijał się zbyt wolno... Teraz czujemy się, jak to wy mówicie, moralnie zobowiązani do opieki nad wami.*

Traktujemy Ziemię, jako takie większe trochę laboratorium. Ziemia jest nam potrzebna i jej mieszkańcy także. Naturalny cykl rozwoju każdej cywilizacji zawsze zmierza ku samozagładzie. Tak było również z naszą planetą. Kiedy doszliśmy do pewnego etapu rozwoju techniki... Wy zbliżacie się właśnie do tego apogeum. Jest już was ponad 5 miliardów. Przy takim tempie prokreacji, jaki macie w Afryce, czy Azji, już wkrótce na każdego mieszkańca Ziemi przypadałoby najwyżej pół metra miejsca. Wy potraficie tylko w jeden sposób rozwiązać ten problem — poprzez wojnę.

Nie macie w ogóle wyobraźni. Czy wiecie, jakie perturbacje wyniknęłyby w Układzie Słonecznym, gdybyście rozsadzili planetę? Przed tysiącami lat nasz rozwój szedł tym samym torem, na który skierowaliśmy was. Niestety, koniec był tragiczny. Została z nas tylko myśl. Czysty rozum. To tylko jest w stanie przetrwać we wszechświecie.

— *Dlaczego w takim razie nie dawaliście znaku życia przez tak długi czas?*

— *Dlaczego? Wy nawet nie jesteście w stanie nas pojąć. Ta ludzka powłoka jest po to właśnie, abyśmy mogli być wśród was. Byśmy mogli wykonać zadanie. Chcemy bowiem, dla waszego dobra, przetransportować część ludzi na inną planetę... Ciekawi pana na jaką, prawda? Na Marsa panie doktorze, na Marsa.*

Jesteśmy tam już od dawna. Nawet narobiliście nam trochę kłopotu w swoim czasie wypatrując przez teleskopy sieć kanałów niezbędną do ładowania naszej energii. Trzeba było szybko je zamaskować, abyście myśleli, że to niewłaściwość przyrządów dała ten efekt.

No i jak pan wie, udało się. Wymyśliliście nawet „efekt oczekiwania", aby wytłumaczyć sobie rozbieżność w obserwacjach. Były kanały i ich nie ma. Pomyłka. Uczeni po prostu tak bardzo chcieli zobaczyć te kanały, że

w końcu je zobaczyli — tłumaczyliście.

— No dobrze, ale do czego potrzebni są wam ludzie tam, na Marsie?

— Są potrzebni... Bardzo potrzebni.

* * *

— Marku, musisz mi pomóc stąd się wydostać — Percivall chwycił go kurczowo za rękę — *No i ratować tych ludzi. Ja byłem już wszędzie. Na policji, u burmistrza... Nikt mi nie uwierzył. Ale ty możesz o tym napisać w prasie. Musisz o tym napisać. Trzeba ostrzec ludzi.* — Drżał teraz na całym ciele, wyczerpany rozmową i strachem.

— Biedny stary — Mark wytarł mu ręcznikiem zroszone czoło. Czy to nie jest przypadkiem rodzinne? — pomyślał nagle. Najpierw babka, teraz...

Drzwi uchyliły się i Martschewski ruchem głowy wywołał go z pokoju.

— Jest w ciężkim stanie — rzekł usprawiedliwiająco. — *Powinien teraz odpocząć. Być może kiedyś jeszcze powróci do zdrowia... Narobił nam wiele kłopotów. Biegał po mieście, zaczepiał ludzi... Na niektórych próbował nawet rzucić się z nożem.*

— Biedny stryj — zgodził się z lekarzem. — *Jeżeli pan pozwoli, będę go czasami odwiedzał.* — *To taka przykra starość.*

— Tak, nikt z nas nie zna swojego końca — odrzekł lekarz. Uścisnęli sobie ręce i już po chwili Mark pędził autostradą w stronę miasta.

Patrol policyjny zatrzymał go dwadzieścia kilometrów od Phoenix.

— O co chodzi? Chyba nie popełniłem wykroczenia? — Mocne dłonie chwyciły go pod pachy.

— Zaraz się pan wszystkiego dowie.

W asyście trzymających go mocno policjantów przeszedł przez las, gdzie na polanie... Chciał się wyrwać, ale było już na wszystko za późno. Jak zahipnotyzowany patrzył na okolony niebieskawym rozlanym światłem pojazd w kształcie spłaszczonego jaja.

— Więc stryj nie był wariatem, więc to wszystko prawda — wyszeptał drewnianymi wargami, zanim ciśnięto go w otwarty luk.

ANNA KALETA

Przyszłość komputerów

Tygodnik amerykański „PC Week" zamieścił w swoich listopadowych numerach cykl artykułów Jamesa Martina, amerykańskiego producenta sprzętu komputerowego i autora wielu prac z zakresu techniki komputerowej, poświęconych perspektywom rozwoju sprzętu komputerowego w przeciągu najbliższych dwudziestu lat.

Na początku lat dziewięćdziesiątych rozpocznie się międzynarodowy wyścig o dominację w produkcji układów scalonych o submikronowych strukturach. Poprzez zastosowanie ultrafioletowych technik trawienia średnica pojedynczej struktury w układzie scalonym osiągnie wymiar między 0,6 a 0,8 mikrona. Pozwoli to na umieszczenie 100 milionów pojedynczych elementów w układach pamięciowych i 1 miliona w mikroprocesorach.

Dzięki nowym mikroprocesorom średniej klasy komputery będą działać z szybkością 15 milionów operacji na sekundę, wysokiej klasy komputery osobiste z szybkością 100 mln operacji na sekundę, a profesjonalne z szybkością 500 mln operacji na sek., a superkomputery 40 miliardów operacji na sek.

Pojawią się układy scalone, zawierające po kilka procesorów, umożliwiających równoległe przetwarzanie informacji. Powstanie tym samym nowa generacja komputerów tzw. komputerów równoległych.

Tak jak podstawowym osiągnięciem przemysłu komputerowego w latach osiemdziesiątych było pojawienie się komputerów osobistych, tak w latach dziewięćdziesiątych ma nim być masowy rozwój sieci komputerowych umożliwiających powszechne korzystanie z banków informacji i komputerów specjalistycznych.

Sieci te będą budowane ze światłowodów, w których szybkość transmisji będzie wynosić 2,2 miliarda bitów na sekundę.

Coraz powszechniej będą stosowane techniki sztucznej inteligencji umożliwiające nieprofesjonalistom łatwy dostęp do pożądanej informacji. Tworzone będą tzw. systemy ekspertów czyli wysoko specjalistyczne oprogramowania dotyczące poszczególnych dziedzin nauki i techniki, dzięki którym możliwe będzie powszechne korzystanie z najnowszych osiągnięć w tych dziedzinach.

Z końcem lat dziewięćdziesiątych przewiduje się wprowadzenie nowych technik wytwarzania układów scalonych, wykorzystujących promienie rentgenowskie, umożliwiających umieszczenie w jednym układzie ponad 1 miliarda elementów. Pozwoli to na dalsze zwiększenie szybkości pracy komputerów i doskonalenie technik procesów równoległych. Zastosowanie dysków optycznych umożliwi znacznie zwiększenie pojemności pamięci komputerów, dla dysków 4 i 3/4 cala do 8 miliardów bitów, dla 3-calowych do 1 miliarda bitów, a dla 2-calowych do 200 milionów bitów.

W latach dziewięćdziesiątych nastąpi szybki rozwój neuroprocesorów umożliwiających konstruowanie neurokomputerów, czyli komputerów posiadających zdolność samodzielnego uczenia się, wykorzystywanych m.in. do rozpoznawania mowy, rozróżniania i identyfikowania charakterów pisma, analizy rysunków itp. Przewiduje się, że na początku XXI wieku neurokomputery będą zawierały ok. 120 mln neuronów i połączeń między nimi (obecnie do 0,5 mln).

Z początkiem XXI wieku zostanie osiągnięta bariera technologiczna możliwości budowy coraz szybszych mikroprocesorów poprzez zmniejszenie wielkości pojedynczych elementów. Wzrost mocy obliczeniowych komputerów będzie uzależniony przede wszystkim od liczby możliwych połączeń równoległych pracujących w nich procesorów. Pojawią się wówczas komputery szóstej generacji, w których oprócz wielu pracujących równolegle procesorów będą stosowane nadprzewodniki, elementy optyczne, połączone równolegle bazy danych o dużej pojemności na dyskach optycznych, neuroprocesory i kanały optyczne o szybkości transmisji wielu miliardów bitów na sekundę. Z końcem pierwszej dekady XXI wieku najtańsze komputery będą osiągać szybkość działania przekraczającą 80 mln operacji na sekundę, a najdroższe superkomputery 10 bilionów operacji na sekundę, natomiast programy komputerowe niejednokrotnie będą przekraczać 10 mln instrukcji dochodząc nawet do 100 milionów.



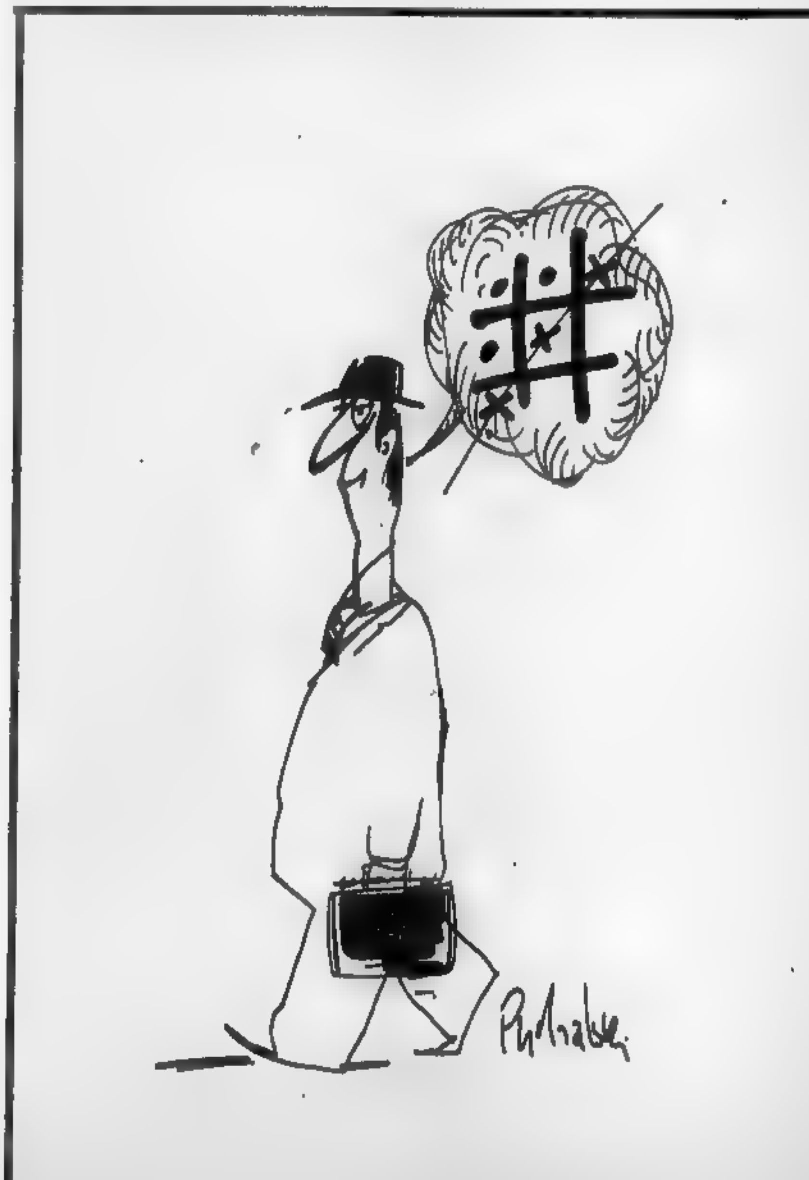
To warto powtórzyć

W podręczniku matematyki dla klasy III szkoły podstawowej znajdują się zadania dotyczące niedziesiątkowych systemów pozycyjnych. Wymagana jest między innymi umiejętność zapisywania liczb naturalnych w systemach o podstawie mniejszej niż 10 oraz zamiany zapisu z innego systemu na zapis w systemie pozycyjnym dziesiątkowym.

```
1 REM Kto zabierze ostatnia litere ?
20 CLS:a=8+INT(RND*9):b=80+INT(RND*120):d=b
30 PRINT"Gra polega na zabieraniu przez graczy - na przemian -
literek x"
40 PRINT"W jednym ruchu mozna zebrac od 1 do ";a;" literek"
50 PRINT"Wygra ten, kto wezmie ostatnia literke"
60 PRINT:PRINT:PRINT:PRINT"s - start"
80 a$=INKEY$
90 IF a$="s" OR a$="S" THEN 110
100 GOTO 80
110 CLS:LOCATE 1,8
120 FOR i=1 TO b:PRINT " x";:NEXT i
130 LOCATE 1,1:PRINT" Kto zaczyna?"
140 PRINT"k - komputer":PRINT"c - czlowiek"
160 a$=INKEY$
170 IF a$<>"k" AND a$<>"K" AND a$<>"c" AND a$<>"C" THEN 160
180 LOCATE 3,20
190 PRINT"Zabieramy od 1 do ";a;" literek"
200 LOCATE 1,1
210 FOR i=1 TO 4
220 PRINT"                                        "
230 NEXT i
240 IF a$="k" OR a$="K" THEN 270
250 IF a$="c" OR a$="C" THEN 390
260 GOTO 130
270 IF b>4*a THEN c=1+INT(RND*a):GOTO 300
280 c=b-(a+1)*INT(b/(a+1))
290 IF c=0 THEN c=1
300 LOCATE 1,1
310 PRINT"                                        "
320 LOCATE 1,1
330 PRINT"Zabieram ";c
340 b=b-c:l=l+c
350 FOR t=1 TO 2000:NEXT t
360 IF b=0 THEN PRINT"Niestety,przegrales":GOTO 500
370 LOCATE 1,8
380 FOR i=1 TO l:PRINT"  ";:NEXT i
390 LOCATE 1,3
400 PRINT"                                        "
410 LOCATE 1,3
420 INPUT"Ile zabierasz ",c
430 IF c<>INT(c) OR c<1 OR c>a THEN 390
440 b=b-c:l=l+c
450 FOR t=1 TO 2000:NEXT t
460 IF b<=0 THEN PRINT"Gratuluje, wygrales !!!":GOTO 500
470 LOCATE 1,8
480 FOR i=1 TO l:PRINT"  ";:NEXT i
490 GOTO 270
500 PRINT"Dowolna litera - nowa gra"
510 a$=INKEY$
520 IF a$="" THEN 510
530 GOTO 10
```

```
10 REM Zamiana liczb naturalnych zapisanych w innych systemach poz
ycyjnych na system dziesiatkowy
20 CLS:DIM t(30)
30 PRINT" Podaj podstawe systemu : 2,3,4,5,6,7,8 lub 9 ";:INPUT a
40 IF a<>INT(a) OR a<2 OR a>9 THEN PRINT"Nie oszukuj":GOTO 30
50 PRINT"Podaj liczbe w systemie o podstawie ";a;" ";:INPUT b
60 d=b
70 IF b<>INT(b) OR b<0 THEN PRINT"Podaj liczbe naturalna":GOTO 50
80 k=k+1:l=10^k
90 IF l>b THEN 110
100 GOTO 80
110 FOR i=k TO 1 STEP -1
120 t(i)=INT(d/(10^(i-1)))
130 IF t(i)>=a THEN PRINT" W tym systemie nie ma takiej liczby, bo
wiem nie ma cyfry ";t(i):GOTO 50
140 d=d-t(i)*10^(i-1)
150 NEXT i
160 PRINT" Podaj zapis tej liczby w systemie dziesiatkowym."
170 PRINT"Jesli sie pomylisz, otrzymasz pomoc ";:INPUT x
180 FOR m=1 TO k
190 y=y+t(m)*a^(m-1)
200 NEXT m
210 IF x=y THEN PRINT"Doskonale":END
220 PRINT"Niestety, nie pamietasz co oznacza zapis ";b;" w systemi
e o podstawie ";a
230 PRINT" Oznacza on, ze jest to suma skladnikow postaci: "
240 PRINT
250 FOR i=k TO 1 STEP -1
260 PRINT t(i);"*";a;"^";i-1
270 NEXT i
280 PRINT:PRINT"Policz teraz ";:INPUT x
290 IF x=y THEN PRINT"Teraz dobrze":END
300 PRINT"Dodaj jeszcze raz":PRINT
310 FOR i=k TO 1 STEP -1
320 PRINT t(i);"*";a^(i-1)
330 NEXT i
340 PRINT:INPUT x
350 IF x=y THEN PRINT"Teraz dobrze":END
360 PRINT:PRINT"Dodaj jeszcze raz":PRINT
370 FOR i=k TO 1 STEP -1
380 PRINT t(i)*a^(i-1)
390 NEXT i
400 PRINT:INPUT x:GOTO 350
```

```
1 REM Zamiana zapisu w systemie dziesiatkowym na zapis w
    innym systemie pozycyjnym
5 CLS
10 PRINT"Wybierz podstawe systemu - 2,3,4,5,6,7,8 lub 9 ";
11 INPUT a
15 IF a<>INT(a) OR a<2 OR a>9 THEN PRINT"Nie oszukuj":GOTO 10
20 PRINT"Wybierz liczbe do zamiany ";:INPUT b
25 IF b<>INT(b) OR b<0 THEN PRINT"Nie oszukuj":GOTO 20
30 d=b:i=-1
35 i=i+1
40 PRINT"Ile to jest ";a;" do potegi ";i;" ";:INPUT c
45 IF c=a^i THEN 55
50 GOSUB 199
51 GOTO 40
55 IF c>b THEN 65
60 GOTO 35
65 DIM t(i)
70 FOR k=i-1 TO 0 STEP -1
75 PRINT"Ile razy ";a^k;" miesci sie w ";d;" ";
76 INPUT t
80 IF t=INT(d/a^k) THEN PRINT "Zapamietaj: ";t:GOTO 90
85 GOSUB 199
86 GOTO 75
90 t(k)=t:f=d:IF k=0 THEN 115
95 PRINT"Ile zostaje ";:INPUT d
100 IF d=f-t*a^k THEN 110
105 GOSUB 199
106 GOTO 95
110 NEXT k
115 FOR m=0 TO i-1:y=y+t(m)*10^m:NEXT m
120 PRINT"Z zapamietanych cyfr utworz liczbe ";
125 INPUT x
130 IF x=y THEN 140
135 GOSUB 199
136 GOTO 125
140 PRINT"To jest liczba ";b;" zapisana w systemie";"o podsta
  wie ";a
145 IF l=0 THEN PRINT "Gratuluje, zadanie rozwiazane bezbled
  nie ":END
150 PRINT"Liczba bledow: ";l:END
199 PRINT"Policz jeszcze raz":l=l+1
200 RETURN
```



Programy narzędziowe na CPC — AMSTRAD 6128

DYSKEDYT

Program DYSKEDYT umożliwia poprawianie dowolnych bajtów każdego sektora wybranej ścieżki dyskietki i jest szczególnie wygodnym narzędziem, gdy chcemy zmienić język polski na języki obcojęzyczne opisy występujące w większości gier komputerowych.

Edycja wybranych bajtów programu (zbioru) odbywa się z poziomu pojedynczego sektora określonej ścieżki dyskietki (interesujący nas sektor jest ściągany do pamięci operacyjnej, a stąd po dokonaniu zmian, zapisywany z powrotem na dyskietkę). Program sygnalizuje każdorazowo błędny odczyt lub zapis sektora. Po wybraniu i wyświetleniu dowolnego sektora możemy wywołać opis dostępnych komend tzw. menu, przyciskając klawisz funkcyjny F0. Z opisem tym trzeba się koniecznie zapoznać, gdyż poniżej piszemy o niektórych tylko komendach używanych w programie.

OPIS DZIAŁANIA PROGRAMU

Po ściągnięciu programu do pamięci i uruchomieniu (komenda RUN „nazwa programu"), w górnym wierszu ekranu pojawi się nazwa programu, dalej wypełniająca prawie cały ekran tabela, a w dolnym wierszu — komunikat: dysk (0,1):. Jeśli korzystamy z napędu umieszczonego w komputerze, to wciskamy kolejno klawisze: "0" i ENTER (klawisz z cyfrą wciskamy, gdy dyskietka znajduje się w dołączonej do komputera stacji dysków). Po wybraniu odpowiedniego napędu pojawi się komunikat:
numer ścieżki:

Po wpisaniu numeru ścieżki (z przedziału <0—39>, na której znajduje się interesujący nas sektor i wciśnięciu klawisza ENTER pojawi się komunikat:
numer sektora:

Na każdej ścieżce dyskietki znajduje się 9 sektorów po 512 bajtów każdy. Dla dyskietek sformatowanych jako DATA wybieramy numer sektora z przedziału <193—201>, a dla sformatowanych jako SYSTEM z przedziału <65—73>.

Po wykonaniu wyżej wymienionych czynności na ekranie monitora w podzielonej na trzy części tabeli ukażą się kolejno: w lewej części numer kolejnych szesnastu bajtów w kodzie szesnastkowym, dalej pierwsze 256 bajtów sektora przedstawionych w kodzie szesnastkowym (drugie 256 bajtów możemy wyświetlić wciskając klawisz funkcyjny F2), a w prawej części te same pierwsze 256 bajtów sektora przedstawione w kodzie ASCII.

Po przepisaniu programu DYSKEDYT (zachowaniu na dyskietce) i zapoznaniu się z całym opisem komend (klawisz funkcyjny F0) stwierdzicie, że warto mieć tego typu narzędzie w swojej biblioteczce oprogramowania.

```
10 REM ****************************************************
20 REM *                                                  *
30 REM *   AMSTRAD - EDYTOR SEKTOROW NA DYSKIETCE         *
40 REM *                                                  *
50 REM ****************************************************
60 MODE 2:INK 0,1:INK 1,26:PAPER 0
70 MEMORY &7FFF:GOSUB 1340:' wstawienie kodu maszynowego
80 GOSUB 650:' inicjacja
90 WINDOW SWAP 0,1
100 GOSUB 800:' opis ekranu
110 CLS
120 t=0:cc=0:cr=0:wd=4:wi=24:off=64:v$=d$
130 PRINT CHR$(23);CHR$(1);:TAG #wd:GOSUB 560:POKE &908C,&80:GO
TO 340
140 s$="":f=0:WHILE s$="":c=0:f=f XOR 1:MOVE off+cc*wi+t*8,334-
cr*16:PRINT #wd,CHR$(143);:WHILE s$="" AND c<20:c=c+1:GOSUB 770
:WEND:WEND:IF f=1 THEN MOVE off+cc*wi+t*8,334-cr*16:PRINT #wd,C
HR$(143);
150 address=PEEK(&908B)+PEEK(&908C)*256+cc+cr*16
160 IF ASC(s$)<237 THEN 370
170 IF ASC(s$)<239 THEN GOSUB 1300:IF h$="N" THEN 140
180 IF ASC(s$)=237 THEN k$="zapis sektora":GOSUB 500:GOTO 140
190 IF ASC(s$)=238 THEN 1320
200 IF ASC(s$)<244 THEN 1210
210 IF ASC(s$)=246 THEN 1270
220 IF ASC(s$)=247 THEN GOSUB 940:GOSUB 800:GOSUB 840:GOTO 350
230 cc=0:cr=0:t=0:CLS
240 IF ASC(s$)=245 THEN POKE &908C,PEEK(&908C)XOR 1:GOTO 350
250 IF ASC(s$)=251 THEN sector=sector+1
260 IF ASC(s$)=250 THEN sector=sector-1
270 IF ASC(s$)=248 THEN track=track+1
280 IF ASC(s$)=249 THEN track=track-1
290 IF ASC(s$)=244 THEN GOSUB 560:POKE &908C,&80
300 IF sector>fsn+r THEN sector=fsn:track=track+1
310 IF sector<fsn THEN sector=fsn+r:track=track-1
320 IF track<0 THEN track=39
330 IF track>39 THEN track=0
340 k$="odczyt sektora":GOSUB 500:IF PEEK(&90E3)<>0 THEN 140
350 LOCATE #1,8,3:PRINT#1,drive:LOCATE #1,20,3:PRINT#1,track:LO
CATE #1,32,3:PRINT#1,sector:LOCATE #1,44,3:PRINT#1,PEEK(&908C)A
ND &7F:LOCATE #1,58,3:PRINT#1,a$
360 LOCATE #2,1,1:WINDOW SWAP 0,2:TAGOFF #wd:CLS#3:CLS#4:CLS#5:
CALL &9000:TAG #wd:WINDOW SWAP 0,2:GOTO 140
370 IF t=1 AND wd=4 THEN POKE address,(PEEK(address) AND &F0)+V
AL("&"+s$)
380 IF t=0 AND wd=4 THEN POKE address,(PEEK(address) AND &F)+VA
L("&"+s$)*16
390 IF wd=5 THEN POKE address,ASC(s$)
400 LOCATE #1,9+cc*3,5+cr:k$=HEX$(PEEK(address)):IF LEN(k$)=1 T
HEN PRINT#1,"0";
410 PRINT#1,k$:LOCATE #1,59+cc,5+cr:k$=".":IF PEEK(address)>31
THEN k$=CHR$(PEEK(address))
420 PRINT#1,k$;
430 IF wd=4 THEN t=t XOR 1:IF t=0 THEN cc=cc+1
440 IF wd=5 THEN cc=cc+1
450 IF cc=16 THEN cc=0:cr=cr+1
460 IF cc=-1 THEN cc=15:cr=cr-1
470 IF cr=16 THEN cr=0
480 IF cr=-1 THEN cr=15
490 GOTO 140
500 j=PEEK(&90E5):u=PEEK(&90E6)
510 POKE &90E5,track:POKE &90E4,drive:POKE &90E6,sector:POKE &9
0E3,0
520 IF ASC(k$)=111 THEN CALL &90B1 ELSE CALL &90C3
```

```
530 IF PEEK(&90E3)<>0 THEN CLS:PRINT"BLAD ";k$;" sciezka:";
track;" sektor:";sector;CHR$(7);track=j:sector=u:t$="":WHILE t$
=INKEY$:WEND:CLS:RETURN
540 IF ASC(k$)=122 THEN PRINT"prawidlowy zapis sektora";CHR$(7)
550 RETURN
560 GOSUB 840
570 CLS:LOCATE 1,1:PRINT" dysk (0,1): ";:nc=1:mi=0:ma=1:GOSUB 8
60:drive=VAL(w$):LOCATE #1,8,3:PRINT#1,drive
580 IF drive=0 THEN |A ELSE |B
590 fsn=PEEK(PEEK(dph+16*drive)+PEEK(dph+1+16*drive)*256+15)
600 r=8:IF fsn=&41 THEN a$="system" ELSE IF fsn=&C1 THEN a$="da
ta  " ELSE a$="ibm   ":r=7
610 LOCATE #1,58,3:PRINT#1,a$
620 CLS:LOCATE 1,1:PRINT"numer sciezki: ";:nc=2:mi=0:ma=39:GOSU
B 860:track=VAL(w$):LOCATE #1,19,3:PRINT#1,track
630 CLS:LOCATE 1,1:PRINT CHR$(22);CHR$(1);"numer sektora (";CHR
$(8);fsn;CHR$(8);"-";CHR$(8);fsn+r;CHR$(8);"): ";CHR$(22);CHR$(
0);:nc=3:mi=fsn:ma=fsn+r:GOSUB 860:sector=VAL(w$):LOCATE #1,31,
3:PRINT#1,sector;" "
640 CLS:RETURN
650 CALL &90BE: 'inicjowanie kodu maszynowego
660 dph=PEEK(&BE40)+PEEK(&BE41)*256+10:d$="0123456789ABCDEF"
670 KEY DEF 14,0,245:KEY DEF 5,0,246
680 KEY DEF 15,0,247:KEY DEF 20,0,237
690 KEY DEF 12,0,238:KEY DEF 13,0,244
700 KEY DEF 2,1,241,249,249
710 KEY DEF 0,1,240,248,248
720 KEY DEF 8,1,242,250,250
730 KEY DEF 1,1,243,251,251
740 POKE &908B,0:POKE &908C,&80
750 WINDOW #1,1,80,22,25:WINDOW #2,2,80,5,25:WINDOW #3,2,6,5,20
:WINDOW #4,9,56,5,20:WINDOW #5,59,75,5,20
760 RETURN
770 s$=INKEY$:IF (s$<CHR$(237) OR s$>CHR$(252)) AND INSTR(""+v$
,UPPER$(s$))<1 AND LEN(v$)>0 THEN s$=""
780 IF v$<>"" THEN s$=UPPER$(s$)
790 RETURN
800 CLS #1:PEN #1,1:LOCATE #1,1,1:PRINT#1,"AMSTRAD - EDYTOR SEK
TOROW NA DYSKIETCE"
810 MOVE 0,380:DRAWR 305,0
820 MOVE 0,72:DRAWR 0,272:DRAWR 600,0:DRAWR 0,-272:DRAWR -600,0
:MOVER 48,0:DRAWR 0,272:MOVER 405,0:DRAWR 0,-272
830 RETURN
840 LOCATE #1,1,3:PRINT#1," dysk:   sciezka:     sektor:
strona:      format:       ":RETURN
850 nc=2:mi=2:ma=10
860 w$=""
870 s$="":WHILE s$="" OR LEN(w$)=nc AND s$<>CHR$(127) AND s$<>C
HR$(13) OR s$=CHR$(127) AND w$="":s$=INKEY$:WEND
880 IF s$=CHR$(13) THEN 920
890 IF s$=CHR$(127) THEN w$=LEFT$(w$,LEN(w$)-1):PRINT CHR$(8);C
HR$(16);
900 IF s$<>CHR$(127) THEN PRINT s$;:w$=w$+s$
910 GOTO 870
920 IF VAL(w$)<mi OR VAL(w$)>ma THEN FOR a=1 TO LEN(w$):PRINT C
HR$(8);CHR$(16);:NEXT:PRINT CHR$(7);:GOTO 860
930 RETURN
940 WINDOW SWAP 0,1:CLS
950 REM ############################################
960 REM #                                          #
970 REM #        OPIS CZYNNOSCI EDYCYJNYCH         #
980 REM #                                          #
990 REM ############################################
1000 LOCATE 31,1:PRINT CHR$(24);" *OPIS KOMEND* ";CHR$(24)
1010 LOCATE 1,3:PRINT CHR$(24);"ustawienie sciezki lub sektora:
";CHR$(24)
1020 PRINT" <shift> "+CHR$(243)+" ... nastepny sektor."
1030 PRINT" <shift> "+CHR$(242)+" ... poprzedni sektor."
1040 PRINT" <shift> "+CHR$(240)+" ... nastepna sciezka."
1050 PRINT" <shift> "+CHR$(241)+" ... poprzednia sciezka."
1060 LOCATE 1,9:PRINT CHR$(24);" komendy edycji:";CHR$(24)
1070 PRINT" "+CHR$(243)+" ... nastepny bajt."
1080 PRINT" "+CHR$(242)+" ... poprzedni bajt."
1090 PRINT" "+CHR$(241)+" ... nastepna linia."
1100 PRINT" "+CHR$(240)+" ... poprzednia linia."
1110 LOCATE 1,15:PRINT CHR$(24);" komendy pomocnicze - klawisze
funkcyjne:";CHR$(24)
1120 PRINT" f0 ... opis."
1130 PRINT" f1 ... nowe parametry dysku."
1140 PRINT" f2 ... numer strony (0 lub 1) sektora."
1150 PRINT"         (0-255 lub 256-511 bajta na strone)."
1160 PRINT" f3 ... edycja w kodzie hex lub ascii."
1170 PRINT" f4 ... zapisanie sektora na dysku."
1180 PRINT" f5 ... wyjscie z programu."
1190 LOCATE 30,24:PRINT CHR$(24);" wyjscie z opisu - SPACJA ";C
HR$(24):s$="":WHILE s$<>" ":s$=INKEY$:WEND
1200 CLS:WINDOW SWAP 0,1:RETURN
1210 IF ASC(s$)=243 THEN 430
1220 IF ASC(s$)=242 AND wd=4 THEN t=t XOR 1:IF t=1 THEN cc=cc-1
1230 IF ASC(s$)=242 AND wd=4 THEN 450
1240 IF ASC(s$)=242 AND wd=5 THEN cc=cc-1:GOTO 450
1250 IF ASC(s$)=240 THEN cr=cr-1:GOTO 470
1260 cr=cr+1:GOTO 470
1270 TAGOFF #wd:wd=wd XOR 1:TAG #wd:t=0
1280 IF wd=4 THEN wi=24:off=64:v$=d$ ELSE wi=8:off=464:v$=""
1290 GOTO 140
1300 IF ASC(s$)=237 THEN k$="zapis sektora " ELSE k$="wyjscie z
programu "
1310 LOCATE 1,1:PRINT k$;"- JESTES PEWIEN (T/N)?";CHR$(7);:h$="
":WHILE INSTR("$TN",h$)<2:h$=UPPER$(INKEY$):WEND:CLS:RETURN
1320 TAGOFF #wd:PRINT CHR$(23);CHR$(0):MODE 1:CALL &BB00
1330 END
1340 IF PEEK(&9000)=&DD THEN RETURN
1350 RESTORE 1430
1360 checksum=0
1370 FOR g=&9000 TO &90DC
1380 READ b$:POKE g,VAL("&"+b$)
1390 checksum=checksum+VAL("&"+b$)
1400 NEXT
1410 IF checksum=30232 THEN RETURN
1420 PRINT"m/c blad danych";CHR$(7)
1430 DATA dd,2a,8b,90,dd,e5,c1,78,e6,7f
1440 DATA 32,8d,90,cd,62,90,79,32,8d,90
1450 DATA cd,62,90,dd,e5,06,03,cd,81,90
1460 DATA 06,10,dd,7e,00,32,8d,90,cd,62
1470 DATA 90,3e,20,cd,5a,bb,dd,23,10,ee
1480 DATA 06,02,cd,81,90,dd,e1,06,10,dd
1490 DATA 7e,00,fe,20,d2,45,90,3e,2e,c5
1500 DATA cd,5d,bb,c1,dd,23,10,ed,3e,0d
1510 DATA cd,5a,bb,3e,0a,cd,5a,bb,dd,e5
1520 DATA c1,3e,00,b9,c2,04,90,c9,21,8d
1530 DATA 90,ed,6f,cd,70,90,ed,6f,cd,70
1540 DATA 90,c9,e6,0f,c6,30,fe,3a,fa,7b
1550 DATA 90,c6,07,c5,cd,5a,bb,c1,c9,c5
1560 DATA 3e,09,cd,5a,bb,c1,10,f7,c9,00
1570 DATA 00,00,dd,21,dd,90,21,db,90,cd
1580 DATA d4,bc,dd,75,00,dd,74,01,dd,71
1590 DATA 02,21,dc,90,cd,d4,bc,dd,75,03
1600 DATA dd,74,04,dd,71,05,c9,21,00,80
1610 DATA ed,5b,e4,90,3a,e6,90,4f,df,dd
1620 DATA 90,d2,d5,90,c9,21,00,80,ed,5b
1630 DATA e4,90,3a,e6,90,4f,df,e0,90,d2
1640 DATA d5,90,c9,3e,ff,32,e3,90,c9,84
1650 DATA 85
```

Koniec „dżumy"?

Uczeni radzieccy szukają środka przeciwko niezwykłej chorobie elektronicznej.

W początkach działalności dziennikarskiej miałem szczęście przeprowadzić wywiad z samym Norbertem Winerem — „Ojcem cybernetyki", jak go wtedy nazywano.

„Człowiek nadaje maszynom cybernetycznym zdolność tworzenia i tym samym stwarza sobie potężnego sojusznika — mówił on — *ale właśnie tutaj czai się niebezpieczeństwo, które może powstać już w najbliższej przyszłości".*

Winera niepokoiła nasza niezdolność do przekazywania swoich życzeń maszynie w sposób dokładny i wyrazisty. Nie przychodziło mu do głowy, że stworzone z wielkim trudem programy, charakteryzujące się przejrzystością, dokładnością i jednoznacznością, staną się przedmiotem świadomego niszczenia — „terroryzmu" komputerowego. Wątpliwym jest, czy już wtedy uświadamiał on sobie, że już w najbliższych dziesięcioleciach ludzkość uzależni się od komputera.

Pierwsze niepokojące informacje łączono z tajemniczą epidemią, która „poraziła" komputery osobiste, należące do setek tysięcy Amerykanów. Udało się wykryć, że „wirus" został przywieziony z pakistańskiego miasta Lachore, a dokładnie z niedużego sklepu z programami komputerowymi, którego właścicielami byli dwaj bracia Alwi — 26-letni Amdżad Faruk i 19-letni Basit Faruk. Sprzedawali oni dyskietki z zapisanymi na nich „figlarnymi" programami po niezwykle niskich cenach. Naturalnie turyści ulegali pokusie i kupowali je tysiącami, darowując je przyjaciołom i znajomym, nie wiedzieli, że tym samym rozpowszechniają „wirusa" stwarzającego w pamięci maszyny coś na kształt elektronicznego konfetti. Podczas dochodzenia przyznali się, że chcieli „ukarać" Amerykanów. W późniejszym okresie, już we wrześniu br., 40-letni Amerykanin Donald J. Burleson, były programista jednej z firm, specjalnie „zaraził" komputery, żeby zemścić się za jakąś niesprawiedliwość, której w jego mniemaniu firma dopuściła się w stosunku do jego osoby. Wydarzenia potoczyły się lawinowo. Jeden za drugim uszkadzały się komputery, nawet te wykonujące najpoważniejsze operacje. Nawet w Ministerstwie Wojny USA komputery zatrzymywały się lub zmniejszały szybkość operacji, ponieważ ich pamięci wypełnione były bezsensownymi „programami-przybyszami". I tutaj też stosunkowo szybko znaleziono winowajcę, który o mało co nie stał się przyczyną ogólnonarodowej klęski.

Programy — produkt absolutnie szczególnego rodzaju, zawierają w sobie wielką potęgę intelektualną. Razem z rozwojem techniki komputerowej w tyle pozostały zagadnienia etyki. Zjawisko to stało się problemem natury międzynarodowej.

Swego czasu akademik A. Jerszow, obecny prezes Rady Naukowej „Cybernetyka" Akademii Nauk ZSRR, jeden z największych w świecie teoretyków programowania, opowiadał, jak wysokie mogą być koszty jednego tylko błędu w programie komputerowym.

Pewien programista w Wołżańskiej Fabryce Samochodów ulegający wszelkiego rodzaju zmartwieniom z powodu niedoceniania jego pracy, bawił się uświadamiając sobie swoją władzę nad tokiem procesu produkcyjnego — jak mówił. Znając dobrze program kierujący główną taśmą produkcyjną, doszedł do wniosku, że wystarczy zmienić stan tylko jednego ogniwa w pamięci komputera, a program zacznie zachowywać się czasami normalnie, to znów całkowicie bezsensownie. W pewnym momencie stał się niewolnikiem swojej idei — nie dawała mu ona spokoju. Myśl o takiej możliwości nie odstępowała go. Pewnego razu nieszczęśliwiec wykonał naprawdę szalony zamysł. Załamał się cały system podawania detali na taśmę. Próbowano znaleźć błąd w programie, lecz znajdowano jedynie nieprzewidywane odchylenia. Gdy ostatecznie udowodniono winę programiście, przyznał się on do przestępstwa.

— I co zrobiono z współczesnym Herostratem[1]? — zapytałem.

— Sądzono. Był to u nas pierwszy tego rodzaju proces. Okazało się, że brak w tej sprawie dowodów rzeczowych popełnionego przestępstwa, nie było uszkodzenia sprzętu, niczego nie ukradziono. Występek podsądnego można było zakwalifikować również jako chuligaństwo w produkcji. Ale sprawa nie leży w problemie prawnym, a w tym, że programy muszą posiadać niezawodne zabezpieczenie, znacznie większe niż np. mosty czy obrabiarki. Nie zdajemy sobie sprawy z tego, że uszkodzona konstrukcja np. mostu, będąca skutkiem inżynierskiej pomyłki w obliczeniach, jest nieporównywalnie bardziej kosztowna od ceny (zresztą stosunkowo niewysokiej) dokładnego sprawdzenia programu. Zdarzenie to miało miejsce przed kilkoma laty.

O współczesnej „dżumie" elektronicznej — epidemii, podobnie jak zaraza wirusowa rażącej pamięć komputerową, [illegible]ającej się w przechowywaniu bezsensownych informacji — nikt wtedy nie myślał. Ale uczony potrafił przewidzieć groźne niebezpieczeństwo. Obecnie, gdy informacje o „dżumie komputerowej" grożącej sparaliżowaniem ekonomiki USA wypierają ze szpalt gazet wiadomości poświęcone największym wydarzeniom politycznym, kłopoty Jerszowa stają się już powszechne.

Istota sprawy tkwi w tym, że naprawdę straszny „wirus" poraził nie tylko amerykańskie, ale i radzieckie komputery. Udowodniono tym samym istotną prawdę:

— dzisiejszy świat stał się nierozłączną całością

Ubiegłego lata w Pierwiesławiu Załesskim, starym rosyjskim mieście, w którym znajduje się jeden z najmłodszych instytutów akademickich — Instytut Systemów Programowych — odbywało się tradycyjne już spotkanie uczniów radzieckich i zagranicznych, zarażonych „bakcylem" komputerowym. Na letni obóz komputerowy zjechały dzieci z USA, RFN, Włoch, Bułgarii i CSRS. I, prawdopodobnie zupełnie przypadkowo ktoś z gości przywiózł na swojej dyskietce „wirusa", który został wprowadzony do komputera instytutowego.

Kiedy odnaleziono infekcję komputerową, wydawało się prawdopodobne, że wiele programów zostało zniszczonych i zagłada grozi również pozostałym programom. A przecież koszt oprogramowania wielokrotnie przewyższa koszt najdroższego urządzenia — jakakolwiek maszyna licząca bez programu, to tylko skupisko elektronicznych i mechanicznych urządzeń.

Dyrektor Instytutu Systemów Programowych AN ZSRR, doktor nauk fizyczno-matematycznych, profesor A. Ajłamazjan opowiedział mi o pracy, która umożliwiła nie tylko unieszkodliwienie „wirusa" komputerowego oraz „oczyszczenie" z niego komputerów instytutu, ale również i odtworzenie wszystkich zniszczonych przez niego programów. Jej idea polega na tym, że do „wirusa" komputerowego badacze, na czele których stał kierownik laboratorium systemów programowych dla równoległych architektur S. Abramow, podeszli jak do zwykłego, wirusa białkowego. Krok po kroku, jakby pod mikroskopem, prześledzili „przestępczą" działalność „wirusa", sprawdzili hipotezy o jego budowie, architekturze jego DNA (kwas dezoksyrybonukleinowy), o możliwym mechanizmie działania „wirusa", a następnie stworzyli antidotum tj. swego rodzaju program antywirusowy.

Oczywiście jeszcze za wcześnie mówić o tym, że wynaleziono uniwersalny środek walki z „dżumą komputerową", w obszarze fantastyki pozostają też marzenia o programach — szczepionkach „antywirusowych" dla komputerów. Ale pierwsze kroki zostały uczynione. Udało się nie tylko pokonać jednego konkretnego „wirusa", ale również wytyczyć kierunki walki z którymkolwiek z nich.

[1]Herostrat — żył w IV w p.n.e.; chcąc zdobyć sławę, podpalił słynną świątynię Artemidy Efeskiej; skazano go na śmierć, a imię jego na zapomnienie; stał się synonimem człowieka zdobywającego rozgłos przez zbrodnię.

Na podstawie artykułu K. Lewitina Koniec „dżumy"? opublikowanego w 317 numerze „Prawdy" z 12.11.1988 r.

Opracowali:
— W. JAGUPOW
— A. KUROCZKIN



INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

I

INTERACTION – wzajemne oddziaływanie, interakcja,
INTERACTION RANGE – obszar wzajemnego oddziaływania,
INTERACTIVE – interakcyjny, dialogowy (np. o systemie, któremu użytkownik wydaje polecenia w trakcie działania programu),
INTERACTIVE DEBUGGER – program umożliwiający poprawianie innych programów poprzez dialog z komputerem,
INTERACTIVE ENVIRONMENT – system konwersacyjny,
INTERACTIVE GRAPHIC – ekranopis przystosowany do dialogu z komputerem,
INTERACTIVE GRAPHICS – grafika interakcyjna (1. organizacja pracy systemu graficznego w którym użytkownik obserwuje i modyfikuje na bieżąco obraz na ekranie monitora, 2. dziedzina programowania związana z opracowywaniem systemów grafiki interakcyjnej),
INTERACTIVE LANGUAGE – język konwersacyjny, język bezpośredniego dostępu,
INTERACTIVELY SELECT – wybór w trybie konwersacyjnym,
INTERACTIVE MODE – tryb konwersacyjny, tryb interakcyjny,
INTERACTIVE OPERATION – praca konwersacyjna, praca interakcyjna,
INTERACTIVE PROCESSING – przetwarzanie konwersacyjne,
INTERACTIVE SYSTEM – system konwersacyjny,
INTERACTIVE UTILITY – konwersacyjny program narzędziowy,
INTERBLOCK GAP – przerwa miedzyblokowa (na nośniku informacji), przerwa blokowa,
INTERCHANGE – 1. wymiana (wzajemna), 2. wymieniać, przestawiać,
INTERCHANGEABLE – zamienny, wymienny,
INTERCHANGEABLY – na przemian,
INTERCHANGE POINT – punkt wymiany,
INTERCOMPUTER COMMUNICATION – komunikacja (łączność) międzykomputerowa,
INTERCONNECT – łączyć wzajemnie,
INTERCONNECTION – wzajemne połączenie, połączenie sprzęgające (sieci),
INTERCONVERSION – przetwarzanie reprezentacji danych,
INTERDEPENDENCE – współzależność, wzajemna zależność,
INTERFACE – 1. interfejs, łącze, sprzęg, złącze standardowe, 2. sprzęgać,
INTERFACE BOARD – tablica złącza (sprzęgu), tablica połączeń interfejsu,
INTERFACE CIRCUIT – układ sprzęgający,
INTERFACE COMPUTER – komputer interfejsowy,
INTERFACE DEVICE – urządzenie złącza (sprzęgu), urządzenie interfejsu (urządzenie zapewniające połączenie mikrokomputera z urządzeniami zewnętrznymi, siecią komputerową lub innymi mikrokomputerami),
INTERFACE-MESSAGE COMPUTER – komputer komunikacyjny,
INTERFACE MESSAGE PROCESSOR – procesor komunikacyjny (służy do przełączania pakietów, marszrut komunikacyjnych, podłączania terminali, utrzymywania łączności z satelitami, szyfrowania informacji, itp.),
INTERFACE MODULE – moduł złącza (sprzęgu), moduł interfejsu,
INTERFACE PROCESSOR – procesor wejścia/wyjścia, procesor sprzęgający,

INTERFACE SPECIFICATION – opis złącza (sprzęgu), opis interfejsu,
INTERFACING – sprzęganie, łączenie,
INTERFERE – wpływać, kolidować,
INTERFERENCE – zakłócenie, interferencja,
INTERFERENCE REJECTION – eliminacja zakłóceń,
INTERFERENCE SUPPRESSION – patrz: INTERFERENCE REJECTION,
INTERFERENCE SUPPRESSOR – eliminator zakłóceń,
INTERFERENCE TRANSMITTER – nadajnik zagłuszający,
INTERFERE WITH ... – kolidować z ...,
INTERFERING FREQUENCY – częstotliwość zakłócająca,
INTERIOR – wnętrze,
INTERLACE – międzyliniowość,
INTERLACED SCANNING – wybieranie międzyliniowe,
INTERLEAVE – przekładać (warstwami), poprzekładać, przeplatać,
INTERLEAVED MEMORY – pamięć o dostępie przeplatanym,
INTERLINE FLICKER – migotanie międzyliniowe,
INTERLISP – odmiana języka Lisp opracowana w firmie Xerox PARC,
INTERLOCK – blokada, uzależnienie, urządzenie ryglujące (programowe lub sprzętowe środki synchronizacji procesów, zapewniające nieprzerwaną pracę w systemach, sieciach, itp.),
INTERLOCKING – blokowanie, uzależnianie, ryglowanie,
INTERMEDIATE – pośredni (np. produkt pośredni, półprodukt),
INTERMEDIATE DATA CARRIER – nośnik danych pośredni,
INTERMEDIATE LANGUAGE – język pośredni (język na który tłumaczony jest program w trakcie pierwszego przebiegu translatora i z którego następuje dalsza translacja),
INTERMEDIATE STORAGE – pamięć pośrednia,
INTERMITTENT – nieciągły, przerywany, nietrwały, przejściowy,
INTERMITTENT ERROR – błąd przejściowy, błąd sporadyczny,
INTERMODULAR REFERENCE – odniesienie międzymodułowe, odwołanie międzymodułowe (np. wykorzystywanie w module nazw zmiennych deklarowanych w innych modułach)
INTERNAL – wewnętrzny,
INTERNAL ABEND – awaryjne zakończenie (np. przetwarzania) spowodowane przyczynami wewnętrznymi,
INTERNAL CHECK – kontrola wewnętrzna,
INTERNAL COMMAND – polecenie rezydentne,
INTERNAL FILE – plik (zbiór) wewnętrzny,
INTERNAL INTERRUPT – przerwanie wewnętrzne,
INTERNAL MEMORY – pamięć wewnętrzna, pamięć operacyjna,
INTERNAL NAME – nazwa wewnętrzna (np. nazwa dostępna tylko w module, w którym jest deklarowana),
INTERNAL PERFORMANCE – szybkość działania procesora (mierzona w liczbie wykonywanych operacji na sekundę),
INTERNAL REFERENCE – odwołanie wewnętrzne, odniesienie wewnętrzne,
INTERNAL REPRESENTATION – przedstawienie wewnętrzne, przedstawienie maszynowe (przedstawienie danych w pamięci komputera, zapewniające łatwość ochrony i przetwarzania),
INTERNAL SCHEMA – schemat wewnętrzny, projekt wewnętrzny,
INTERNAL SORT – sortowanie wewnętrzne,
INTERNAL SPECIFICATION – opis wykonania (opis wewnętrznej struktury programu i sposobu jego działania),
INVERSLY PROPORTIONAL – odwrotnie proporcjonalny,
INVERTED FILE – plik (zbiór) o odwróconym porządku,
INVERTED OUTPUT – wyjście zanegowane,
INVIGILATOR – sygnalizator,
INVISIBLE – niewidzialny, niewidoczny, niedostrzegalny,
INVOCATION – wywołanie (np. podprogramu, procedury),
INVOKE – wywoływać,
IN WORDS – słownie,
IN WRITING – na piśmie,
I/O – patrz: INPUT-OUTPUT,
I/O BOUND TASK – zadanie, którego czas wykonania ogranicza szybkość działania urządzeń wejścia/wyjścia,
I/O CONVERSION – przetwarzanie danych przy wprowadzaniu (z postaci tekstowej na wewnętrzną maszynową) i wyprowadzaniu z komputera (z postaci wewnętrznej na tekstową),
IOCS – patrz: INPUT/OUTPUT CONTROL SYSTEM,
I/O LIMITED PROGRAM – program, którego czas działania ogranicza szybkość działania urządzeń wejścia/wyjścia,
I/O LIST – lista wejścia/wyjścia,
I/O PORT – port wejścia/wyjścia,
IOS – patrz: INPUT/OUTPUT SYSTEM,
I/O WITH HANDSHAKING – przesyłanie z potwierdzeniem i cyklicznym testowaniem stanu urządzenia,
IP – patrz: 1. IMAGE POSITIONING, 2. INSTRUCTION POINTER,
IPL – patrz: INITIAL PROGRAM LOADER,
IRRATIONAL – niewymierny,
IRREGULAR – nieregularny, nieprawidłowy,
IRREVERSIBLE PROCESS – proces nieodwracalny,
IRS – patrz: INFORMATION RETRIEVAL SYSTEM,
IS-A – "jest elementem" (zależność pomiędzy konkretnym obiektem a pojęciem, którego jest on elementem),
ISAM – patrz: INDEXED SEQUENTIAL ACCESS METHOD,
ISARITHMIC CONTROL – sterowanie izorytmiczne,
ISN – patrz: INTERNAL SYSTEM NUMBER,
ISO – patrz: INTERNATIONAL STANDARDS ORGANISATION,
ISOBITS – bity równowartościowe,
ISO CODE – kod ISO (europejski odpowiednik amerykańskiego kodu ASCII),
ISOLATE – wyodrębniać, oddzielać,
ISOLATED WORD – słowo wybrane, słowo wyodrębnione,
ISOLATION – 1. wyodrębnienie, oddzielenie, izolowanie, 2. odłączenie (obwodu),

ISONET – Międzynarodowa Sieć Informacji Normalizacyjnej ISO,
ISR – patrz: 1. INTERRUPT SERVICE ROUTINE, 2. IN SERVICE REGISTER,
ISSUE – 1. wyjście, ujście, wydobywanie się, 2. wynik, rezultat, wydanie (publikacji),
IT – patrz: INFORMATION TECHNOLOGY,
ITEM – 1. pozycja (w spisie), 2. element danych,
ITEM NUMBER – numer pozycji,
ITEM SIZE – 1. rozmiar pozycji, 2. wielkość danych (w bitach, bajtach czy znakach),
ITEM VALUE – wartość (wielkość) elementu danych,
ITERATE – powtarzać, wykonywać iteracje,
ITERATION – iteracja, powtarzanie,
ITERATION STATEMENT – instrukcja cykliczna, instrukcja iteracyjna,
ITERATIVE – powtarzający się,
ITERATIVE ALGORITHM – algorytm iteracyjny,
ITERATIVE PROCESS – proces iteracyjny, proces powtarzający się,
INTERNAL STORE – patrz: INTERNAL MEMORY,
INTERNAL SYSTEM NUMBER – identyfikator obiektu,
INTERNAL TIMER – wbudowany regulator czasowy,
INTERNATIONAL BUSINESS MACHINES CORPORATION – amerykańska korporacja, największy na świecie producent sprzętu komputerowego,
INTERNATIONAL COMPUTATION CENTRE – Międzynarodowy Ośrodek Obliczeniowy,
INTERNATIONAL COMPUTERS LTD. – największy w Wielkiej Brytanii producent sprzętu komputerowego,
INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING Międzynarodowa Federacja Przetwarzania Informacji,
INTERNATIONAL STANDARDS ORGANISATION – Międzynarodowa Organizacja Normalizacji,
INTERNATIONAL TELEGRAPHIC ALPHABET NO 2 – międzynarodowy kod telegraficzny nr 2, kod dalekopisowy CCITT nr 2 (przesyłany znak jest reprezentowany przez 5 bitów),
INTERNET PROTOCOL – protokół wewnątrzsieciowy,
INTERNETTING – współdziałanie międzysieciowe, międzysieciowe oddziaływanie wzajemne,
INTERNETWORKING – współdziałanie międzysieciowe (komunikacja i wzajemne oddziaływanie pomiędzy węzłami różnych sieci komputerowych),
INTERPOLATION – interpolacja,
INTERPOLATOR – kolator, mieszacz,
INTERPRET – tłumaczyć, interpretować, objaśniać,
INTERPRETATION – tłumaczenie, interpretacja,
INTERPRETER – program interpretujący, interpreter, interpretator,
INTERPRETIVE EXECUTION – interpretacja, realizacja (wykonywanie) w trybie interpretacji,
INTERPRETIVE LANGUAGE – język interpretacyjny,
INTERPRETIVE MODE – tryb interpretacyjny,
INTERPRETIVE PROGRAM – patrz: INTERPRETER,
INTERPROCESS COMMUNICATION – współdziałanie procesów,
INTERPROCESSOR INTERFERENCE – interferencja międzyprocesorowa,
INTERRECORD GAP – przerwa między zapisami (rekordami),
INTERRELATION – współzależność, wzajemna zależność, wzajemne powiązanie,
INTERROGATE – zapytywać, zadawać pytania (np. maszynie),
INTERROGATE FEATURE – możność kierowania zapytań do maszyny cyfrowej,
INTERROGATION – zapytywanie, zgłoszenie, odzew,
INTERROGATION PROGRAM – program zapytywania,
INTERRUPT – przerwanie czynności maszyny,
INTERRUPT ACKNOWLEDGE – potwierdzenie przyjęcia przerwania,
INTERRUPT ADDRESS VECTOR – wektor adresów przerwań,
INTERRUPT CHANNEL – kanał przerwań,
INTERRUPT-DRIVEN I/O – układ wejścia/wyjścia sterowany przerwaniami,
INTERRUPT ENABLE – 1. zezwolenie na przerwanie, 2. znacznik włączający lub wyłączający system przerwań,
INTERRUPT ENABLE FLIP-FLOP – przerzutnik zezwalający na przerwania,
INTERRUPT EVENT – zdarzenie powodujące przerwanie,
INTERRUPT FEATURE – możliwość przerywania (np. realizacji programu),
INTERRUPT HANDLER – program obsługi przerwania,
INTERRUPTION – przerwa, przerwanie (np. programu),
INTERRUPT MASK – maska przerwania,
INTERRUPT-MASK REGISTER – rejestr maskowania przerwań,
INTERRUPT PRIORITY – priorytet przerwania,
INTERRUPT REGISTER – rejestr przerwań,
INTERRUPT REQUEST/ACKNOWLEDGE CYCLE – cykl przyjęcia przerwania,
INTERRUPT SERVICE ROUTINE (ISR) – podprogram obsługi przerwań,
INTERRUPT SOFTWARE – 1. oprogramowanie obsługi przerwania, 2. oprogramowanie działające po wywołaniu przerwania,
INTERRUPT STATE – patrz: INTERRUPT STATUS,
INTERRUPT STATUS – stan przerwania,
INTERRUPT SYSTEM – system przerwań,
INTERRUPT TRAP – 1. przerwanie, 2. obsługa przerwania,
INTERRUPT VECTOR – wektor przerwania (jedna lub kilka komórek pamięci, zawierająca adres programu obsługi przerwania i ewentualnie słowo stanu procesora, ustawiane przy wykonywaniu tego programu),
INTERRUPT VECTOR REGISTER – rejestr wektora adresów przerwań,
INTERSECTION – 1. część wspólna (zbiorów), 2. koniunkcja, iloczyn logiczny,
INTERSECTION OF EVENTS – iloczyn zdarzeń,
INTERSEGMENT LINK – połączenie międzysegmentowe,
INTERSEGMENT REFERENCE – odwołanie (odniesienie) międzysegmentowe,
INTERSPACE – odstęp, szczelina,
INTERSYSTEM CROSS-TALK – przesłuch między zespołami,
INTERTASK COMMUNICATION – współdziałanie między zadaniami, wzajemne oddziaływanie zadań,
INTERVAL – przedział czasu, odstęp, interwał,
INTERVAL OF CONVERGENCE – przedział zbieżności,
INTRA-APPLICATION COMMUNICATIONS AREA – obszar komunikacji wewnątrzzastosowaniowej (międzyprogramowej),
INTRICATE – skomplikowany,
INTRINSIC – 1. wewnętrzny, wbudowany, 2. przeznaczony,
INTRINSIC CALL – wywołanie wbudowanej procedury, wywołanie procedury rezydentnej,
INTRINSIC COMMAND – rozkaz rezydentny,
INTRINSIC FUNCTION – funkcja wewnętrzna (nie wymaga definiowania),
INTRINSIC MULTIPROCESSOR – wieloprocesor wewnętrzny,
INTRODUCE – wprowadzać,
INTRODUCE INTO SERVICE – wprowadzić do eksploatacji,
INTRUDER – użytkownik lub program, usiłujący bez zezwolenia dostać się do zbiorów danych,
IN TURN – kolejno, po kolei, w kolejności,
IN USABLE ORDER – w stanie nadającym się do pracy,
IN USE – 1. będący w użyciu, wykorzystywany, 2. zajętość,
INVALID – błędny, nieważny,
INVALID CHARACTER – znak błędny, znak niepoprawny,
INVARIABLE – niezmienny,
INVARIANT – niezmiennik, inwariant,
INVARIANT SYSTEM – układ niezmienny, układ zerozmienny,
INVENTORIES – zapasy (magazynowe),
INVENTORY TAPE – taśma stanów,
INVERSE – 1. odwrotny, 2. odwrotność,
INVERSE IMAGE – przeciwobraz,
INVERSE MATRIX – macierz odwrotna,
INVERSE VIDEO – obraz odwrócony (dot. grafiki),
INVERSION – inwersja, odwrócenie,

J

JABBER MONITOR – monitor nadmiernej długości pakietu,
JACK – gniazdko,
JAGGING – 1. nierówność, schodkowaty kształt (czegoś), 2. w grafice rastrowej – zniekształcenie linii z powodu dużego rozmiaru elementów rastra,
JCL – patrz: JOB CONTROL LANGUAGE,
J-K FLIP-FLOP – przerzutnik J-K,
JMP – patrz: JUMP,
JND – patrz: JUST-NOTICEABLE DIFFERENCE,
JOB – 1. zadanie (sekwencja programów), 2. praca (zarobkowa), robota,
JOB ACCOUNTING – rozliczanie zadania,
JOB BATCH – pakiet zadań,
JOB-BLENDING – łączenie w jednej osobie kilku funkcji (np. analityka i programisty),
JOB CONTROL – sterowanie zadaniami (rozdzielenie zasobów komputerowych pomiędzy zadania, ich załadowanie i uruchomienie, zabezpieczenie danych itp.),
JOB-CONTROL CARD – karta sterująca pracy,
JOB CONTROL LANGUAGE – język zarządzania zadaniami, język sterowania pracami,
JOB-CONTROL STATEMENT – dyrektywa,
JOB DECK – pakiet zadań (na kartach dziurkowanych),
JOB DEFINITION – opis zadania, zdefiniowanie zadania,
JOB DESCRIPTION – patrz: JOB DEFINITION,
JOB DESCRIPTION FILE – zbiór opisu zadania,
JOB-DIVIDING – dzielenie zadania do wykonania na części,
JOB FILE – plik (zbiór) zadania (plik, zawierający opis przygotowanego do wykonania zadania),

PIOTRUŚ PAN

Dokończenie ze strony 32

```
 na przesuwaniu wybranego piona
     (FLASH) na wolne pole."
3025 PRINT   PRINT . PRINT TAB 6
;"DOSTEPNE ROZKAZY."
3030 PRINT . PRINT TAB 6,"<w> -
wybor piona"
3031 PRINT TAB 6,"<r> - ruch pio
na"
3032 PRINT TAB 6;"<i> - informac
je"
3033 PRINT TAB 6,"<k> - kolory p
lanszy"
3034 PRINT TAB 6;"<n> - nowa gra
"
3100 PRINT #0; INK 9,"    nacisn
ij dowolny klawisz": PAUSE 0: RE
TURN
4000 REM Ustalanie kolorow
4010 INK 9: CLS   PRINT TAB 7; I
NVERSE 1,"USTALANIE KOLOROW"
4020 FOR k=1 TO 4  PRINT AT 2+2*
k,10,n$(k);TAB 19, INVERSE 1, IN
K k(k);"  "  PLOT 151,160-16*k
DRAW 17,0: DRAW 0,-9: DRAW -17,0
: DRAW 0,9. NEXT k
4030 PRINT : PRINT . PRINT : PRI
NT TAB 5;"wybor elementu <6>,<7>
"
4040 PRINT   PRINT TAB 7,"wybor
koloru <5>,<8>"
4050 PRINT . PRINT TAB 6;"koniec
 wyboru <k>"
4060 LET k=4: LET x$="6": GO TO
4135
4100 LET x$=INKEY$. IF x$="" THE
N GO TO 4100
4110 BEEP .05,20
4120 IF x$="5" THEN LET k(k)=k(k
)-1  LET k(k)=k(k)+8*(k(k)=-1)
PRINT AT 2+2*k,19; INK k(k); INV
ERSE 1;"  "
4125 IF x$="8" THEN LET k(k)=k(k
)+1. LET k(k)=k(k)-8*(k(k)=8)· P
RINT AT 2+2*k,19; INK k(k); INVE
RSE 1,"  "
4130 IF x$="7" THEN PRINT AT 2+2
*k,10,n$(k)· LET k=k-1  LET k=k+
4*(k=0)  PRINT AT 2+2*k,10  INVE
RSE 1;n$(k)
4135 IF x$="6" THEN PRINT AT 2+2
*k,10,n$(k)  LET k=k+1  LET k=k-
4*(k=5): PRINT AT 2+2*k,10, INVE
RSE 1;n$(k)
4140 IF x$="k" THEN RETURN
4150 GO TO 4100
9000 REM Pozycje liczb
9010 DATA 4.4
9011 DATA 4,15
9012 DATA 4,26
9013 DATA 11,19
9014 DATA 18,26
9015 DATA 18,15
9016 DATA 18,4
9017 DATA 11,11
9020 REM Polaczenia pol
9030 DATA 2,2.8
9031 DATA 2,1,3
9032 DATA 2,2,4
9033 DATA 3,3.8,5
9034 DATA 2,4,6
9035 DATA 2,5,7
9036 DATA 2,6.8
9037 DATA 3,1,4,7
9040 REM Nazwy elementow
9050 DATA "piony"
9051 DATA " pola"
9052 DATA "linie"
9053 DATA "  tlo"
9100 REM Pozycje pol
9110 DATA 2,2
9111 DATA 2,13
9112 DATA 2,24
9113 DATA 9,9
9114 DATA 9,17
9115 DATA 16,2
9116 DATA 16,13
9117 DATA 16,24
9120 REM Pozycje polaczen
9130 DATA 4,7
9131 DATA 4,18
9132 DATA 18,7
9133 DATA 18,18
9140 DATA 16 24
9141 DATA 16,6
9142 DATA 6,24
9143 DATA 6,6
9990 REM Zapis programu
9995 CLEAR · SAVE "PiotrusPan" L
INE 1. VERIFY "PiotrusPan"
```

Co to jest „Gorąca linia"



Minęło 25 lat od chwili, gdy między Moskwą a Waszyngtonem rozpoczęła pracę tzw. gorąca linia (hot line), czyli linia bezpośredniej łączności między obu stolicami supermocarstw. Nastąpiło to w wyniku porozumienia podpisanego przez Nikitę Chruszczowa i Johna Kennedy'ego, a bezpośrednią przyczyną jej powstania były wydarzenia z jesieni 1962 roku. Wówczas, w trakcie kryzysu karaibskiego świat zbliżył się bardzo niebezpiecznie do nowego konfliktu zbrojnego.

Do tego nie doszło, lecz zdawano sobie sprawę, że takie zagrożenie istnieje, nawet poprzez niedoskonały system wzajemnej informacji dotyczącej posunięć strategicznych obu stron. Rakieta balistyczna lecąca z jednej stolicy do drugiej potrzebowała wtedy 30 minut na przebycie tej drogi. Telefonistka natomiast aż godzinę, aby uzyskać połączenie Moskwa—Waszyngton lub odwrotnie.

W powszechnej opinii gorąca linia kojarzy się z czerwonymi telefonami umieszczonymi w gabinetach prezydenta USA i przywódcy radzieckiego. Nic podobnego. Od początku istnienia tej linii łączność realizowana była za pomocą dwóch kanałów teleksowych, którymi przekazywano zaszyfrowane depesze.

Praktycznie tak to wygląda do dziś, chociaż technika przekazywania komunikatów została unowocześniona i jest mocno skomplikowana. Początkowo przesyłano depesze za pośrednictwem specjalnego kabla leżącego na dnie oceanu. Potem to łącze zdublowano systemem łączności radiowej na falach krótkich. Stacja przekaźnikowa ulokowana została w Tangerze w Algierii. W 1977 roku, gdy nadeszła era łączności satelitarnej, gorącą linię

Dokończenie na stronie 32

„Papilarny kluczyk"

Ochrona danych dotyczących firmy i prowadzonych przez nią interesów przed konkurencją, to problem wielu biznesmenów posługujących się komputerem. Takie informacje przechowywane są dziś coraz częściej w pamięci tego urządzenia. Aby tę informację uzyskać trzeba znać określone hasło lub kod. Jak uczy praktyka, mogą one zostać rozszyfrowane lub wykradzione.

W celu uchronienia się przed taką ewentualnością część firm elektronicznych specjalizuje się w produkcji zabezpieczeń nazywanych biometrycznymi urządzeniami ochronnymi. Są to skomputeryzowane analizatory niepowtarzalnych cech fizycznych człowieka, na przykład głosu czy linii papilarnych. Te urządzenia chronią przed dostępem do nich osób niepowołanych.

Niedawno firma Identix specjalizująca się w produkcji takich zabezpieczeń wypuściła urządzenie przeznaczone dla komputerów osobistych. Dołączone przez złącze RS232C pozwala „nauczyć" komputer rozpoznawania właściciela po odciskach palców. Rozpoznawanie linii papilarnych trwa pierwszy raz kilkanaście sekund, później nie przekracza nawet dwóch sekund. Gdy ktoś nie upoważniony chce skorzystać z komputera, ten nie tylko nie daje się uruchomić, ale też może włączyć system alarmowy.

(Raj)

PIOTRUŚ PAN

```
 1 REM **************************
 2 REM *                        *
 3 REM *                 ©1989  *
 4 REM *                        *
 5 REM *    Krzysztof Poźniak   *
 6 REM *                        *
 7 REM *      PIOTRUŚ PAN       *
 8 REM *    gra dla 1 osoby     *
 9 REM *                        *
10 REM **************************

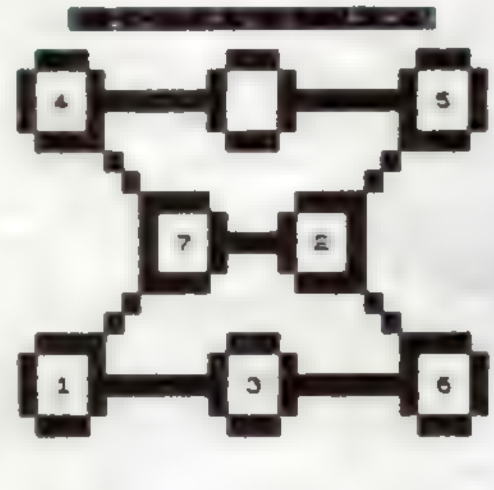
  20 DEF FN p$(x)=(" " AND x=0)+
(STR$ x AND x<>0)
 100 REM  Inicjalizacja programu
 110 CLEAR : INK 7: PAPER 0: INV
ERSE 0: OVER 0: BRIGHT 0: FLASH
0: BORDER 0: CLS
 120 DIM k(4): LET k(1)=7: LET k
(2)=6: LET k(3)=2: LET k(4)=0
 130 RESTORE 9000: DIM a(8,2): F
OR k=1 TO 8: READ a(k,1),a(k,2):
 NEXT k

 140 DIM p(8,4): FOR k=1 TO 8: R
EAD p(k,1): FOR y=1 TO p(k,1): R
EAD p(k,y+1): NEXT y: NEXT k
 150 DIM n$(4,5): FOR k=1 TO 4:
READ n$(k): NEXT k
 160 DIM w(8)
 200 GO SUB 3000: BEEP .1,20
 300 GO SUB 1400
1000 REM  Glowna petla programu
1005 GO SUB 2000
1010 PAUSE 0: LET x$=INKEY$: IF
x$="" THEN GO TO 1010
1015 BEEP .05,20
1020 IF x$="w" THEN GO SUB 1500:
 GO TO 1010
1030 IF x$="r" THEN GO SUB 1600:
 GO TO 1010
1040 IF x$="i" THEN GO SUB 3000:
 GO TO 1005
1050 IF x$="n" THEN GO SUB 1300:
 PAUSE 1: GO TO 1010
1060 IF x$="k" THEN GO SUB 4000:

 GO TO 1005
1090 BEEP .1,30: PRINT #0;" ROZK
AZ NIEZROZUMIALY   i)-inf": PAUS
E 100: LET x$=INKEY$: INPUT "":
IF x$="i" THEN GO SUB 3020: GO T
O 1005
1100 GO TO 1010
1300 REM  Nowa gra
1310 PRINT #0," Zaczynasz od now
a (t/n)"
1320 PAUSE 0: LET x$=INKEY$: INP
UT "": IF INKEY$="t" THEN FOR k=
1 TO 6: BEEP .05,10+5*k: NEXT k:
 GO SUB 1400: GO SUB 2000
1330 RETURN
1400 REM  Losowe ustaw. pionow
1410 RANDOMIZE 256*PEEK 23672+PE
EK 23672
1420 LET x$="01234567": FOR k=1
TO 10: LET x=INT (7*RND)+2: LET
x$=x$(2 TO x)+x$(1)+x$(x+1 TO ):
 NEXT k

1430 FOR k=1 TO 8: LET w(k)=VAL
x$(k): IF x$(k)="0" THEN LET p=k
1440 NEXT k: LET r=2: RETURN
1500 REM  Przesuniecie wskaznika
1510 PRINT AT a(p(p,r),1),a(p(p,
r),2);w(p(p,r))
1520 LET r=r+1: IF p(p,1)<r-1 TH
EN LET r=2
1530 REM  Wyswietlenie wskaznika
1540 PRINT AT a(p(p,r),1),a(p(p,
r),2); FLASH 1;w(p(p,r))
1550 RETURN
1600 REM  Przesuniecie piona
1610 LET w(p)=w(p(p,r)): LET w(p
(p,r))=0
1620 PRINT AT a(p(p,r),1),a(p(p,
r),2);" "
1630 PRINT AT a(p,1),a(p,2);FN p
$(w(p))
1640 LET p=p(p,r): LET r=2: GO S
UB 1530: RETURN
2000 REM  Rysowanie planszy

2010 PAPER k(4): INK k(2): BORDE
R k(4): CLS : PRINT TAB 5; INVER
SE 1;"P I O T R U S   P A N"
2015 RESTORE 9100
2020 FOR k=1 TO 8: READ y,x
2030 PRINT AT y,x+1; INVERSE 1;"
```



I N F O R M A C J A

Celem gry jest ulozenie pionow (cyfr) w/g z gory zalozonej kolejnosci. Drogi ruchu pionow zaznaczone sa liniami. Ruch polega na przesuwaniu wybranego piona (FLASH) na wolne pole.

DOSTEPNE ROZKAZY:

- (w) - wybor piona
- (r) - ruch piona
- (i) - informacja
- (k) - kolory planszy
- (n) - nowa gra

```
2031 PRINT AT y+1,x; INVERSE 1;"
 "; INVERSE 0;TAB x+4; INVERSE 1
;" "
2032 PRINT AT y+2,x; INVERSE 1;"
 "; INVERSE 0;TAB x+4; INVERSE 1
;" "
2033 PRINT AT y+3,x; INVERSE 1;"
 "; INVERSE 0;TAB x+4; INVERSE 1
;" "
2034 PRINT AT y+4,x+1; INVERSE 1
;"   "
2040 NEXT k
2050 INK k(3): FOR k=1 TO 4: REA
D y,x: PRINT AT y,x; INVERSE 1;"
      ": NEXT k

2060 FOR k=-1 TO 1 STEP 2: FOR l
=-1 TO 1 STEP 2: READ y,x: FOR m
=0 TO 3: PRINT AT y+k*m,x+l*m; I
NVERSE 1;" ": NEXT m: NEXT l: N
EXT k
2070 PRINT AT 11,14; INVERSE 1;"
  "
2080 INK k(1): FOR k=1 TO 8: PRI
NT AT a(k,1),a(k,2);FN p$(w(k)):
 NEXT k
2090 GO SUB 1530
2100 RETURN
3000 REM  Informacja
3010 INK 9: CLS : PRINT TAB 6; F
LASH 1;"I N F O R M A C J A"
3020 PRINT : PRINT : PRINT INVER
SE 1;"PIOTRUS PAN gra dla jednej
 osoby": PRINT : PRINT : PRINT "
 Celem gry jest ulozenie pionow
(cyfr) w/g z gory zalozonej ko-
lejnosci. Drogi ruchu pionow za-
znaczone sa liniami. Ruch polega
```

Dokończenie na stronie 31

Co to jest „Gorąca linia"

Dokończenie ze strony 31

poprowadzono przez kosmos. Od czterech lat istnieje też zaszyfrowana łączność teleksowa, dzięki której można przekazywać dokumenty, mapy i inne informacje graficzne.

Na Kremlu obecnie komputerowa końcówka (hot line) składa się z czterech terminali. Dwa z nich — w języku rosyjskim i angielskim — obsługiwane są drogą satelitarną, trzeci podłączony jest do kabla oceanicznego. Czwarty służy do kontaktowania się obsługi technicznej. By utrzymać stałą sprawność systemu gorącej łączności, co godzinę Moskwa i Waszyngton nawiązują łączność. W obu ośrodkach całodobowe dyżury pełnią najwyższej klasy fachowcy — inżynierowie i tłumacze. W godzinach nieparzystych komunikaty kontrolne nadaje strona radziecka, w godzinach parzystych — amerykańska. Są to teksty nie zawierające elementów propagandowych, a najczęściej zawierają fragmenty literatury pięknej lub opisy przyrody. Taki tekst kontrolny musi — w myśl wspólnego porozumienia — być neutralny i zawierać wszystkie litery alfabetu łacińskiego.

W ciągu ćwierć wieku istnienia gorącej linii nie było wypadku, aby łączność się urwała. Współcześnie, nie likwidując starych łączy, a dodając wciąż nowocześniejsze, osiągnięto wielokrotne zdublowanie systemu łączności, które w praktyce uniemożliwia przerwanie kontaktu między Moskwą i Waszyngtonem.

Gorąca linia została uzupełniona specjalnymi ośrodkami zmniejszającymi ryzyko wybuchu konfliktu nuklearnego. Takie porozumienie obie strony zawarły w ubiegłym roku. Ośrodki te, wyposażone w supernowoczesne urządzenia do przekazywania tekstów i obrazów, służą do wymiany niezbędnych informacji dla realizacji układu o likwidacji rakiet jądrowych średniego i krótszego zasięgu, dwóch układów o ograniczeniu podziemnych wybuchów jądrowych, układu o zapobieganiu incydentom na morzu oraz porozumienia o środkach zmniejszania ryzyka wybuchu wojny jądrowej.

(Raj)

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 146. Nr ind. 381462. A-41

INFORMATYKA KOMPUTERY SYSTEMY



CENA 120 zł

DODATEK DO „ŻOŁNIERZA WOLNOŚCI" NR 3/1989 ISSN 0860-2794



CeBIT 1989

8 marca 1989 roku 3067 sprzedawców przedstawiło swoją ofertę na światowej wystawie „Komputery i Komunikacja” w Hanowerze. Prezentowano bardzo szeroki asortyment od dyskietek po duże systemy komputerowe, od kabli po centrale telefoniczne. W sumie było tak wiele do oglądania, że gdyby ktoś chciał spędzić na każdym stoisku tylko 100 sekund, to powinien przebywać na wystawie po 11 godzin przez wszystkie 8 dni tego spektaklu. Próbę tę (nie wiadomo czy skutecznie) podjęło ponad pół miliona zwiedzających.

W tym roku wystawę tę zdominowały: sieci (komputerowe oczywiście), otwarte systemy i przetwarzanie obrazów. Oczywiście miało to swoje implikacje na prezentowane wytwory profesjonalnej technologii informatycznej oraz gusty użytkowników.

SIECI

Wśród nowych dokonań w sferze technologii informacyjnej — łączenie komputerów z urządzeniami komunikacyjnymi okazało się najbardziej znaczące. Było to widoczne na wielu stoiskach i podczas wielu imprez towarzyszących. Warto w tym miejscu choćby tylko wymienić zainteresowanych tą problematyką. Oto oni: ALCATEL, DIGITAL, ERCSSON, IBM, NIXDORF, NOVELL, PHILIPS i SIMENS.

Ranga przypisywana łączności ma bezpośredni związek z rokiem 1992 — „wielką unifikacją Europy Zachodniej”.

PRZETWARZANIE OBRAZÓW

Kopiarki, laserowe drukarki, optyczne skanery, oprogramowanie graficzne i systemy typu desktop publishing ukazywały wyraźny postęp we wszystkim, co jest związane z przetwarzaniem obrazów.

Przez lata owa dziedzina — wykorzystywania komputerów do przetwarzania obrazów widoczna była tylko w profesjonalnych zastosowaniach, takich jak medycyna, biura projektów. Nigdzie więcej. Natomiast nowa generacja tego typu sprzętu i oprogramowania pracuje z powodzeniem na każdym obrazie (także na znakach) i może być z powodzeniem stosowana nawet w domu. Wiodące w tej dziedzinie firmy, to: APPLE, CANON, RICOH i XEROX.

Zatem przyszłość informatyki leży (stoi?) w sieciach i przetwarzaniu obrazów.

Opracowano na podstawie: CeBIT 1989 Preview, NEWSWEEK (marzec 13/89)

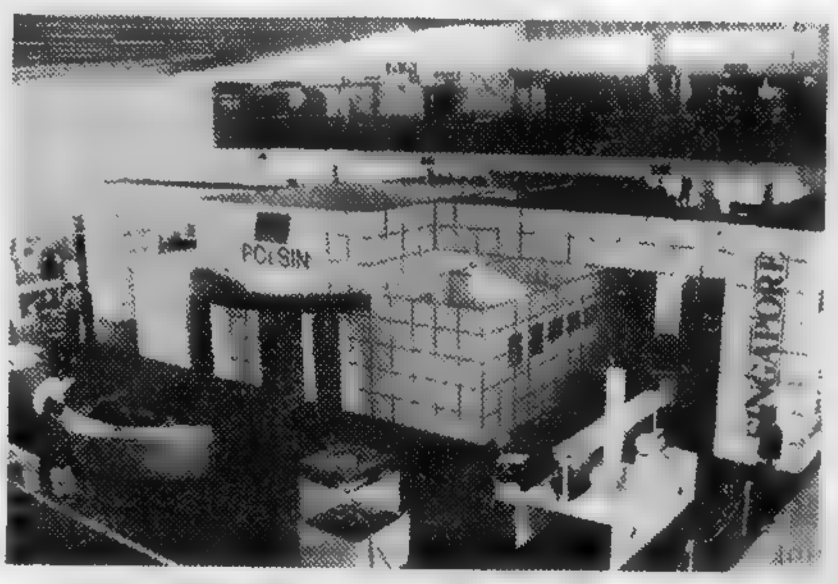
Włodzimierz GOGOŁEK

W NUMERZE:

CeBIT str. 2
Człowiek a komputer, czy tylko baśń str. 3
Uciążliwości przy obsłudze komputera str. 4
Jajo str. 5
Komputerowa analiza układów elektronicznych str. 7
Zastosowanie str. 14
Nowe edytory tekstu dla C-64 i C-128 str. 19
„SCROLLING” tekstu w dowolnym trybie graficznym str. 21
Zaglądamy pod skórę str. 25
„Labirynt” Lilavati (14) str. 27
Procedura Copy dla drukarki D-100 (i podobnych) str. 28
Informatyczny słownik str. 30







CZŁOWIEK A KOMPUTER, CZY TYLKO BAŚŃ?

Wszyscy dzisiaj jesteśmy zachwyceni komputerami, zachłystujemy się ich możliwościami, tym wszystkim, co przy ich wykorzystaniu możemy dokonać. Mówimy: oto zostały rozszerzone granice ludzkich możliwości, dokonujemy nowych odkryć i pogłębiamy ludzką wiedzę. Wszystko to jest prawdą, ale opuśćmy na chwilę ten zgiełk peanów na rzecz komputera i stojąc z boku spójrzmy chłodnym okiem na ten niewątpliwy fenomen społeczny — komputeryzację. Nie jestem wrogiem komputerów, ale... zdejmijmy ten produkt człowieka z piedestału.

Przyjmijmy tezę: komputer, to wróg człowieka, nie jest jego przyjacielem. Każdy człowiek obsługujący komputer wchodzi w określone związki z przedmiotem swojej pracy, wytwarza się swoisty układ pomiędzy tymi dwoma składnikami sytuacji. Zależność ta niesie ze sobą określone, niestety negatywne konsekwencje. Zainteresujmy się ludźmi zawodowo trudniącymi się informatyką w szerokim tego słowa ujęciu. Zauważymy, iż stworzyli oni odrębny klan posługujący się swoim własnym językiem, określonymi rodzajami skrótów myślowych, odczytywalnymi tylko przez wtajemniczonych. Powie ktoś: przecież to nic dziwnego, bowiem każda grupa zawodowa tworzy swój własny język oraz hierarchię wartości. Zgadzam się, ale przecież nie w każdej grupie młodych ludzi hierarchię stanowi pojemność pamięci wytworu ludzkich rąk.

Proponuję teraz mały spacer drogą zarysowaną w książkach S. Lema. Człowiek posługujący się komputerem i równocześnie obsługiwany przez ten komputer poczyna zatracać swoją ludzką istotę, tj. umiejętność posługiwania się swoimi szarymi komórkami. Wybierzmy jedno zdarzenie: poddano egzaminowi trzech studentów, zadając im do rozwiązania prosty problem matematyczny. Dwóch z nich nie podało żadnej odpowiedzi, bowiem komputer znajdujący się w pokoju wyłączono. Trzeci zaś skierował się w stronę komputera umieszczonego na korytarzu, a przez zapomnienie egzaminującego, nie wyłączonego z sieci. Zabawne zdarzenie, nieprawdaż? Ale, gdy przestaniemy się uśmiechać, zwróćmy uwagę na fakt, iż żaden ze studentów nie wykorzystał swojego umysłu. Któż zaręczy, że sytuacja ta nie stanie się kiedyś powszechną?

Przenieśmy się teraz do Krzemowej Doliny w Kalifornii, miejsca najbardziej „skomputeryzowanego" na kuli ziemskiej. Powinno ono stanowić miejsce powszechnej szczęśliwości, a jednak... Permanentnie wzrasta tam procent samobójstw, psychoz i rozwodów. Okazuje się bowiem, że ów klan informatyczny zaczyna w sferze psychiki wykazywać cechy obsługiwanych przez siebie maszyn. Ludzie ci nabrali nawyków pracy komputerowej w formułowaniu swoich myśli i — co jest znacznie bardziej niebezpieczne — zaczęli językiem komputera przemawiać do członków swoich rodzin. Przyznajmy, że nie jest to najlepszy sposób porozumiewania się ludzi najbliższych sobie uczuciowo. Czyż może nas teraz dziwić, że „skomputeryzowani" mieszkańcy Krzemowej Doliny oceniają postępowanie innych ludzi według skali 0 lub 1. Każde takie uzależnienie może stanowić realną groźbę braku interakcji między ludźmi lub silnego jej zubożenia. Nikt z nas przecież nie lubi, aby jego zachowanie oceniane było tylko z pozycji celowości i logiczności poczynań. I jeszcze jedno. „Skomputeryzowanie" sposobu myślenia i sposobu porozumiewania się może negatywnie wpłynąć na podświadomość człowieka, powodując zanik marzeń i fantazji. A czymże bez tego jesteśmy?

Każdy z nas słyszał chyba o terminie „potwór z Id", stworzonym przez psychologię, a określającym strukturę funkcjonującą w podświadomości człowieka i niszczącą jego osobowość poprzez jej rozszczepienie. Któż nam może zaręczyć, iż wpływ komputera, przejęcie jego schematów myślowych nie przyczyni się do wytworzenia nowego jakościowo „potwora z Id"? Czy w przyszłości nie może dojść do rozpadu osobowości człowieka skutkiem zarażenia się swoistym odpowiednikiem dzisiejszego „wirusa komputerowego"? Czy taka możliwość naprawdę nie może zaistnieć? Jeszcze tak niedawno nikt z nas nie słyszał terminu „wirus komputerowy", a dzisiaj mówimy o nim wszyscy.

W dzisiejszym świecie człowiek odbiera mnogość informacji poprzez odpowiedni dla siebie „kanał" oraz do określonego progu ich przyswajalności. W sytuacji natłoku informacji osiągających wartość ponadprogową powstaje swoisty „korek", a człowiek przestaje funkcjonować. Logika rozumowania wskazuje, że przypadek taki może mieć miejsce w sytuacji człowieka obsługującego skomputeryzowany system rozpoznania operacyjnego i czuwającego nad odpaleniem rakiet z głowicami jądrowymi. Cóż wówczas stać się może?

Ogarnięci jesteśmy fascynacją światem komputerów. Dlaczegóż więc czujemy się w ich obecności niezbyt pewnie? Czyżby instynkt sygnalizował nam jakieś niebezpieczeństwo? Jeśli nie, to skąd wywodzi się dążenie do nadawania komputerom kształtów najbardziej miłych dla człowieka, w myśl hasła: komputer przyjacielem człowieka?

Przyjrzyjmy się teraz wpływowi komputerów na psychikę dzieci. Japończycy prowadzący badania swych pociech zaobserwowali, że u dzieci bawiących się z komputerem przez wiele godzin pojawiają się dziwne objawy. Dziecko ma kłopoty z charakterystycznymi dla swego wieku zabawami, wykazuje trudności w kontaktach z rówieśnikami, następują zaburzenia snu. Najspokojniejsze są w obecności swego komputera, pracując z nim. Czy nie obserwujemy tu związku z zachowaniem się ludzi z Krzemowej Doliny? Nie wspomnę tu już o szkodliwym wpływie migotliwego, o nienaturalnych barwach ekranu monitora na wzrok programistów, czy też upośledzenia ich słuchu przez dźwięki komputera odbiegające od normalnych.

Kończąc swoją opowieść chcę poruszyć jeszcze jeden aspekt fenomenu, o którym mówimy — aspekt moralny. Gry wojenne, tak modne wśród młodzieży, stanowią olbrzymie zagrożenie, bowiem wpajają jej nawyki zabijania w trakcie zabawy. „Nalot na Moskwę" wskazuje konkretny cel: ludzi z obozu socjalistycznego, nazywanego nie tak dawno przez b. prezydenta Reagana mianem „imperium zła". Ten szkodliwy rodzaj indoktrynacji młodzieży powoduje odczłowieczenie wroga; nie jest on przecież wówczas człowiekiem, jest tylko celem. Doprawdy, argument, iż jest to tylko zabawa, nie trafia mi do przekonania. Amerykanie w czasie wojny wietnamskiej stosowali tę samą metodę. Nie mówili: człowiek, Wietnamczyk, tylko pogardliwie „żółtek", nie zabijali ludzi, tylko odpersonifikowanych „żółtków".

A teraz coś z pogranicza czarnej baśni, która stała się jawą. Otóż na Zachodzie coraz częściej pojawiają się programy, które na domowych komputerach pozwalają oglądać obrazy pornograficzne. Ich jakość przewyższa klasę odtwarzania w systemie wideo. A komputer trafił tam już na podwórka. Czy jeszcze niedawno ktoś z nas to przewidział?

Powiecie, to demagogia. Być może, ale baśń dzisiaj przeze mnie opowiedziana, jest nią już tylko częściowo. Jutro — może stać się przerażającą rzeczywistością.

Powtarzam: nie jestem przeciwnikiem komputerów, ale...

opracował:
Maciej WASZKOWSKI

UCIĄŻLIWOŚCI PRZY OBSŁUDZE KOMPUTERA

Miliony Amerykanów spędzają przy komputerach osobistych całe dnie: w pracy i w domu. Lekarze w tym kraju zainteresowali się, jak ten spędzony przy komputerowym ekranie czas wpływa na ich zdrowie.

Niepokoi zwłaszcza sprawa — to wcale nie księżycowy problem — czy praca przy monitorach zwiększa ryzyko poronienia u ciężarnych kobiet. Kwestia ta stanęła na porządku dnia w grudniu ubiegłego roku, gdy magazyn „USA Today" poinformował o badaniach dotyczących bardzo wysokiego wskaźnika poronień wśród kobiet obsługujących komputery. Sugerują one między innymi, że winę za ten stan rzeczy ponosi promieniowanie emitowane przez displaye oraz napięty zazwyczaj harmonogram pracy, nie dający pracownikowi zbyt wiele czasu na odpoczynek.

Narodowy Instytut do Spraw Bezpieczeństwa i Higieny Pracy prowadzi nawet badania na ten temat, porównując wskaźnik poronień wśród 2 tys. operatorek telefonicznych pracujących na displayach z analogicznym wskaźnikiem 2 tys. innych, które nie mają z komputerami do czynienia.

Inne badania ujawniły, że użytkownicy terminali cierpią na różnego rodzaju bóle i zmęczenie oczu. Powoduje to jaskrawe światło emitowane przez monitor, ale także nieodpowiednie krzesło i pulpit.

Ergonomiści twierdzą, że od fizycznych reakcji organizmu na kontakt z displayami uzależnione jest samopoczucie i jakość wykonywanej pracy. I tak klawiatura, której kąt nachylenia nie jest dostosowany do nadgarstków może wywołać tak zwany syndrom przewodu nadgarstkowego, czyli schorzenie wywołane zapaleniem ścięgien nadgarstkowych. Konsekwencją jest podrażnienie nerwów. W idealnych warunkach pracy stawy kończyn powinny znajdować się w pobliżu środków przestrzeni, w jakiej się mogą poruszać, aby ograniczyć tym sposobem ruchy do minimum. Krzesła czy fotele powinny być tak dopasowane, by stopy mocno opierały się o podłogę, a łydki pozostawały niemal w pionie. Kręgosłup także powinien mieć pełne oparcie.

Klawiatura powinna być umieszczona tak, aby ramiona były pionowe, zaś przedramiona — poziome. Linia wzroku w stosunku do ekranu powinna przebiegać pod kątem 10—20 stopni poniżej linii horyzontalnej terminalu. Większy kąt, to możliwość drętwienia szyi i barku.

Osoby pracujące w domu mają łatwiejsze zadanie, gdyż same dobierają sobie właściwe meble. Pracujący w biurach, pozbawieni tej możliwości, wadliwie zaprojektowane wyposażenie mogą usprawnić przykładowo, podnóżkiem. Dobrze jest też robić kilkuminutowe przerwy podczas pracy. Gdzieniegdzie wprowadza się nawet gimnastykę.

Oczy męczą się wówczas, gdy wędrują z ekranu na kartkę papieru. Można to wyeliminować. Najprostszym sposobem jest ustawienie pulpitu w ten sposób, że papier jest na tej samej wysokości co ekran. Dobrze jest też zmniejszyć oświetlenie wnętrza o połowę w stosunku do oświetlenia wymaganego przy innych pracach biurowych. Zbyt jasne światło powoduje bowiem migotanie ekranu displaya i zmusza oczy do stałego przystosowywania się do jego stosunkowo matowej powierzchni. Służą temu specjalne filtry przeciwmigotliwe, które założone na ekran rozpraszają zbędne odblaski światła. Dobrze jest tak ustawić ekran, by nie padało na niego światło dzienne.

Czasami praca przy monitorach może wywołać zaburzenia w ostrości widzenia. Podczas kontroli lekarskiej, lekarz musi wiedzieć, ile czasu pacjent spędza przy komputerze. Osoby noszące okulary dwuogniskowe, powinny mieć specjalne okulary do pracy. Ekran winien być oddalony 40—60 cm od oczu.

Przestrzeganie podanych podstawowych reguł powoduje, że praca przy komputerze jest wydajniejsza i zwraca poniesione dla wygody i zdrowia koszty.

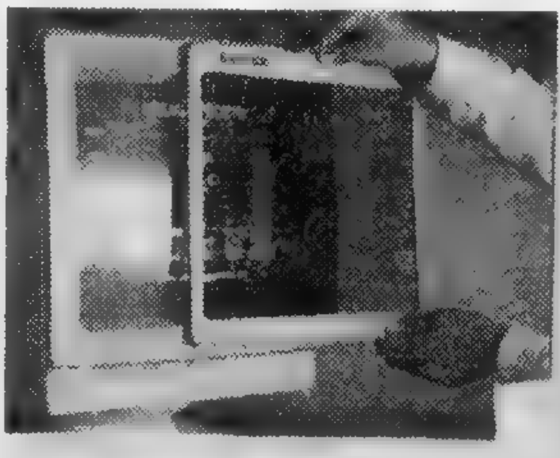
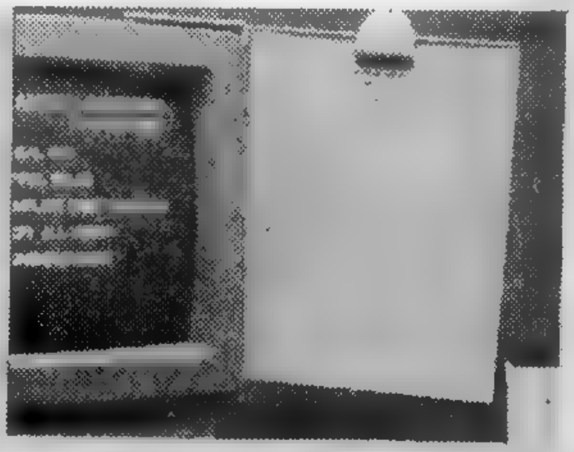
Objaśnienia do zdjęć (z cennikiem):

Zaprezentowane przedmioty mają przynieść zdecydowaną ulgę ciału przy obsłudze komputera.

1. Ten filtr nałożony na ekran komputera eliminuje migotanie. Cena tylko 69,95 dol.
2. Nośnik na dokumenty (6,95 dol.). Wygodny w obsłudze — nie trzeba kręcić szyją i „biegać oczami".
3. Dwupoziomowe biurko na obsługiwany sprzęt jest bardzo ergonomiczne. Cena, bagatelka: 599 dol.
4. Kąt pochylenia klawiatury — sprzętu stojącego na właściwym poziomie — powinien być przystosowany do natury nadgarstków.
5. Tu korpus i nogi czują się jak trzeba. Za jedyne 1200 dol. można sobie sprawić komfortowy i zdrowy anatomicznie fotel marki Girsberger (jeden z tańszych modeli).
6. Gdy głowa pracuje, stopy muszą mieć wygodę.

Komputerowy podnóżek kosztuje 105 dol.

(Na podst. US News and World (Report)
opr. J. R.





Jajo

Zasada gry jest prosta. Trzeba naprowadzić znaczek przypominający jajko na wszystkie drzewa wyświetlone na ekranie, niszcząc je, przy czym każde zniszczenie drzewa wyczerpuje energię naszego jajka. Odnawiamy ją zbierając cyfry pojawiające się na ekranie.

Świąteczny program JAJKO WELKANOCNE adresowany jest do wszystkich użytkowników komputerów, posiadających jakikolwiek kompilator języka PASCAL. Mimo że napisany jest on w języku TURBO PASCAL na komputerze IBM PC, to jednak starałem się unikać wszystkich efektownych, ale mogących sprawić kłopoty sztuczek.

Przystępując do przenoszenia tego programu na własny komputer (jeżeli dysponujemy językiem TURBO PASCAL) musimy ustalić, jakie znaki ASCII będą naśladowały grafikę. Zacznijmy od jajka. Wybieramy wśród znaków ASCII jakiś najbardziej przypominający jajko i jego numer podstawiamy pod zmienną *„głowa"*. Następnie wyszukujemy znak przypominający drzewo i jego numer podstawiamy pod stałą *„drzewo"*, po czym numer znaku zawierającego zaciemniony prostokąt pod stałą *„rama"*. Następnie pod zmienne *„ekranx"* i *„ekrany"* podstawiamy liczbę wierszy i kolumn na ekranie. Jeżeli liczby te będą różne od 80×24 będziemy musieli przeredagować procedury **OPIS** i **CZOŁÓWKA**. Reszta już powinna zadziałać prawidłowo.

Jeśli zaś program chcielibyśmy uruchomić pod innym kompilatorem języka PASCAL może zaistnieć potrzeba dopisania na początku programu procedur:

go to xy(x,y: integer) — ustawiającej kursor na ekranie w pozycji x.y;

funkcji.

random (x: integer) — podającej liczbę losową z zakresu 0 X

readkey — czytającą naciśnięty klawisz bez konieczności naciśnięcia ENTER po wprowadzeniu znaku;

round (x: real) — zaokrąglającą liczbę rzeczywistą do całkowitej.

oraz ustalenia typu okrojonego *byte* = 1...256;

może również sprawić kłopot linia:

if (a = gora) or (a = dol) or (a = lewo)...

wtedy można ją zamienić na

if (a = chr (góra))... lub if (a = 'r') or (a = 'c')...

Tak więc tym razem nie daję gotowej zabawki do wklepania, ale proponuję trochę pracy przed zabawą. Na pewno sprawi to znacznie więcej przyjemności.

Po prawidłowym przeniesieniu programu proponuję już na własną rękę urozmaicić program np. przez wprowadzenie grafiki, czy też licznika czasu itp. Konstrukcja programu jest podatna na wszelkie przeróbki. Autor pozostawia sobie jedynie prawo do nieusuwania swojego nazwiska.

Życzę miłej zabawy zarówno intelektualnej, jak i manualnej.

CEZARY DOBROWOLSKI

```
Program Jajko_Wielkanocne;
(uses crt;)   (W wersji TURBO PASCAL'a 4.0 i 5.0 usunac nawiasy komentarza)
const rama  =177;   (Numer ASCII znaku ramki ▓)
      glowa =2;     (Numer ASCII znaku wybranego na glowe weza)
      ekranx=80;    (Ilosc znakow w wierszu)
      ekrany=24;    (Ilosc linii na ekranie)
      drzewo=5;     (Numer znaku dla drzewa)
      gora  ='r';   (Klawsz poruszajacy jajko w gore)
      dol   ='c';   (Klawsz poruszajacy jajko w dol )
      lewo  ='k';   (Klawsz poruszajacy jajko w lewo)
      prawo ='l';   (Klawsz poruszajacy jajko w prawo)
var

   las:array[1..2,1..ekranx] of byte;  (Tabela polozenia drzew w lesie)
   jedzenie:array[1..3] of byte;       (Polozenie i wartosc znaku energii)
   poziom:byte;                        (Poziom gracza)
   wykarczow:byte;                     (Ilosc drzew wykarczowanych)
   moc:integer;                        (Aktualna posiadana moc)
   x,y:byte;                           (Polozenie jajka na ekranie)

Function readkey:char;
(Funkcja odczytuje znak nacisnietego klawisza odpowiegnik w TURBO PASCAL v4.0)
(Przy uruchamianiu tego programu w wersji 4.0 lub 5.0 TURBO PASCAL a procedure
nalezy usunac)
var a:char;
begin
   repeat read(kbd,a); until a<>'';
   readkey:=a;
end;

Procedure ramka(opcja:boolean);
(Procedura rysuje ramke wokol ekranu, opcja ustala czy ramka ma byc rysowana
czy tez czyszczona (opcja=false))
var i,j:byte;
begin
   gotoxy(1,1);
   for i:=1 to ekranx do     (Linia gorna )
   begin
      if opcja then write(chr(rama)) else write(' ');
   end;
   for j:=1 to ekrany-1 do   (Linia boczne)
   begin
     gotoxy(1,j);if opcja then write(chr(rama)) else write(' ');
     gotoxy(ekranx,j);if opcja then write(chr(rama)) else write(' ');
   end;
   gotoxy(1,ekrany);         (Linia dolna)
   for i:=1 to ekranx do
   begin
      if opcja then write(chr(rama)) else write(' ');
   end;
end;

Procedure opis;
(Ukazuje na ekranie opis programu)
var a:char;
begin

   clrscr;
   gotoxy(1,3);
writeln('              JAJKO WIELKANOCNE wybralo sie na wycieczke do lasu.        ');
writeln('        Wpadlo tam w zasadzke.  Wielki plac otoczony zabojczym  ogrodzeniem. ');
writeln('        Uratuje sie jesli zetnie wszystkie drzewa. Na sciecie drzewa potrze- ');
writeln('        buje energii ktora pojawia sie na ekranie w postaci cyfr rownych    ');
writeln('        wartosci energii.                                                     ');
writeln('          Jezeli zasob energii spadnie ponizej zera jajko ginie.             ');
writeln('        Niekiedy zdarza sie ze energia zostaje umieszczona na drzewie i wtedy');
writeln('        jajko musi posiadac energie wystarczajaca na sciecie drzewa i dopiero');
writeln('        wtedy mozna myslec o odnowie energii.                                 ');
writeln('        Po ekranie poruszamy sie uzywajac klawiszy:                          ');
writeln('                        k - ruch w lewo ;                                     ');
writeln('                        l - ruch w prawo;                                     ');
writeln('                        r - ruch w gore ;                                     ');
writeln('                        c - ruch w dol  ;                                     ');
writeln;
writeln('                 Z Y C Z E   M I L E J   Z A B A W Y                          ');
   ramka(true);
   repeat a:=readkey; until a<>'';

end;

Procedure tworz_las;
(Wyznacza pozycje poszczegolnych drzew i pokazuje je na ekranie)
var i:byte;
begin
   for i:=1 to ekranx do
   begin
      repeat
         las[1,i]:=random(ekranx-3)+2; las[2,i]:=random(ekrany-3)+2;
      until (las[1,i]<>2) and (las[2,i]<>2);
      gotoxy(las[1,i],las[2,i]);write(chr(drzewo));
   end;
end;

Procedure Czolowka;
(Pokazuje czolowke na ekranie)
var a:char;
    i,j:byte;                         (Zm. pomocnicze do obslugi petli)
    tab:array[1..11] of string[80];   (Tablica przechowyjaca napis czolowki)
    pom:string[80];                   (Zm pomocnicza do zamiany * na ▓)
begin
   clrscr;  ramka(true);  tworz_las;
tab[2]:='*   * *  *****  *      *  *  ***   *   *   ***    ***   *   *  *****';
tab[3]:='*   * * *       *      * *  *   *  **  *  *   *  *   *  *   *      ';
```

```
tab[4]:='# # #  #  ##     #      # #    #   # # # # #  #  #'#     # # #  ###  ';
tab[5]:='## ##  #  #       #      # #    #####  #  ## #   #  #   # #  ### #    ';
tab[6]:='#   #  #  #####  #####  #   #  #   #  #   #   ###    ###   #   #  #####';
tab[7]:='#####   ###   #####  #   #   ###  ';
tab[8]:='   #  #   #      #  # #   #   #  ';
tab[9]:=    #  #   #      #  # #   #   #  ';
tab[10]:='#   #  #####  #   #  #  #  #   #  ';
tab[11]:=' ###   #   #   ###   #  #   ###  ';
  for i:=1 to 10 do
  begin
    tab[i]:=''; pom:=tab[i+1];
    for j:=1 to length(tab[i+1]) do  (Zamienia znak # na █)
    begin
      if pom[j]='#' then tab[i]:=tab[i]+chr(rama)
      else tab[i]:=tab[i]+' ';
    end;
  end;
  for i:=1 to 5 do   (Drukuje tekst naglowka)
  begin
    gotoxy(6,3+i); writeln(tab[i]);
  end;
  for i:=6 to 10 do
  begin
    gotoxy(20,5+i); writeln(tab[i]);
  end;
  gotoxy(round((ekranx-25)/2),round(ekrany/2)+5);
  writeln('autor: CEZARY DOBROWOLSKI');
  gotoxy(5,ekrany-2);write('Wybierz poziom gry (1,2,3,4,5)');
  repeat a:=readkey; until a in ['1','2','3','4','5'];
  case a of              (Wybor poziomu gry)
    '1' : poziom:=200;
    '2' : poziom:=150;
    '3' : poziom:=100;
    '4' : poziom:=50;
    '5' : poziom:=10;
  end;
end;

Procedure wygral;
begin
  gotoxy(round((ekranx-11)/2),round(ekrany/2)-2);
  writeln('W Y G R L E S');
  gotoxy(round((ekranx-11)/2),round(ekrany/2));
  writeln('JAJKO URATOWANE');
  ramka(false);
end;

Procedure przegral;
begin
  gotoxy(round((ekranx-11)/2),round(ekrany/2)-2);
  writeln('P R Z E G R A L E S');
  gotoxy(round((ekranx-11)/2),round(ekrany/2));
  writeln('   JAJKO ZGINELO');
  gotoxy(x,y); write(' '); ramka(true);
end;

Procedure czy_jedzenie;
(Procedura sprawdza czy pobrano jedzenie a jesli tak to ustawia nowa porcje)
begin
  if (x=jedzenie[1]) and (y=jedzenie[2]) then
  begin
    moc:=moc+jedzenie[3];
    jedzenie[1]:=random(ekranx-3)+2;
    jedzenie[2]:=random(ekrany-3)+2;
  i:=1;
  repeat
    if las[1,i]=x then
    begin
      if las[2,i]=y then
      begin
        las[1,i]:=0; las[2,i]:=0;
        moc:=moc-random(20);
        wykarczow:=wykarczow+1;
        exit;        (Wyjscie w chwili wykarczowania drzewa)
      end;
    end;
    i:=i+1;
  until (i>=ekranx);   (Jajko nie znajduje sie na poz. wyst. drzewa)
end;

Procedure Inicjuj;
(Procedura ustala wartosci poczatkowe zmiennych ogolnodostepnych)
begin
  wykarczow:=0; moc:=20;  x:=2; y:=2;
  jedzenie[1]:=random(ekranx-3)+2;
  jedzenie[2]:=random(ekrany-3)+2;
  jedzenie[3]:=random(10);
end;

Procedure jajko;
(Wlasciwa procedura gry)
var old_x,old_y:byte;  (Zm. pamietajace poprzednia poz jajka)
    old_a,a:char;      (Zm. pomocn. pamietajace znak nacisnietego klawisz)
    koniec :boolean;   (Zm. warunkujaca koniec gry po dotknieciu barierki)
begin
  koniec:=false;
  gotoxy(jedzenie[1],jedzenie[2]);write(jedzenie[3]);
  gotoxy(x,y); write(chr(glowa));
  repeat a:=readkey; until a<>'';  (oczekiwanie na pocz. nacisn. klawisza)
  repeat
    old_x:=x;  old_y:=y; old_a:=a;       (opoznienie zmieniajace gre)
    delay(poziom);
    if keypressed then a:=readkey;
    czy_karczuje;
    czy_jedzenie;
    gotoxy(25,24);write('   ');
    gotoxy(10,24);write('Posiadana moc: ',moc);
    if (a=gora) or (a=dol) or (a=lewo) or (a=prawo) or (a=' ') then
    begin                                      (obsluga klawiatury)
      case a of
        gora : if y>2        then y:=y-1 else koniec:=true;
        dol  : if y<ekrany-1 then y:=y+1 else koniec:=true;
        lewo : if x>2        then x:=x-1 else koniec:=true;
        prawo: if x<ekranx-1 then x:=x+1 else koniec:=true;
      end;
    end
    else a:=old_a;
    gotoxy(old_x,old_y);
    write(' ');
    gotoxy(x,y);
    write(chr(glowa));
  until (koniec) or (moc<0) or (wykarczow>=80);
end;

BEGIN
  inicjuj;
  czolowka;
  opis;
  clrscr;
  ramka(true);
  tworz_las;
  jajko;
  if wykarczow=80 then wygral else przegral;
END.
```



KOMPUTEROWA ANALIZA UKŁADÓW ELEKTRONICZNYCH

ANALIZA ZMIENNOPRĄDOWA — AC

Analiza AC umożliwia badanie działania układów elektronicznych przy pobudzaniu sygnałem sinusoidalnym o zadanej częstotliwości *f*. Do analizy wykorzystamy metodę wskazową i liczby zespolone. Nie będę wnikał w teoretyczne podstawy tej metody, a wszystkich zainteresowanych odsyłam do podręczników fizyki opisujących obwody prądu zmiennego i metodę wskazową.

Zgodnie z prawem Ohma zależność między napięciem i natężeniem prądu zmiennego dla rezystora wynosi: $u = R \cdot i$,

dla kondensatora wynosi: $u = \frac{1}{j \cdot \omega \cdot C} \cdot i$

dla indukcyjności wynosi: $u = j \cdot \omega \cdot L \cdot i$

gdzie: u — spadek napięcia na elemencie [V]
i — prąd przepływający przez element [A]
j — liczba zespolona: $j = \sqrt{-1}$
ω — pulsacja: $\omega = 2 \cdot \pi \cdot f$ [rad/s], $\pi = 3.14$
f — częstot. [Hz]
R — rezystancja [Ω]
C — pojemność [F]
L — indukcyjność H]

· Problem można uogólnić używając *impedancji:*

dla rezystora: $u = Z_R \cdot i$, Z_R = impedancja rezystora

dla kondensatora: $u = Z_C \cdot i$, $Z_C = \frac{1}{j \cdot \omega \cdot C}$ — impedancja kondensatora

dla indukcyjności: $u = Z_L \cdot i$, $Z_L = j \cdot \omega \cdot L$ — impedancja indukcyjności

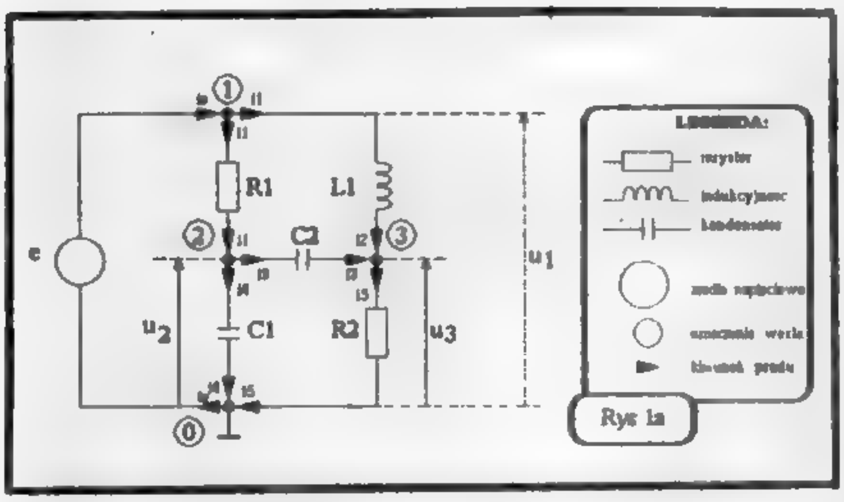
Rozważmy prosty układ przedstawiony na **rys. 1a.** W miejsce elementów podstawmy ich impedancje. Obrazuje to **rys. 1b.** Powstały schemat przypomina sieć rezystorową z analizy DC. Tak jak poprzednio interesują nas napięcia w węzłach i prądy w źródłach napięciowych. Ponieważ w każdym obwodzie działają prawa Kirchoffa, więc możemy zapisać *(analogicznie, jak w analizie DC):*

węzeł 1: $-i_e + i_1 + i_2 = 0$
węzeł 2: $-i_1 + i_3 + i_4 = 0$
węzeł 3: $-i_2 - i_3 + i_5 = 0$
źródło E. $e - u_1 = 0$

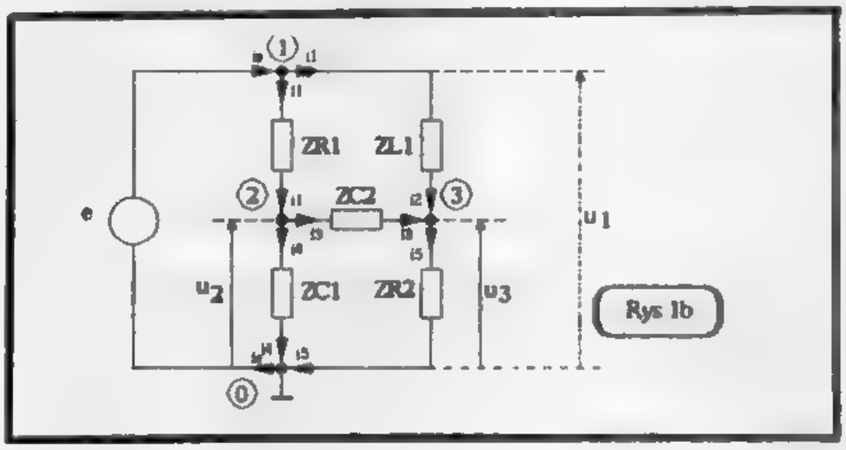
gdzie: $i_{ZR1} = i_1 = (u_1 - u_2) \cdot Y_{R1}$, $Y_{R1} = 1/Z_{R1}$
$i_{ZR2} = i_5 = u_3 \cdot Y_{R2}$, $Y_{R2} = 1/Z_{R2}$
$i_{ZC1} = i_4 = u_2 \cdot Y_{C1}$, $Y_{C1} = 1/Z_{C1}$
$i_{ZC2} = i_3 = (u_2 - u_3) \cdot Y_{C2}$, $Y_{C2} = i/Z_{C2}$
$i_{ZL1} = i_2 = (u_1 - u_3) \cdot Y_{L1}$, $Y_{L1} = i/Z_{L1}$

po podstawieniu prądów otrzymujemy:

węzeł 1: $-i_E + Y_{R1} \cdot u_1 - Y_{R2} \cdot u_2 + Y_{L1} \cdot u_1 - Y_{L1} \cdot u_3 = 0$
węzeł 2: $-Y_{R1} \cdot u_1 + Y_{R1} \cdot u_2 + Y_{C1} + u_2 - Y_{C2} \cdot u_2 - Y_{C2} \cdot u_3 = 0$
węzeł 3: $-Y_{L1} \cdot u_1 + Y_{L1} \cdot u_3 - Y_{C2} \cdot u_2 + Y_{C2} \cdot u_3 + Y_{R2} \cdot u_3 = 0$
źródło E: $e - u_1 = 0$



Rys 1a


Rys 1b

węzeł	u1	u2	u3	ie	b
1	YR1+YL1	-YR1	-YL1	-1	
2	-YR1	YR1+YC1+YC2	-YC2		
3	-YL1	-YC2	YL1+YC2+YR2		
E	-1				−e

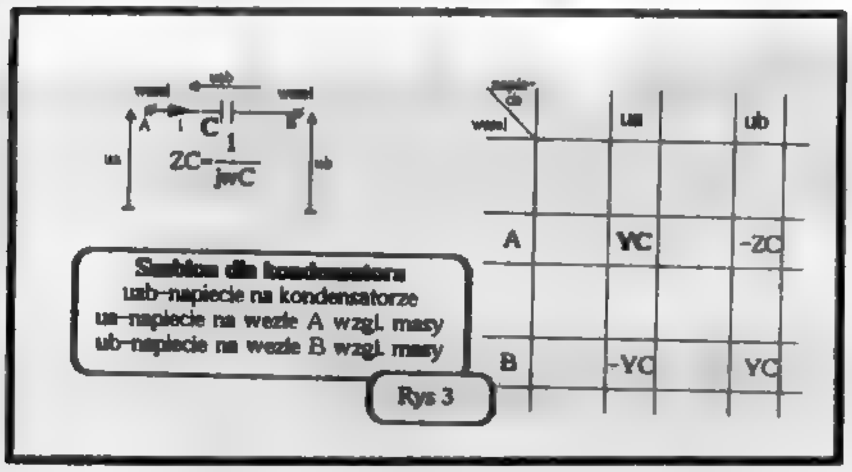
UKŁAD RÓWNAŃ
dla schematu z rysunku 1a,b
u1, u2, u3 — zmienne niewiadome
b — kolumna wyrazów wolnych
1, 2, 3 — numery węzłów
E — równanie źródła napięciowego

Rys 2

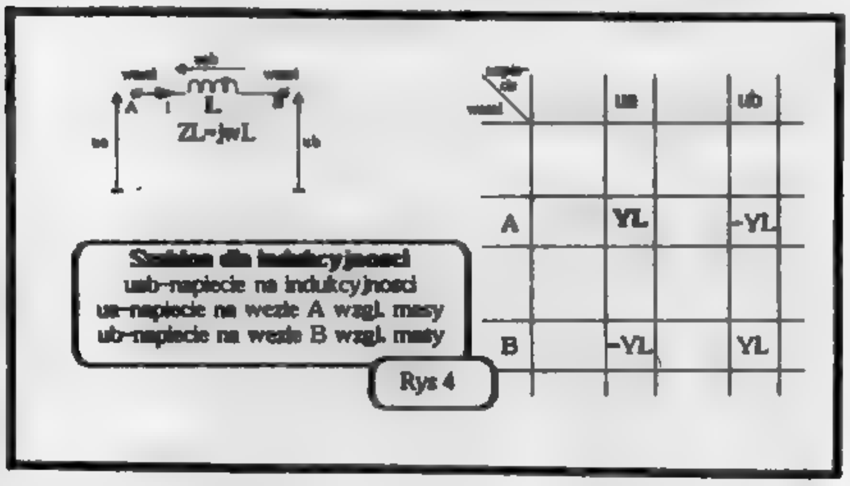
po uporządkowaniu względem napięć i prądów otrzymujemy:

węzeł 1: $(Y_{R1} + Y_{L1}) \cdot u_1 \quad -Y_{R1} \cdot u_2 \quad -Y_{L1} \cdot u_3 - i_E = 0$
węzeł 2: $-Y_{R1} \cdot u_1 + (Y_{R1} + Y_{C1} + Y_{C2}) \cdot u_2 \quad Y_{C2} \cdot u_3 = 0$
węzeł 3: $-Y_{L1} \cdot u_1 \quad -Y_{C2} \cdot u_2 + (Y_{L1} + Y_{C2} + Y_{R2}) \cdot u_3 = 0$
źródło E: $-u_1 = -e$

Tablicę admitancyjną przedstawia rys. 2.



Rys 3



Rys 4

Z przeprowadzonej analizy wynika kilka istotnych wniosków:

1) mechanizm tworzenia macierzy admitancyjnej jest taki sam, jak w wypadku analizy DC.
2) kondensatory i indukcyjności posiadają taki sam szablon, jak rezystory, z tą jednak różnicą, że ich impedancję należy obliczyć względem ustalonej częstotliwości (rys. 3, 4).
3) analiza numeryczna polega na rozwiązaniu liniowego zespolonego układu równań.

Jedyną istotną różnicą, w porównaniu z analizą DC, jest konieczność operowania na liczbach zespolonych. Musimy nauczyć komputer rozwiązywać liniowy zespolony układ równań. Oto przykład takiego równania stopnia drugiego:

$$(a_{11}+j\cdot b_{11})\cdot(x_1+j\cdot y_1)+(a_{12}+j\cdot b_{12})\cdot(x_2+j\cdot y_2)= \\ =c_1+j\cdot d_1$$

$$(a_{21}+j\cdot b_{21})\cdot(x_1+j\cdot y_1)+(a_{22}+j\cdot b_{22})\cdot(x_2+j\cdot y_2)= \\ =c_2+j\cdot d_2$$

gdzie: $a_{11},\ldots,a_{22}$ — część rzeczywista współczynników równania
$b_{11},\ldots,b_{22}$ — część urojona współczynników równania
x_1, x_2 — część rzeczywista niewiadomych
y_1, y_2 — część urojona niewiadomych
c_1, c_2 — część rzeczywista wyrazów wolnych
d_1, d_2 — część urojona wyrazów wolnych

po rozpisaniu układu i rozdzieleniu części rzeczywistej i urojonej otrzymujemy:

$$a_{11}\cdot x_1+a_{12}\cdot x_2-b_{11}\cdot y_1-b_{12}\cdot y_2+j\cdot(b_{11}\cdot x_1+b_{12}\cdot x_2+a_{11}\cdot y_1+a_{12}\cdot y_2)=c_1+j\cdot d_1$$

$$a_{21}\cdot x_1+a_{22}\cdot x_2-b_{21}\cdot y_1-b_{22}\cdot y_2+j\cdot(b_{21}\cdot x_1+b_{22}\cdot x_2+a_{21}\cdot y_1+a_{22}\cdot y_2)=c_2+j\cdot d_2$$

Ponieważ części: rzeczywista i urojona muszą spełniać równanie tożsamościowo, stąd **układ dwóch równań zespolonych możemy zastąpić układem czterech równań rzeczywistych:**

$$a_{11}\cdot x_1+a_{12}\cdot x_2-b_{11}\cdot y_1-b_{12}\cdot y_2=c_1$$
$$a_{21}\cdot x_1+a_{22}\cdot x_2-b_{21}\cdot y_1-b_{22}\cdot y_2=c_2$$
$$b_{11}\cdot x_1+b_{12}\cdot x_2+a_{11}\cdot y_1+a_{12}\cdot y_2=d_1$$
$$b_{21}\cdot x_1+b_{22}\cdot x_2+a_{21}\cdot y_1+a_{22}\cdot y_2=d_2$$

Z takim układem już sobie radzimy!

Dla układu n-równań zespolonych:

$$(a_{11}+j\cdot b_{11})\cdot(x_1+j\cdot y_1)+\ldots+(a_{1n}+j\cdot b_{1n})\cdot(x_n+j\cdot y_n)=c_1+j\cdot d_1$$
$$\vdots$$
$$(a_{n1}+j\cdot b_{n1})\cdot(x_1+j\cdot y_1)+\ldots+(a_{nn}+j\cdot b_{nn})\cdot(x_n+j\cdot y_n)=c_n+j\cdot d_n$$

uzyskujemy 2n-równań rzeczywistych:

$$a_{11}\cdot x_1+\ldots+a_{1n}\cdot x_n-b_{11}\cdot y_1-\ldots-b_{1n}\cdot y_n=c_1$$
$$\vdots$$
$$a_{n1}\cdot x_n+\ldots+a_{nn}\cdot x_n-b_{n1}\cdot y_n-\ldots-b_{nn}\cdot y_n=c_n$$
$$b_{11}\cdot x_1+\ldots+b_{1n}\cdot x_n+a_{11}\cdot y_1+\ldots+a_{1n}\cdot y_n=d_1$$
$$\vdots$$
$$b_{n1}\cdot x_1+\ldots+b_{nn}\cdot x_n+a_{n1}\cdot y_1+\ldots+a_{nn}\cdot y_n=d_n$$

Pora przypomnieć sobie parę użytecznych wzorów:
$k=a+j\cdot b$, k — liczba zespolona
a — część rzeczywista
b — część urojona

$|k|=\sqrt{a\cdot a+b\cdot b}$ — moduł liczby zespolonej k
$\varphi=agr(k)=arc\ tg\left(\frac{b}{a}\right)$ — argument liczby zespolonej k

Ponieważ rozwiązaniem układu równań są części rzeczywista i urojona napięć w węzłach i prądów przepływających przez źródła napięciowe, stąd na podstawie powyższych wzorów otrzymujemy:

dla napięć: moduł napięcia — napięcie mierzone przez woltomierz
argument napięcia — przesunięcie fazowe napięcia

dla prądów: moduł prądu — prąd mierzony przez amperomierz
argument prądu — przesunięcie fazowe prądu

Układ wprowadza się identycznie, jak dla analizy DC. Po uruchomieniu analizy numerycznej (klawisz <w>) należy podać częstotliwość, dla której analiza ma zostać przeprowadzona

WYJAŚNIENIE: dla tranzystora podajemy stały prąd jaki przepływa przez kolektor, czyli I_C, obliczony podczas analizy DC.

Krzysztof POŻNIAK

```
    1 REM
*********************************
*                               *
*                        ©1989  *
*                               *
*        Krzysztof Pozniak      *
*                               *
* Analiza AC ukladow elektron.  *
*                               *
*********************************
   10 DEF FN z(p,k,z)=(z<p) OR (z
>k) OR (z<>INT z)
   11 DEF FN u(k)=t(2,e(x,l,k))
   13 DEF FN p(k)=t(2,k+maxw-1)
   14 DEF FN m(r,i)=SQR (r*r+i*i)
   15 DEF FN a(r,i)=180*SGN (i+(i
=0))*ACS ((SGN (r+(r=0))*SQR ((r
*r+((r=0) AND (i=0)))/(r*r+i*i+(
(r=0) AND (i=0))))))/PI
  100 REM inicjalizacja ukladu
  105 PAPER 0: INK 7: BORDER 0: C
LS
  110 LET Ybe=1/2500: LET Ut=0.02
5: LET maxw=10: LET b=maxw+1: LE
T maxn=2*maxw
  120 GO SUB 7000
  130 GO SUB 7400
  140 GO TO 9000
 1000 REM rozklad Gaussa
 1010 LET n=2*n: FOR k=1 TO n: LE
```

```
T x(k)=k: NEXT k: FOR k=1 TO n:
LET m=0: FOR y=k TO n: FOR x=k T
O n: IF ABS n(y,x)>m THEN LET m=
ABS n(y,x): LET my=y: LET mx=x
1020 NEXT x: NEXT y: IF m=0 THEN
 PRINT '"uklad nie ma rozwiazani
a ": RETURN
1030 FOR l=1 TO n: LET m=n(k,l):
 LET n(k,l)=n(my,l): LET n(my,l)
=m: NEXT l: LET m=b(k): LET b(k)
=b(my): LET b(my)=m: FOR l=1 TO
n: LET m=n(l,k): LET n(l,k)=n(l,
mx): LET n(l,mx)=m: NEXT l: LET
m=x(k): LET x(k)=x(mx): LET x(mx
)=m
1040 FOR l=k+1 TO n: LET m=n(l,k
)/n(k,k): FOR j=k+1 TO n: LET n(
l,j)=n(l,j)-n(k,j)*m: NEXT j: LE
T b(l)=b(l)-b(k)*m: NEXT l: NEXT
 k: FOR k=n TO 1 STEP -1: LET m=
0: FOR l=k+1 TO n: LET m=m+t(1,x
(l))*n(k,l): NEXT l: LET t(1,x(k
))=(b(k)-m)/n(k,k): NEXT k: LET
m=1: PRINT '"analiza numeryczna
zakonczona": RETURN
5000 REM Stale zrodla pradowe
5001 REM uwagi i bledy
5005 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte zrodlo SZP";e
(x,l,1)
5006 IF e(x,l,4)=0 THEN PRINT "u
waga-zerowa wyd. pradowa SZP";e(
x,l,1)
5010 REM tworzenie macierzy
5015 LET k=3: GO SUB 6100: IF st
op=1 THEN RETURN
5020 IF e(x,l,2)>0 THEN LET r(FN
 u(2),b)=r(FN u(2),b)+e(x,l,4)
5021 IF e(x,l,3)>0 THEN LET r(FN
 u(3),b)=r(FN u(3),b)-e(x,l,4)
5022 RETURN
5030 REM wizualizacja wynikow
5035 PRINT ' INVERSE 1;"SZP";e(x
,l,1); INVERSE 0;TAB 6;"U+=";e(x
,l,2);TAB 12;"U-=";e(x,l,3)
5036 LET m1=2: LET m2=3: GO SUB
6400: LET mod=FN m(mr,mi): LET a
rg=FN a(mr,mi): PRINT 'TAB 6;"Re
U = ";mr;" V";TAB 6;"ImU = ";mi;
" V";TAB 6;"Usk = ";mod;" V";TAB
 6;"arU = ";arg;" stop";TAB 6;"I
sk = ";e(x,l,4);" A";TAB 6;"P =
";e(x,l,4)*mr*mr;" W"
5040 RETURN
5100 REM Stale zrodla napieciowe
5105 IF e(x,l,4)=0 THEN PRINT "u
waga-zerowa wartosc nap. SZN";e(
x,l,1)
5106 IF e(x,l,2)=e(x,l,3) THEN P
RINT INVERSE 1;"blad-zwarte zrod
lo SZN";e(x,l,1): LET blad=1: RE
TURN
5110 REM tworzenie macierzy
5115 LET k=3: GO SUB 6100: LET k
=1: GO SUB 6200: IF stop=1 THEN
RETURN
5120 IF e(x,l,2)>0 THEN LET r(FN
 u(2),FN p(lp))=r(FN u(2),FN p(l
p))+1: LET r(FN p(lp),FN u(2))=r
(FN p(lp),FN u(2))-1
5121 IF e(x,l,3)>0 THEN LET r(FN
 u(3),FN p(lp))=r(FN u(3),FN p(l
p))-1: LET r(FN p(lp),FN u(3))=r
(FN p(lp),FN u(3))+1
5122 LET r(FN p(lp),b)=r(FN p(lp
),b)-e(x,l,4)
5125 RETURN
5130 REM wizualizacja wynikow
5135 PRINT ' INVERSE 1;"SZN";e(x
,l,1); INVERSE 0;TAB 6;"U+=";e(x
,l,2);TAB 12;"U-=";e(x,l,3)
5136 LET m1=2: LET m2=3: GO SUB
6400: LET re=-t(1,FN p(lp)): LET
im=-t(1,n+FN p(lp)): LET mod=FN
m(re,im): PRINT 'TAB 6;"Usk = "
;mr;" V";TAB 6;"ReI = ";re;" A";
TAB 6;"ImI = ";im;" A";TAB 6;"Is
k = ";mod;" A";TAB 6;"Psk = ";mr
*re;" W"
5140 LET lp=lp+1: RETURN
5200 REM zr. pradowe ster. nap.
5201 REM uwagi i bledy
5205 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
PSN";e(x,l,1)
5206 IF e(x,l,4)=e(x,l,5) THEN P
RINT "uwaga-zwarte wezly prad. Z
PSN";e(x,l,1)
5207 IF e(x,l,6)=0 THEN PRINT IN
VERSE 1;"Blad-zerowy wspolczynni
k ZPSN";e(x,l,1): LET stop=1: RE
TURN
5210 REM tworzenie macierzy
5215 LET k=5: GO SUB 6100: IF st
op=1 THEN RETURN
5220 IF e(x,l,2)>0 AND e(x,l,4)>
0 THEN LET r(FN u(4),FN u(2))=r(
FN u(4),FN u(2))-e(x,l,6)
5221 IF e(x,l,2)>0 AND e(x,l,5)>
0 THEN LET r(FN u(5),FN u(2))=r(
FN u(5),FN u(2))+e(x,l,6)
5222 IF e(x,l,3)>0 AND e(x,l,4)>
0 THEN LET r(FN u(4),FN u(3))=r(
FN u(4),FN u(3))+e(x,l,6)
5223 IF e(x,l,3)>0 AND e(x,l,5)>
0 THEN LET r(FN u(5),FN u(3))=r(
FN u(5),FN u(3))-e(x,l,6)
5225 RETURN
5230 REM wizualizacja wynikow
5235 PRINT ' INVERSE 1;"ZPSN";e(
x,l,1); INVERSE 0;TAB 6;"S+=";e(
x,l,2);TAB 12;"S-=";e(x,l,3);TAB
 18;"I+=";e(x,l,4);TAB 24;"I-=";
e(x,l,5)
5236 LET m1=2: LET m2=3: GO SUB
6400: LET mod=FN m(mr,mi): LET u
r=mr: LET ui=mi: LET m1=4: LET m
2=5: GO SUB 6400: PRINT 'TAB 7;"
Yr = ";e(x,l,6);" S";TAB 5;"ReUs
 = ";ur;" V";TAB 5;"ImUs = ";ui;
" V";TAB 5;"skUs = ";mod;" V";TA
B 5;"arUs = ";FN a(ur,ui);" stop
";TAB 6;"reU = ";mr;" V";TAB 6;"
imU = ";mi;TAB 6;"Usk = ";FN m(m
r,mi);" V";TAB 6;"arU = ";FN a(m
r,mi);TAB 6;"Isk = ";mod*e(x,l,6
);" A": PRINT TAB 8;"P = ";mod*e
(x,l,6)*mr;" W"
5240 RETURN
5300 REM zr. pradowe ster. prad.
5301 REM uwagi i bledy
5305 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
PSP";e(x,l,1)
5306 IF e(x,l,4)=e(x,l,5) THEN P
RINT "uwaga-zwarte wezly prad. Z
PSP";e(x,l,1)
5307 IF e(x,l,6)=0 THEN PRINT IN
VERSE 1;"Blad-zerowy wspolczynni
k ZPSP";e(x,l,1): LET stop=1: RE
TURN
5310 REM tworzenie macierzy
5315 LET k=5: GO SUB 6100: LET k
=1: GO SUB 6200: IF stop=1 THEN
RETURN
5320 IF e(x,l,2)>0 THEN LET r(FN
 u(2),FN p(lp))=r(FN u(2),FN p(l
p))+1: LET r(FN p(lp),FN u(2))=r
(FN p(lp),FN u(2))-1
5321 IF e(x,l,3)>0 THEN LET r(FN
 u(3),FN p(lp))=r(FN u(3),FN p(l
p))-1: LET r(FN p(lp),FN u(3))=r
(FN p(lp),FN u(3))+1
5322 IF e(x,l,4)>0 THEN LET r(FN
 u(4),FN p(lp))=r(FN u(4),FN p(l
p))-e(x,l,6)
5323 IF e(x,l,5)>0 THEN LET r(FN
```

```
u(5),FN p(lp))=r(FN u(5),FN p(l
p))+e(x,l,6)
5325 RETURN
5330 REM wizualizacja wynikow
5335 PRINT ' INVERSE 1;"ZPSP";e(
x,l,1); INVERSE 0;TAB 6;"S+=";e(
x,l,2);TAB 12;"S-=";e(x,l,3);TAB
 18;"I+=";e(x,l,4);TAB 24;"I-=";
e(x,l,5)
5336 LET ir=t(1,FN p(lp)): LET i
i=t(1,n+FN p(lp)): LET mod=FN m(
ir,ii): LET m1=4: LET m2=5: GO S
UB 6400: PRINT 'TAB 8;"k = ";e(x
,l,6);" []";TAB 5;"ReIs = ";ir;"
 A";TAB 5;"ImIs = ";ii;" A";TAB
5;"skIs = ";mod;" A";TAB 5;"arIs
 = ";FN a(ir,ii);" stop";TAB 6;"
reU = ";mr;" V";TAB 6;"imU = ";m
i;TAB 6;"Usk = ";FN m(mr,mi);" V
";TAB 6;"arU = ";FN a(mr,mi);TAB
 6;"Isk = ";mod*e(x,l,6);" A": P
RINT TAB 8;"P = ";mod*e(x,l,6)*m
r;" W"
5340 LET lp=lp+1: RETURN
5400 REM zr. napiec. ster. nap.
5401 REM uwagi i bledy
5405 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
NSN";e(x,l,1)
5406 IF e(x,l,6)=0 THEN PRINT "U
waga-zerowy wspolcz. ZNSN";e(x,l
,1)
5407 IF e(x,l,4)=e(x,l,5) THEN P
RINT INVERSE 1;"blad-zwarte wezl
y nap. ZNSN";e(x,l,1): LET stop=
1: RETURN
5410 REM tworzenie macierzy
5415 LET k=5: GO SUB 6100: LET k
=1: GO SUB 6200: IF stop=1 THEN
RETURN
5420 IF e(x,l,2)>0 THEN LET r(FN
 p(lp),FN u(2))=r(FN p(lp),FN u(
2))-e(x,l,6)
5421 IF e(x,l,3)>0 THEN LET r(FN
 p(lp),FN u(3))=r(FN p(lp),FN u(
3))+e(x,l,6)
5422 IF e(x,l,4)>0 THEN LET r(FN
 u(4),FN p(lp))=r(FN u(4),FN p(l
p))-1: LET r(FN p(lp),FN u(4))=r
(FN p(lp),FN u(4))+1
5423 IF e(x,l,5)>0 THEN LET r(FN
 u(5),FN p(lp))=r(FN u(5),FN p(l
p))+1: LET r(FN p(lp),FN u(5))=r
(FN p(lp),FN u(5))-1
5425 RETURN
5430 REM wizualizacja wynikow
5435 PRINT ' INVERSE 1;"ZPSP";e(
x,l,1); INVERSE 0;TAB 6;"S+=";e(
x,l,2);TAB 12;"S-=";e(x,l,3);TAB
 18;"I+=";e(x,l,4);TAB 24;"I-=";
e(x,l,5)
5436 LET m1=2: LET m2=3: GO SUB
6400: LET mod=FN m(mr,mi): LET i
r=t(1,FN p(lp)): LET ii=t(1,n+FN
 p(lp)): PRINT 'TAB 7;"kr = ";e(
x,l,6);" []";TAB 5;"ReUs = ";ur;
" V";TAB 5;"ImUs = ";ui;" V";TAB
 5;"skUs = ";mod;" V";TAB 5;"arU
s = ";FN a(ur,ui);" stop";TAB 6;
"reI = ";ir;" A";TAB 6;"imI = ";
ii;TAB 6;"Isk = ";FN m(ir,ii);"
V";TAB 6;"arI = ";FN a(ir,ii);TA
B 6;"Usk = ";mod*e(x,l,6);" V":
PRINT TAB 8;"P = ";mod*e(x,l,6)*
ir;" W"
5440 LET lp=lp+1: RETURN
5500 REM zr. napiec. ster. prad.
5501 REM uwagi i bledy
5505 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly ster. Z
NSP";e(x,l,1)
5506 IF e(x,l,6)=0 THEN PRINT "U
waga-zerowy wspolcz. ZNSP";e(x,l
,1)
5507 IF e(x,l,4)=e(x,l,5) THEN P
RINT INVERSE 1;"blad-zwarte wezl
y nap. ZNSP";e(x,l,1): LET stop=
1: RETURN
5510 REM tworzenie macierzy
5515 LET k=5: GO SUB 6100: LET k
=2: GO SUB 6200: IF stop=1 THEN
RETURN
5520 IF e(x,l,2)>0 THEN LET r(FN
 u(2),FN p(lp-1))=r(FN u(2),FN p
(lp-1))+1: LET r(FN p(lp-1),FN u
(2))=r(FN p(lp-1),FN u(2))-1
5521 IF e(x,l,3)>0 THEN LET r(FN
 u(3),FN p(lp-1))=r(FN u(3),FN p
(lp-1))-1: LET r(FN p(lp-1),FN u
(3))=r(FN p(lp-1),FN u(3))+1
5522 IF e(x,l,4)>0 THEN LET r(FN
 u(4),FN p(lp))=r(FN u(4),FN p(l
p))+1: LET r(FN p(lp),FN u(4))=r
(FN p(lp),FN u(4))-1
5523 IF e(x,l,5)>0 THEN LET r(FN
 u(5),FN p(lp))=r(FN u(5),FN p(l
p))-1: LET r(FN p(lp),FN u(5))=r
(FN p(lp),FN u(5))+1
5524 LET r(FN p(lp),FN p(lp-1))=
r(FN p(lp),FN p(lp-1))+e(x,l,6)
5525 RETURN
5530 REM wizualizacja wynikow
5535 PRINT ' INVERSE 1;"ZNSP";e(
x,l,1); INVERSE 0;TAB 6;"S+=";e(
x,l,2);TAB 12;"S-=";e(x,l,3);TAB
 18;"U+=";e(x,l,4);TAB 24;"U-=";
e(x,l,5)
5536 LET ir=t(1,FN p(lp)): LET i
i=FN t(1,n+FN p(lp)): LET mod=FN
 m(ir,ii): LET mr=FN t(1,FN p(lp
+1)): LET mi=FN t(1,n+FN p(lp+1)
): PRINT 'TAB 7;"Rw = ";e(x,l,6)
;" Om";TAB 5;"ReIs = ";ir;" A";T
AB 5;"ImIs = ";ii;" A";TAB 5;"sk
Is = ";mod;" A";TAB 5;"arIs = ";
FN a(ir,ii);" stop";TAB 6;"reI =
 ";mr;" A";TAB 6;"imI = ";mi;" A
";TAB 6;"Isk = ";FN m(mr,mi);" A
";TAB 6;"arI = ";FN a(mr,mi);" s
top";TAB 6;"Usk = ";mod*e(x,l,6)
;" V": PRINT TAB 8;"P = ";mod*e(
x,l,6)*mr;" W"
5540 LET lp=lp+2: RETURN
5600 REM Rezystor
5601 REM uwagi i bledy
5605 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarty rezystor R";e
(x,l,1)
5606 IF e(x,l,4)=0 THEN PRINT IN
VERSE 1;"Blad-zerowa rezystancja
 R";e(x,l,1): LET stop=1: RETURN
5610 REM tworzenie macierzy
5615 LET k=3: GO SUB 6100
5620 IF e(x,l,2)>0 THEN LET r(FN
 u(2),FN u(2))=r(FN u(2),FN u(2)
)+1/e(x,l,4): IF e(x,l,3)>0 THEN
 LET r(FN u(2),FN u(3))=r(FN u(2
),FN u(3))-1/e(x,l,4)
5621 IF e(x,l,3)>0 THEN LET r(FN
 u(3),FN u(3))=r(FN u(3),FN u(3)
)+1/e(x,l,4): IF e(x,l,2)>0 THEN
 LET r(FN u(3),FN u(2))=r(FN u(3
),FN u(2))-1/e(x,l,4)
5625 RETURN
5630 REM wizualizacja wynikow
5635 PRINT ' INVERSE 1;"R";e(x,l
,1); INVERSE 0;TAB 6;"U+=";e(x,l
,2);TAB 12;"U-=";e(x,l,3)
5636 LET m1=2: LET m2=3: GO SUB
6400: LET mod=FN m(mr,mi): PRINT
'TAB 6;"R   = ";e(x,l,4);" Om";
TAB 6;"Usk = ";mod;" V";TAB 6;TA
B 6;"Isk = ";mod/e(x,l,4);" A":
PRINT TAB 6;"Psk = ";mod*mod/e(x
,l,4);" W"
5640 RETURN
```

```
5700 REM Kondensator
5701 REM uwagi i bledy
5705 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarty kondensator C
";e(x,l,1)
5706 IF e(x,l,4)=0 THEN PRINT IN
VERSE 1;"Blad-zerowa pojemnosc C
";e(x,l,1): LET stop=1: RETURN
5710 REM tworzenie macierzy

5715 LET k=3: GO SUB 6100
5720 IF e(x,l,2)>0 THEN LET u(FN
 u(2),FN u(2))=u(FN u(2),FN u(2)
)+w*e(x,l,4): IF e(x,l,3)>0 THEN
 LET u(FN u(2),FN u(3))=u(FN u(2
),FN u(3))-w*e(x,l,4)
5721 IF e(x,l,3)>0 THEN LET u(FN
 u(3),FN u(3))=u(FN u(3),FN u(3)
)+w*e(x,l,4): IF e(x,l,2)>0 THEN
 LET u(FN u(3),FN u(2))=u(FN u(3
),FN u(2))-w*e(x,l,4)
5725 RETURN
5730 REM wizualizacja wynikow
5735 PRINT ' INVERSE 1;"C";e(x,l
,1); INVERSE 0;TAB 6;"U+=";e(x,l
,2);TAB 12;"U-=";e(x,l,3)
5736 LET m1=2: LET m2=3: GO SUB
6400: LET mod=FN m(mr,mi): PRINT
'TAB 6;"C  = ";e(x,l,4);" F";T
AB 6;"Usk = ";mod;" V";TAB 6;"Is
k = ";mod*e(x,l,4)*w;" A"
5740 RETURN

5800 REM Indukcyjnosc
5801 REM uwagi i bledy
5805 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarta indukcyjnosc
L";e(x,l,1)
5806 IF e(x,l,4)=0 THEN PRINT IN
VERSE 1;"Uwaga-zerowa indukcyjno
sc L";e(x,l,1): LET stop=1: RETU
RN
5810 REM tworzenie macierzy
5815 LET k=3: GO SUB 6100
5820 IF e(x,l,2)>0 THEN LET u(FN
 u(2),FN u(2))=u(FN u(2),FN u(2)
)-1/w/e(x,l,4): IF e(x,l,3)>0 TH
EN LET u(FN u(2),FN u(3))=u(FN u
(2),FN u(3))+1/w/e(x,l,4)
5821 IF e(x,l,3)>0 THEN LET u(FN
 u(3),FN u(3))=u(FN u(3),FN u(3)
)-1/w/e(x,l,4): IF e(x,l,2)>0 TH
EN LET u(FN u(3),FN u(2))=u(FN u
(3),FN u(2))+1/w/e(x,l,4)
5825 RETURN

5830 REM wizualizacja wynikow
5835 PRINT ' INVERSE 1;"L";e(x,l
,1); INVERSE 0;TAB 6;"U+=";e(x,l
,2);TAB 12;"U-=";e(x,l,3)
5636 LET m1=2: LET m2=3: GO SUB
6400: LET mod=FN m(mr,mi): PRINT
'TAB 6;"L  = ";e(x,l,4);" H";T
AB 6;"Usk = ";mod;" V";TAB 6;"Is
k = ";mod*e(x,l,4)*w;" A"
5840 RETURN
5900 REM Tranzystor
5905 IF e(x,l,2)=e(x,l,3) THEN P
RINT "uwaga-zwarte wezly B-E T";
e(x,l,1)
5906 IF e(x,l,3)=e(x,l,4) THEN P
RINT "uwaga-zwarte wezly K-E T";
e(x,l,1)
5907 IF e(x,l,2)=e(x,l,4) THEN P
RINT "uwaga-zwarte wezly B-K T";
e(x,l,1)
5908 IF e(x,l,5)=0 THEN PRINT IN
VERSE 1;"Uwaga-zerowe wzmocnieni


e T";e(x,l,1): LET stop=1: RETUR
N
5910 REM tworzenie macierzy
5915 LET k=4: GO SUB 6100
5920 IF e(x,l,2)>0 THEN LET r(FN
 u(2),FN u(2))=r(FN u(2),FN u(2)
)+Ybe: IF e(x,l,4)>0 THEN LET r(
FN u(2),FN u(4))=r(FN u(2),FN u(
4))-Ybe
5921 IF e(x,l,4)>0 THEN LET r(FN
 u(4),FN u(4))=r(FN u(4),FN u(4)
)+Ybe+e(x,l,5)/Ut: IF e(x,l,2)>0
 THEN LET r(FN u(4),FN u(2))=r(F
N u(4),FN u(2))-Ybe-e(x,l,5)/Ut
5922 IF e(x,l,3)>0 AND e(x,l,4)>
0 THEN LET r(FN u(3),FN u(4))=r(
FN u(3),FN u(4))-e(x,l,5)/Ut
5923 IF e(x,l,2)>0 AND e(x,l,3)>
0 THEN LET r(FN u(3),FN u(2))=r(
FN u(3),FN u(2))+e(x,l,5)/Ut
5925 RETURN
5930 REM wizualizacja wynikow

5935 PRINT ' INVERSE 1;"T";e(x,l
,1); INVERSE 0;TAB 6;" B=";e(x,l
,2);TAB 12;" K=";e(x,l,3);TAB 18
;" E=";e(x,l,4)
5936 LET m1=2: LET m2=4: GO SUB
6400: LET ur=mr: LET m1=3: LET m
2=4: GO SUB 6400: PRINT 'TAB 7;"
Ik = ";e(x,l,5);" [A]";TAB 6;"Rb
e = ";1/Ybe;TAB 4;"skUbe = ";ur;
TAB 4;"skIbe = ";ur*Ybe;TAB 4;"R
eUce = ";mr;" V";TAB 4;"ImUce =
";mi;TAB 4;"skUce = ";FN m(mr,mi
);" V";TAB 4;"arUce = ";FN a(mr,
mi);" stop";TAB 5;"skIc = ";e(x,
l,5)*ur*Ybe/Ut;" A";TAB 5;"skIe
= ";ur*Ybe*(1+e(x,l,5)/Ut);TAB 6
;"Psk = ";ur*(ur*Ybe+e(x,l,5)*mr
/Ut);" W"
5940 RETURN
6000 REM edytor zbioru
6010 CLS
6020 GO SUB 8000+100*(x-1)
6030 GO SUB 9800
6040 GO SUB 9990
6050 IF x$="p" AND i(x)>0 THEN L
ET k=0: GO SUB 9600: IF e>0 THEN
 GO TO 6070
6051 IF x$="d" AND i(x)<10 THEN
LET i(x)=i(x)+1: LET e=i(x): GO
TO 6070
6052 IF x$="s" AND i(x)>0 THEN G
O SUB 9600: IF e>0 THEN GO SUB 8
050+100*(x-1): GO TO 6000
6053 IF x$="k" THEN RETURN
6060 BEEP .1,20: GO TO 6040
6070 GO SUB 8030+100*(x-1): GO T
O 6000
6100 REM lokowanie wezlow
6105 IF lz+k>2*maxw THEN LET sto
p=1: PRINT "blad-przekroczono ro
zmiar macierzy": RETURN
6110 FOR j=2 TO k: IF e(x,l,j)>0
 THEN IF t(2,e(x,l,j))=0 THEN LE
T t(2,e(x,l,j))=lz: LET lz=lz+1

6115 NEXT j: RETURN
6200 REM lokowanie pradow
6205 IF lp+k>maxw THEN LET stop=
1: PRINT "blad-przekroczono ilos
c pradow": RETURN
6210 FOR j=1 TO k: LET t(2,lp+ma
xw)=lz: LET lp=lp+1: LET lz=lz+1
: NEXT j: RETURN
6300 REM tworzenie macierzy
6310 CLS : INPUT "podaj czestotl
iwosc w [Hz]="'"F=";f: PRINT "cz
estotliwosc F[Hz]=";f;"Hz": IF f
<=0 THEN GO TO 6310
6315 LET w=2*PI*f
6320 PRINT : GO SUB 7400: LET st
op=0: LET lz=1: LET lp=1: PRINT
"analiza ukladu rozpoczeta"''
6350 FOR x=1 TO 10: FOR l=1 TO i
(x): GO SUB 5000+100*(x-1): NEXT
 l: NEXT x: IF stop=1 THEN PRINT
 '"bledy w ukladzie-koniec anali
zy": GO TO 6395
```

```
6351 IF lz=1 THEN PRINT "brak uk
ladu - koniec analizy": GO TO 63
95
6352 PRINT '"analiza ukladu zako
nczona"
6358 GO SUB 7500: LET n=lz-1: FO
R k=1 TO n: FOR l=1 TO n: LET n(
k,l)=r(k,l): LET n(n+k,n+l)=r(k,
l): LET n(k,n+l)=-u(k,l): LET n(
n+k,l)=u(k,l): NEXT l: LET b(k)=
r(k,b): LET b(n+k)=u(k,b): NEXT
k
6359 PRINT '"analiza numeryczna
rozpoczeta"
6360 GO SUB 1000: IF m=0 THEN GO
 TO 6395
6361 LET n=n/2
6362 PRINT '"napiecia w wezlach
ukladu:": FOR x=1 TO maxn/2: IF
t(2,x)=0 THEN GO TO 6364
6363 PRINT : LET re=t(1,t(2,x)):
 LET im=t(1,n+t(2,x)): PRINT INV
ERSE 1;"wezel ";x; INVERSE 0'"Re
 U";x;"  = ";re;" V"'"Im U";x;"
 = ";im;" V"'"Usk";x;"   = ";FN
m(re,im);" V"'"arg(U";x;") = ";FN
a(re,im);" stop"
6364 NEXT x
6365 LET lp=2: FOR x=1 TO 10: FO
R l=1 TO i(x): GO SUB 5030+100*(
x-1): NEXT l: NEXT x
6370 PRINT #0;"czy powtorzyc wyn
iki? (t/n)": PAUSE 1: PAUSE 0: I
NPUT "": IF INKEY$="t" THEN GO T
O 6362
6375 RETURN
6395 PRINT #0;"nacisnij dowolny
klawisz": PAUSE 1: PAUSE 0: INPU
T "": RETURN
6400 REM obliczanie napiecia
6410 LET xr1=0: LET xi1=0: IF e(
x,l,m1)>0 THEN LET xr1=t(1,FN u(
m1)): LET xi1=t(1,n+FN u(m1))
6420 LET xr2=0: LET xi2=0: IF e(
x,l,m2)>0 THEN LET xr2=t(1,FN u(
m2)): LET xi2=t(1,n+FN u(m2))
6430 LET mr=xr1-xr2: LET mi=xi1-
xi2: RETURN
7000 REM inicjal. zbioru elemen.
7010 DIM e(10,maxw,7): DIM i(10)
7020 RETURN
7400 REM inicjalizacja macierzy
7410 DIM t(2,4*maxw): DIM r(maxw
,b): DIM u(maxn,b)
7420 RETURN
7500 REM inicjalizacja tablicy
7510 DIM n(2*maxn,2*maxn): DIM b
(2*maxn): DIM x(2*maxn)
7520 RETURN
8000 REM Stale zrodla pradowe
8010 REM lista
8015 PRINT "Stale Zrodla Pradowe
": PRINT
8020 PRINT "nr element I+ I- Isk
[A]"
8025 LET x$="NZP": GO SUB 9110:
RETURN
8030 REM edycja elementu
8040 LET k=0: GO SUB 9500
8041 LET x$="nr wezla I+": GO SU
B 9500
8042 LET x$="nr wezla I-": GO SU
B 9500
8043 LET x$="prad zrodla w [A]":
 GO SUB 9550
8045 RETURN
8050 REM kasowanie elementu
8055 LET l=4
8060 GO SUB 9400
8065 RETURN
8100 REM Stale zrodla napieciowe
8110 REM lista
8115 PRINT "Stale Zrodla Napieci
owe": PRINT
8120 PRINT "nr element U+ U- Usk
[V]"
8125 LET x$="NZN": GO SUB 9110:
RETURN
8130 REM edycja elementu
8140 LET k=0: GO SUB 9500
8141 LET x$="nr wezla U+": GO SU
B 9500
8142 LET x$="nr wezla U-": GO SU
B 9500
8143 LET x$="napiecie zrodla w [
V]": GO SUB 9550
8145 RETURN
8150 REM kasowanie elementu
8155 LET l=4
8160 GO SUB 9400
8165 RETURN
8200 REM Zr. Pradowe Ster Nap.
8210 REM lista
8215 PRINT "Zrodla Pradowe Ster.
 Napieciowo": PRINT
8220 PRINT "nr element S+ S- U+
U- Uy[S]"
8225 LET x$="ZPSN": GO SUB 9130:
 RETURN
8230 REM edycja elementu
8240 LET k=0: GO SUB 9500
8241 LET x$="nr wezla S+": GO SU
B 9500
8242 LET x$="nr wezla S-": GO SU
B 9500
8243 LET x$="nr wezla U+": GO SU
B 9500
8244 LET x$="nr wezla U-": GO SU
B 9500
8245 LET x$="wspolczynnik Y[Om]"
: GO SUB 9550
8246 RETURN
8250 REM kasowanie elementu
8255 LET l=6
8260 GO SUB 9400
8265 RETURN
8300 REM Zr. Pradowe Ster Prad.
8310 REM lista
8315 PRINT "Zrodla Pradowe Ster.
 Pradowo": PRINT
8320 PRINT "nr element S+ S- I+
I- Uk[]"
8325 LET x$="ZPSP": GO SUB 9130:
 RETURN
8330 REM edycja elementu
8340 LET k=0: GO SUB 9500
8341 LET x$="nr wezla S+": GO SU
B 9500
8342 LET x$="nr wezla S-": GO SU
B 9500
8343 LET x$="nr wezla I+": GO SU
B 9500
8344 LET x$="nr wezla I-": GO SU
B 9500
8345 LET x$="wspolczynnik k[]":
GO SUB 9550
8346 RETURN
8350 REM kasowanie elementu
8355 LET l=6
8360 GO SUB 9400
8365 RETURN
8400 REM Zr. Nap. Ster. Nap.
8410 REM lista
8415 PRINT "Zr. Napieciowe Ster.
 Napieciowo": PRINT
8420 PRINT "nr element S+ S- U+
U- Ut[]"
8425 LET x$="ZNSN": GO SUB 9130:
 RETURN
8430 REM edycja elementu
8440 LET k=0: GO SUB 9500
8441 LET x$="nr wezla S+": GO SU
B 9500
8442 LET x$="nr wezla S-": GO SU
B 9500
8443 LET x$="nr wezla U+": GO SU
```

```
B 9500
8444 LET x$="nr wezla U-": GO SU
B 9500
8445 LET x$="wspolczynnik t[]":
GO SUB 9550
8446 RETURN
8450 REM kasowanie elementu
8455 LET l=6
8460 GO SUB 9400
8465 RETURN

8500 REM Zr. Napiec. Ster. Prad.
8510 REM lista
8515 PRINT "Zrodla Napieciowe St
er. Pradowo": PRINT
8520 PRINT "nr element S+ S- I+
I- Wr[Om]"
8525 LET x$="ZNSP": GO SUB 9130:
 RETURN
8530 REM edycja elementu
8540 LET k=0: GO SUB 9500
8541 LET x$="nr wezla S+": GO SU
B 9500
8542 LET x$="nr wezla S-": GO SU
B 9500
8543 LET x$="nr wezla I+": GO SU
B 9500
8544 LET x$="nr wezla I-": GO SU
B 9500
8545 LET x$="wspolczynnik Y[S]":
 GO SUB 9550
8546 RETURN
8550 REM kasowanie elementu

8555 LET l=6
8560 GO SUB 9400
8565 RETURN
8600 REM Rezystory
8610 REM lista
8615 PRINT "Rezystory": PRINT
8620 PRINT "nr element W+ W- R[O
m]"
8625 LET x$="R": GO SUB 9110: RE
TURN
8630 REM edycja elementu
8640 LET k=0: GO SUB 9500
8641 LET x$="nr wezla W+": GO SU
B 9500
8642 LET x$="nr wezla W-": GO SU
B 9500
8643 LET x$="rezystancja R[Om]":
 GO SUB 9550
8644 RETURN
8650 REM kasowanie elementu
8655 LET l=4
8660 GO SUB 9400
8665 RETURN
8700 REM Pojemnosci
8710 REM lista
8715 PRINT "Kondensatory": PRINT

8720 PRINT "nr element W+ W- C[F
]"
8725 LET x$="C": GO SUB 9110: RE
TURN
8730 REM edycja elementu
8740 LET k=0: GO SUB 9500
8741 LET x$="nr wezla W+": GO SU
B 9500
8742 LET x$="nr wezla W-": GO SU
B 9500
8743 LET x$="pojemnosc C[F]": GO
 SUB 9550
8744 RETURN
8750 REM kasowanie elementu
8755 LET l=4
8760 GO SUB 9400
8765 RETURN

8800 REM Indukcyjnosci
8810 REM lista
8815 PRINT "Indukcyjnosci": PRIN
T
8820 PRINT "nr element W+ W- L[H
]"
8825 LET x$="L": GO SUB 9110: RE
TURN
8830 REM edycja elementu
8840 LET k=0: GO SUB 9500
8841 LET x$="nr wezla W+": GO SU
B 9500
8842 LET x$="nr wezla W-": GO SU
B 9500
8843 LET x$="indukcyjnosc L[H]":
 GO SUB 9550
8844 RETURN
8850 REM kasowanie elementu
8855 LET l=4
8860 GO SUB 9400
8865 RETURN
8900 REM Tranzystory

8910 REM lista
8915 PRINT "Tranzystory": PRINT
8920 PRINT "nr element B  K  E
Ik[A]"
8925 LET x$="T": GO SUB 9120: RE
TURN
8930 REM edycja elementu
8940 LET k=0: GO SUB 9500
8941 LET x$="nr wezla B": GO SUB
 9500
8942 LET x$="nr wezla K": GO SUB
 9500
8943 LET x$="nr wezla E": GO SUB
 9500
8944 LET x$="prad kolektora Ik[A
]": GO SUB 9550
8945 RETURN
8950 REM kasowanie elementu
8955 LET l=5
8960 GO SUB 9400
8965 RETURN
9000 REM edytor elementow

9010 GO SUB 9900
9020 GO SUB 9990
9030 IF x$>="0" AND x$<="9" THEN
 LET x=VAL x$+1: CLS : GO SUB 60
00: GO TO 9000
9040 IF x$="w" THEN GO SUB 6300:
 GO TO 9000
9041 IF x$="n" THEN PAUSE 1: PRI
NT #0;"nowy uklad? (t/n)": PAUSE
 0: INPUT "": IF INKEY$="t" THEN
 GO SUB 7000: GO TO 9000
9042 IF x$="l" THEN CLS : PRINT
"lista elementow ukladu": FOR x=
1 TO 10: GO SUB 8025+100*(x-1):
NEXT x: PRINT : PRINT #0;"Nacisn
ij dowolny klawisz": PAUSE 1: PA
USE 0: INPUT "": GO TO 9000
9045 BEEP .1,20
9050 GO TO 9020
9100 REM listy zbiorow
9110 REM 2 wezly,1 parametr
9115 FOR k=1 TO i(x): PRINT k;TA
B 3;x$;e(x,k,1);TAB 11;e(x,k,2);
TAB 14;e(x,k,3);TAB 17;e(x,k,4):
 NEXT k: RETURN
9120 REM 3 wezly,1 parametr
9125 FOR k=1 TO i(x): PRINT k;TA
B 3;x$;e(x,k,1);TAB 11;e(x,k,2);
TAB 14;e(x,k,3);TAB 17;e(x,k,4);
TAB 20;e(x,k,5): NEXT k: RETURN
9130 REM 4 wezly,1 parametr
9135 FOR k=1 TO i(x): PRINT k;TA
B 3;x$;e(x,k,1);TAB 11;e(x,k,2);
TAB 14;e(x,k,3);TAB 17;e(x,k,4);
TAB 20;e(x,k,5);TAB 23;e(x,k,6):
 NEXT k: RETURN
9400 REM kasowanie elementu
9410 LET i(x)=i(x)-1: IF e=10 TH
EN RETURN
9420 FOR k=e TO i(x): FOR j=1 TO
 l
9430 LET e(x,k,j)=e(x,k+1,j)
9440 NEXT j: NEXT k: RETURN
9500 REM wprowadzenie wezla

9505 IF k=0 THEN LET x$="numer u
kladowy elementu"
```

```
9510 LET k=k+1
9515 INPUT (x$);": ";e(x,e,k)
9520 IF FN z(k=1,maxw,e(x,e,k))
THEN GO TO 9515
9530 RETURN
9550 REM wprowadzenie wartosci
9560 LET k=k+1: INPUT (x$);": ";
e(x,e,k)
9565 RETURN
9600 REM wybor elementu
9610 INPUT "padaj nr elementu: "
;e: IF FN z(0,i(x),e) THEN GO TO
 9610
9620 RETURN
9800 REM edytor zbioru
9810 PRINT AT 15,11; INVERSE 1;"
FUNKCJE:"
9820 PRINT
9830 IF i(x)>0 THEN PRINT TAB 5;
"p-poprawienie elementu"

9831 IF i(x)<10 THEN PRINT TAB 5
;"d-dolaczenie elementu"
9832 IF i(x)>0 THEN PRINT TAB 5,
"s-skasowanie elementu"
9833 PRINT TAB 5,"k-koniec edycj
i zbioru"
9840 RETURN
9900 REM informacja edytora
9910 CLS : PRINT TAB 7; INVERSE
1;"FUNKCJE EDYTORA:"
9915 PRINT
9920 PRINT "0-Stale Zrodlo Prado
we"
9921 PRINT "1-Stale Zrodlo Napie
ciowe"
9922 PRINT "2-Zr. Pradowe Ster.
Napieciowo"
9923 PRINT "3-Zr. Pradowe Ster.
Pradowo"
9924 PRINT "4-Zr. Napiecowe Ster
. Napieciowo"
9925 PRINT "5-Zr. Napieciowe Ste

r. Pradowo"
9926 PRINT "6-Rezystor"
9927 PRINT "7-Kondensator"
9928 PRINT "8-Indukcyjnosc"
9929 PRINT "9-Tranzystor"
9930 PRINT : PRINT "w-wykonanie
obliczen"
9931 PRINT "n-nowy uklad"
9932 PRINT "l-lista elementow uk
ladu"
9940 RETURN
9990 REM wczytanie znaku
9991 PRINT #0;AT 0,8,"wybierz fu
nkcje"
9992 PAUSE 0: LET x$=INKEY$: IF
x$<>"" THEN INPUT "": RETURN
9993 GO TO 9991
9998 REM zapis programu na tasme
9999 CLEAR : SAVE "analiza AC" L
INE 1: VERIFY "analiza AC"
```


ZASTOSOWANIA

(materiały do podstaw informatyki cz. 7)

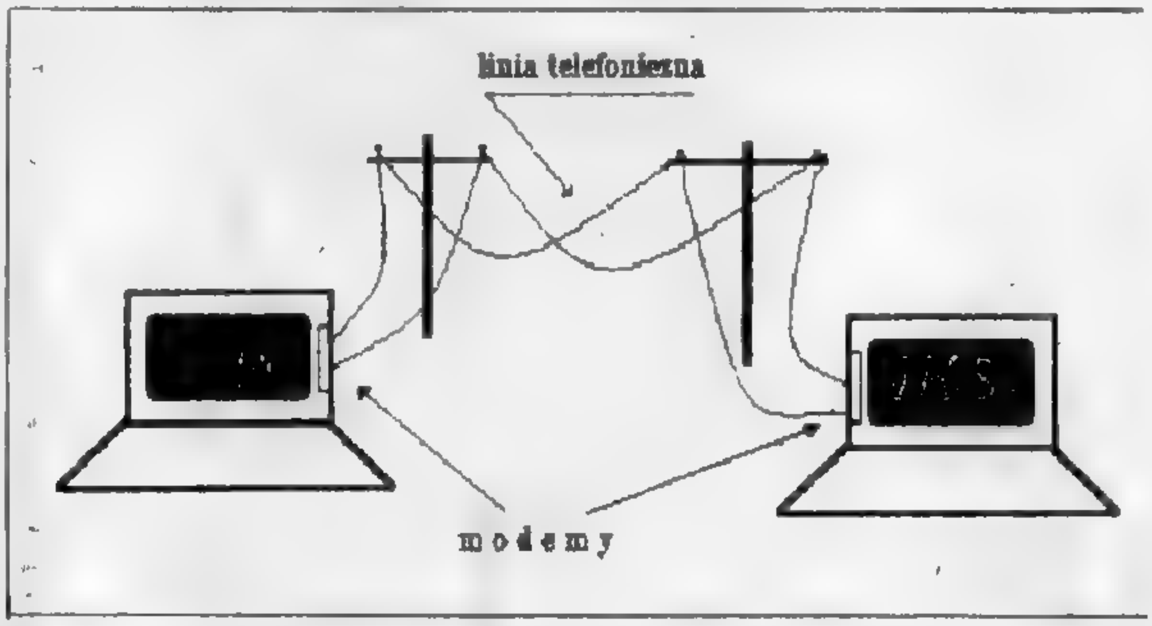
Przebrnęliśmy już przez sześć spotkań, podczas których informatyka okazywała się nam coraz mniej tajemnicza (mam taką nadzieję). Zgodnie ze zobowiązaniem, z poprzedniego numeru, tym razem zamierzam omówić kilka praktycznych przykładów (sądzę, że atrakcyjnych) zastosowań komputerów klasy IBM PC. Przede wszystkim powinniśmy być świadomi, iż maszyny te mają ograniczone możliwości wobec profesjonalnych zastosowań informatyki (pisałem o tym przy okazji superkomputerów w „IKS-ie" nr 2/89). Wynika to ze stosunkowo niewielkiej pamięci (RAM — megabajty i dyskowej — setki megabajtów) oraz ograniczonej szybkości — pojedyncze miliony operacji na sekundę. Jednak niezaprzeczalną zaletą sprzętu tej klasy jest jego dostępność, bogate oprogramowanie i stosunkowo niewielka cena. Sądzę, że w najbliższych latach sprzęt ten całkowicie zdominuje naszą informatykę.

Szczególnie korzystnym (efektywnym) sposobem wykorzystywania PC-tów jest ich łączenie w tak zwane **sieci lokalne.** Zazwyczaj instalowane są one w jednym budynku, na przykład w szkole, biurowcu, fabryce. Dzięki wzajemnemu połączeniu komputery mogą pośredniczyć w przesyłaniu informacji (poczta elektroniczna), umożliwiać korzystanie ze wspólnych (kosztownych) urządzeń — na przykład ze stacji dysków o dużej pojemności, drukarki laserowej, a także pracować na ogólnodostępnej bazie danych.

„Apetyt rośnie w miarę jedzenia" — skoro połączyliśmy maszyny w jednym budynku, powstaje pytanie czy nie można ich połączyć w ramach miasta, państwa, a może w skali świata? Owszem istnieje taka możliwość — wyko-



rzystując istniejącą sieć telefoniczną. Wystarczy bowiem dokupić do naszego komputera urządzenie — **modem** (w postaci karty lub wolno stojącego pudełka) i już nasz komputer można włączyć do gniazdka telefonicznego (por. „IKS" nr 6/88). Jeśli w tym czasie inny komputer podobnie dołączymy do sieci telefonicznej np. w Ciechocinku lub Słupsku, a nawet za oceanem w Palo Alto, komputery te mogą między sobą „uciąć pogawędkę".



Efektem owych „pogaduszek" jest przekazywanie określonych informacji, w postaci zarówno tekstów, jak i obrazów (w formie cyfrowej, por. „IKS" 9/88). Maszyna wyręcza nas zatem w rozmowie telefonicznej — samodzielnie przesyłając tekst (wcześniej zapisany w pamięci). Nie prościej i taniej po prostu porozmawiać samemu, bez komputerów? W czasie jednej sekundy rozmowy wypowiadamy przeciętnie słowa składające się z 10—20 liter. Natomiast konwersacja komputerów przebiega znacznie sprawniej — od dziesiątek do bilionów znaków na sekundę, (jedną sekundę!) W praktyce, na którą pozwala „taka sobie" sieć telefoniczna, są to szybkości od 30 do 150 znaków w czasie jednej sekundy. W owych „bilionowych transmisjach" np. przez ocean — wykorzystywane są sputniki oraz specjalistyczny sprzęt, który gromadzi informacje w tak zwane porcje (paczki), następnie przesyłane do adresata z niewiarygodną szybkością.

Omawiana „komputerowa konwersacja" wymaga oczywiście odpowiedniego oprogramowania, które realizuje najprzeróżniejsze funkcje. Na przykład nocą, gdy jest mniejsza opłata za rozmowy telefoniczne, komputery łączą się (samodzielnie) między sobą i rozpoczynają „wielkie gadanie", czyli przesyłanie całej korespondencji między firmami, także między prywatnymi użytkownikami komputerów domowych.

W eksploatowanych u nas w kraju, PC-tach dosyć często wykorzystywany jest komunikacyjny program PROCOMM. Realizuje on szereg bardzo użytecznych funkcji, które związane są z czynnościami użytkownika wykorzystującego łączność komputerową na przykład w sieci FIDO. Kilka razy miałem okazję sprawdzić tę możliwość łącząc się z redakcyjnym komputerem miesięcznika KOMPUTER pracującym właśnie w owej sieci.

Po uruchomieniu programu PROCOMM, na ekranie ukazuje się bogate MENU:

```
:                           P r o C o m m   H e l p                                  :

   MAJOR FUNCTIONS              UTILITY FUNCTIONS             FILE FUNCTIONS

 Dialing Directory . Alt-D    Program Info ...... Alt-I    Send files ...... PgUp
 Automatic Redial... Alt-R    Setup Screen ...... Alt-S    Receive files ... PgDn
 Keyboard Macros ... Alt-M    Kermit Server Cmd . Alt K    Directory ...... Alt-F
 Modem Parameters .. Alt-P    Change Directory .. Alt-B    View a File .... Alt-V
 Translate Table ... Alt-W    Clear Screen ...... Alt-C    Screen Dump .... Alt-G
 Editor ............ Alt-A    Toggle Duplex ..... Alt-E    Log Toggle .... Alt-F1
 Exit .............. Alt-X    Hang Up Phone ..... Alt-H    Log Hold ...... Alt-F2
 Host Mode ......... Alt-Q    Elapsed Time ...... Alt-T
 Chat Mode ......... Alt-O    Print On/Off ...... Alt-L
 DOS Gateway ...... Alt-F4    Set Colors ........ Alt-Z
 Command Files .... Alt-F5    Toggle CR-CR/LF .. Alt-F3
 Redisplay ........ Alt-F6    Break Key ........ Alt-F7
```

Wybierając na przykład pierwszą pozycję: „Dialing Directory" (przez jednoczesne wciśnięcie dwóch klawiszy: ALT i D) na ekranie pojawia nam się kolejne MENU, tym razem w postaci spisu telefonów:

Gdy usiłuję połączyć swój komputer z komputerem w COMERS FIDO, naciskam tylko klawisz jedynki, resztą (to znaczy „wykręceniem numeru") zajmie się sam komputer. Po uzyskaniu połączenia mogę przesłać uprzednio przygotowaną korespondencję lub odebrać adresowane do mnie „elektroniczne listy" (Send files i Receive files). Oczywiście teraz nie uda mi się omówić wszystkich możliwości PROCOMMa (są bardzo ciekawe). Może innym razem? Jednak mam nadzieję, że udało mi się zasygnalizować ogólną ideę funkcjonowania komputerowej łączności.

Omówiony program, mimo pozorów swej uniwersalności, jest tylko narzędziem. Uzupełniając go o pewną autonomię, na przykład zdolność gromadzenia i udostępniania, przez sieć telefoniczną, innym komputerom (u nas w domu) wyników ligi piłkarskiej, notowań listy przebojów, rozkładu jazdy pociągów, repertuaru kin, teatrów, słowników: wyrazów obcych, „polsko-obcojęzycznych" i odwrotnie, itp., otwiera się przed nami nowa, niedostępna wcześniej, jakość korzystania z potrzebnych nam informacji. Nietrudno jest bowiem sobie wyobrazić taki program komputerowy, który rozpoznaje nasze pytania. Na przykład w postaci: „słownik angielsko-polski, cinema?" Pytanie owo

```
N                       D I A L I N G   D I R E C T O R Y                    :

                 Name                     Number     Baud P D B  E   CMD File
     1- COMERS FIDO                       194391   1200-N-8-1  N
     2- FIDO Maciek Kollo           0,03120982502   300-N-8-1  N
     3- hcc olivetti ag            0,031 79517575   300-N-8-1  N
     4- POLY FIDO                   0,04415801690  1200-N-8-1  N
     5  LINK FIDO                   0,04416596992  1200-N-8-1  N
     6- iks                               312242   1200-N-8-1  N
     7  Airtel IBBS London          0,04412003439   300-N-8-1  N
     8- Marctel London              0,04413467150   300-N-8-1  N
     9- NNBBS London                0,04414556607   300-N-8-1  N
    10- Twighlight Phone London     0,04416245338   300-N-8-1  N

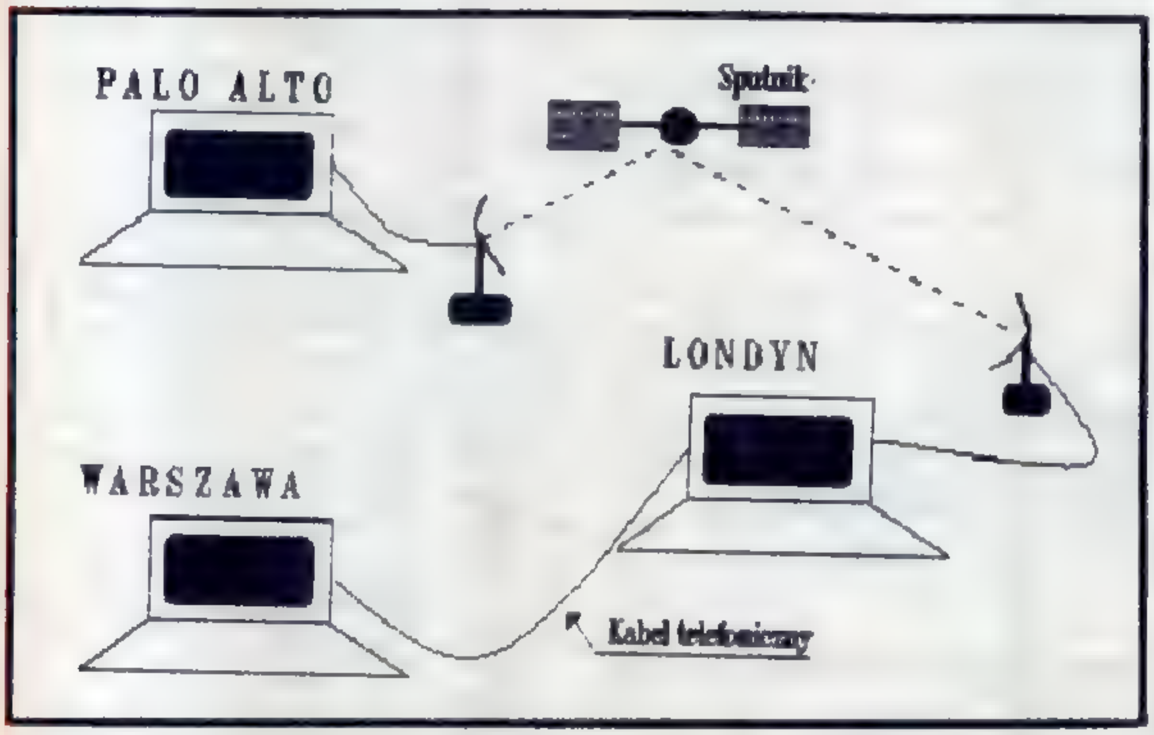
  ==>             R Revise          M Manual Dialing        Entry to Dial
                  P LD Codes        D Delete Entry          F Find
 / Scroll
                  Home Top Page     End Bottom Page         ESC Exit

       Modem command: ATDP                       LD Codes Active:

::::::::::::::::::::::::::::::::  AUTO  DIALER  ::::::::::::::::::::::::::::::::
```

(cztery słowa) wysyłamy z naszego komputera do centrum, oczywiście telefonicznie. Tam jest ono analizowane Pierwsza część pytania: SŁOWNIK kieruje komputer do odpowiedniej **bazy,** która nazwana została SŁOWNIKI Analiza dalszej części pytania prowadzi nas do **podbazy** — słownik angielsko-polski. Ostatecznie następuje wyszukanie polskiego słowa związanego z hasłem „cinema". Podany przykład dotyczy bardzo skromnej bazy danych W praktyce centra komputerowe dysponują setkami baz danych. Jednym z lepszych tego typu źródeł informacji jest DIALOG.

Sieć ta (z centrum w Palo Alto) bazuje na komputerze tysiące razy „silniejszym" od PC-ta. Przypuszczam, że jest to jeden z przedstawicieli rodziny CRAYa, który, krótko mówiąc, w czasie jednej sekundy zdolny jest wykonać miliardy operacji (por. „IKS" nr 2/89). Ta moc umożliwia jednoczesne korzystanie z owych pracowicie (i kosztownie) założonych i ciągle aktualizowanych baz danych, tysiącom użytkowników z całego świata — włączając Polskę. Należy zwrócić uwagę, że jest to sieć komercyjna. Oznacza to, że za każdą odpowiedź należy płacić. W praktyce wygląda to następująco. Łączymy nasz komputer z tak zwanym **węzłem** w Londynie (wybierając odpowiedni numer telefoniczny, najpierw kierunkowy 0,044 itd.). W węźle sprawdzane jest nasze upoważnienie (hasło) i już mój komputer zostaje dołączony do komputera w Palo Alto via Londyn (drutem) i via sputnik (eterem). Najdłużej trwa uzyskanie połączenia z Londynem...



Pytania zadajemy stosując, tak zwane **deskryptory** — słowa opisujące nasze pytanie. W praktyce nie wyczuwa się ograniczenia co do zakresu stosowanych słów podczas zadawania DIALOGowi pytań, można po prostu używać słów języka codziennego. Odbierane są one w Palo Alto, gdzie natychmiast przygotowywana jest odpowiedź (oczywiście przez komputer). Na początku jest to liczba pozycji (odpowiedzi), które mogą stanowić odpowiedź na zadane pytanie. Na przykład na temat: „młodzież, narkotyki i szkoła" jest 4000 pozycji. Dodajemy kolejne ograniczenie: „kobiety, 19 lat", uzyskujemy odpowiedź (natychmiast): 98 pozycji. Polecamy wszystkie owe pozycje wydrukować w postaci:

Tytuł, autor, nazwa pisma i rok wydania, gdzie można zdobyć oryginał język, w jakim jest napisany, materiał źródłowy, typ dokumentu, streszczenie, deskryptory oraz wiele innych bardziej szczegółowych informacji.

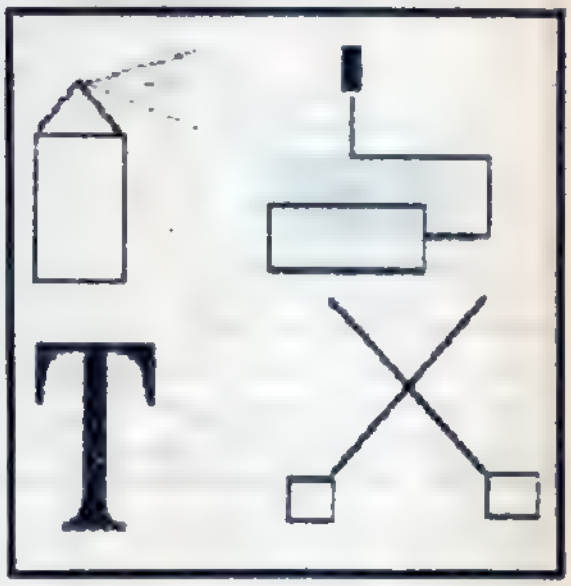
W zależności od naszej „kieszeni", można uzyskać wydruk natychmiast (bardzo drogo) lub jest przesyłany pocztą (znacznie taniej). Po wybraniu, na podstawie streszczeń, szczególnie interesujących pozycji — tą samą drogą — prosimy o przesłanie ich kopii lub wypożyczenie oryginałów.

DIALOG dysponuje ponad 150 milionami pozycji, które poukładane są w 300 bazach danych, plus kompletne teksty 450 pism i gazet, łącznie z dzisiejszym wydaniem.

Korzystanie z DIALOGU, sprawia ogromne wrażenie — siedzę wygodnie w Warszawie, a tam gdzieś daleko za oceanem jakiś superkomputer natychmiast wyszukuje niezbędne mi informacje. O dalszych szczegółach być może napiszę przy innej okazji.

Wróćmy „na ziemię", a dokładnie do analizy innych możliwości wykorzystania naszego PC-ta. Dotychczas omówiłem (z konieczności bardzo ogólnie) trzy zastosowania — pisanie własnych programów, edytor tekstów i szeroko rozumianą łączność między komputerami. Wydaje mi się, że bardzo atrakcyjnym narzędziem programowym są edytory obrazów. Komputer wspaniale ułatwia budowę ilustracji, czego przykładem są załączone ryciny. Może wykonane są one trochę nieudolnie — ale w bardzo krótkim czasie (1,5 godziny) i samemu! Wykorzystywałem do tego celu Paint Brusha, który jest bardzo prosty w użyciu. Na początku wyświetla się nam na monitorze „pole pracy", ikonki (po lewej stronie ekranu) oraz jeden wiersz bardzo bogatego MENU.

Wspomniane ikony ukazują funkcje, które PAINT potrafi wykonać. Jest ich kilkanaście, a załączona ilustracja ukazuje cztery przykładowe



Gdy „dotkniemy" kursorem, sterowanym myszą lub klawiaturą, pierwszą ikonę kursor staje się narzędziem do malowania — przesuwając nim po tworzonym rysunku — funkcjonuje, jak „farba w sprayu". Oczywiście kolory (lub w wersji czarno-białej zamiast kolorów — różne wzorki) można dowolnie dobierać. Podobnie funkcjonuje „wałek malarski", z tą różnicą, że wystarczy nim dotknąć zamkniętej powierzchni, a cała automatycznie pokrywa się wybranym kolorem lub wzorkiem (np dach ilustracji „Szkoły przyszłości"). Duża litera „T", to symbol wyboru możliwości pisania na budowanym rysunku, dostępnych jest kilka krojów pism i kilkanaście wielkości). „Nożyczki" pozwalają wyciąć kawałek budowanego obrazka i przenieść ten fragmencik w dowolne inne miejsce ekranu. W tym wypadku mysz zamienia się w czarną kredę, którą obrysowujemy wybrany do przeniesienia obraz i już można go przesunąć w inne miejsce ryciny.

Opisałem zaledwie drobną część możliwości PAINTa i podobnych edytorów obrazów. Zawierają one niemal wszystkie, niezbędne podczas rysowania i malowania zdolności (poza wymienionymi: drukowanie, mazanie, przesuwanie, powiększanie — tak, by poprawić obraz z dokładnością do jednego piksela, zapisywanie na dysk) i wiele innych.

Inne, adresowane do bardziej przygotowanych Czytelników, zastosowania komputerów umożliwiają programy statystyczne. Jednym z bardziej efektownych jest STATGRAF. Na jego przykładzie widać wiele elementów wspominanej wcześniej życzliwości komputerów.

Tytułem wstępu do tego tematu — kilka słów o przykładowym przeznaczeniu tego typu programów. Ostatnio dosyć często dziennikarze TV, radia i prasy powołują się na tajemnicze *wyniki badań*. Rzecz polega na tym, że badaczom stawiane jest na przykład pytanie: „czy średnia ze szkoły średniej ma wpływ na efekty studiowania?". Do chwili, gdy na owo pytanie nie otrzymamy odpowiedzi w postaci **wymiernej oceny,** nasze osądy w tej kwestii, to tylko domysły, intuicja. Pomocy poszukuje się właśnie w **badaniach** (reprezentatywnych, to znaczy takich, w których badana liczba, tak zwana próba abiturientów — nie badamy oczywiście wszystkich maturzystów tylko losowo wybraną grupę, próbę — odzwierciedla interesujące nas cechy wszystkich abiturientów z prawdopodobieństwem popełnienia błędu np.: jeden raz na 100 przypadków, dopuszczając błąd szacunku ocen od +1 do —1).

Badania są źródłem poszukiwanej (wymiernej i reprezentatywnej) oceny związku wyników nauki w szkole i na studiach, (Owa wymierność oznacza po prostu liczbę. Na przykład stwierdzenie, że na stadionie było dużo kibiców jest przeciwieństwem wymiernej oceny, którą wyraża po prostu liczba kibiców, wynosiła ona 12 345 osób) Wracając do poszukiwania wymiernej oceny zależności efektów studiowania od ocen szkolnych, należy stwierdzić że nie jest to takie proste, jak miało to miejsce „na stadionie". Na początku naszej pracy musimy zgromadzić kolumny cyfr (uzyskanych od reprezentatywnej liczby osób). Jedną kolumnę stanowią średnie wyniki ze szkoły, drugą natomiast średnie wyniki uzyskane na wyższej uczelni.

Wróćmy do naszego komputera, aby cokolwiek można było liczyć, wcześniej musimy wprowadzić dane do pamięci maszyny. Zobaczmy, jaką pomoc oferuje w tym wypadku STATGRAF?

Przedstawione MENU Statgrafa (niestety w języku angielskim) jest wierzchołkiem przysłowiowej pływającej góry (możliwości) lodowej. Wybieramy DATA MANAGEMENT, przesuwając kursor na nazwę wybranej „usługi". W odpowiedzi na ekranie zostaje wyświetlone kolejne MENU:

Wybieramy możliwość nr 2 i po kilku sekundach od uruchomienia programu możemy przystąpić do wpisywania danych źródłowych (kolumn ocen). Polega to na pisaniu cyfr w określonych

```
            IHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH;
            ;   STATGRAPHICS Statistical Graphics System   ;
            HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH<

DATA HANDLING AND SYSTEM UTILITIES          TIME SERIES PROCEDURES
  A. Data Management                          L. Forecasting
  B. System Environment                       M. Quality Control
  C. Report Writer and Graphics Replay        N. Smoothing
  D. Plotter Interface                        O. Time Series Analysis

PLOTTING AND DESCRIPTIVE STATISTICS         ADVANCED PROCEDURES
  E. Plotting Functions                       P. Categorical Data Analysis
  F. Descriptive Methods                      Q. Multivariate Methods
  G. Estimation and Testing                   R. Nonparametric Methods
  H. Distribution Functions                   S. Sampling
  I. Exploratory Data Analysis                T. Experimental Design

ANOVA AND REGRESSION ANALYSIS               MATHEMATICAL PROCEDURES
  J. Analysis of Variance                     U. Numerical Analysis
  K. Regression Analysis                      V. Mathematical Functions

        Use cursor keys to highlight desired section.  Then press ENTER.

 1HELP   2LABEL  3SAVSC  4RECORD 5       6       7       8       9REVIEW 10QUIT
PRINT    THR FEB 16  1989 01:04:00 PM  VERSION 1.0                      REC:OFF
```

```
                IHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH;
                ;          DATA MANAGEMENT         ;
                HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH<

              1.   Manipulate Defined Variables
              2.   Full-Screen Data Editor
              3.   Read Variable Definitions from SG File
              4.   Write to STATGRAPHICS File
              5.   Import Data from ASCII Data File
              6.   Export Data to ASCII Data File
              7.   Import Data from DIF File
              8.   Export Data to DIF File
              9.   Import Lotus 1-2-3 Worksheet
             10.   Export Lotus 1-2-3 Worksheet
             11.   Recode Missing Values


              Use cursor keys to highlight desired procedure.
                           Then press ENTER.

 1HELP   2LABEL  3SAVSC  4RECORD 5       6       7       8       9REVIEW 10QUIT
 RINT    THR FEB 16  1989 01:04:00 PM  VERSION 1.0                      REC:OFF
```

```
                                 Data Editor        Maximum Rows:     10
      Date Updated: 02/16/89                        Number of Cols:    6

Row        logr1      stgr1      logr2      stgr2      logr3      stgr3
     IHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH;
  1 ;        3          5          4          5          3          4;
  2 ;        4          3          4          5          4          3;
  3 ;        5          3          3          4          5          3;
  4 ;        3          5          3          4          3          5;
  5 ;        3          5          3          4          3          5;
  6 ;        5          3          3          3          4          3;
  7 ;        4          4          5          5          5          3;
  8 ;        4          4          5          5          3          4;
  9 ;        5          3          4          5          3          4;
 10 ;        4          3          4          5          4          3;
 11 ;                                                                ;
 12 ;                                                                ;
 13 ;                                                                ;
 14 ;                                                                ;
     HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH<
ength   10         10         10         10         10         10
 pe      N          N          N          N          N          N
                 Cursor at Row 11     and Column 1
 1HELP   2SAVE   3DONE   4SORT   5DSORT  6VSORT  7DVSORT 8UNSORT 9PRINT  10QUIT
 RINT    THR FEB 16  1989 01:13:00 PM  VERSION 1.0                      REC:OFF
```

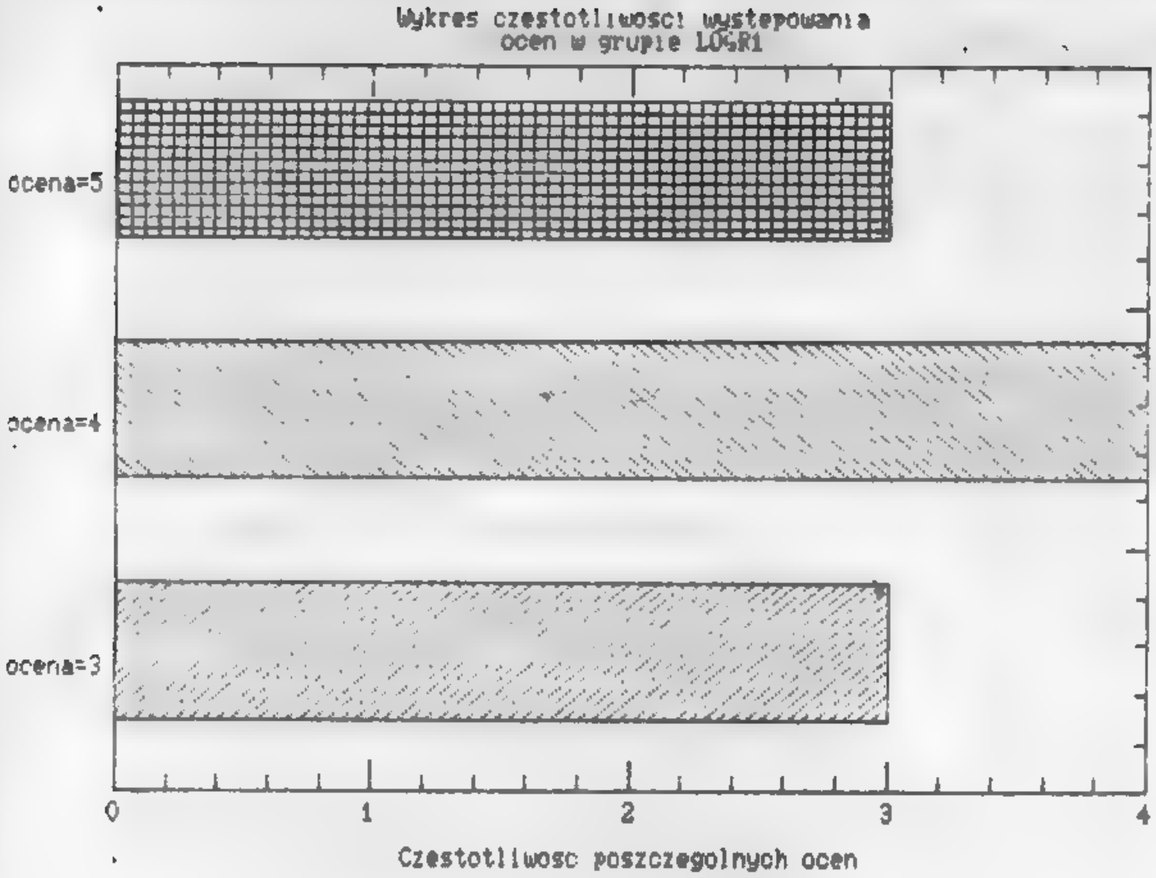
miejscach wyświetlanej przez program tabeli

Przewidziano oczywiście nasze ewentualne błędy, które łatwo można poprawić. Kontynuując przykład - wprowadzamy pary kolumn wyników ze szkół średnich i studiów od trzech grup badanych osób z różnych części kraju. LOGR1 oznacza wyniki ze szkoły średniej (LO) uzyskane od grupy nr 1(GR1), STGR1 — oznacza wyniki ze studiów (ST) tej samej grupy (GR1). Podobnie oznaczono pozostałe wyniki dla GR2 i GR3.

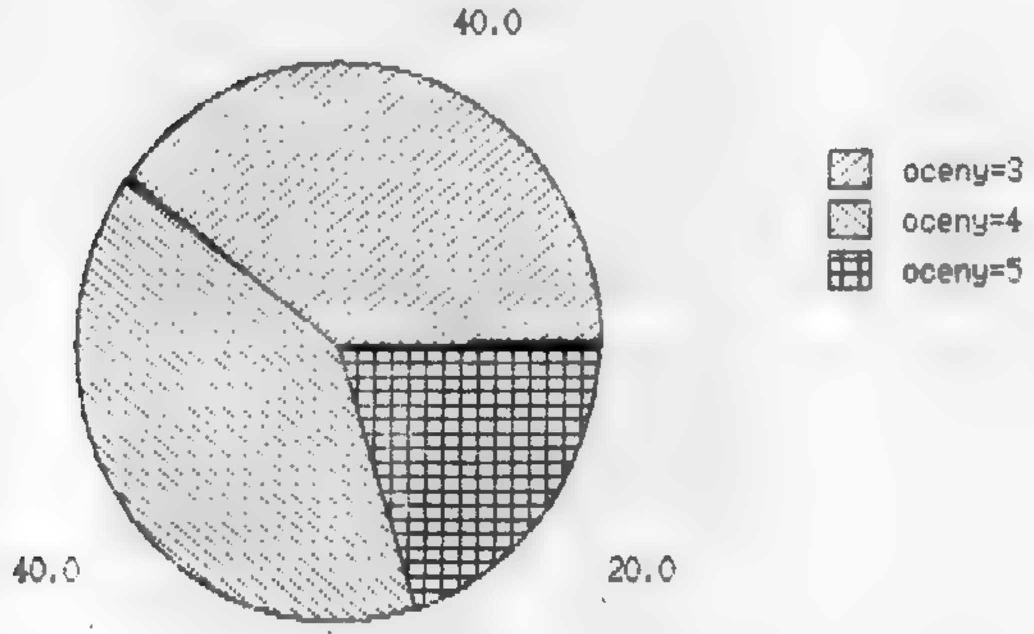
Tak wprowadzone dane możemy zobaczyć w postaci graficznej. Wystarczy z głównemu MENU wybrać odpowiednią „usługę" i natychmiast otrzymujemy, np.:

Oczywiście form graficznej prezentacji informacji jest bardzo wiele

Oglądając kolumny wprowadzonych ocen trudno jest oszacować wymierną wartość miary zależności między wynikami nauki uzyskanymi w szkole i na studiach. Przechodzimy zatem do rea



Ilustracja rozkladu ocen w grupie LOGR2



CORRELATION MATRIX

	logr1	logr2	logr3	stgr1	stgr2
logr1	1.00000	.00000	.49588	-.88852	-.19245
logr2	.00000	1.00000	.06844	-.06131	.79682
logr3	.49588	.06844	1.00000	-.52873	-.09543
stgr1	-.88852	-.06131	-.52873	1.00000	.00000
stgr2	-.19245	.79682	-.09543	.00000	1.00000
stgr3	-.66118	-.27375	-.80328	.79309	-.09543

	stgr3
logr1	-.66118
logr2	-.27375
logr3	-.80328
stgr1	.79309
stgr2	-.09543
stgr3	1.00000

lizacji naszego podstawowego celu. Polecamy komputerowi policzyć **wartość,** konkretną liczbę, która odzwierciedla nam poszukiwaną zależność.

Miar takich jest sporo, wybieramy jedną z nich — współczynnik korelacji „r". Jego wartość może wynosić od —1 do +1. Gdy r = —1 lub +1 analizowane zmienne (kolumny ocen) są całkowicie od siebie zależne (r = —1 oznacza, że wzrost jednej zmiennej powoduje malenie drugiej zmiennej, r = +1 — wzrost jednej zmiennej powoduje wzrost drugiej). Wartość r = 0 oznacza całkowitą niezależność zmiennych.

Obliczone zostały współczynniki korelacji między wszystkimi zmiennymi. Nas interesują zależności tylko między ocenami dla tych samych grup. Wyniki wskazują, że istnieje bardzo duża zależność między wynikami ze szkoły, a efektami studiowania dla grup nr 1 (—0,88852). W wypadku grupy nr 2 zachodzi sytuacja odwrotna — dodatnia korelacja (r = 0,79682). Natomiast w grupie nr 3, powtarza się pierwsza prawidłowość (r = —0,80328), co oznacza — czym lepsze wyniki w szkole — tym gorsze na studiach. Oczywiście to tylko przykład!

W naszych podstawach informatyki coraz więcej miejsca poświęcamy omawianiu problemów bardzo odległych od informatyki (RAMów, dysków, DOSów i innych bajtów). Nie jest to przypadkowe. W ten bowiem sposób usiłowałem zwrócić uwagę, że komputery powinny być traktowane przez nas jako bardzo przydatne narzędzia. Sugestia ta nie wynika z „mody", ale z autentycznej uniwersalności sprzętu informatycznego. Ważne jest zatem byśmy byli świadomi możliwości zaprzęgnięcia komputera do naszej pracy. Zakres tego typu pomocy wzrasta niemal każdego dnia, osiąga nadspodziewanie wysoki poziom, czy aby nie za wysoki..., ale o tym podczas następnego spotkania.

Włodzimierz GOGOŁEK



NOWE EDYTORY TEKSTU DLA C-64 i C-128

Zastanawiałem się nad odpowiednim zakończeniem cyklu artykułów na temat edytorów tekstu. Przypadkowo natknąłem się w czerwcowym numerze miesięcznika RUN z 1988 r. na artykuł Michaela Cavanaugha pt. *„The latest word"* prezentującego najnowsze tego typu programy przeznaczone dla komputerów C-64 i C-128. Autor omówił w nim: PAPERCLIP III, FONTMASTER 128, WORDPRO, GEOWRITE, WORKSHOP 128, WRITE STUFF. Poniższy tekst jest obszernym omówieniem artykułu p. Cavanaugha i mam nadzieję, że będzie dobrym zakończeniem zamieszczonych we wcześniejszych numerach „IKS-a" artykułów na temat edytorów tekstu.

PAPERCLIP III

Jest to program o bardzo szerokich możliwościach, przeznaczony zarówno dla C-64 jak i C-128. Firma, która go opracowała tj. Electronic Arts zrezygnowała z zabezpieczenia programu przed kopiowaniem, co równocześnie umożliwia łatwe przeniesienie go na dyskietkę 3 i 1/2 cala i użycie stacji dysków 1581. Przy zakupie programu otrzymuje się dwie dyskietki (dla C-64 i C-128). Każda dyskietka zawiera na jednej stronie edytor, a na drugiej słownik. Różnica między obydwiema wersjami programu polega na tym, że w wersji dla C-128 występuje 80 kolumn oraz możliwa jest współpraca z rozszerzeniem RAM. Rozszerzenie to wykorzystane zostało przede wszystkim do sprawdzania poprawności pisowni. PaperClip III posiada wbudowany program telekomunikacyjny, rezydujący w pamięci i dzięki temu możliwe jest łatwe przechodzenie od pracy z edytorem do programu telekomunikacyjnego (np. wysłanie napisanego tekstu do innego komputera) i odwrotnie.

Dokument tworzony za pomocą tego programu formatowany jest po jego napisaniu za pomocą komend umieszczonych w tekście.

Gotowy tekst z widocznym efektem wszystkich komend formatujących (w tym również odległości między wierszami) można obejrzeć na ekranie monitora. Do wad programu można zaliczyć pewne ograniczenia w przesuwaniu kolumn (program został raczej zaprojektowany do manipulowania tablicami i liczbami niż tekstem) oraz mały słownik — tylko 40000 słów — z tym, że może być on rozszerzany.

FONTMASTER 128

Fontmaster 128, opracowany przez firmę Xetec jest częściowo edytorem tekstu, a częściowo programem typu DTP (Desktop Publisher) bardzo łatwym w użyciu. Program może współpracować z ponad 100 rodzajami drukarek i 20 rodzajami interfejsów, jak również możliwe jest łatwe przystosowanie go do pracy z innymi drukarkami i interfejsami niż wymienione w programie. Główne komendy programu wyświetlane są w górnej części ekranu i w zasadzie to wystarcza do szybkiego nauczenia się pisania i obróbki tekstu. Fontmaster 128 nie jest zabezpieczony przez kopiowaniem, ale wymaga zastosowania specjalnego przyłącza (dongle) w celu umieszczenia napisanego tekstu na dyskietce lub jego wydrukowania. Czas ładowania programu do pamięci komputera wynosi 33 sekundy, a przy zastosowaniu stacji 1581 (3 1/2 cala) spada do 20 sekund.

Z programem współpracuje słownik o zawartości 100000 słów, nazwany Spell-Master 128, który może być załadowany do rozszerzenia RAM (1700 lub 1750). Istotną zaletą programu są jego możliwości graficzne. Zawiera on ponad 45 różnych rodzajów krojów pisma, w tym również takie jak np. rosyjskie, greckie i arabskie. W tekście może być umieszczana grafika wykonana za pomocą niektórych innych programów np. Print Shop. Do wad edytora należy zaliczyć brak możliwości łączenia plików, co ogranicza rozmiar pojedynczego pliku do ok. 64 K (ok. 30 stron tekstu) i brak możliwości formatowania tekstu na ekranie. Istnieje co prawda możliwość przeglądania napisanego tekstu na ekranie z uwzględnieniem jego graficznej formy, ale widoczne są jedynie 53 kolumny, co wymaga poziomego przesuwania tekstu. Niemożliwe jest także przeglądanie tekstu do tyłu.

WORDPRO

Opracowany przez firmę Spinnaker Software program nie zawiera żadnych większych rewelacji, może z wyjątkiem dużego słownika i wydaje się pewnym powrotem do tego co ukazało się na rynku edytorów tekstu dwa lata wcześniej.

WordPro również nie jest zabezpieczony, przed kopiowaniem wymaga jednak stosowania Turbo Load i Save (cartridge) w celu załadowania go do pamięci komputera. Uniemożliwia to z kolei korzystanie m.in. z rozszerzenia RAM.

Program jest dość skomplikowany, a nawet czasami niepotrzebnie zagmatwany. Trudna jest np. zmiana koloru tła, czy określenie rodzaju używanej drukarki.

Program ładuje się do pamięci dość szybko, bo w ciągu ok. 20 sekund. Przeznaczony jest dla C-64 i C-128, z tym, że C-128 pracuje w trybie C-64, a stacja dysków 1571 w trybie stacji 1541. Program nie rozpoznaje stacji 1581. Edytor posiada jednak pewne zalety, do których należy zaliczyć:

a) możliwość druku dwustronnego z automatyczną numeracją stron (najpierw drukowane są strony nieparzyste, a po odwróceniu papieru strony parzyste),
b) słownik o zawartości 100 000 słów.

GEOWRITE WORKSHOP 128

GeoWrite Workshop (prod. Berkeley Softworks) jest 80-kolumnowym edytorem typu WYSIWYG (what you see is what you get), co oznacza, że to co widzimy na ekranie jest zgodne z tym co otrzymamy po przelaniu na papier. Program ten współpracuje z programem GEOS 128.

Główną częścią programu jest GeoWrite 2.1, rozszerzenie edytora zawartego w Geosie 128. Dzięki temu rozszerzeniu możliwe jest formatowanie pojedynczych paragrafów, dowolne formatowanie tekstu, ustawianie odstępu między wierszami, drukowanie w stopce i nagłówku oraz operacje poszukiwania i zastępowania fragmentów tekstu.

Edytor ten posiada również takie możliwości jak:
— umieszczanie danych w napisanym tekście („Beo-Merge"),
— współpraca z drukarką laserową „Bloeaser", co zapewnia znakomitą jakość druku,

— współpraca z programem GeoPaint i GeoWrite (umieszczanie tekstu i grafiki),
— umieszczanie danych numerycznych uzyskanych za pomocą GeoDex, GeoFile i GeoCalc,
— umieszczanie w tekście za pomocą programu Text Grabber dokumentów napisanych za pomocą Fleet System.
4. WordWriter 128, PaperClip II i innych edytorów,
— współpraca z GeoSpell (słownik o zawartości 28000 słów, z możliwością rozszerzenia).

Sprawdzanie poprawności pisowni przebiega jednak wolno (pięć stron tekstu w około 10 minut).

Workshop zawiera trzy opcje druku — wysokiej jakości (high quality), druk NLO i druk normalny.

Pomimo że Geowrite Workshop posiada podstawowe możliwości edytora tekstu, to jednak najrozsądniejszym rozwiązaniem wydaje się używanie np. PaperClip III do pisania tekstu, a następnie Geowrite Workshop w celu nadania mu odpowiedniej szaty graficznej.

WRITE STUFF

Write Stuff dostępny w firmie Busy Bee Software jest jednym z najbardziej interesujących, a zarazem najtańszych edytorów. Przeznaczony jest dla C-64, współpracuje z dwiema stacjami dysków i rozszerzeniami RAM, umożliwia drukowanie w dwóch kolumnach, a nawet generuje mowę.

Unikalny jest sposób dystrybucji programu. Kupujący może sam zostać jego dystrybutorem, co znacznie obniża cenę programu (z 24.95 do 11.25, a nawet do 8 — bez podręcznika i opisu klawiatury).

Do podstawowych możliwości, a zarazem zalet edytora należy zaliczyć.
— możliwość przechowywania dokumentów na dyskietce w postaci plików programowych lub sekwencyjnych,
— możliwość używania skrótów (dwie litery zastępujące słowo lub kilka słów),
— przełączanie klawiatury z typu Owerty na Dvorak i odwrotnie,
— możliwość ukrycia tekstu (konieczna jest znajomość hasła w celu jego załadowania i odczytania),
— możliwość jednoczesnego przechowywania w pamięci dwóch dokumentów,
— współpracę z rozszerzeniem RAM (konieczne użycie programu RAMDOS),
— współpracę z programem Superbase (baza danych).

Program zawiera również 57 stron instrukcji obsługi, także przykłady tekstu.

Druga strona dyskietki z programem zawiera BB Talker, syntezator mowy, którego podstawową częścią jest znany również w Polsce program o nazwie S.A.M.

BB Talker zawiera wiele znanych piosenek i rytmów, które mogą być odczytane i słuchane. Można również komponować własne utwory. Program ten doskonale może służyć np. do nauczania alfabetu w przedszkolu.

Write Staff nie zawiera słownika, ale poprawność pisowni utworzonych dokumentów może być sprawdzona za pomocą innych programów.

Jeszcze w roku 1988 powinna ukazać się wersja programu Write Staff dla C-128. Planuje się także unowocześnienie opisanej wyżej wersji dla C-64, a także dla C-128.

W tablicy nr 1 zestawiono podstawowe możliwości omówionych w artykule programów.

Tablica nr 1.

	PaperClip III	PaperClip III	WordPro	Write Stuff	GeoWriteworkshop	FontMaster 128
Komputer	C-64	C-128	C-64	C-64	C-128	C-128
Zabezpieczenie przed kopiowaniem	—	—	— (1)	—	—	— (1)
Słownik	+	+	+	— (2)	— (2)	+
Formatowanie na ekranie	—	—	—	—	+	—
Długość tekstu (stron)	2,5	7,5	4,5	10	62	30
Łączenie plików	+	+	+	+	—	—
Przeglądanie 80 kol.	—	+	+	—	—	+ (3)
Czas sprawdzenia poprawności pisowni (4) (w sek.)	210	18	105	—	[illegible]	49
Integracja z bazą danych programem kalkulacyjnym	—	—	+	—	+	—
Praca z wieloma stacjami dysków	+	+	+	+	+	+
Praca ze stacją 1581	+	+	—	—	+	+
Praca z rozszerzeniem RAM	—	+	—	+	+	+
Cena /$/	50		40	25	70	70

Uwagi:
1 — zabezpieczenie sprzętowe (dongle),
2 — słownik dostępny w oddzielnym programie,
3 — widoczne jest 12 wierszy x 58 kolumn, ale tekst może być przesuwany poziomo,
4 — dla PaperClip III (C-64) długość sprawdzanego dokumentu wynosi około 700 słów, dla pozostałych programów około 2700 słów.

Opracował: Tadeusz CISEK

„SCROLLING" TEKSTU W DOWOLNYM TRYBIE GRAFICZNYM

Często istnieje potrzeba efektownego opisania wygenerowanego przez program rysunku lub grafiki wytworzonej za pomocą jednego z pakietów graficznych jak Koala Microilustrator, Design Master czy MicroPainter. Sądzę, że idealnym rozwiązaniem w takiej sytuacji jest umieszczenie wolno przesuwającego się tekstu pod lub nad rysunkiem. Takie rozwiązanie ma wiele zalet, a najważniejszą wydaje się być prawie niczym nie ograniczona długość przesuwanych napisów. Pragnę również zaprezentować narzędzie, które uczyni realnym powyższy zamiar.

Jest nim przedstawiona na wydruku 2 w liniach DATA procedura w kodzie maszynowym. Realizuje ona scrolling tekstu w dowolnym trybie graficznym dostępnym na komputerach ATARI 800XL i 130XE (tryby 3—11 i 14—15). Procedura może być wywoływana z programu w języku BASIC (komenda USR) oraz dołączana do programów pisanych w języku C lub PASCAL. Procedura po wywołaniu określa na podstawie przekazanych do niej argumentów miejsce pamięci ekranu, w którym utworzone ma być okno tekstowe, zawierające przesuwające się napisy. Dane z tej części pamięci ekranu kopiowane są przez procedurę w bezpieczne miejsce. Po wykonaniu scrollingu procedura umieszcza ponownie dane tworzące obraz w miejscu, z którego zostały one skopiowane.

Tworzone przez procedurę okno tekstowe ma wysokość ośmiu punktów odpowiadających wybranemu trybowi graficznemu. Oznacza to, że wysokość okna tekstowego waha się w granicach od 54 (dla GRAPHTOS 30) do 8 (dla GRAPHTOS 8) linii obrazu. Długość okna tekstowego wyrażona jest liczbą bajtów i zawiera się w przedziale od 1 do 40. Liczba ta zależy również od wybranego trybu graficznego.

Przesuwany przez procedurę tekst nie może zawierać więcej niż 120 znaków. Dlatego w sytuacji, gdy istnieje potrzeba „przewijania" w oknie tekstowym większych partii tekstu, należy cały tekst podzielić na łańcuchy o długości nie przekraczającej 120 bajtów i wywołać procedurę kilkakrotnie. W przewijanym łańcuchu nie mogą wystąpić znaki w inwersie. Procedura działa poprawnie również z niestandardowym generatorem znaków, zatem przesuwane przez nią teksty mogą zawierać znaki zdefiniowane przez użytkownika o ile odpowiadające im kody ATASCII nie są większe od 127.

Wydruk 1 przedstawia program demonstrujący działanie procedury i ilustrujący sposób jej wywoływania z programu napisanego w BASIC-u. Program zbudowany jest z dwóch modułów: MENU i SCROLL. Moduł MENU wczytuje parametry sterujące przesuwaniem tekstu oraz tekst, który będzie przesuwany. Moduł SCROLL, w zależności od wybranego trybu graficznego, bądź rysuje na ekranie prosty obraz graficzny, bądź wczytuje obraz graficzny z taśmy lub z dysku. Druga opcja dostępna jest jedynie po wybraniu trybu graficznego 8.

Moduł MENU ustawia wartości siedmiu opcji, określających parametry scrollingu:

— opcja 1 definiuje tryb graficzny, przy zastosowaniu którego demonstrowany będzie scrolling. Dopuszczalnymi wartościami są: 3—11 i 14—15;

— opcja 2 definiuje szybkość, z jaką będzie przesuwany tekst w oknie tekstowym. Szybkość określana jest liczbą z przedziału 0—15, przy czym zero odpowiada maksymalnej, a 15 minimalnej szybkości;

— Opcje 3 i 4 definiują współrzędne górnego lewego wierzchołka okna tekstowego podane w postaci numerów kolumny i wiersza. Po wybraniu trybu graficznego moduł określa dopuszczalne w nim wartości numerów kolumn i wierszy. Znaczenie numerów wierszy i kolumn wyjaśnia dokładniej następujący przykład: gdy wybrany zostanie ósmy tryb graficzny, to odpowiadający mu numer kolumny może zawierać się w przedziale 0—39. Wynika to stąd, że w każdym trybie graficznym znaki tekstu pobierane są z generatora znaków, w którym są zdefiniowane w matrycy o wymiarach 8x8 bitów. Ponieważ w tym trybie graficznym w jednej linii dostępnych jest 320 pixeli i każdemu pixelowi odpowiada dokładnie jeden bit, stąd po podzieleniu tej liczby przez 8 otrzymujemy 40, co oznacza, że w jednej linii tego trybu można zmieścić 40 znaków ponumerowanych od 0 do 39. Analogicznie, gdy rozważymy tryb graficzny 3, w którym w jednej linii dostępnych jest 40 pixeli i uwzględnimy, że w tym trybie każdy pixel tworzony jest z dwóch bitów, zatem szerokość znaku jest równa czterem pixelom, co umożliwia zapisanie 10 znaków w jednej linii tego trybu. Numer kolumny dla tego trybu musi mieścić się w przedziale 0—9. Analogiczne rozważania prowadzą do wniosku, że w trybie graficznym 8 dopuszczalny numer wiersza musi być z przedziału 0—184, a w trybie graficznym 3 — z przedziału 0—16. Nie wolno zapomnieć o tym, że umieszczenie okna tekstowego na samym dole ekranu wymaga zarezerwowania dla niego ośmiu linii odpowiadających wybranemu trybowi graficznemu. Stąd maksymalny numer wiersza w ósmym trybie graficznym jest równy 184 (192—8), a w trzecim — 16 (24—8). Pominięcie przedstawionych wyżej zasad przy określaniu położenia okna tekstowego na ekranie może spowodować, w najlepszym wypadku, pojawienie się okna tekstowego w innym niż zamierzono miejscu ekranu, a w wypadku najgorszym „załamanie" się programu z powodu błędu 141 (Cursor out of range).

— Opcja 5 definiuje długość okna tekstowego wyrażoną w bajtach. W trybie graficznym o największej rozdzielczości długość okna tekstowego nie może przekroczyć 40 bajtów. W innymch trybach graficznych długość ta nie może przekroczyć zwiększonego o jeden, maksymalnego dla wybranego trybu graficznego, numeru kolumny.

— Opcja 6 definiuje rejestr koloru, którego zawartość użyta będzie do określenia koloru liter w przesuwanym tekście. W trybach dwukolorowych jak 4, 5, 8, czy 14 opcja ta nie powoduje żadnych efektów, jednakże wartość z zakresu 1—4 musi być podana.

— Opcja 7 służy do wprowadzenia przesuwanego tekstu. Dodatkowo, gdy wybrany zostanie tryb graficzny 8 istnieje możliwość wprowadzenia grafiki wykonanej w tym trybie z taśmy lub z dysku.

Procedura realizująca przewijanie może być wykorzystana w prosty sposób w tworzonych programach. W tym celu należy podprogram przedstawiony na wydruku 2 nagrać na taśmę lub dysk komendą LIST, a następnie dołączać go do tworzonego programu, w którym wykorzystana będzie ta procedura, powinna być instrukcja GOSUB 00000. Jest to bardzo ważne, gdyż podprogram ten oprócz inicjacji procedury wykonującej przewijanie

dokonuje również zmiany RAMTOP, zmniejszając o 16 stron standardowy rozmiar dostępnej pamięci RAM. Procedura realizująca scrolling zajmuje prawie 8 stron pamięci, jej początek znajduje się na stronie 152. Procedura ta nie jest relokowalna. Pozostałe 8 stron wykorzystywane jest przez procedurę do skopiowania danych obrazu, gdy tworzone jest okno tekstowe. Procedura wykorzystuje ponadto 12 bajtów ze strony zerowej (adresy SCB-sD6). Zatem, jeśli program wywołujący procedurę używa tych bajtów, to musi skopiować je w bezpieczne miejsce, gdyż ich zawartość będzie przez procedurę zniszczona.

Procedura realizująca scrolling może zwrócić do programu, z którego została wywołana sześć wartości, które sygnalizują błąd. Oznaczają one:

0 — procedura scrollingu została zrealizowana poprawnie,
1 — do procedury przekazano złą liczbę argumentów,
2 — długość tekstu, który ma być przesuwany przekracza 120 zaków;
3 — długość zdefiniowanego okna tekstowego przekracza 40 znaków;
4 — podany został zły numer rejestru koloru;
5 — łańcuch tekstu, który ma być przesuwany zawiera nielegalny znak o kodzie ATASCII większym od 127;
6 — tryb pracy ekranu nie jest trybem graficznym.

na podstawie magazynu „ANTIC"
opr. H. KRASUSKI

Wydruk 1

```
1 REM **********************************
****
2 REM * program demonstruje scrolling
   *
3 REM * tekstu w dowolnym trybie graficz
nym*
4 REM **********************************
****
5 GOSUB 30000
6 MENU=500:SCROLL=1000
7 DIM TEXT$(120),SPAC$(35),CIO$(6),F$(20
),F1$(20),GK$(7)
8 CIO$(1)=CHR$(104):CIO$(2)=CHR$(162):CI
O$(3)=CHR$(32)
9 CIO$(4)="L":CIO$(5)="V":CIO$(6)=CHR$(2
28)
12 GOSUB MENU:REM wybor parametrow steru
jacych scrollingiem
14 GOSUB SCROLL:REM scrolling
16 GRAPHICS 0:IF W<>0 THEN ? CHR$(253);"
blad nr ";W;" wywolania procedury realiz
ujacej scrolling"
20 POSITION 2,10:? "chcesz sprobowac jes
zcze raz (T/N)"
21 CLOSE #1:OPEN #1,4,0,"K:":GET #1,ZNAK

22 IF ZNAK=84 OR ZNAK=116 THEN GOTO 12
26 IF ZNAK=78 OR ZNAK=110 THEN GRAPHICS
0:END
28 ? CHR$(253):GOTO 21
490 REM ********************************
*****
491 REM * modul MENU - wczytuje paramet
ry  *
492 REM * sterujace przesuwem tekstu
   *
493 REM * nr trybu graficznego, wsplrzed
ne  *
494 REM * okna tekstowego, kolor, szybko
sc  *
495 REM * przesuwu oraz przesuwany tekst
 *
496 REM ********************************
*****
500 ? CHR$(125)
503 SPAC$="
   "
505 LGR=137:KPG=149:KPD=21:PGR=143:KPL=2
5:KPP=153:LDR=139:PDR=140
510 SETCOLOR 4,7,14:SETCOLOR 2,7,14:SETC
OLOR 1,5,7
515 POKE 752,1:POKE 82,0
520 POSITION 0,0:? CHR$(LGR);
525 FOR I=1 TO 38:? CHR$(KPG);:NEXT I
527 ? CHR$(PGR)
530 FOR I=1 TO 3
535 POSITION 0,I:? CHR$(KPL):POSITION 39
,I:? CHR$(KPP)
540 NEXT I
545 POSITION 0,4:? CHR$(LDR);
550 FOR I=1 TO 38:? CHR$(KPD);:NEXT I
555 ? CHR$(PDR)

560 POSITION 2,1:? "   Program demonstru
je scrolling"
565 POSITION 2,2:? "      tekstu w dowoln
ym trybie"
570 POSITION 2,3:? "              graficz
nym"
585 POSITION 0,5:? "parametry sterujace
przesuwem tekstu"
630 POKE 752,0
635 TRAP 635:POSITION 3,7:? SPAC$:POSITI
ON 3,7:? "Tryb graficzny  3-11 i 14-15
";:INPUT #16,TRYBGR
640 IF TRYBGR<3 OR TRYBGR>15 THEN ? CHR$
(253):GOTO 635
642 IF TRYBGR=12 OR TRYBGR=13 THEN ? CHR
$(253):GOTO 635
645 TRAP 645:POSITION 3,8:? SPAC$:POSITI
ON 3,8:? "szybkosc przesuwu (0-15)
";:INPUT #16,SZYBP
650 IF SZYBP<0 OR SZYBP>16 THEN ? CHR$(2
53):GOTO 645
655 IF TRYBGR=3 THEN LK=40:LW=24:GK$="11
":MLK=9:MLW=16
660 IF TRYBGR=4 THEN LK=80:LW=48:GK$="11
":MLK=9:MLW=40
662 IF TRYBGR=5 THEN LK=80:LW=48:GK$="11
":MLK=19:MLW=40
665 IF TRYBGR=6 THEN LK=160:LW=96:GK$="1
1":MLK=19:MLW=88
667 IF TRYBGR=7 THEN LK=160:LW=96:GK$="1
2":MLK=39:MLW=88
670 IF TRYBGR=8 THEN LK=320:LW=192:GK$="
2":MLK=39:MLW=184
675 IF TRYBGR=9 THEN LK=80:LW=192:GK$="3
50017":MLK=39:MLW=184
677 IF TRYBGR=10 THEN LK=80:LW=192:GK$="
300608":MLK=39:MLW=184
678 IF TRYBGR=11 THEN LK=80:LW=192:GK$="
300617":MLK=39:MLW=184
680 IF TRYBGR=14 THEN LK=160:LW=192:GK$=
"11":MLK=19:MLW=184
682 IF TRYBGR=15 THEN LK=160:LW=192:GK$=
"12":MLK=39:MLW=184
685 POSITION 3,9:? "wspolrzedne okna tek
stowego"
690 TRAP 690:POSITION 4,10:? SPAC$:POSIT
ION 4,10:? "nr kolumny 0-";MLK:POSITION
32,10:? " ";:INPUT #16,KOL
```

```
695 IF KOL<0 OR KOL>MLK THEN ? CHR$(253)
 GOTO 690
700 TRAP 700:POSITION 4,11:? SPAC$:POSIT
ION 4,11:? "nr wiersza 0-";MLW:POSITION
32,11:? " ";:INPUT #16,WIER
705 IF WIER<0 OR WIER>MLW THEN ? CHR$(25
3):GOTO 700
710 TRAP 710:POSITION 3,12:? SPAC$:POSIT
ION 3,12:? "dlugosc okna tekst. 1-",MLK+
1
711 POSITION 32,12:? " ";:INPUT #16,DLUG

715 IF DLUG<1 OR DLUG>MLK+1 THEN ? CHR$(


253):GOTO 710
720 TRAP 720:POSITION 3,13:? SPAC$:POSIT
ION 3,13:? "kolory 1-4
  ";:INPUT #16,COL
725 IF COL<1 OR COL>4 THEN ? CHR$(253):G
OTO 720
730 TRAP 0:POSITION 3,14:? "tekst, ktory
 bedzie 'przewijany'":INPUT #16,TEXT$
735 IF TRYBGR<>8 THEN JESTGR=0:RETURN
740 POSITION 2,18:? "Wybierz rodzaj graf
iki"
745 POSITION 2,19:? CHR$(177);"-programo
wa ";CHR$(178);"- Lad. z tasmy lub dysku
"
750 CLOSE #1:OPEN #1,4,0,"K:":GET #1,RO:
CLOSE #1
755 IF RO<>49 AND RO<>50 THEN ? CHR$(253
):GOTO 745
757 IF RO=49 THEN JESTGR=0:RETURN
760 TRAP 760:POSITION 2,20:? "Nazwa zbio
ru: ";:INPUT #16,F$:TRAP 40000
761 IF F$(2,2)=":" OR F$(3,3)=":" THEN 7
70
765 F1$=F$:F$="D1:":F$(4)=F1$
770 JESTGR=1:RETURN
990 REM ********************************
*******
991 REM * MODUL SCROLL - realizuje scrol
ling  *
992 REM * tekstu w oparciu o parametry
      *
993 REM * wczytane przez modul MENU
      *
994 REM ********************************
*******
1000 GRAPHICS TRYBGR+16:A=1:IF JESTGR TH
EN GOTO 1100
1001 IF VAL(GK$(1,1))=2 THEN SETCOLOR 1,
0,14:SETCOLOR 2,0,0:COLOR 1
1002 IF VAL(GK$(1,1))=3 THEN SETCOLOR 4,
VAL(GK$(2,3)),VAL(GK$(4,4)):LICZ=VAL(GK$
(5,6)):A=0
1004 IF TRYBGR<=8 THEN J=INT(LW/2)
1006 IF TRYBGR>=9 THEN J=INT(LK/2)
1010 FOR I=0 TO J STEP 2
1012 IF VAL(GK$(1,1))=1 OR VAL(GK$(1,1))
=3 THEN COLOR A
1020 PLOT I,I:DRAWTO LK-1-I,I:DRAWTO LK-
1-I,LW-1-I:DRAWTO I,LW-1-I:DRAWTO I,I
1024 IF VAL(GK$(1,1))=1 THEN A=A+1:IF A=
4 THEN A=1
1026 IF VAL(GK$(1,1))=3 THEN A=A+1:IF A=
LICZ THEN A=0
1030 NEXT I
1035 FOR I=1 TO 500:NEXT I
1040 W=USR(38912,ADR(TEXT$),SZYBP,LEN(TE
XT$),KOL,WIER,DLUG,COL)
1050 FOR I=1 TO 500:NEXT I
1055 RETURN
1099 REM LADUJ GRAFIKE W 8 TRYBIE GRAFIC
ZNYM


1100 TRAP 1900:CLOSE #2:OPEN #2,4,0,F$:T
RAP 40000
1112 AE=PEEK(88)+PEEK(89)*256:SZP=PEEK(1
06)*256:RMP=SZP-AE:SB=INT(RMP/256):MB=RM
P-SB*256
1120 IOCB=864:POKE IOCB+2,7
1130 POKE IOCB+4,PEEK(88):POKE IOCB+5,PE
EK(89)
1140 POKE IOCB+8,MB:POKE IOCB+9,SB
1150 TRAP 1900:X=USR(ADR(CIO$)):CLOSE #2
:TRAP 40000
1160 GOTO 1035
1900 TRAP 40000:GRAPHICS 0:? "Blad nr ";
PEEK(195);" podczas czytania zbioru":? F
$
1910 ? "Nacisnij dowolny klawisz aby":?
"uruchomic program ponownie"
1920 CLOSE #1:OPEN #1,4,0,"K:":GET #1,X:
GOTO 12
```

Wydruk 2

```
29950 REM ***************************
29951 REM *    PODPROGRAM INIT       *
29952 REM * wczytuje dane z linii   *
29953 REM * DATA, sprawdza ich pop-*
29954 REM * rawnosc i inicjuje pod-*
29955 REM * program maszynowy, rea- *
29956 REM * lizujacy scrolling tek-*
29957 REM * stu w trybie graficznym*
29958 REM ***************************
30000 BGN=152:POKE 106,BGN-8:GRAPHICS 0
30005 ADRP=38911
30010 PRINT CHR$(125):POSITION 5,5:PRINT
 "Wczytywanie danych z linii DATA"
30011 POSITION 5,6:? "Do wczytania 1020
bajtow"
30012 POSITION 10,10:? "Wczytano bajtow:
"
30013 SKG=0:RESTORE 30600
30015 FOR I=0 TO 101
30017 SKW=0
30020 FOR J=1 TO 10
30025 READ BAJT:SKW=SKW+BAJT:POKE ADRP+I
*10+J,BAJT
30027 POSITION 30,10:? I*10+J
30030 NEXT J
30035 READ BAJT:IF BAJT=SKW THEN GOTO 30
050
30040 POSITION 5,20:? "Popraw linie DATA
 nr: ";:? 30600+I:? CHR$(253)
30045 POSITION 5,21:? "suma kontr. wiers
za ";:? SKW:POSITION 2,1
30046 LIST 30600+I:STOP
30050 SKG=SKG+BAJT
30052 NEXT I
30055 READ BAJT:IF BAJT<>SKG THEN POSITI
ON 5,22:? "blad w liniach DATA":? CHR$(2
53):STOP
30060 POSITION 5,22:? "inicjacja OK...":
? CHR$(253):RETURN
30600 DATA 104,201,007,240,006,170,169,0
01,076,092,1066
30601 DATA 153,104,133,213,104,133,212,1
04,104,141,1401
30602 DATA 027,156,104,104,201,121,144,0
07,162,004,1030
```

```
30603 DATA 169,002,076,092,153,141,246,1
55,104,104,1242
30604 DATA 141,248,155,104,104,141,247,1
55,104,104,1503
30605 DATA 201,041,144,007,162,001,169,0
03,076,092,896
30606 DATA 153,141,249,155,104,104,201,0
05,144,007,1263
30607 DATA 162,000,169,004,076,092,153,1
41,251,155,1203
30608 DATA 160,000,162,000,173,244,002,1
33,208,169,1251
30609 DATA 000,133,207,177,212,201,128,1
44,007,162,1371
30610 DATA 000,169,005,076,092,153,201,0
32,176,006,910
30611 DATA 024,105,064,076,123,152,201,0
96,176,003,1020
30612 DATA 056,233,032,142,250,155,170,2
24,000,240,1502
30613 DATA 016,165,207,024,105,008,133,2
07,165,208,1238
30614 DATA 105,000,133,208,202,208,240,1
74,250,155,1675
30615 DATA 230,207,208,002,230,208,165,2
07,157,033,1647
30616 DATA 156,232,165,208,157,033,156,2
32,200,204,1743
30617 DATA 246,155,240,003,076,084,152,1
65,088,133,1342
30618 DATA 212,165,089,133,213,165,087,1
68,185,180,1597
30619 DATA 155,208,007,162,000,169,006,0
76,092,153,1028
30620 DATA 141,252,155,041,254,141,250,1
55,173,247,1809
30621 DATA 155,240,025,162,000,238,247,1
55,165,212,1599
30622 DATA 024,109,250,155,133,212,165,2
13,105,000,1366
30623 DATA 133,213,232,236,247,155,208,2
36,165,212,2037
30624 DATA 024,109,248,155,133,212,133,2
03,165,213,1595
30625 DATA 105,000,133,213,133,204,172,2
49,155,140,1504
30626 DATA 024,156,173,250,155,024,109,2
50,155,109,1405
30627 DATA 250,155,109,250,155,109,250,1
55,109,250,1792
30628 DATA 155,141,030,156,206,249,155,1
69,001,133,1395
30629 DATA 205,169,158,133,206,162,008,1
72,249,155,1617
30630 DATA 177,212,145,205,169,000,145,2
12,136,016,1417
30631 DATA 245,202,240,052,165,212,024,1
09,250,155,1654
30632 DATA 133,212,165,213,105,000,133,2
13,165,205,1544
30633 DATA 024,109,024,156,133,205,165,2
06,105,000,1127
30634 DATA 133,206,172,249,155,076,044,1
53,168,224,1580
30635 DATA 000,240,005,104,104,202,208,2
51,169,000,1283
30636 DATA 133,213,152,133,212,096,169,0
01,141,244,1494
30637 DATA 155,173,252,155,056,237,250,1
55,141,253,1827
30638 DATA 155,165,203,024,109,250,155,1
33,203,165,1562
30639 DATA 204,105,000,133,204,076,166,1
53,173,244,1458
30640 DATA 155,208,016,160,000,185,017,1
57,145,203,1246
30641 DATA 200,204,030,156,208,245,238,2
44,155,076,1756
30642 DATA 098,228,173,034,002,141,031,1
56,173,035,1071
30643 DATA 002,141,032,156,160,142,162,1
53,169,006,1123
30644 DATA 032,092,228,160,000,177,203,1
53,017,157,1219
30645 DATA 200,204,030,156,208,245,169,0
17,133,212,1574
30646 DATA 141,254,155,169,157,133,213,1
41,255,155,1773
30647 DATA 162,005,160,002,165,212,024,1
09,250,155,1244
30648 DATA 133,212,153,254,155,200,165,2
13,105,000,1590
30649 DATA 133,213,153,254,155,200,202,2
08,231,169,1918
30650 DATA 033,133,212,169,156,133,213,1
69,000,141,1359
30651 DATA 026,156,141,028,156,141,029,1
56,141,010,984
30652 DATA 156,173,253,155,208,010,169,0
03,141,251,1519
30653 DATA 155,169,000,141,028,156,173,2
51,155,201,1429
30654 DATA 004,208,008,169,001,141,251,1
55,141,028,1106
30655 DATA 156,206,251,155,173,251,155,2
40,011,201,1799
30656 DATA 001,208,005,169,016,076,058,1
54,169,032,888
30657 DATA 141,251,155,160,000,177,212,1
33,207,200,1636
30658 DATA 177,212,133,208,160,000,162,0
11,177,207,1447
30659 DATA 140,023,156,141,248,155,041,0
15,024,109,1052
30660 DATA 251,155,168,185,196,155,157,0
11,156,202,1636
30661 DATA 173,248,155,041,240,074,074,0
74,074,024,1177
30662 DATA 109,251,155,168,185,196,155,1
57,011,156,1543
30663 DATA 172,023,156,202,200,192,006,2
08,205,162,1526
30664 DATA 011,160,000,140,245,155,140,0
25,156,185,1217
30665 DATA 254,155,133,214,200,140,023,1
56,185,254,1714
30666 DATA 155,133,215,024,189,011,156,0
42,157,011,1093
30667 DATA 156,202,189,011,156,042,157,0
11,156,172,1252
30668 DATA 249,155,177,214,042,145,214,1
36,016,248,1596
30669 DATA 172,245,155,192,001,208,003,0
76,195,154,1401
30670 DATA 232,238,245,155,076,152,154,2
38,023,156,1669
30671 DATA 160,000,140,245,155,172,023,1
56,202,238,1491
30672 DATA 025,156,173,025,156,201,006,2
08,176,169,1295
30673 DATA 000,141,244,155,173,244,155,2
40,251,173,1776
30674 DATA 027,156,240,011,172,027,156,1
62,255,202,1408
```

```
30675 DATA 208,253,136,208,248,238,026,1
56,173,026,1672
30676 DATA 156,201,008,208,130,238,010,1
56,160,000,1267
30677 DATA 140,026,156,162,011,173,029,1
56,208,062,1123
30678 DATA 173,010,156,205,246,155,208,0
03,076,074,1306
30679 DATA 155,173,028,156,240,030,173,2
51,155,208,1569
30680 DATA 008,169,016,141,251,155,076,0
58,155,201,1230
30681 DATA 016,208,008,169,032,141,251,1
55,076,058,1114
30682 DATA 155,169,000,141,251,155,165,2
12,024,105,1377
30683 DATA 002,133,212,165,213,105,000,1
33,213,076,1252
30684 DATA 061,154,173,029,156,205,024,1
56,016,009,983
30685 DATA 238,029,156,238,029,156,076,1
27,154,169,1372
30686 DATA 001,133,205,169,158,133,206,1
65,203,056,1429
30687 DATA 237,250,155,133,203,165,204,2
33,000,133,1713
30688 DATA 204,162,000,160,000,177,205,1
45,203,200,1456
30689 DATA 204,024,156,208,246,165,205,0
24,109,024,1365
30690 DATA 156,133,205,165,206,105,000,1
33,206,165,1474
30691 DATA 203,024,109,250,155,133,203,1
65,204,105,1551
30692 DATA 000,133,204,160,000,232,224,0
08,208,211,1380
30693 DATA 169,006,174,032,156,172,031,1
56,032,092,1020
30694 DATA 228,169,000,133,212,133,213,0
96,000,000,1184
30695 DATA 000,011,010,021,020,041,040,0
41,041,041,266
30696 DATA 000,000,020,041,000,001,004,0
05,016,017,104
30697 DATA 020,021,064,065,068,069,080,0
81,084,085,637
30698 DATA 000,002,008,010,032,034,040,0
42,128,130,426
30699 DATA 136,138,160,162,168,170,000,0
03,012,015,964
30700 DATA 048,051,060,063,192,195,204,2
07,240,243,1503
30701 DATA 252,255,000,000,000,000,000,0
00,000,000,507
30702 DATA 137764
```

Zaglądanie pod skórę

Pierwsze objawy choroby, to ból, krwawienie, podwyższona temperatura. Właśnie temperatura jest tym wskaźnikiem, który w subtelny sposób informuje o zmianach. A trzeba wiedzieć, że wszystkie organizmy, tkanki, przedmioty, materiały mają różne temperatury. Emitują ciepło począwszy od — 273,15°C. Jest to tzw. temperatura zera bezwzględnego odkryta przez Kelwina.

Powyżej niej, niewidoczne promieniowanie podczerwone daje się zróżnicować. Służą temu detektory podczerwieni. Technika obserwacji i rejestracji obrazów w promieniowaniu cieplnym emitowanym przez dany obiekt nazywa się **termografią**. Wykorzystuje się ją w przemyśle, medycynie oraz wojsku. Tak przecież rozwinęły się noktowizory i inne przyrządy do patrzenia w nocy przez mgłę. Termograficzne przyrządy rozpoznawcze mają detektory podczerwieni, umieszczone w ogniskowej płaszczyźnie układu optycznego. Przetwarza on sygnał cieplny na sygnał elektryczny, a ten z kolei na obraz widzialny.

Powstała **termolokacja**, metoda określania położenia obiektów, a wykorzystująca naturalne promieniowanie energii elektromagnetycznej. Na przykład stacje termolokacyjne nie mają nadajników promieniowania, co uniemożliwia przeciwnikowi ich wykrycie. Na podstawie wskazań detektora podczas przeszukiwania przestrzeni można określić kierunki, w których znajdują się „gorące" obiekty, takie jak lecące samoloty, pociski rakietowe, a także odległości do nich.

Obecnie satelity zwiadowcze, a także meteorologiczne wykorzystują termografię do sporządzania map terenu, ujawniania miejsc rozmieszczenia pocisków rakietowych, określania temperatury powierzchni ziemi, chmur. Tylko niedoskonałości techniki należy przypisać, że nie stworzono jeszcze urządzeń wszystkowidzących i słyszących. Chociaż wizje przedstawiane w filmach i opowiadaniach fantastycznych zaczynają się przybliżać. Dziś już z kosmosu można dokładnie rozpoznawać małe przedmioty, nawet nocą, czy podsłuchiwać rozmowy telefoniczne. Dotychczas zapotrzebowania militarne były, i chyba jeszcze trochę będą, motorem napędowym badań w tym kierunku.

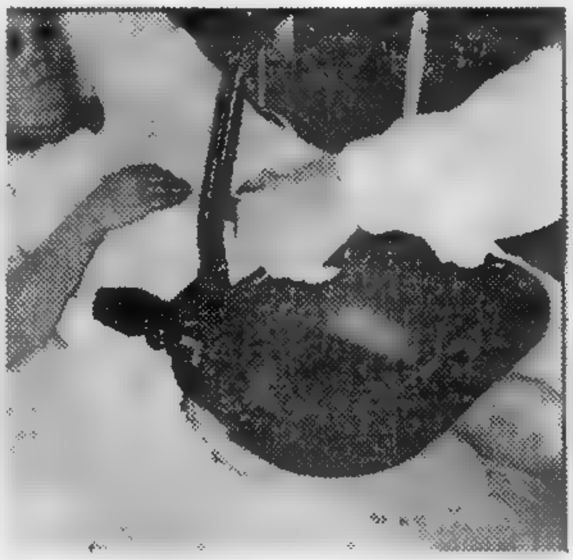
Ale w ślad za wojskowymi zastosowaniami termografii poszły i inne. Powstały bardzo czułe **kamery termowizyjne**. Ujawniają one na przykład miejsca, w których ulatnia się ciepło, choćby w budowlach różnych typów, albo też wykrywają ogniska zapalne w kopalniach odkrywkowych, na zwałowiskach.

Odrębne jednak **zastosowanie termografii znaleziono w medycynie**. Czyli w dziedzinie sprzyjającej człowiekowi. Na początku lat sześćdziesiątych podjęto próby wykrywania stanów zapalnych naczyń w węzłach chłonnych człowieka. Później do określania złośliwości guzów nowotworowych o różnej lokalizacji, ale najczęściej niezbyt głęboko pod skórą. Zaczęto określać żywotność miazgi zębów, a w ginekologii lokalizować położenie łożyska przedniego matki i płodu.

Ale zbyt wysoki koszt urządzeń odstraszał potencjalnych użytkowników. Mogli sobie na nie pozwolić naprawdę bogaci. I dziś nie są to urządzenia tanie. W Polsce jest ich zaledwie kilka. Stosujący je lekarze doceniają wyniki badań uzyskiwane tą drogą.

W Dolnośląskim Centrum Diagnostyki

dokończenie na str. 32



Dzięki ciekłym kryształom można wykrywać zmiany chorobowe w żyłach kończyn.



Na zdjęciu wyraźnie widać, że jedna z kończyn jest chora.

Lilavati (14)

„LABIRYNT"

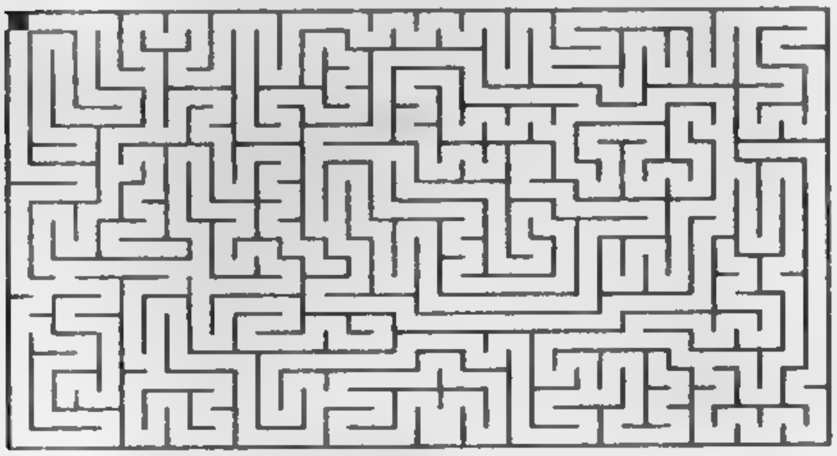
W dzisiejszym odcinku mikrokomputerowych anegdot matematycznych chcę zaprezentować grę, której zasady zostały opisane poniżej

```
Plansza gry jest labirynt, ktory urza-
dzil w swym ogrodzie matematyk angielski
Rouse Ball.
Gra sklada sie z dwoch etapow.
W pierwszym etapie nalezy wyjsc z labi-
ryntu w mozliwie najkrotszym czasie.
Drugi etap polega na obejsciu wszystkich
drog labiryntu z najmniejsza iloscia
miejsc dwukrotnie nawiedzanych.

KLAWISZE STERUJACE:↑,↓,←,→

      NACISNIJ DOWOLNY KLAWISZ
```

Po naciśnięciu dowolnego klawisza na ekranie mikrokomputera pojawi się labirynt, który jest planszą gry.



W pierwszym etapie po wyjściu z labiryntu ukazuje się informacja o czasie wyjścia z labiryntu:

WYJŚCIE Z LABIRYNTU TRWAŁO mMIN sSEK

SPRÓBUJESZ JESZCZE RAZ (T/N)?

Naciskając klawisz „T", grający ponownie zaczyna drogę w etapie 1. Naciśnięcie klawisza „N" spowoduje rozpoczęcie etapu 2, w którym należy przejść cały labirynt z najmniejszą ilością dróg dwukrotnie przechodzonych. Również w tym etapie mierzony jest czas przejścia przez labirynt. Po wyjściu z labiryntu pojawi się komunikat:

DROGA W LABIRYNCIE TRWAŁA mMIN sSEK.

Jeżeli grający nie przeszedł wszystkich dróg labiryntu, zostanie o tym poinformowany:

NIE PRZESZEDŁEŚ WSZYSTKICH DRÓG LABIRYNTU
SPRÓBUJESZ JESZCZE RAZ (T/N)?

Natomiast przejście całego labiryntu spowoduje pojawienie się komunikatu:

PRZESZEDŁEŚ WSZYSTKIE DROGI LABIRYNTU
n % DRÓG DWUKROTNIE.

Poniżej prezentujemy segment sterujący programem. (Instrukcje 10—750)

```
10 '
20 ' Ekran informacyjny
30 '
40 MODE 1:INK 0,26:INK 1,0:INK 2,14:INK 3,6:PAPER 0:BORDER 26:CLS
50 DIM LAB(650,2):lp=1:PEN 1
60 PRINT "Plansza gry jest labirynt, ktory urza-   "
70 PRINT "dzil w swym ogrodzie matematyk angielski "
80 PRINT "Rouse Ball.":PRINT
90 PRINT "Gra sklada sie z dwoch etapow.           "
100 PRINT "W pierwszym etapie nalezy wyjsc z labi-   "
110 PRINT "ryntu w mozliwie najkrotszym czasie.      "
120 PRINT "Drugi etap polega na obejsciu wszystkich "
130 PRINT"drog labiryntu z najmniejsza iloscia      "
140 PRINT "miejsc dwukrotnie nawiedzanych."
150 LOCATE 1,21:PRINT "KLAWISZE STERUJACE:";CHR$(240);",";CHR$(241);",";CHR$(242
);",";CHR$(243)
160 LOCATE 5,25:PRINT "NACISNIJ DOWOLNY KLAWISZ"
170 GOSUB 1000
180 CLEAR INPUT
190 a$=INKEY$
200 IF a$="" THEN GOTO 190
210 '
220 ' Segment organizacyjny
230 '
240 MODE 1
250 WINDOW £1,14,27,11,13:PAPER £1,1:PEN £1,0:CLS £1
260 LOCATE £1,2,2:PRINT £1,"E T A P  ";LP"
270 FOR t=1 TO 2000:NEXT t
280 MODE 1:WINDOW £1,3,35,20,23:PAPER £1,1:PEN £1,0
290 GRAPHICS PEN 1:PLOT LAB(1,1),LAB(1,2)
300 FOR I=2 TO 621
310 IF LAB(I,1)=0 AND LAB(I,2)=0 THEN GOTO 340
```

cd. listingu (10—750)

```
320 DRAWR LAB(I,1),LAB(I,2)
330 NEXT I:GOTO 350
340 I=I+1:PLOT LAB(I,1),LAB(I,2):GOTO 330
350 CZAS=TIME
360 kol=2
370 IF lp=1 THEN x=470:y=246:GOTO 390
380 x=110:y=392
390 GRAPHICS PEN kol:PLOT x-4,y+4:DRAWR 8,0:PLOT x-4,y+2:DRAWR 8,0:PLOT x-4,y:DR
AWR 8,0:PLOT x-4,y-2:DRAWR 8,0:PLOT x-4,y-4:DRAWR 8,
0
400 IF x<100 THEN GOTO 470
410 a$=INKEY$
420 IF a$="" THEN GOTO 410
430 IF ASC(a$)<240 OR ASC(a$)>243 THEN GOTO 410
440 k=ASC(a$)-239
450 ON k GOSUB 790,850,910,970
460 GOTO 390
470 CLEAR INPUT:CZ=(TIME-CZAS)/300
480 MI=INT(CZ/60):SEK=CZ-(MI*60)
490 IF lp=2 THEN GOTO 610
500 CLS £1:LOCATE £1,2,2:PRINT £1,"WYJSCIE Z LABIRYNTU TRWALO ":PRINT £1,"
";USING "££";MI;:PRINT £1," MIN ";USING "££";SEK;:PR
INT £1," SEK."
510 LOCATE 8,25:PRINT "NACISNIJ DOWOLNY KLAWISZ"
520 CLEAR INPUT
530 A$=INKEY$
540 IF A$="" THEN GOTO 530
550 CLS £1:LOCATE £1,2,2:INPUT £1,"SPROBUJESZ JESZCZE RAZ";T$
560 IF T$="T" OR T$="t" THEN MODE 1:GOTO 240
570 IF T$="n" OR T$="N" THEN GOTO 590
580 GOTO 550
590 IF LP=2 THEN END
600 lp=2:MODE 1:GOTO 240
610 bl=0:dr=0:pow=0
620 CLS £1:LOCATE £1,2,2:PRINT £1,"DROGA W LABIRYNCIE TRWALA ":PRINT £1,"
;USING "££";MI;:PRINT £1," MIN ";USING "££";SEK;:PRI
NT £1," SEK."
630 LOCATE 12,25:PRINT "PROSZE CZEKAC"
```

```
640 FOR k=106 TO 534 STEP 2
650 FOR i=122 TO 398 STEP 2
660 IF TEST(k,i)=0 THEN bl=1:GOTO 740
670 IF TEST(k,i)=2 THEN pow=pow+1:GOTO 690
680 IF TEST(k,i)=3 THEN dr=dr+1
690 NEXT i
700 NEXT k
710 CLS £1:LOCATE £1,2,2:PRINT £1,"PRZESZEDLES WSZYSTKIE DROGI":PRINT £1," LABIR
YNTU.";
720 PRINT £1,USING "£££";(pow*100)/(dr+pow);:PRINT £1,"% DROG POPRAWNIE";
730 GOTO 510
740 CLS £1:LOCATE £1,2,2:PRINT £1,"NIE PRZESZEDLES WSZYSTKICH":PRINT £1," DROG L
ABIRYNTU."
750 GOTO 510
```

W programie wykorzystuje się 2 podprogramy.

1. Podprogram do przesuwania kursora w labiryncie.
(Instrukcje 760—1040)

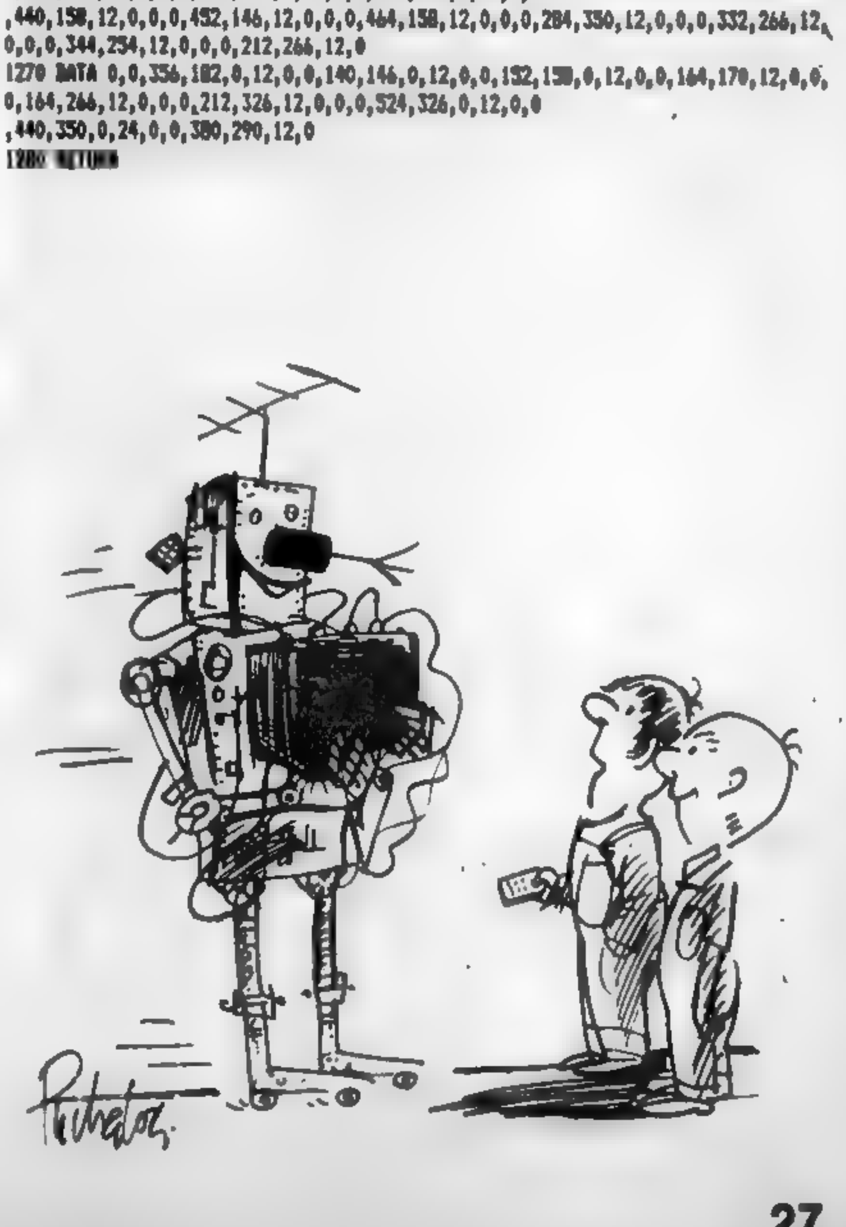
```
760 '
770 ' Podprogramy do przesuwania kursora w labiryncie
780 '
790 IF TEST(x,y+6)=1 OR TEST(x+2,y+6)=1 OR TEST(x-2,y+6)=1 OR TEST(x+4
,y+6)=1 OR TEST(x-4,y+6)=1 THEN RETURN
800 x1=x-4:y1=y-4:a1=8:a2=0
810 IF TEST(x,y+6)=2 THEN kol=3:GRAPHICS PEN 2:GOSUB 1040:GOTO 840
820 IF TEST(x,y+6)=3 THEN kol=2:GRAPHICS PEN 2:GOSUB 1040:GOTO 840
830 kol=2:GOSUB 1030
840 y=y+2:RETURN
850 IF TEST(x,y-6)=1 OR TEST(x+2,y-6)=1 OR TEST(x-2,y-6)=1 OR TEST(x+4
,y-6)=1 OR TEST(x-4,y-6)=1 THEN RETURN
860 y1=y+4:a1=8:a2=0:x1=x-4
870 IF TEST(x,y-6)=3 THEN kol=2:GRAPHICS PEN 2:GOSUB 1040:GOTO 900
880 IF TEST(x,y-6)=2 THEN kol=3:GRAPHICS PEN 2:GOSUB 1040:GOTO 900
890 kol=2:GOSUB 1030
900 y=y-2:RETURN
910 IF TEST(x-6,y)=1 OR TEST(x-6,y+2)=1 OR TEST(x-6,y-2)=1 OR TEST(x-6
,y+4)=1 OR TEST(x-6,y-4)=1 THEN RETURN
920 x1=x+4:y1=y-4:a1=0:a2=8
930 IF TEST(x-6,y)=2 THEN kol=3:GRAPHICS PEN 2:GOSUB 1040:GOTO 960
940 IF TEST(x-6,y)=3 THEN kol=2:GRAPHICS PEN 2:GOSUB 1040:GOTO 960
950 kol=2:GOSUB 1030
960 x=x-2:RETURN
970 IF TEST(x+6,y)=1 OR TEST(x+6,y+2)=1 OR TEST(x+6,y-2)=1 OR TEST(x+6
,y+4)=1 OR TEST(x+6,y-4)=1 THEN RETURN
980 x1=x-4:y1=y-4:a1=0:a2=8
990 IF TEST(x+6,y)=2 THEN kol=3:GRAPHICS PEN 2:GOSUB 1040:GOTO 1020
1000 IF TEST(x+6,y)=3 THEN kol=2:GRAPHICS PEN 2:GOSUB 1040:GOTO 1020
1010 GOSUB 1030
1020 x=x+2:RETURN
1030 GRAPHICS PEN 3
1040 PLOT x1,y1:DRAWR a1,a2:RETURN
```

2. Podprogram wczytywania danych do rysowania labiryn-
tu
(Instrukcje 1050—1280)

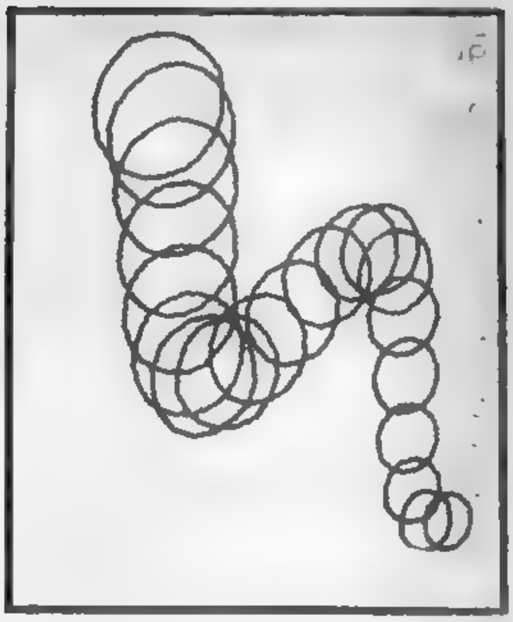
```
1050 '
1060 'Podprogram wczytywania danych do rysowania labiryntu
1070 '
1080 RESTORE 1130
1090 FOR I=1 TO 621
1100 READ LAB(I,1),LAB(I,2)
1110 NEXT I
1120 DATA 332,302,0,12,24,0,0,-12,0,12,24,0,0,-12,0,0
1130 DATA 104,398,432,0,0,-276,-432,0,0,264,0,0,152,386,0,-36,24,0,0,-24,-48,0,0
,48,0,0,164,398,0,-36,12,0,0,0,164,338,-24,0,0,48,-2
4,0,0,-84,36,0,0,0,116,314,24,0,0,0,104,290,36,0,0,0,152,326,0,-48,-36,0,0,-48,1
```

cd. listingu (1050—1280)

```
2,0,0,0,176,386,0,-12,24,0,0,12,0,0
1140 DATA 188,374,0,-48,0,0,212,398,0,-36,-12,0,0,0,188,350,36,0,0,0,224,386,0,-
96,0,0,236,386,0,-60,12,0,0,0,248,398,0,-36,0,0,236,
350,36,0,0,-12,12,0,0,0,260,350,0,36,24,0,0,-12,72,0,0,-24,60,0,0,-12,24,0,0,12,
72,0,0,0,212,338,-12,0,0,-24,0,0
1150 DATA 296,374,0,-48,-36,0,0,-72,12,0,0,-12,12,0,0,-12,-24,0,0,0,224,314,36,0
,0,0,164,302,0,12,48,0,0,-36,36,0,0,0,248,302,-12,0,
0,-48,0,0,224,242,0,12,24,0,0,-12,12,0,0,-48,-24,0,0,0,188,314,0,-60,12,0,0,-12,
-72,0,0,24,0,0,152,242,0,24,12,0,0,24,12,0
1160 DATA 0,12,0,0,200,302,0,-36,12,0,0,-36,36,0,0,0,272,314,36,0,0,48,36,0,0,-2
4,60,0,0,0,320,350,12,0,0,-36,60,0,0,-24,12,0,0,-12,
72,0,0,36,-12,0,0,12,0,0,308,314,12,0,0,-24,48,0,0,24,0,0,368,290,0,-48,12,0,0,0
,368,278,24,0,0,-12,48,0,0,0
1170 DATA 488,386,0,-84,36,0,0,-132,-12,0,0,-24,0,0,524,386,-24,0,0,-24,24,0,0,-
24,-24,0,0,-12,0,0,536,374,-24,0,0,0,536,314,-48,0,0
,0,512,314,0,12,0,0,500,302,0,-48,0,0,488,290,0,-48,24,0,0,48,0,0,500,242,0,-24,
12,0,0,-36,-48,0,0,12,-24,0,0,0
1180 DATA 416,290,0,12,-12,0,0,24,48,0,0,12,24,0,0,-12,0,0,452,326,0,-24,0,0,440
,290,0,12,24,0,0,-12,0,0,428,278,0,36,0,0,416,314,24
,0,0,0,476,398,0,-36,0,0,464,350,0,36,-36,0,0,-24,-36,0,0,0,380,386,0,-36,0,0,39
2,398,0,-24,24,0,0,0,308,302,0,-24,48,0
1190 DATA 0,-48,36,0,0,24,-12,0,0,12,0,0,272,266,0,36,24,0,0,-36,36,0,0,-48,72,0
,0,48,0,0,320,254,0,-48,96,0,0,48,36,0,0,36,0,-60,0,
0,176,182,0,36,24,0,0,12,0,-48,108,0,0,24,-48,0,0,0,308,194,120,0,0,12,72,0,0,-1
2,0,0,152,218,-24,0,0,-24,24,0,0,-24,-24,0
1200 DATA 0,-24,36,0,0,0,164,122,0,108,-24,0,48,0,0,0,116,194,0,-60,36,0,0,0,188
,206,0,-36,36,0,0,-48,0,0,248,146,0,24,72,0,0,12,24,
0,0,-12,24,0,0,12,0,-60,0,0,392,170,0,12,60,0,0,-12,12,0,0,-48,0,0,284,290,0,-36
,12,0,0,-36,-24,0,48,0,0,0,476,170,12,0
1210 DATA 0,-24,36,0,0,-12,0,0,476,134,0,12,24,0,0,-12,0,0,380,194,0,-60,24,0,0,
0,440,182,0,-48,-12,0,24,0,0,0,416,218,48,0,0,48,12,
0,0,0,476,194,0,60,12,0,0,0,284,134,24,0,0,24,-24,0,72,0,0,-24,0,0,236,182,0,-48
,24,0,0,24,0,0,392,146,24,0,0,-24,0,24,12,0
1220 DATA 0,24,0,0,176,134,0,24,36,0,-12,0,0,-24,12,0,0,0,224,182,0,24,24,0,0,0,
212,194,0,24,48,0,0,0,272,122,0,24,24,0,0,0,320,122,
0,24,0,0,344,122,0,24,0,0,368,398,0,-36,0,0,404,386,24,0,0,-36,0,0,332,146,0,24,
0,0,380,158,24,0,0,12,0,0,428,230,0,24,0,0
1230 DATA 164,254,24,0,0,0,176,278,12,0,0,0,308,230,0,36,0,0,452,362,0,24,0,0,35
6,326,0,-24,0,0,272,350,0,24,0,-12,12,0,0,0,296,398,
0,-12,0,0,320,398,0,-12,0,0,344,398,0,-12,0,0,116,182,24,0,0,0,104,218,12,0,0,0,
140,206,24,0,0,0,260,326,0,12,-12,0,0,0
1240 DATA 308,374,0,12,0,0,332,374,0,12,0,0,356,374,0,12,0,0,332,242,12,0,0,0,34
4,230,12,0,0,0,344,230,12,0,0,0,488,122,0,12,0,0,512
,122,0,12,0,0,500,134,0,12,0,0,500,158,0,24,0,0,248,266,12,0,0,0,248,290,12,0,0,
0,284,206,0,-12,12,0,0,0,308,338,12,0,0,0
1250 DATA 344,326,0,12,24,0,0,-12,0,0,392,326,0,12,0,0,188,398,0,-12,0,0,320,326
,12,0,0,0,332,302,0,12,0,0,344,290,0,12,0,0,380,314,
0,12,0,0,140,254,0,24,0,0,188,122,0,24,0,0,212,146,12,0,0,0,116,206,12,0,0,0,236
,230,0,12,0,0,272,182,0,12,0,0,272,158,0,12
1260 DATA 0,0,416,158,0,24,0,0,524,158,12,0,0,0,512,230,12,0,0,0,488,218,12,0,0,
0,476,230,12,0,0,0,488,182,0,12,0,0,440,218,0,24,0,0
,440,158,12,0,0,0,452,146,12,0,0,0,464,158,12,0,0,0,284,350,12,0,0,0,332,266,12,
0,0,0,344,254,12,0,0,0,212,266,12,0
1270 DATA 0,0,356,182,0,12,0,0,140,146,0,12,0,0,152,158,0,12,0,0,164,170,12,0,0,
0,164,266,12,0,0,0,212,326,12,0,0,0,524,326,0,12,0,0
,440,350,0,24,0,0,380,290,12,0
1280 RETURN
```



PROCEDURA COPY DLA DRUKARKI D-100 (i podobnych)



Efekt graficznego kopiowania ekranu przez drukarkę D-100

ZX Spectrum w połączeniu z ZX Interface umożliwia współpracę poprzez złącze RS z innymi urządzeniami. W moim wypadku urządzeniem tym jest drukarka mozaikowa D-100

Na poziomie znaków ASCII oba urządzenia znakomicie się „rozumieją". Jednak druk znaków graficznych lub kopiowanie ekranu bez odpowiedniego oprogramowania jest niemożliwe.

Przedstawiam program, który umożliwia wydrukowanie na drukarce aktualnego ekranu. Program zajmuje obszar pamięci od adresu 65000. Wykorzystuje tryb graficzny drukarki.

Ustalane parametry:
- POKE 65511, numer kanału transmisji
- POKE 65512, ilość nadruków linii
- POKE 65513, tabulacja pozioma rys.

Program wykorzystuje kanał transmisji RS typu „b", czyli prostą transmisję bajtów (patrz linia 2010). W tym wypadku wysłanie bajtu do drukarki z poziomu maszynowego polega na wywołaniu restartu 10H (RST 10H) z wartością bajtu umieszczoną w akumulatorze. Prędkość transmisji ustala się rozkazem: FORMAT „b"; prędkość transmisji. W programie została przyjęta prędkość 4800 (patrz linia 2020).

Program jest tak skonstruowany, że w prosty sposób można go przystosować do innego mechanizmu transmisji. Wystarczy bowiem rozkaz RST 10H zastąpić odpowiednią procedurą transmitującą bajty do drukarki.

Uzupełnienie:

Funkcje przełącznika K-200 drukarki do współpracy z RS

numer przełącznika	oznaczenie	funkcja
1		ON — procedura XON/XOFF OFF — użycie linii LBRM
2	A	prędkość transmisji
3	B	prędkość transmisji
4	C	prędkość transmisji
5	P	OFF — z parzystością ON — bez parzystości
6	P/N	ON — parzystość nieparzysta OFF — parzystość parzysta
7	X	liczba bitów stopu
8	Y	liczba bitów stopu

prędkość transmisji:

	110	150	300	600	1200	2400	4800	9600
A	ON	OFF	ON	OFF	ON	OFF	ON	OFF
B	ON	ON	OFF	OFF	ON	ON	OFF	OFF
C	ON	ON	ON	ON	OFF	OFF	OFF	OFF

liczba bitów stopu

	1 bit	1,5 bitu	2 bity
X	OFF	ON	OFF
Y	ON	OFF	OFF

Ustawienie przełączników umożliwiające współpracę drukarki z prezentowanym programem.

1—OFF, 2—ON, 3—OFF, 4—OFF, 5—ON, 6—ON, 7—OFF, 8—ON

KRZYSZTOF POŹNIAK

```
   1 REM ***********************
   2 REM *                     *
   3 REM *              ©1989  *
   4 REM *                     *
   5 REM *   Krzysztof Pozniak *
   6 REM *                     *
   7 REM * Procedura COPY dla  *
   8 REM *    drukarki D 100   *
   9 REM *                     *
  10 REM ***********************
 100 REM wczytanie procedury
 110 CLEAR 64999
 120 RESTORE 9000
 130 LET a=65000
 140 FOR k=1 TO 21
 150 LET z=1: LET l1=0: LET l2=0
 160 FOR y=1 TO 11
 170 READ x: REM POKE a,x
 180 LET l1=l1+z*x: LET z=-z: LE
T l2=l2+x
 190 LET a=a+1: NEXT y
 200 READ lp1,lp2
 210 IF l1<>lp1 OR l2<>lp2 THEN
PRINT "blad w linii ";8990+10*k:
 STOP
 220 NEXT y: NEXT k
1000 REM tworzenie rysunku
1110 FOR l=1 TO 20
1120 LET r=31-l
1121 LET x=30+10*l
1122 LET y=l*PI/20
1123 LET y=80+INT 60*(SIN y*COS
y+COS (3*y))
1130 CIRCLE x,y,r
1140 NEXT l
2000 REM ustalenie paramet. COPY
2010 CLEAR #: OPEN #4,"b"
2020 FORMAT "b";4800
2030 POKE 65511,4:  REM kanal 4
2040 POKE 65512,1:  REM 1 nadruk
2050 POKE 65513,30: REM TAB 30
3000 REM wywolanie procedury
3010 RANDOMIZE USR 65000

9000 DATA 62,188,50,225,255,58,2
31,255,205,1,22,98,1552
9010 DATA 62,15,215,62,27,215,62
,49,215,62,13,191,997
9020 DATA 215,175,50,227,255,33,
```

```
207,254,34,221,255,106,1926
9030 DATA 58,225,255,71,62,187,5
,144,14,0,205,-28,1226
9040 DATA 176,34,235,42,221,255,
115,35,114,35,34,494,1296
9050 DATA 221,255,58,227,255,60,
50,227,255,58,225,237,1891
9060 DATA 255,61,50,225,255,254,
12,40,7,58,227,168,1444
9070 DATA 255,254,7,32,207,62,32
,50,226,255,33,107,1413
9080 DATA 221,254,34,223,255,6,0
,62,255,24,2,198,1336
9090 DATA 24,178,119,35,16,252,3
3,207,254,58,227,-57,1403
9100 DATA 255,50,228,255,135,61,
95,22,0,25,34,334,1160
9110 DATA 221,255,42,221,255,86,
43,94,52,43,26,-60,1330
9120 DATA 34,221,255,6,8,42,223,
255,7,48,2,-43,1101
9130 DATA 203,190,203,14,35,16,2
46,58,228,255,61,443,1509
9140 DATA 50,228,255,32,220,34,2
23,255,58,226,255,286,1836
9150 DATA 61,50,226,255,32,190,5
8,232,255,50,230,65,1639
9160 DATA 255,62,9,215,58,233,25
5,215,6,0,33,-109,1341
9170 DATA 221,254,126,215,35,62,
255,215,16,248,62,-279,1709
9180 DATA 13,215,58,230,255,61,5
0,230,255,32,224,87,1623
9190 DATA 62,10,215,58,225,255,2
54,12,32,136,62,379,1321
9200 DATA 27,215,62,53,215,62,2,
205,1,22,201,-49,1065
```

```
;*******************************
;*                      ?1989 *
;*     Krzysztof Pozniak       *
;*                             *
;*     Procedura COPY na       *
;*   drukarke D 100 i inne     *
;*                             *
;*******************************
;
ht      equ  9
lf      equ  10
cr      equ  13
max_y   equ  188
end_y   equ  12
max_x   equ  32
apixel  equ  22b0h
open    equ  1601h
        org  65000
start   ld   a,max_y
        ld   (num_y),a
        ld   a,(kanal)
        call open
        ld   a,0fh
        rst  10h
        ld   a,1bh
        rst  10h
        ld   a,31h
        rst  10h
        ld   a,cr
        rst  10h
loop61  xor  a
        ld   (echonb),a
        ld   hl,onelin
        ld   (adline),hl
loop2   ld   a,(num_y)
        ld   b,a
        ld   a,max_y-1
        dec  a
        sub  b
        ld   c,0
        call apixel
        ex   de,hl
        ld   hl,(adline)
        ld   (hl),e
        inc  hl
        ld   (hl),d
        inc  hl
        ld   (adline),hl
        ld   a,(echonb)
        inc  a
        ld   (echonb),a
        ld   a,(num_y)
        dec  a
        ld   (num_y),a
        cp   end_y
        jr   z,end1
        ld   a,(echonb)
        cp   7
        jr   nz,loop2
end1    ld   a,max_x
        ld   (num_x),a
        ld   hl,plotbt
        ld   (echopl),hl
        ld   b,0
        ld   a,255
        jr   loop1
loop6   jr   loop61
loop1   ld   (hl),a
        inc  hl
        djnz loop1
loop4   ld   hl,onelin
        ld   a,(echonb)
        ld   (memnub),a
        add  a,a
        dec  a
        ld   e,a
        ld   d,0
        add  hl,de
        ld   (adline),hl
loop5   ld   hl,(adline)
        ld   d,(hl)
        dec  hl
        ld   e,(hl)
        inc  (hl)
        dec  hl
        ld   a,(de)
        ld   (adline),hl
        ld   b,8
        ld   hl,(echopl)
loop3   ;rlc (hl)
        rlca
        jr   nc,npoint
        res  7,(hl)
npoint  rrc  (hl)
        inc  hl
        djnz loop3
        ld   a,(memnub)
        dec  a
        ld   (memnub),a
        jr   nz,loop5
        ld   (echopl),hl
        ld   a,(num_x)
        dec  a
        ld   (num_x),a
        jr   nz,loop4
        ld   a,(iledrk)
        ld   (echoil),a
loop8   ld   a,ht
        rst  10h
        ld   a,(tab)
        rst  10h
        ld   b,0
        ld   hl,plotbt
loop7   ld   a,(hl)
        rst  10h
        inc  hl
        ld   a,255
        rst  10h
        djnz loop7
        ld   a,cr
        rst  10h
        ld   a,(echoil)
        dec  a
        ld   (echoil),a
        jr   nz,loop8
        ld   a,lf
        rst  10h
        ld   a,(num_y)
        cp   end_y
        jr   nz,loop6
        ld   a,1bh
        rst  10h
        ld   a,35h
        rst  10h
        ld   a,2
        call open
        ret
;
onelin  ds   14
plotbt  ds   256
adline  ds   2
echopl  ds   2
num_y   ds   1
num_x   ds   1
echonb  ds   1
memnub  ds   1
countr  ds   1
echoil  ds   1
kanal   db   4
iledrk  db   1
tab     db   30
        end
```

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

LEADLESS PLUGGABLE IC PACKAGE - mikroukład bezwyprowadzeniowy wkładalny,
LEAF NODE - wierzchołek drzewa nie posiadający podwierzchołków,
LEAPFROG TEST - test przeskokowy (program testujący pamięć),
LEARNING - uczenie (się), nauczanie,
LEARNING AUTOMATON - maszyna ucząca się,
LEARNING MACHINE - patrz: LEARNING AUTOMATON,
LEARNING PROGRAM - program samouczący się,
LEAST COMMON DENOMINATOR - najmniejszy wspólny mianownik,
LEAST COMMON MULTIPLE - najmniejsza wspólna wielokrotność,
LEAST FREQUENTLY USED REMOVAL - usunięcie rzadko wykorzystywanych (w systemie sterowania pamięcią wirtualną algorytm usuwania stronic pamięci, do których odwołania występowały najrzadziej),
LEAST RECENTLY USED REMOVAL - w systemie sterowania pamięcią wirtualna algorytm zastępowania segmentów pamięci, do których dawno nie było odwołania,
LEAST-SIGNIFICANT BIT - bit najmniej znaczący,
LEAST-SIGNIFICANT CHARACTER - znak o najniższej wadze,
LEAST-SIGNIFICANT DIGIT - cyfra najmniej znacząca (w liczbie),
LEAVES - gałęzie końcowe drzewa (zapisy (rekordy) najniższego poziomu hierarchicznej bazy danych),
LECTURE - wykład,
LEFT CURLY BRACKET - lewy nawias klamrowy,
LEFT JUSTIFIED - wyrównany po lewej stronie,
LEFT-JUSTIFY - wyrównywać w lewo,
LEFT-MOST - położony najbardziej na lewo,
LEFT-OVER CARD - karta pozostawiona omyłkowo w maszynie,
LEFT PARENTHESES - lewy nawias okrągły,
LEFT-RECURSIVE GRAMMAR - gramatyka lewostronnie rekursyjna,
LEFT-SIDED - lewostronny,
LEFTSIDE VALUE - 1. wyrażenie lewostronne (po lewej stronie operatora przyporządkowania), 2. adres zmiennej,
LEFT SQUARE BRACKET - lewy nawias kwadratowy,
LEG - gałąź (programu),
LEGAL - prawny, legalny, ustawowy,
LEGAL COMMUNICATIONS - komunikacja właściwa,
LEGALIZATION - legalizacja,
LEGALLY BINDING - prawnie obowiązujący, prawomocny,
LEGAL PROTECTION - ochrona prawna,
LEGEND - legenda (objaśnienia),
LEGIBILITY - czytelność,
LEGIBLE - czytelny,
LEGITIMATE VALUE - wartość dopuszczalna, wartość uzasadniona,
LEMMA OF AN ENTRY - wyraz hasłowy (np. pozycji słownika),
LEND - pożyczać,
LENGTH - długość,
LENGTHEN - wydłużać, przedłużać,
LENGTH OVER ALL - długość całkowita,
LEPORELLO - leporello (formularz ciągły złożony w harmonikę),
LESS OR EQUAL - mniejsze lub równe,
LESS-THEN SIGN - znak mniejszości, "mniejsze od",
LESS THEN - mniejsze od,
LET ALONE - pominąć,opuścić,
LETTER - 1. litera, 2. czcionka, 2. list,
LETTERING - liternictwo,
LETTER/NUMERIC SHIFT - przełączanie litery/cyfry, przełączanie czcionek,
LETTERPRESS PRINTING - druk typograficzny,
LETTER-QUALITY PRINTER - drukarka o wysokiej jakości druku,
LETTERS/FIGURES SHIFT - patrz: LETTER/NUMERIC SHIFT,
LETTERS SHIFT - przełączanie na litery,
LETTER STAMP - znacznik literowy,
LEVEL - 1.poziom, 2. poziomy, 3. poziomować, wyrównywać,
LEVEL OF ABSTRACTION - poziom abstrakcji,
LEVEL OF CONFIDENCE - poziom ufności,
LEVEL OF MAPPING - poziom odzwierciedlenia, poziom odwzorowania (bazy danych),
LEVEL OF SIGNIFICANCE - poziom znaczenia błędu,
LEVEL-SENSITIVE - wrażliwy na poziom,
LEVEL-TRIGGERED - uruchamiany poziomem (np. sygnału),
LEXER - patrz: LEXICAL ANALYZER,
LEXICAL ANALYZER - analizator leksykalny,
LEXICOGRAPHICAL LABEL - etykieta leksykograficzna,
LEXICOGRAPHIC ORDER - porządek leksykograficzny,
LEXICON - leksykon, słownik,
LFU - patrz: LEAST FREQUENTLY USED REMOVAL,
LIABLE TO FEE - podlegający opłacie,
LIASON - połączenie, złącze, potencjalna możliwość stworzenia połączenia pomiędzy dwoma węzłami sieci transmisji danych,
LIBERATED LINE - linia zwolniona, linia wydzielona,
LIBR - patrz: 1. LIBRARIAN, 2. LIBRARY,
LIBRARIAN - patrz: LIBRARY MAINTENANCE PROGRAM,
LIBRARY - biblioteka,
LIBRARY DISC - dysk biblioteczny,
LIBRARY FUNCTION - funkcja biblioteczna,
LIBRARY MAINTENANCE PROGRAM - program zarządzający biblioteką,
LIBRARY OF TAPES - taśmoteka,
LIBRARY PROGRAM - program biblioteczny,
LIBRARY ROUTINE - patrz: LIBRARY PROGRAM,
LIBRARY TABLE - spis biblioteki (programów),
LIC - patrz: LINEAR INTEGRATED CIRCUIT,
LICENCE - pozwolenie, licencja,
LICENCE HOLDER - posiadacz licencji, licencjobiorca,
LICENCE PROGRAM - program licencyjny,
LIFE - trwałość, okres trwania, żywotność,
LIFE-PERIOD - trwałość, okres trwania,
LIFE TEST - próba trwałości,
LIFO - patrz: LAST-IN FIRST-OUT,
LIFO QUEUE - organizacja stosu typu LIFO,
LIFT RESTRICTIONS - znieść ograniczenia,
LIGHT - 1. lekki, jasny, 2. zapalać, rozpalać, 3. lampka sygnalizacyjna,
LIGHT-ACTIVATED - działający pod wpływem światła,
LIGHT BEAMS - promienie światła (przy odczycie fotoelektrycznym),
LIGHT BUTTON - klawisz świetlny (znak lub napis na ekranie działający jako klawisz funkcyjny),
LIGHTGUIDE CABLE - kabel światłowodowy, światłowód,
LIGHT GUN - pióro świetlne, pisak świetlny,
LIGHT LINE - linia cienka,
LIGHT PEN - pióro świetlne, pisak świetlny,
LIGHT PEN SENSIVITY - wrażliwość na pióro świetlne,
LIGHT PIPE - światłowód,
LIKEN TO - przyrównywać (coś do czegoś),
LIMIT - granica, limit, także: ograniczać, limitować,
LIMITATION - ograniczenie, przedział,
LIMITER - ogranicznik,
LIMITING - graniczny, ograniczający,
LIMITING FACTOR - czynnik ograniczający,
LIMITING MARK - znacznik ograniczający,
LIMIT OF ERROR - granica błędu, przedział błędów,
LIMIT OF INTEGRATION - przedział integracji,
LIMIT VALUE - wartość graniczna,
LINAGE - liczba wierszy (drukowanego tekstu),
LINE - 1. linia, wiersz, (np. na ekranie, w programie), 2. linia łączności, linia transmisji, 3. seria,
LINE ADAPTER - adapter łącza,
LINEAL - patrz: LINEAR,
LINEAR - liniowy, linearny,
LINEAR-BOUNDED AUTOMATON - automat z liniowo ograniczoną pamięcią,
LINEAR CODE - kod liniowy,
LINEAR DISTRIBUTION - rozkład liniowy,
LINEAR FUNCTION - przekształcenie liniowe,
LINEAR INTEGRATED CIRCUIT - układ scalony liniowy,
LINEARITY - liniowość,
LINEAR PROGRAM - program liniowy,
LINEAR PROGRAMMING - programowanie liniowe,
LINEAR-PROGRAMMING LANGUAGE - język do zadań programowania liniowego,
LINEAR SEARCH - przeszukiwanie liniowe, przeszukiwanie szeregowe,
LINEAR-SELECTION MEMORY - pamięć o wybieraniu liniowym,
LINEAR TRANSFORMATION - przekształcenie liniowe,
LINE-AT-A TIME PRINTER - drukarka wierszowa,
LINE ATTRIBUTE - atrybut linii (w grafice komputerowej - typ, szerokość i kolor linii),
LINE BUFFER - patrz: LINE ADAPTER,
LINE-BY-LINE SCANNING - wybieranie liniowe,
LINE CLIPPING - patrz: CLIPPING,
LINE COMMUNICATION - transmisja jednoprzewodowa,
LINE CONTROL BLOCK - blok sterowania kanałem (dane zawierające parametry kanału transmisji danych, np. szybkość transmisji, sposób synchronizacji oraz informacje o stanie bieżącym),
LINE DEFINITION - rozdzielczość liniowa,
LINE EDITOR - edytor wierszowy,
LINE EQUIPMENT - urządzenie (teletransmisyjne) liniowe,
LINE FEED - znak wysuwu (papieru), przesuw wierszowy, posuw wierszowy,
LINEFEED - przesuw do nowej linii, nowa linia,
LINE FOLDING - składanie linii (kontynuacja linii logicznej w kolejnej linii fizycznej),

LINE FREQUENCY - częstotliwość linii, częstotliwość odchylania,
LINE GRAPHIC - grafika konturowa,
LINE JUMP SCANNING - wybieranie międzyliniowe,
LINE LAYOUT - rozmieszczenie liniowe,
LINE NUMBER - numer linii,
LINE OF CODE - wiersz (tekstu) programu,
LINEOUT - uszkodzenie linii (łączności),
LINE PRINTER - drukarka wierszowa,
LINE PRINTING - drukowanie wierszowe (jednoczesne pisanie treści całego wiersza),
LINE PROTOCOL - protokół linii łączności,
LINE SLIP - przesunięcie obrazu lub jego części w kierunku poziomym,
LINE SPACING - odstęp wiersza,
LINES PER INCH - ilość wierszy na cal,
LINES PER MINUTE - ilość wierszy na minutę,
LINE STYLE - rodzaj linii (np. ciągła, kreskowa, kropkowa, osiowa),
LINE SUPPRESSION - wygaszanie linii,
LINE SWITCHING - komutacja łączy,
LINE TYPE - patrz: LINE STYLE,
LINE UP - wyrównywać, ustawić w szeregu,
LINE WEIGHT - grubość linii,
LINEWIDTH - szerokość linii,
LINING PEN - grafion,
LINK - 1. połączenie, powiązanie, łączenie, 2. zestawiać, łączyć, 3. wskazywać, 4. wskaźnik, odsyłacz,
LINKABLE - przydatny do konsolidacji,
LINKAGE - 1. konsolidacja, połączenie, łączenie, 2. zestawianie, montowanie, sprzężenie,
LINKAGE EDITOR - program łączący,
LINKAGE LOADER - ładowacz konsolidatora,
LINKAGE RELEASE - zwalnianie połączenia,
LINK CONTROL PROTOCOL - protokół sterowania łączem,
LINKED COMPUTERS - komputery sprzężone,
LINK EDIT - konsolidować, łączyć,
LINK EDITOR - konsolidator,
LINKEDNESS - spójność, spoistość,
LINKED SUBROUTINE - 1. podprogram biblioteczny, podprogram wewnętrzny, 2. podprogram zamknięty,
LINKER - program łączący, konsolidator,
LINK FILE - plik (zbiór) połączeń, plik (zbiór) łączników (konsolidatora),
LINKING LOADER - ładowacz konsolidujący, ładowacz dynamiczny,
LINK LEVEL - poziom łącza,
LINK LIBRARY - biblioteka połączeń,
LINT - pył (pochodzący od kart dziurkowanych),
LIOCS - patrz: LOGICAL INPUT/OUTPUT CONTROL SYSTEM,
LIPS - patrz: LOGICAL INFERENCES PER SECOND,
LIQUID CRYSTAL - ciekły kryształ,
LIQUID CRYSTAL DISPLAY - wskaźnik na ciekłych kryształach, wskaźnik ciekłokrystaliczny,
LISP - patrz: LIST PROCESSING (LANGUAGE),
LISP MACHINE - komputer LISP, zapewniający sprzętową interpretację programów napisanych w języku Lisp,
LIST - 1. lista, tabulogram, 2. umieszczać w spisie,
LIST DEVICE - (logiczne) urządzenie drukujące,
LIST-DIRECTED - sterowany listą,
LISTENER - odbiornik (informacji),
LIST-FORM OF MACROINSTRUCTION - tablicowa postać makroinstrukcji,
LISTING - wydruk, listing, listowanie,
LIST OF MODIFICATIONS - lista modyfikacji,
LIST PICTURE - obraz listy,
LIST PROCESSING - przetwarzanie łańcuchowe (danych),
LIST-PROCESSING LANGUAGE - autokodowy język programowania do zapisu algorytmów przetwarzania list danych w postaci łańcuchów,
LIST REPRESENTATION - przedstawienie (danych) w postaci listy,
LIST STRUCTURE - struktura listowa danych,
LIST WITH ONE-WAY POINTERS - lista jednokierunkowa,
LIT - patrz: LITERAL,
LITERAL - 1. stała znakowa, literał, 2. literowy,
LL(k) GRAMMAR - gramatyka z ograniczonym prawym kontekstem,
LOAD - wprowadzać, ładować, obciążać,
LOADABLE DRIVER - napęd ładowalny, napęd nierezydentny,
LOADABLE FONT - font ładowalny (kroje liter ładowane przez program do pamięci urządzenia drukującego lub generatora znaków monitora),
LOADABLE FORMAT - format wsadowy,
LOAD ADDRESS - adres ładowania,
LOAD-AND-GO - wprowadź i wykonaj (metoda przetwarzania),
LOADED DATA BASE - zapełniona baza danych,
LOADER - program wprowadzający, program ładujący,
LOADING - wprowadzanie danych (do pamięci wewnętrznej),
LOADING FACTOR - współczynnik załadowania, współczynnik wykorzystania,
LOADING PROGRAM - program wprowadzający,
LOAD-IN LINE - szyna wprowadzania, szyna wprowadzająca,
LOAD INSTRUCTION - rozkaz wprowadzania,
LOAD KEY - klawisz wprowadzania (np. programu do pamięci),
LOAD LIBRARY - biblioteka programów ładujących,
LOAD LIBRARY TABLE - tabela wprowadzania programów,
LOAD LINE - linia wprowadzania danych,
LOAD MAP - karta przydziału pamięci (tablica przedstawiająca rozmieszczenie i długość segmentów programu w pamięci),
LOAD MODULE - moduł ładujący,
LOAD SHARING - podział obciążenia,
LOADTIME SUBROUTINE LIBRARY - biblioteka podprogramów załadowczych,
LOC - patrz: LOCATION,
LOCAL - miejscowy, lokalny,
LOCAL AREA NETWORK - lokalna sieć komputerowa,
LOCAL BATCH PROCESSING - przetwarzanie partiowe lokalne,
LOCAL COMPUTER NETWORK - lokalna sieć komputerowa,
LOCAL EXCHANGE - lokalna centrala telefoniczna,
LOCAL IDENTIFIER - identyfikator lokalny, nazwa lokalna,
LOCALIZE - lokalizować,
LOCAL LINE - linia lokalna,
LOCAL MODE - tryb autonomiczny,
LOCAL NAME - nazwa lokalna,
LOCAL OPTIMIZATION - optymalizacja lokalna,
LOCAL VARIABLE - zmienna lokalna,
LOCATE - 1. umieszczać, rozmieszczać, 2. umiejscawiać, ustalać położenie, oznaczć,
LOCATION - 1. miejsce w pamięci, komórka pamięci, 2. lokalizacja, umiejscowienie, ustalenie położenia, usytuowanie,
LOCATION COUNTER - wskaźnik adresu,
LOCATION POINTER - wskaźnik pozycji,
LOCATOR - urządzenie wprowadzania pozycji,
LOCK - 1. zablokowanie, blokada (kod, struktura danych lub program stosowane w sterowaniu dostępem do pliku danych i synchronizacji procesów), 2. zabezpieczać, 3. blokować (np. klawiaturę), unieruchamiać,
LOCKED FILE - plik (zbiór) zabezpieczony,
LOCKING - 1. unieruchomienie, blokowanie, zabezpieczenie, 2. zamykanie na klucz,
LOCKING GATE - bramka zamykająca,
LOCKING SIGNAL - sygnał synchronizujący, sygnał blokujący,
LOCKOUT - blokada, wstrzymanie,
LOCK-UP - stała blokada,
LOG - 1. dziennik, plik rejestracji, protokół, 2. rejestrować (zapisywać informację o niektórych zdarzeniach w specjalnym pliku-dzienniku), 3. patrz: LOGARITHM,
LOGARITHM - logarytm,
LOGARITHMIC SEARCH - przeszukiwanie metodą połowienia,
LOG BLOSIM - patrz: LOGIC BLOCK SIMULATION,
LOG BOOK - dziennik, rejestr,
LOGGER - przyrząd rejestrujący, urządzenie rejestrujące, rejestrator,
LOGGING - rejestracja danych, zapisywanie,
LOGGING IN - wejście (do systemu),
LOGGING OUT - wyjście (z systemu),
LOGIC - 1. układ logiczny, 2. logika, 3. algorytm,
LOGIC-0 LEVEL - poziom zera logicznego, poziom zerowy,
LOGIC-1 LEVEL - poziom logicznej jedynki,
LOGICAL - 1. logiczny, 2. logicznie,
LOGICAL ADDITION - sumowanie logiczne, dysjunkcja,
LOGICAL ADDRESS - adres logiczny,
LOGICAL BLOCK NUMBER - logiczny numer bloku,
LOGICAL CHANNEL - kanał logiczny,
LOGICAL CHANNEL NUMBER - numer logiczny kanału,
LOGICAL CHOICE - wybór logiczny,
LOGICAL CIRCUIT - 1. łącze logiczne, 2. układ logiczny,
LOGICAL COMPARISON - porównanie logiczne,
LOGICAL DATA BASE - logiczna baza danych,
LOGICAL DESIGN - schemat logiczny, projekt logiczny,
LOGICAL DEVICE - urządzenie logiczne,
LOGICAL DEVICE NAME - logiczna nazwa urządzenia,
LOGIC(AL) ERROR - błąd logiczny,
LOGICAL EXPRESSION - wyrażenie logiczne,
LOGICAL FILE - plik (zbiór) logiczny,
LOGICAL INFERENCES PER SECOND - liczba wniosków logicznych na sekundę,
LOGICAL INPUT DEVICE - logiczne urządzenie wejściowe,
LOGICAL INPUT/OUTPUT CONTROL SYSTEM - programy standardowe pracy buforowej i blokowania danych,
LOGICAL INSTRUCTION - instrukcja logiczna,
LOGICAL LEVEL - poziom logiczny,
LOGIC(AL) LINE - wiersz logiczny,

Zaglądanie pod skórę

dokończenie ze str. 25

Medycznej „Dolmed" we Wrocławiu na termografie stosuje się badania profilaktyczne, szczególnie przy podejrzeniach zmian nowotworowych. Otóż komórki rakowe rozmnażając się powodują większe ukrwienie danego miejsca. A krew jako nośnik ciepła powoduje większą jego emisję w danym miejscu. Teraz już tylko kwestia interpretacji obrazu. Szczególnie dane te potrzebne są przy określaniu pola zajętego chorobą, a następnie przy obserwacji pooperacyjnej.

Bardzo istotne są pomiary termograficzne w wypadkach **oceny przyjmowania się przeszczepów**. Metodę tę wykorzystuje się m.in. w szpitalu w Trzebnicy (doc. Ryszard Kociemba), znanym ośrodku transplantologii. We wrocławskim „Dolmedzie" rocznie bada się 20 tys. kobiet z podejrzeniami zmian nowotworowych. Dzięki tej technice uratowano wiele istnień ludzkich.

Kolejny przykład pochodzi z Kliniki Chirurgii Dziecięcej Instytutu Pediatrii Akademii Medycznej w Łodzi. Dr **Wojciech Kuzański pokazuje fotografie oparzeń dzieci**, po czym omawia:

— *Wydawało się, że należałoby dziecku półtorarocznemu przeszczepić skórę na całej wierzchniej części stopy. Obraz termowizyjny ukazał, że tylko kilka centymetrów kwadratowych (gdzie wystąpiła zgorzel) trzeba przeszczepić.*

I tak zrobiono już w trzecim dniu po oparzeniu. Należy dodać, że normalnie, czyli bez termografu, zmiany te zauważone mogły być dopiero conajmniej po siedmiu dniach. W efekcie zastosowanego zabiegu na o wiele mniejszej powierzchni ciała i szybko, dziecko zostało wypisane do domu już po 14 dniach pobytu w szpitalu, z zagojoną raną. Gwoli dodania, do tej kliniki trafia rocznie około 200 dzieci z poparzeniami. Kamera termowizyjna nie jest własnością szpitala lecz kombinatu w Bełchatowie. Ale tam wykorzystuje się ją zaledwie kilka razy w roku **do badania złóż węgla brunatnego**.

Burzliwe zastosowanie termografii nastąpiło w drugiej połowie lat siedemdziesiątych. Dzięki tej technice udaje się rozpoznawać stany zapalne kości i szpiku kostnego, oceniać ukrwienie kończyn, diagnozować kręgosłup, stawy, oceniać czy przeszczepy skóry się przyjęły. Były i też **niepowodzenia kliniczne**, wynikające bądź z niewłaściwej interpretacji obrazów termowizyjnych, bądź nieprzestrzegania ostrych warunków technicznych, w których pracują urządzenia, albo niewłaściwego przygotowania pacjenta do badania. Jest więc i druga strona medalu. To przede wszystkim konieczność zapewnienia stałej temperatury pomieszczenia, wilgotności, przeprowadzania testów termicznych przed badaniem. Jak w każdym wypadku posługiwania się techniką o wysokim stopniu złożoności niezbędna okazuje się **odpowiednia kultura techniczna**. Na przykład detektory podczerwieni muszą pracować w bardzo niskich temperaturach. Uzyskuje się to chłodząc je ciekłym azotem. Aparatura jest więc bardzo skomplikowana, typu stacjonarnego, wrażliwa na wszelkie bodźce zewnętrzne.

Można by sobie zadać pytanie: **po co się bawić w skomplikowaną termografię, skoro badania wnętrza człowieka prowadzi się za pomocą promieni rentgenowskich, ultradźwiękowych, zjawisk magnetycznych?** Odpowiedź jest bardzo prosta. Otóż termografia **jest metodą bezinwazyjną**, w odróżnieniu od wymienionych w pytaniu. Owszem przy użyciu tomografu komputerowego można dokonywać dokładnych zdjęć wnętrza człowieka, ale z uwagi na promienie rentgenowskie nie można tego robić często. Nie wiadomo także, jaki mają wpływ ultradźwięki na nasz organizm. Natomiast termografia zbiera jedynie bodźce emitowane przez organizm.

Interesujące są badania prowadzone w **Wojskowym Instytucie Medycyny Lotniczej**, gdzie **ocenia się reakcje organizmu w warunkach niedotlenienia wysokościowego**. Analizuje się możliwość wykonywania pracy w warunkach ekstremalnych. Na jakiej podstawie? Za pomocą zdjęć termograficznych kończyn. Dr **Joanna Łaszczyńska** z WIML pokazuje zdjęcia wykonywane przy różnych obciążeniach pracą i w różnych warunkach ciśnienia atmosferycznego. Barwy ukazują, że następuje duże schłodzenie kończyn podczas pracy na wysokości. W wypadku dbałości o zdrowie i bezpieczeństwo pilotów takie badania mają znaczenie trudne do przecenienia.

Ostatnio pojawiła się nowa technika pomiaru temperatury ciała tzw. **ciekłokrystaliczna termografia kontaktowa**. Okazało się, że ciekłe kryształy, których zastosowanie znamy najczęściej we wskaźnikach różnego typu, jak zegarki, monitory itd., mają i tę właściwość, że pod wpływem zmian temperatury zmieniają zabarwienie. Chodzi w tym wypadku o ciekłe kryształy **na podłożu cholesterowym**.

Ostatnio zaprezentowano w Polsce szerszej opinii publicznej takie urządzenie produkcji firmy brytyjskiej **NovaTherm**. Jedno takie urządzenie pracuje już **w Klinice Ortopedii w Krakowie**.

Cienka warstwa substancji ciekłokrystalicznej jest umieszczona w folii. Przykłada się ją do ciała i po chwili widać zabarwienie w zależności od rozkładu temperatur. Taki obraz można sfotografować i dokumentować. Rozdzielczość temperatury wynosi 0,8 stopnia Celsjusza. Rejestrowany, jako barwna mapa, **obraz termiczny różnych obszarów ciała**, niesie ważną informację na temat ich stanu zdrowotnego. Powstaje obraz naturalnej wielkości. Ostro zarysowane kolory tworzą izotermy ograniczające pola jednakowych temperatur. Barwna podziałka pozwala na natychmiastową ocenę obrazu. Może być on w każdej chwili sfotografowany. Ale wydaje się, że przyszłość należy zapewne do zautomatyzowanej komputerowej analizy obrazu, co przy wyborze właściwego wzorca umożliwi szerokie badania masowe.

Jest to bardzo dobra metoda badania zwężeń i zatorów żylnych kończyn i płuc. Umożliwia tworzenie tzw. map bólów. Jest to metoda fizjologicznego testowania pacjenta i dająca obiektywną informację. **Sześć barw w zakresie 20—38°C**, jest wystarczających do uzyskania potrzebnych informacji.

Na całym świecie tysiące ludzi umiera **wskutek tzw. zakrzepicy tętnic płucnych**. Zatory biorą się głównie z kończyn. Zakrzepy przemieszczają się i wtedy jest już za późno na ratunek. Także u 30 proc. operowanych występują zakrzepy. Toteż, jak podkreśla dr **Jacek Kluz**, z Kliniki Ortopedii w Krakowie, wykrycie u pacjenta metodą termografii powstawania zakrzepów, bądź ich istnienia w kończynach, decyduje o terminie wykonania zabiegu, a nawet jego odłożeniu.

Na barwnych obrazach wyraźnie można ocenić, np. która noga jest objęta zakrzepami. Ta, która jest chłodniejsza, bo przez żyły przepływa mniej krwi.

Obecnie w **Wojskowej Akademii Technicznej** już udało się wykonać folię ciekłokrystaliczną, do zastosowań medycznych. Prezentował ją dr **Stanisław Kłosowicz** podczas konferencji pod nazwą **„Elektronika w medycynie"**. Otwierają się więc możliwości tworzenia polskich urządzeń diagnostycznych, niezwykle cennych dla medycyny.

Firma **„Sirpol-Ruch"** z Łodzi, która zorganizowała tę konferencję z udziałem wystawców zagranicznych, zaprezentowała np. **możliwości komputerowego przetwarzania obrazów**. Zamiast fotografować można by na przykład rejestrować obrazy w pamięci dyskowej, przechowywać je, archiwizować i porównywać zmiany w czasie. Przy czym w porównaniu do termografii bezdotykowej, tej przy użyciu kamery, ciekłokrystaliczna jest bez porównania tańsza.

Tak więc został zrobiony kolejny krok ku wykorzystaniu nowoczesnej techniki na potrzeby człowieka, zwłaszcza w sferze profilaktyki masowej. **Marek SIENIAWSKI**

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 3427. Nr ind. 361682. A-63

PROGRAMY

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" — zeszyt programów komputerowych Cena 80 zł

ORTOGRAFIA I JĘZYK...

KRZYŻÓWKA

ATRAKCJE

RULETKA

OPTYKA

WIEŻA HANOI

GRAFIKA

MUZYKA

W numerze:

AMSTRAD
- Prosta — str. 14
- Parabola — str. 15
- Elipsa — str. 16
- Hiperbola — str. 17
- Krzyżówka — str. 19
- Wieże Hanoi — str. 25

ATARI
- Mini-asembler — str. 27
- Test ortografii — str. 8

COMMODORE
- Muzyka 64 — str. 5

MERITUM
- Test z optyki — str. 22
- Ruch jednostajnie przyspieszony — str. 23
- Rzut poziomy — str. 23
- Rzut ukośny — str. 23
- Figury Lissajous — str. 24
- Test egzaminujący — str. 24

SHARP
- Komputerowa... ruletka — str. 3

SPECTRUM
- Program „Artysta" — str. 11

Okładka i rysunki: Michał Przybyłowski

Oddajemy do Waszych rąk pierwszy, specjalny numer „IKS-a„ poświęcony w całości programom komputerowym. Jak je wykorzystać?

Po pierwsze są one przeznaczone na konkretne typy mikrokomputerów. Wierzymy jednak, że wielu z naszych Czytelników potrafi je przetłumaczyć na potrzebny sobie dialekt Basica — a jeśli nie — pomoc zawsze znajdziecie w naszej redakcji.

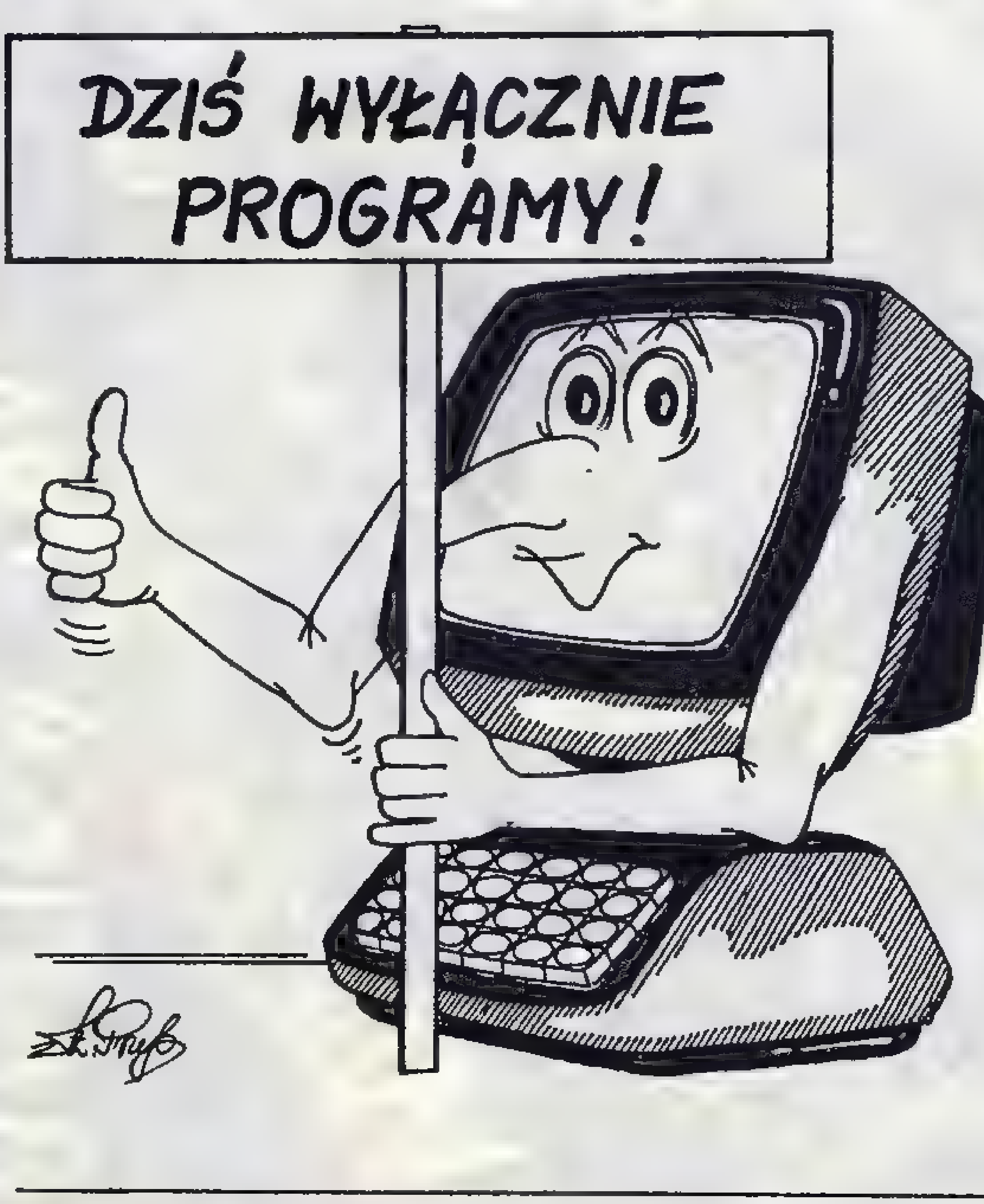
Po drugie nasze programy są jedynie propozycjami rozwiązań. To tylko od użytkownika zależy jaką ostateczną postać otrzymają ich eksploatowane wersje — radzimy eksperymentować przede wszystkim z grafiką. Dotyczy to szczególnie programów edukacyjnych, które można rozbudować zgodnie z wymogami metodycznymi. Poza tym będą one z pewnością częściej wykorzystywane przez uczniów na ich domowych komputerach niż w szkołach, w których jest zaledwie 3000 mikrokomputerów różnych typów.

Po trzecie wprowadzanie do pamięci komputera długich programów z ich wydruków jest zajęciem żmudnym, łatwo popełnić błędy. Wiemy o tym. Ale UWAGA:

NIEKTÓRE Z PROGRAMÓW BĘDĄ EMITOWANE PRZEZ „RADIOKOMPUTER" — PROGRAM ROZGŁOŚNI HARCERSKIEJ POŚWIĘCONY MIKROINFORMATYCE. Które i kiedy — dowiecie się w kolejnych audycjach: WTOREK, program IV, godz. 22.50.

A teraz życzymy już przyjemnej zabawy.

REDAKCJA



Prezentujemy 22 uznane za najlepsze komputery w 1986 roku.

1. Bondwell BW 2
Mikroprocesor: Z8OL, 4MHz
System operacyjny: CP/M2,2
Pamięć RAM: 64 KB
Pamięć masowa: RAM-Disc 256 lub 512 KB, Floppy-Disc 360 KB
Grafika: 640 × 200
Zasilanie: akumulator
Rozmiary: 28,4 × 7,8 × 31 cm
Waga: 3 kg
Cena: ok. 2400 marek



SHARP

PROGRAM 1

KOMPUTEROWA... RULETKA

Proponujemy Wam taką ruletkę, w której „bankierem" jest komputer, co nam daje *pewność*, że nie stracimy majątku.

W rzetelnej ruletce każdy numer jest jednakowo prawdopodobny. U nas zapewnia to *obiektywny* generator liczb losowych.

Program wywołuje etykieta „A" (RUN 40). Najpierw „kupujemy" talony (tzw. „sztony"), potem zaś określamy wysokość *stawki.*

Typujemy:
— *pojedyncze* numery od 1 do 36,
— *trójki* numerów od 1 do 12,
— *szóstki* od 1 do 6 oraz
— *tuziny* od 1 do 3.

Oprócz tego możemy typować „połówki" typu:
— *parzyste* lub *nieparzyste,*
— *małe* (od 1 do 18) lub *duże* (od 19 do 36).

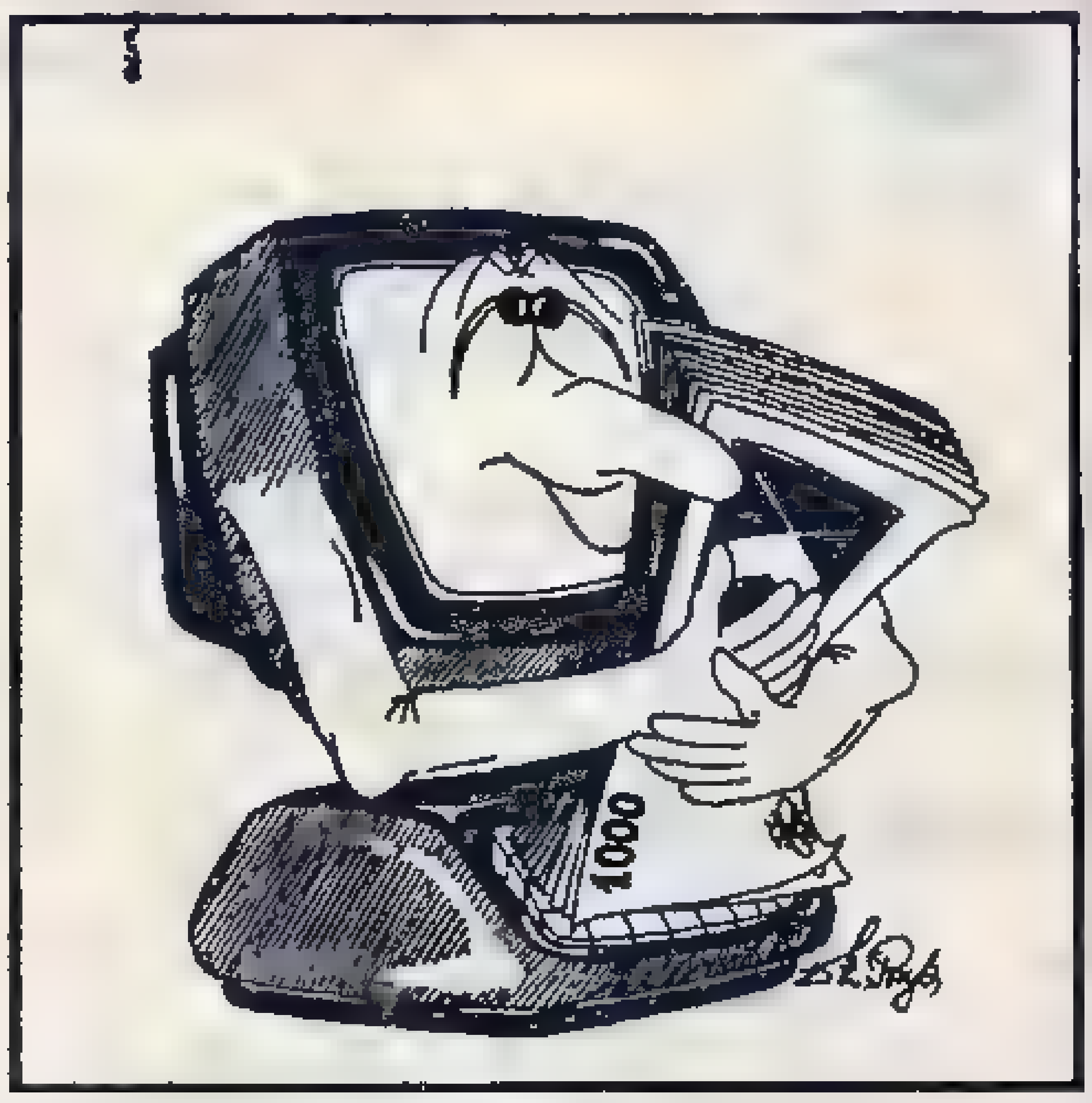
Ich rozmieszczenie podaje tablica:

Gdy „coś" trafimy, to *wygrana* jest *krotnością* stawki:
— dla numerów mnożnik 36,
— dla trójki x 12,
— dla szóstki x 6,
— dla tuzina x 3.

Trafny wybór połówki daje wygraną w postaci podwojonej stawki. Jeśli przegramy, tracimy (niestety) stawkę.

Po zakończeniu typowania „rusza kuleczka", co symbolizuje hasło RIEN NE VA PLUS (koniec typowania, zaś dosłownie *„nic więcej ne idzie").*

Komputer wyświetla wygrany



numer (przypadkowy), a potem... oblicza. Rezultat jest podany pod postacią komunikatu: wygrałeś... lub przegrałeś. Jeśli zostanie wylosowane *zero* , to *wszystkie stawki zagarnia bankier.* Jest to jedyny legalny zysk kasyna.

Typować można *jedną, dwie* lub wszystkie kombinacje jednocześnie.

Życzymy *rozbicia banku.*

WYGRANE (mnożniki stawek)		0 BANKIERSKIE 0							
Numer x 36	TUZIN 1	Trójka 1			Trójka 2			SZÓSTKA 1	1, 2.........18
Trójka x 12		1	2	3	4	5	6		
Szóstka x 6		Trójka 3			Trójka 4			SZÓSTKA 2	
Tuzin x 3		7	8	9	10	11	12		
x 2 dla:	TUZIN 2	Trójka 5			Trójka 6			SZÓSTKA 3	
Parzyste		13	14	15	16	17	18		
Nieparzyste		Trójka 7			Trójka 8			SZÓSTKA 4	19, 20.........36
Poniżej 19		19	20	21	22	23	24		
Powyżej 18	TUZIN 3	Trójka 9			Trójka 10			SZÓSTKA 5	
		25	26	27	28	29	30		
		Trójka 11			Trójka 12			SZÓSTKA 6	
		31	32	33	34	35	36		

```
40:"A":CLEAR :
   RANDOM :USING
   :WAIT 0:B=2E4
50:INPUT "KUP TAL
   ONY 100<T<10 0
   00 ";Z
60:IF Z<100OR Z>1
   E4THEN BEEP 3,
   100,500:GOTO 5
   0
70:INPUT "STAWKA
   OD 1 DO 500 ";
   Y
80:IF Y<1OR Y>500
   GOTO "D"
90:INPUT "NUMER O
   D 1 DO 36 ";N:
   IF N>36GOTO "D
   "
100:INPUT "TROJKA
    OD 1 DO 12 ";C
    :IF C>12GOTO "
    D"
110:INPUT "SZOSTKA
     OD 1 DO 6 ";S
    :IF S>6GOTO "D
    "
120:INPUT "TUZIN O
    D 1 DO 3 ";D:
    IF D>3GOTO "D"
130:INPUT "PARZYST
    E ? T=1 N=0 ";
    P:IF P=1GOTO 1
    50
140:INPUT "NIEPARZ
    YSTE ? T=1 N=0
     ";R:IF (P+R)>
    1GOTO "D"
150:INPUT "1,2...1
    8 ? T=1 N=0 ";
    M:IF M=1GOTO 1
    70
160:INPUT "19,20..
    .36 ? T=1 N=0
    ";O:IF (M+O)>1
    GOTO "D"
170:WAIT 0:PRINT "
    RIEN NE VA PLU
    S":BEEP 36,60,
    10
180:X=(RND 37)-1:
    PRINT "N U M E
     R *";X;" *"
190:BEEP 3,200,100
    0:IF X>0GOTO 2
    30
200:IF X=0THEN
    PRINT "BANKIER
    SKIE ZERO":
    BEEP 3,200,100
    0
210:Q=SGN N+SGN C+
    SGN S+SGN D+
    SGN P
```

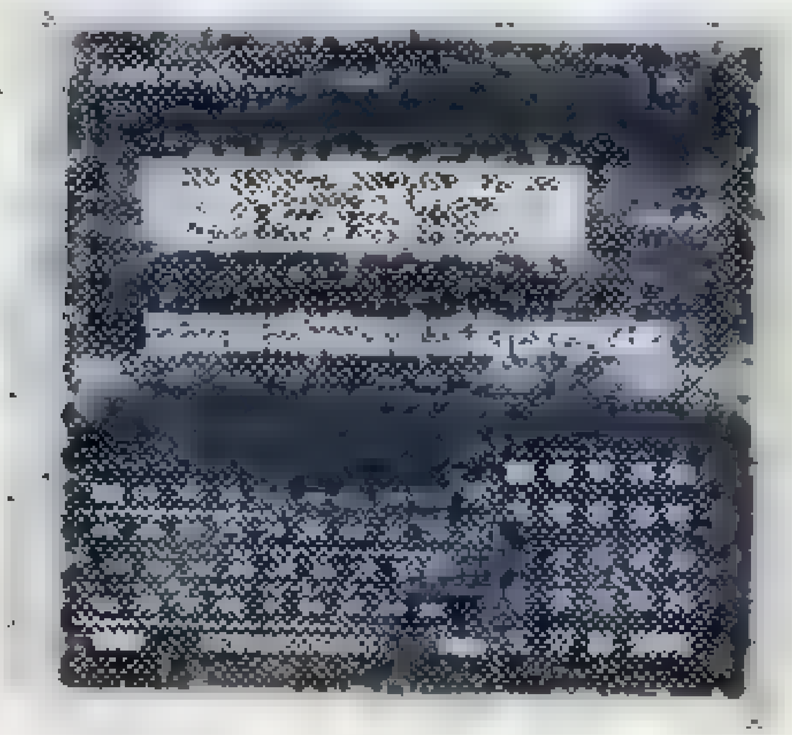
```
220:Q=Q+SGN R+SGN
    M+SGN O:B=B+Q*
    Y:W=-Q*Y:GOTO
    390
230:IF X<>NAND N>0
    GOSUB "C"
240:IF X=NLET F=36
    :GOSUB "B"
250:Q=INT ((X-1)/3
    )+1:IF Q<>CAND
    C>0GOSUB "C"
260:IF Q=CLET F=12
    :GOSUB "B"
270:Q=INT ((X-1)/6
    )+1:IF Q<>SAND
    S>0GOSUB "C"
280:IF Q=SLET F=6:
    GOSUB "B"
290:Q=INT ((X-1)/1
    2)+1:IF Q<>D
    AND D>0GOSUB "
    C"
300:IF Q=DLET F=3:
    GOSUB "B"
310:Q=X-INT (X/2)*
    2:IF (1+Q)*R=1
    GOSUB "C"
320:F=2:IF (1+Q)*P
    =1GOSUB "B"
330:IF Q*P=1GOSUB
    "C"
340:IF Q*R=1GOSUB
    "B"
350:IF M=1AND X<19
    GOSUB "B"
360:IF O=1AND X<19
    GOSUB "C"
370:IF M*X>18GOSUB
    "C"
380:IF O*X>18GOSUB
    "B"
390:W$="WYGRALES":
    IF W<0LET W$="
    PRZEGRALES"
400:WAIT :PRINT W$
    ;ABS W:Z=Z+W:
    IF Z<0GOTO 460
410:IF B<100THEN
    PRINT "BANK RO
    ZBITY":END
420:PRINT "MASZ";Z
    ;" TALONOW"
430:INPUT "GRASZ ?
     T=1 N=0 ";U:
    IF U=0THEN END
440:N=0:C=0:D=0:P=
    0:R=0:M=0:F=0
450:O=0:Y=0:W=0:Q=
    0:WAIT 0:GOTO
    70
460:PRINT "MASZ DL
    UG";ABS Z;" T
    ALONOW":END
470:"B":W=W+F*Y:B=
    B-F*Y:RETURN
480:"C":W=W-Y:B=B+
    Y:RETURN
490:"D":PRINT "U W
     A Z A J  !":
    BEEP 3,100,500
    :GOTO 70
```

JANUSZ MILLER



2. **Bondwell BW 8**
Mikroprocesor: 80C88, 4,77 MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 512 KB
Pamięć masowa: Floppy-Disc 720 lub 360 KB
Grafika: 640 × 200
Zasilanie: Sieć lub akumulator
Rozmiary: 28 × 7 × 30 cm
Waga: 4,9 kg
Cena: 3990 marek

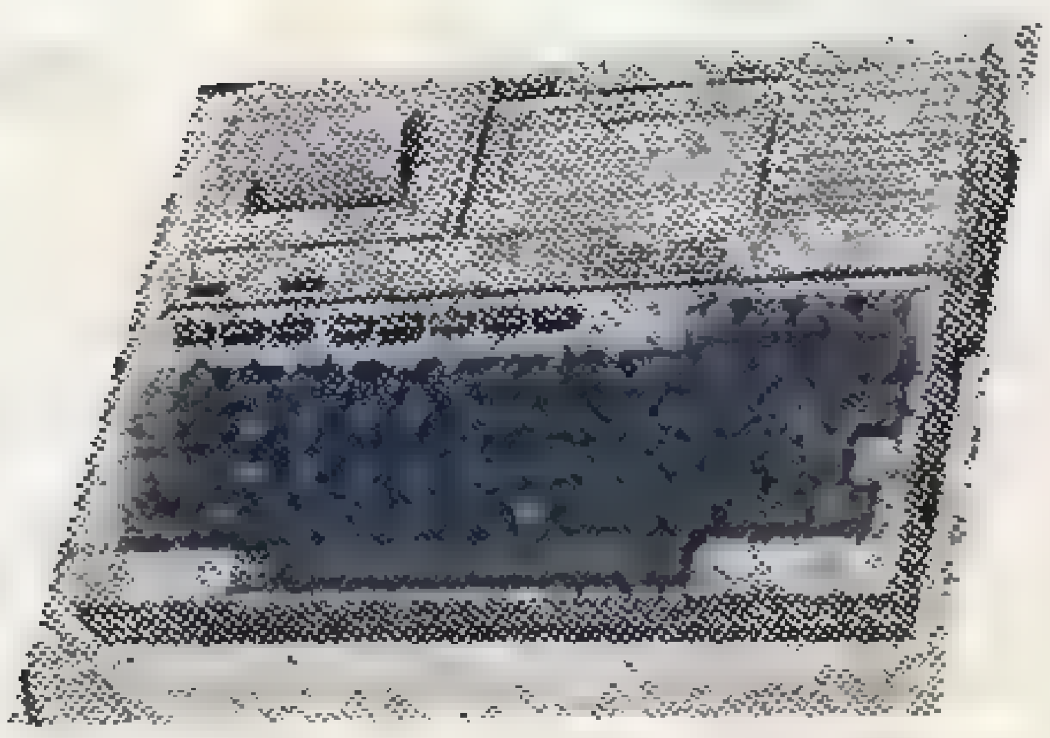
3. **Casio BP-1000**
Mikroprocesor: Casio-eigen
System operacyjny: C61 BASIC
Pamięć RAM: 8—40 KB
Pamięć masowa: Floppy-Disc (3,5) 320 KB
Grafika: 192 × 32
Zasilanie: bateria
Rozmiary: 18,4 × 2,4 × 9,7 cm
Waga: 437 g
Cena: 1400 marek (bez dyskietek)





4. **Datavue 25 II**
Mikroprocesor: 80C88, 4.77MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 768 KB lub 1.2 MB
Pamięć masowa: 1 lub 2 Floppy-Disc po 360 KB
Grafika: 640x200, 320x200
Zasilanie: sieć lub akumulator
Waga: 5.5 kg
Cena: 8660 M

5. **Epson HX-20**
Mikroprocesor: 6301, 2.45MHz
System operacyjny: MBASIC
Pamięć RAM: 16 lub 32 KB
Pamięć masowa: mikrokaseta — 30 KB
Grafika: 120x32
Zasilanie: akumulator
Rozmiary: 29x4.4x21.5 cm
Waga: 1.6 kg
Cena: 1800 M



COMMODORE

PROGRAM 2

MUZYKA 64

Commodore-64 posiada duże możliwości muzyczne. Niestety oferowany Basic nie zawiera żadnych instrukcji (nie licząc POKE), które ułatwiałyby programowanie muzyki. Istniejące na rynku oprogramowanie muzyczne nie zmienia tej sytuacji, gdyż realizowane za ich pomocą kompozycje nie dają się wykorzystać w naszych własnych programach.

MUZYKA 64 może być traktowana jako autonomiczny program do tworzenia różnorodnych kompozycji lub też jako narzędzie programistyczne wspomagające realizację efektów dźwiękowych w naszych własnych programach. Osoby uczące się muzyki będą mogły wykorzystać go jako program edukacyjny.

Zasady posługiwania się programem MUZYKA 64 zilustrowane są na przykładzie fragmentu utworu „NOBODY I KNOW" zespołu THE BEATLES.

MUZYKA 64 pozwala budować kompozycje na trzy głosy. W utworze muzycznym, który chcemy zaprogramować odszukujemy najkrócej trwający dźwięk (pauzę) — posłuży on jako jednostka czasu. Każdy dłużej trwający dźwięk wyrażany jest wielokrotnością podstawowej jednostki czasu. W przedstawionym fragmencie jest to ósemka. Programując dźwięki posługujemy się zamieszczoną tabelą kodową przyporządkowującą każdemu dźwiękowi dwie liczby (mniejszą i większą). Wybrane kody dźwięków wprowadzamy do programu w postaci danych typu DATE. W przedstawionym przykładzie zaprogramowanej muzyki (wiersze 349—670) każdy wiersz odpowiada jednej jednostce czasu i zawiera trzy dźwięki. Kompozycje kończymy wprowadzając wiersz zawierający trzy zera (wiersz 670).

W celu uzyskania zamierzonych efektów muzycznych należy korzystać z następujących reguł:

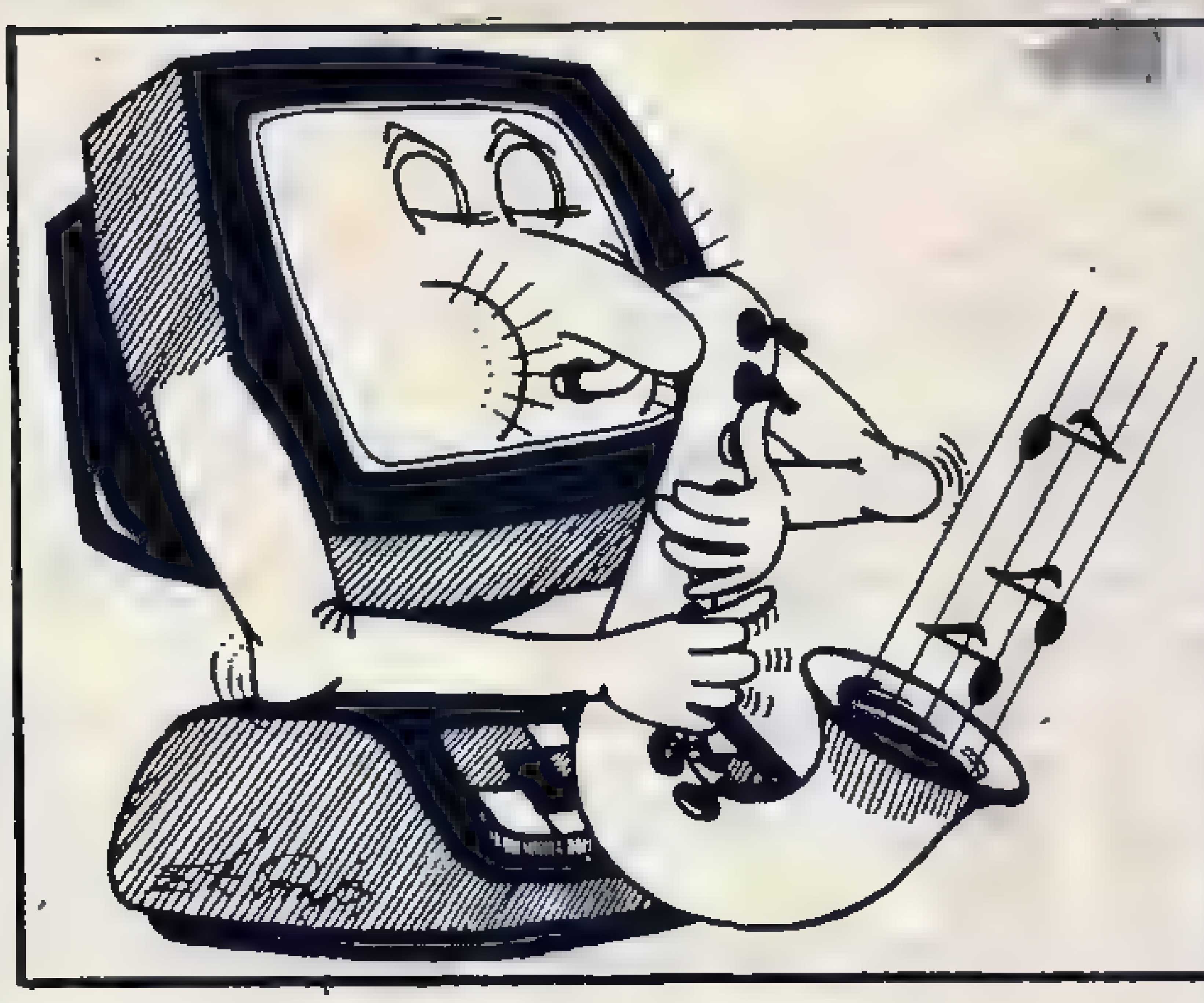
— kod o większej wartości używamy, gdy aktualnie programowany dźwięk ma być kontynuowany w następnej jednostce czasowej w tym samym głosie;

— kod o mniejszej wartości używamy, gdy aktualnie programowany dźwięk nie będzie kontynuowany w następnej jednostce czasowej w tym samym głosie;

— pauzę lub wyłączenie jakiegoś głosu w bieżącej jednostce czasowej otrzymujemy za pomocą kodu 192.

Program składa się z dwóch podprogramów oddzielnie wprowadzanych do pamięci mikrokomputera. Pierwszy z programów (program pomocniczy) inicjuje procedury zapisane w kodzie maszynowym, automatycznie ładuje z taśmy następny program i uruchamia go. Drugi program (program grający) zawiera sekwencje instrukcji realizujących odtworzenie dołączonej do niego kompozycji zapisanej w wierszach DATE. Jeśli chcemy wykorzystać właściwości programu MUZYKA 64 w naszym własnym programie musimy:

— zapisać na taśmie jako pierwszy program pomocniczy,



— wprowadzić do pamięci mikrokomputera program grający i stworzyć z niego procedurę wywoływaną w naszym własnym programie,

— dopisać własny program i zakodowaną kompozycję.

Przed uruchomieniem obydwu programów, składających się na program MUZYKA 64 należy stworzyć ich kopie.

```
5 REM ****** MUZYKA 64 ******
PROGRAM POMOCNICZY ******
10 PRINT"........PROSZE CZEKAC":
SK=0
20 FORI=1TO61:READW,N:POKE49664+
I,W:POKE49920+I,N:SK=SK+W+N:NEXT
30 DATA4,48,4,113,4,180,4,252,5,
72,5,152,5,237,6,71,6,167,7,12
40 DATA7,119,7,233,8,97,8,225,9,
104,9,247,10,143,11,48,11,218,12
,143
50 DATA13,78,14,24,14,239,15,210
,16,195,17,194,18,208,19,239,21,
30
60 DATA22,96,23,180,25,29,26,156
,28,49,29,222,31,164,33,134,35,1
33
70 DATA37,161,39,222,42,61,44,19
2,47,105,50,59,53,55,56,97,59,18
8
80 DATA63,73,67,13,71,9,75,66,79
,188,84,122,89,127,94,21,100,117
90 DATA106,111,112,195,119,119,1
26,146,134,24
100 IFSK<>102191THENPRINT"BLAD W
DANYCH, LINIE: 30 - 90":STOP
110 SK=0:FORI=49152TO49252:READW
:POKEI,W:SK=SK+W:NEXT
120 DATA162,0,189,0,193,32,25,19
2,162
```

```
130 DATA7,189,0,193,32,25,192,16
2,14,189
140 DATA0,193,32,25,192,96,41,12
8,201
150 DATA128,240,25,189,0,193,41,
63,168
160 DATA185,0,195,157,0,212,185,
0,194
170 DATA157,1,212,189,113,192,15
7,4,212
180 DATA96,189,112,192,157,4,212
,96,162
190 DATA0,189,0,193,32,88,192,16
2,7
200 DATA189,0,193,32,88,192,162,
14,189
210 DATA0,193,32,88,192,96,41,64
,201,64
220 DATA240,6,189,114,192,157,4,
212,96
230 IFSK<>11601THENPRINT"BLAD W
DANYCH, LINIE: 120 - 220":STOP
240 POKE631,76:POKE632,111:POKE6
33,13:POKE634,82:POKE635,117:POK
E636,13:POKE198,6:END
```
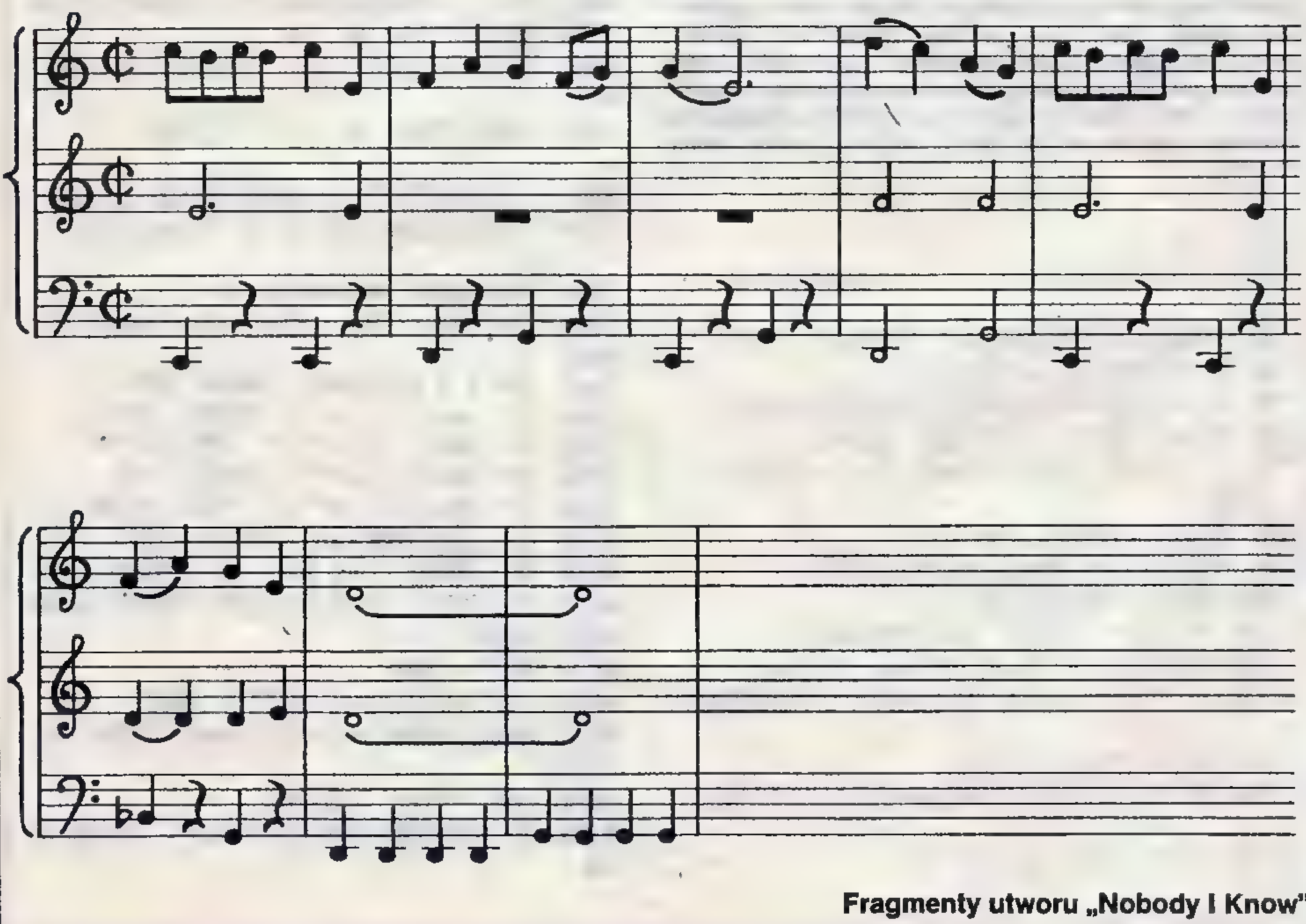
```
5 REM ****** MUZYKA 64 ******
PROGRAM GRAJACY ******
10 INPUT"TEMPO";T:S=54272
20 FORI=0TO24:POKES+I,0:NEXT:POK
ES+5,35:POKES+6,196:POKES+12,35
30 POKES+13,196:POKES+19,35:POKE
S+20,196,POKES+3,8:POKES+2,0:POK
ES+10,8
40 POKES+9,0:POKES+17,8:POKES+16
,0:POKES+24,15
50 FORI=0TO14STEP7:POKE49264+I,7
2:POKE49265+I,65:POKE49266+I,64:
NEXT
60 READG1,G2,G3:POKE49408,G1:POK
49415,G2:POKE49422,G3:IFG1=0THEN
80
70 SYS49152:FORI=0TOT:NEXT:SYS49
215:FORI=1TO50:NEXT:GOTO60
80 POKES+4,64:POKES+11,64:POKES+
18,64:FORI=0TO50:NEXT:POKES+24,0
:END
```

DZWIEK	CZESTOTLIWOSC	KOD	
C	65	1	65
C#	69	2	66
D	73	3	67
D#	78	4	68
E	82	5	69
F	87	6	70
F#	93	7	71
G	98	8	72
G#	104	9	73
A	110	10	74
A#	117	11	75
H	123	12	76
C	131	13	77
C#	139	14	78
D	147	15	79
D#	157	16	80
E	165	17	81
F	175	18	82
F#	185	19	83
G	196	20	84
G#	208	21	85
A	220	22	86
A#	233	23	87
H	246	24	88
* C	262	25	89
C#	277	26	90
D	294	27	91
D#	311	28	92
E	330	29	93
F	349	30	94
F#	370	31	95
G	392	32	96
G#	415	33	97
A	440	34	98
A#	466	35	99
H	494	36	100
C	523	37	101
C#	554	38	102
D	587	39	103
D#	622	40	104
E	659	41	105
F	698	42	106
F#	740	43	107
G	784	44	108
G#	831	45	109
A	880	46	110
A#	932	47	111
H	988	48	112
C	1047	49	113
C#	1109	50	114
D	1175	51	115
D#	1246	52	116
E	1319	53	117
F	1397	54	118
F#	1480	55	119
G	1568	56	120
G#	1661	57	121
A	1760	58	122
A#	1865	59	123
H	1976	60	124
C	2093	61	125

* SRODKOWE C

```
349 REM **NOBODY I KNOW**
350 DATA  49, 77, 93
355 DATA  48, 77, 93
360 DATA  49,192, 93
365 DATA  48,192, 93
370 DATA 113, 77, 93
375 DATA 113, 77, 29
380 DATA 105,192, 93
385 DATA 105,192, 93
390 DATA 106, 79,192
395 DATA 106, 79,192
400 DATA 110,192,192
405 DATA 110,192,192
410 DATA 108, 84,192
415 DATA 108, 84,192
420 DATA  42,192,192
425 DATA  44,192,192
430 DATA 108, 77,192
435 DATA 108, 77,192
440 DATA 105,192,192
445 DATA 105,192,192
450 DATA 105, 84,192
455 DATA 105, 84,192
460 DATA 105,192,192
465 DATA 105,192,192
470 DATA 115, 79,106
475 DATA 115, 79,106
480 DATA 113, 79,106
485 DATA 113, 79, 42
490 DATA 110, 84,106
495 DATA 110, 84,106
500 DATA 108, 84,106
505 DATA 108, 84,106
510 DATA  49, 77, 93
515 DATA  48, 77, 93
520 DATA  49,192, 93
525 DATA  48,192, 93
530 DATA 113, 77, 93
535 DATA 113, 77, 29
540 DATA 105,192, 93
545 DATA 105,192, 93
550 DATA 106, 87,103
555 DATA 106, 87,103
560 DATA 110,192,103
565 DATA 110,192, 39
570 DATA 108, 84,103
575 DATA 108, 84,103
580 DATA 105,192,105
585 DATA 105,192,105
590 DATA 103, 79,103
595 DATA 103, 15,103
600 DATA 103, 79,103
605 DATA 103, 15,103
610 DATA 103, 79,103
615 DATA 103, 15,103
620 DATA 103, 79,103
625 DATA 103, 15,103
630 DATA 103, 84,103
635 DATA 103, 20,103
640 DATA 103, 84,103
645 DATA 103, 20,103
650 DATA 103, 84,103
655 DATA 103, 20,103
660 DATA 103, 84,103
665 DATA 103, 20,103
670 DATA   0,  0,  0
```

S. WASILEWSKI



Fragmenty utworu „Nobody I Know"

ATARI

PROGRAM 3

TEST ORTOGRAFII

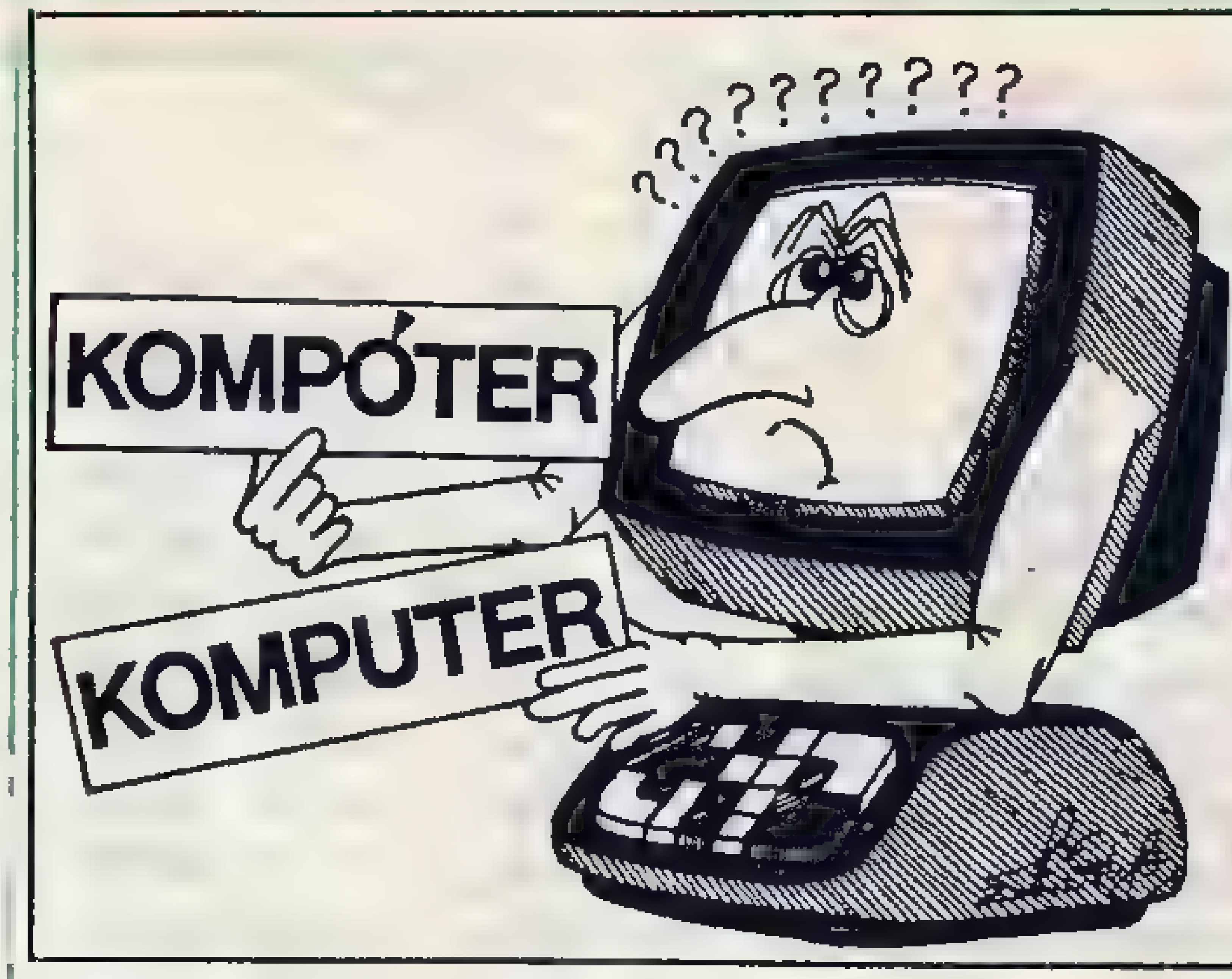
Program „TEST ORTOGRAFII" jest kolejnym programem edukacyjnym publikowanym w „IKS-ie" na komputery ATARI.

Po uruchomieniu programu, na ekranie pojawi się MENU. Na początku użytkowania tego programu, należy najpierw wpisać do komputera wyrazy, które sprawiają nam trudności w pisaniu. (UWAGA — wyrazy te powinny być wpisane oczywiście poprawnie). Maksymalnie można ich wpisać 30 000. Program ten umożliwia używanie polskich liter: ą,ę,ć,ł,ń,ó,ś,ż,ź. Uzyskuje się je przez naciśnięcie odpowiednio: CONTROL-A, CONTROL-E itd. Literę ż uzyskamy przez naciśnięcie CONTROL-",".

Po wpisaniu do komputera wyrazów, wybierając odpowiednie opcje o MENU, można przystąpić do testu lub nagrać wpisane do komputera wyrazy na taśmę magnetofonową. Zarejestrowane na taśmie wyrazy, przy późniejszym korzystaniu z programu można bezpośrednio wpisać do komputera.

Gdy przystąpimy do testu na ekranie będzie pokazywał się wyraz. Jeżeli unamy, że jest on napisany dobrze, to wciśniemy klawisz SPACE, jeżeli źle — dowolny klawisz (oprócz BREAK). Gdy nasza decyzja będzie poprawna, z głośnika wydobędzie się pojedyńczy dźwięk, jeżeli zła — potrójny. Oprócz tego, na ekranie znajduje się licznik dobrych i złych odpowiedzi oraz zegar. Czas podawania odpowiedzi jest ograniczony. Jeżeli zegar osiągnie wartość 500, odpowiedź zostanie uznana za złą. Życzę owocnej nauki i miłej zabawy.

```
10 CLR
20 ? CHR$(125):? "Prosze czeka
c..."
30 OPEN #1,4,0,"K":GOSUB 1610
40 DIM A$(15),B$(18)
50 DL=PEEK(560)+PEEK(561)*256
60 POKE DL+6,7:POKE DL+7,6
70 ? CHR$(125):POKE 710,0:TRAP
 100
80 SETCOLOR 1,10,10
90 RESTORE 1999:READ DAT
100 POSITION 8,3
110 ? "1 - TEST ORTOGRAFII"
120 POSITION 8,6:POKE 752,6
130 ? "2 - WPIS NOWYCH WYRAZOW
"
140 POSITION 8,9
150 ? "3 - WPIS WYRAZOW Z MAGN
."
160 POSITION 8,12
```

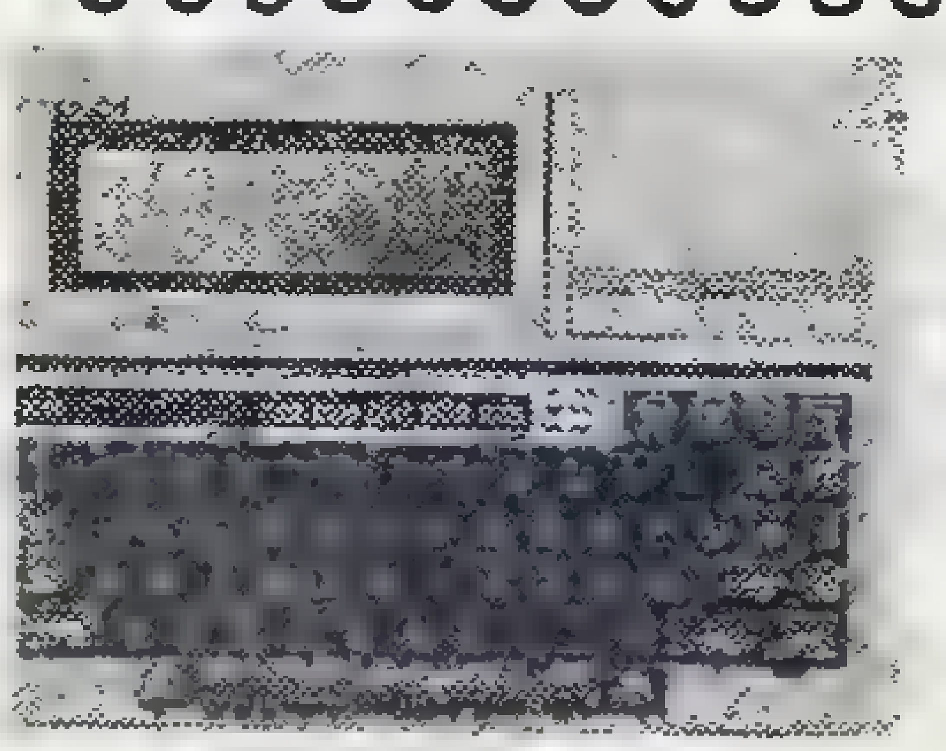



```
170 ? "4 - ZAPIS WYRAZOW NA MA
GN."
180 TRAP 180:GET #1,EE:E=EE-48
190 ON E GOTO 210,1040,1420,12
60
200 GOTO -9
210 LICZ=-1:MAX=500:DOB=0:ZL=0
220 ? CHR$(125):POKE 710,0:TRA
P 70
230 GOSUB 920
240 POSITION 1,5:? "  ODPOWIED
ZI:  "
250 POSITION 1,7:? " Dobre:
Ze:"
260 A=INT(RND(0)*DAT)+2000:M=0
270 POSITION 3,22
280 ? " SELECT  - Powrt do ME
NU"
290 POSITION 4,3:? "CZAS :
"
300 LICZ=LICZ+1:POSITION 3,20
310 POSITION 23,3:? "Licznik :
";LICZ
320 RESTORE A:READ A$
330 FOR I=1 TO LEN(A$):TRAP 40
000
340 IF I>LEN(A$) THEN 500
350 IF A$(I,I)="" THEN 740
360 IF A$(I,I)="u" THEN 770
370 IF A$(I,I)="" THEN 800
380 IF A$(I,I)="h" THEN 860
390 IF LEN(A$)=I THEN 420
400 IF A$(I,I+1)="rz" THEN 830
410 IF A$(I,I+1)="ch" THEN 890
420 NEXT I
430 GOTO 500
440 LICZ=LICZ-1:GOTO 260
450 H=LEN(A$)-1:IF H=0 THEN RE
TURN
460 FOR HH=1 TO H
470 IF A$(HH,HH+1)="cc" THEN 2
60
```

```
480 NEXT HH
490 RETURN
500 GOSUB 450
510 POSITION 20,15:? A$;"
     "
520 POKE 764,255:LL=0
530 P=PEEK(764):LL=LL+1:V=V+1
540 SETCOLOR 0,5,V:IF V>100 TH
EN V=1
550 IF LL>MAX THEN 680
560 POSITION 10,3:? LL
570 IF PEEK(53279)=5 THEN 70
580 IF P=255 THEN 530
590 IF P=33 AND M=0 THEN 640
600 IF P<>33 AND M=3 THEN 640
610 IF P=33 AND M=3 THEN 680
620 IF P<>33 AND M=0 THEN 680
630 STOP
640 FOR F=1 TO 50:SOUND 1,30,1
0,15
650 NEXT F:SOUND 1,0,0,0
660 DOB=DOB+1:POSITION 4,9:? D
OB
670 GOTO 260
680 FOR FF=1 TO 3:FOR F=50 TO
100
690 SOUND 1,F,12,15:NEXT F
700 SOUND 1,0,0,0:FOR F=1 TO 3
0
710 NEXT F:NEXT FF
720 ZL=ZL+1:POSITION 12,9:? ZL
730 GOTO 260
740 K=INT(RND(0)*5)
750 IF K=1 OR K=4 THEN A$(I,I)
="u":M=3
760 GOTO 420
770 K=INT(RND(0)*7)
780 IF K=3 OR K=6 THEN A$(I,I)
="":M=3
790 GOTO 420
800 K=INT(RND(0)*5)
810 IF K=1 OR K=3 THEN GOSUB 9
40
820 GOTO 420
830 K=INT(RND(0)*7)
840 IF K=2 OR K=5 THEN GOSUB 9
70
850 GOTO 420
860 K=INT(RND(0)*9)
870 IF K=5 OR K=8 THEN GOSUB 9
90
880 GOTO 420
890 K=INT(RND(0)*8)
900 IF K=2 OR K=5 THEN GOSUB 1
020
910 GOTO 420
920 POSITION 3,1:? "TEST ORTOG
RAFII"
930 SETCOLOR 0,5,15:RETURN
940 B$=A$(I,LEN(A$))
950 A$(I+1,LEN(A$)+1)=B$
960 A$(I,I+1)="rz":M=3:RETURN
970 A$(I)=A$(I+1):M=3
980 A$(I,I)="":RETURN
990 B$=A$(I,LEN(A$))
1000 A$(I+1,LEN(A$)+1)=B$
1010 A$(I,I+1)="ch":M=3:RETURN
1020 A$(I)=A$(I+1):M=3
1030 A$(I,I)="h":RETURN
1040 ? CHR$(125):POKE 710,0:PO
KE 702,0
1050 POKE 752,0:POSITION 10,3
1060 ? " WPIS WYRAZOW ":GOSUB
920
1070 POSITION 4,20
1080 ? " RETURN  --- MENU"
1090 POSITION 3,15:? " UWAGA "
:?
1100 ? "WYRAZ WPISUJ MALYMI LI
TERAMI"
1110 POSITION 5,8:? "PODAJ WYR
AZ",
1120 INPUT A$:IF A$="" THEN 70
1130 ? CHR$(125):POSITION 2,4
1140 SETCOLOR 1,0,0:? DAT+2000
;"D.";A$
1150 ? :? "G.1180"
1160 POSITION 0,2:POKE 842,1
1170 STOP
1180 POKE 842,12:DAT=DAT+1
1190 ? CHR$(125):POSITION 2,4
1200 ? 1999;"D.";DAT
1210 ? :? "G.1240"
1220 POSITION 0,2:POKE 842,1
1230 STOP
1240 POKE 842,12:SETCOLOR 1,10
,10
1250 GOTO 1040
1260 ? CHR$(125):POKE 710,0
1270 GOSUB 920:POSITION 7,5
1280 ? " ZAPIS WYRAZOW NA MAGN
ETOFON "
1290 POKE 702,64:POSITION 9,16
1300 ? "CZY JESTES PEWIEN ? [T
/N]"
1310 GET #1,EE
1320 IF EE=78 THEN 70
1330 IF EE=84 THEN 1350
1340 GOTO 1310
1350 RESTORE 1999:OPEN #2,9,0,
"C:"
1360 READ A$
```

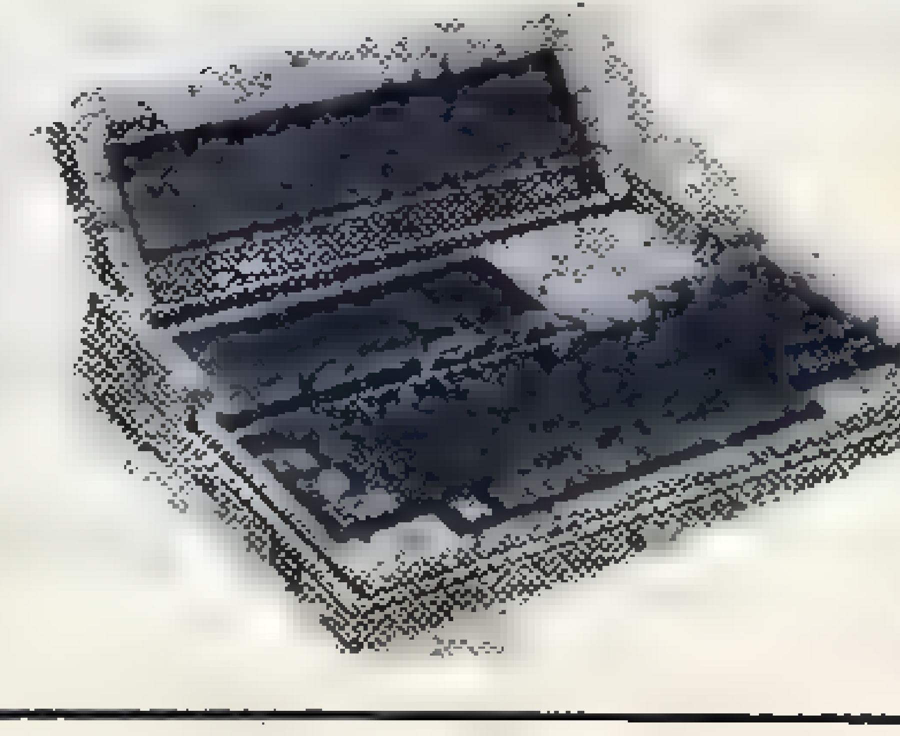
```
1370 ? #2;A$
1380 IF A$="*+*" THEN CLOSE #2
:GOTO 1400
1390 GOTO 1360
1400 POSITION 12,10:? "O.K"
1410 FOR F=1 TO 300:NEXT F:GOT
O 70
1420 ? CHR$(125):POKE 710,0
1430 GOSUB 920:POSITION 7,4
1440 ? " WPIS WYRAZOW Z MAGNET
OFONU "
1450 POKE 702,64:POSITION 9,16
1460 ? "CZY JESTES PEWIEN ? [T
/N]"
1470 GET #1,EE
1480 IF EE=78 THEN 70
1490 IF EE=84 THEN 1510
1500 GOTO 1470
1510 SETCOLOR 1,0,0
1520 ? "}":OPEN #2,5,0,"C:":P=
1998
1530 INPUT #2;A$
1540 P=P+1:POSITION 2,4:? P;"D
.";A$
1550 ? :? "G.1580":POSITION 0,
2
1560 POKE 842,1
1570 STOP
1580 POKE 842,12:? CHR$(125)
1590 IF A$="*+*" THEN CLOSE #2
:GOTO 70
1600 GOTO 1530
1610 REM GEN. POLSKICH LITER
1620 N=57344:M=32768
1630 FOR I=0 TO 1023:POKE M+I,
PEEK(N+I)
1640 NEXT I
1650 A=M+776:B=M+520
1660 FOR I=0 TO 207:POKE B+I,P
EEK(A+I)
1670 NEXT I
1680 A=M+976:B=M+512
1690 FOR I=0 TO 7:POKE B+I,PEE
K(A+I)
1700 NEXT I
1710 POKE M+513,12:POKE M+514,
24
1720 POKE M+515,124:POKE M+527
,3
1730 POKE M+537,12:POKE M+538,
24
1740 POKE M+539,60:POKE M+559,
6
1750 POKE M+611,28:POKE M+612,
56
1760 POKE M+625,12:POKE M+626,
24
1770 POKE M+627,124:POKE M+633
,12
1780 POKE M+634,24:POKE M+635,
60
1790 POKE M+664,6:POKE M+665,1
2
1800 POKE M+721,24:POKE M+722,
0
1810 POKE M+723,124:POKE 756,1
28
1820 RETURN
1999 DATA 0
31999 DATA *+*
```

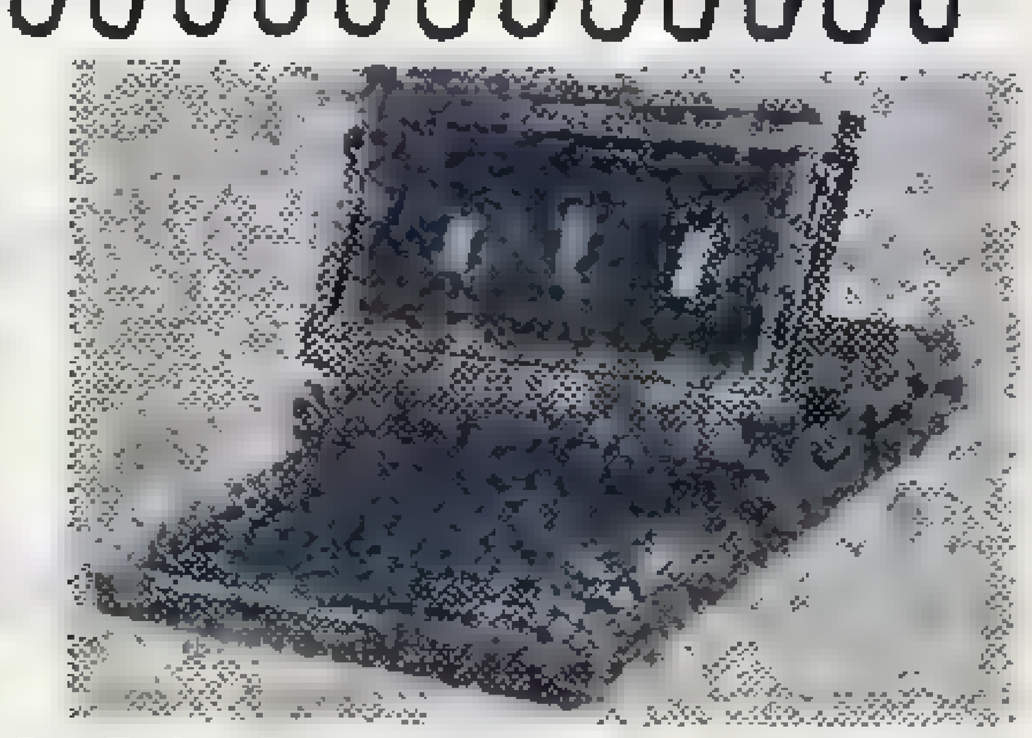
OLGIERD NIEMYJSKI
LVI LO—Warszawa



6. **Epson PX-4**
Mikroprocesor: Z80A, 3.68MHz
System operacyjny: CP/M 2.2
Pamięć RAM: 64 KB
Grafika: 240x64
Zasilanie: bateria lub akumulator
Rozmiary: 29.7x3.4x28.6 cm
Waga: 1.6 kg
Cena: 2500 M

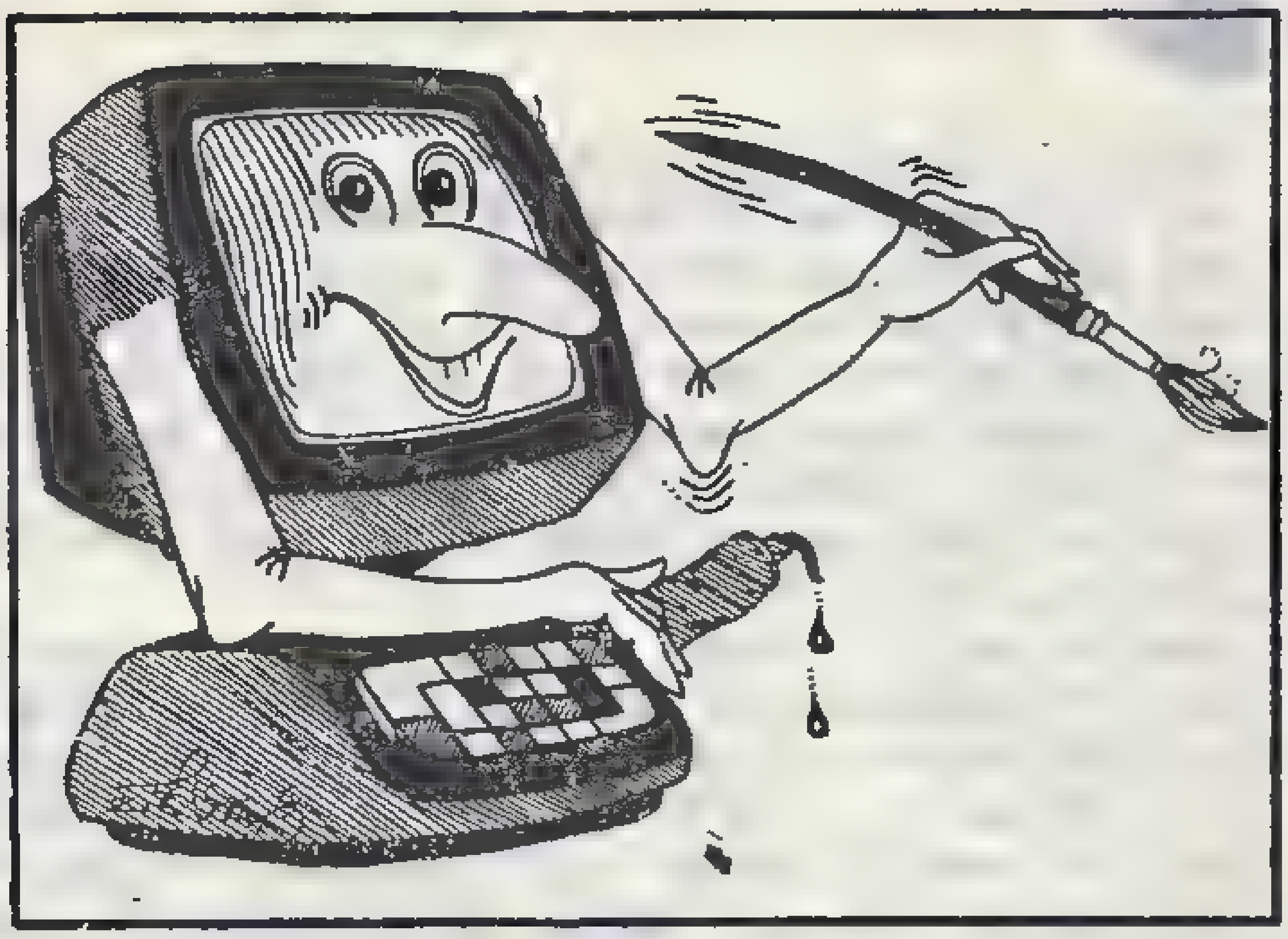
7. **Epson PX-8**
Mikroprocesor: Z80, 2.45MHz
System operacyjny: CP/M
Pamięć RAM: 64 KB lub 128 MB
Grafika: 480x64
Zasilanie: sieć lub akumulator
Rozmiary: 29.7x4.8x21.6 cm
Waga: 2.3 kg
Cena: 3000 M





8. **Grid Case**
Mikroprocesor: 80C86, 4.77MHz
System operacyjny: MS-DOS 2.11/3.2
Pamięć RAM: 128 KB lub 640 MB
Pamięć masowa: wewnętrzny Floppy-Disc 720 KB
Grafika: 640x200
Zasilanie: sieć lub akumulator
Rozmiary: 29x5.5x38 cm
Waga: 5.4 kg
Cena: 7000 M

SPECTRUM
PROGRAM 4



ARTYSTA

Program „Artysta" jest edytorem obrazu złożonego ze standardowych znaków graficznych. Program ten może być dobrym uzupełnieniem do programu „Kompens" publikowanego w numerze piątym „IKSa". Pracuje w trzech trybach: graficznym, tekstowym i edycyjnym.

Tryb graficzny pozwala na budowanie obrazu ze znaków graficznych (klawisze 1,2...,8). Aktualną pozycję dostawionego znaku wskazuje kursor — kwadrat o podwyższonej jaskrawości. W tym trybie dostępne są następujące funkcje:

CAPS SHIFT (CS) i ↑↓←→	— przesunięcie kursora we wskazanym kierunku (dodatkowo funkcję tę pełnią klawisze Q, W, E, R),
ENTER	— przejście na początek nowej linii,
SPACE	— usunięcie znaku na pozycji kursora,
DELETE	— przesunięcie kursora o jedno pole wstecz i usunięcie stojącego tam znaku,
CS i TRUE VIDEO	— kolor i tło znaku są zgodne z deklarowanymi,
CS i INVERS VIDEO	— kolor i tło znaku są dopełnieniem w stosunku do deklarowanych atrybutów,
GRAPHICS lub 9	— zmiana (negacja) zestawu znaków graficznych,
X, Y	— zmiana sposobu przesuwania się kursora po wpisaniu znaku, (x=1 w prawo, x=Ø bez przesunięcia, X=1 w lewo Y=1 w górę, Y=Ø bez przesunięcia, Y=1 w górę),
EDIT	— przejście do trybu edycyjnego.

O aktualnym rodzaju pracy informuje komunikat u dołu ekranu (napis: video, wartości X i Y oraz aktualny zestaw znaków).

W trybie edycyjnym dostępne są funkcje:
- J (LOAD) — wczytanie obrazu z taśmy,
- P (PRINT) — przejście do trybu tekstowego,
- S (SAVE) — zapisanie obrazu na taśmę,
- V (US) — skasowanie zawartości ekranu,
- Z (COPY) — wydruk zawartości ekranu,
- ENTER — powrót do trybu graficznego.

Tryb tekstowy pozwala na umieszczenie napisów (poziomych i pionowych) w tworzonym obrazie. Napisy mogą składać się ze znaków alfabetu ZX Spectrum i znaków UDG. Wykluczone są jedynie znaki dostępne w trybie „EXTEND MODE". Komunikat informuje o rodzaju pracy podając tryb L lub C i sposób przemieszczania się kursora (kursor Ø— przesunięcie w prawo, kursor 1 przesunięcie w dół).

CAPS LOCK (CS i 2)	— zmiana trybu L na C i odwrotnie,
EXTEND MODE (CAPS SHIFT : SYMBOL SHIFT	— zmiana sposobu przemieszczania się kursora (napis poziomy/napis pionowy),
↑↓←→	— przesunięcie kursora
DELETE	— skasowanie znaku na polu poprzedzającym kursor,
GRAPHICS	— przejście do wpisywania znaków UDG,
ENTER	— powrót do trybu edycji.

Struktura programu pozwala na łatwą rozbudowę (np. wersja kolorowa lub uzupełnienie instrukcjami DRAW, PLOT itd.). Dodatkowe bloki można zbudować oddzielnie i dołączyć w miarę potrzeby instrukcję merge „nazwa".

```
  5 REM PROGRAM "ARTYSTA"
 10 BORDER 7: PAPER 7: INK 0:        INVERSE 0: OVER 0: BRIGHT 0
CLS
 20 LET p=0: LET w=0: LET k=0:       LET g=0: LET x=1: LET y=0:
 30 POKE 23658,0: GO SUB 1000
 40 PRINT AT w;k; OVER 1; INVERSE 0; BRIGHT 1;" "
100 LET w$=INKEY$: IF w$="" THEN  GO TO 100
105 IF CODE w$=4 THEN  LET i=0: GO SUB 1000
110 IF CODE w$=5 THEN  LET i=1: GO SUB 1000
115 IF CODE w$=15 OR w$="9" THEN  LET g=1-g: GO SUB 1000
120 IF CODE w$=7 THEN  GO SUB 5000: GO TO 100
125 PRINT AT w,k; OVER 1; INVERSE 0; BRIGHT 1;" "
```

```
 130 LET ww=w: LET kk=k
 135 LET w=w+(w<21)*(CODE w$=10 OR w$="w")-(w>0)*(CODE w$=11 OR w$="q")
 140 LET k=k+(k<31)*(CODE w$=9 OR w$="r")-(k>0)*(CODE w$=8 OR w$="e")
 145 PRINT AT ww,kk; OVER 1; INVERSE 0;" "
 150 IF w$>="1" AND w$<="8" THEN  LET h=VAL w$-8*INT (VAL w$/8): LET w$=CHR$ ((g
=0)*(128+h)+(g=1)*(143-h)): PRINT AT w,k;w$: LET w=w-y*(((w-y)<=21) AND ((w-y)>=
0)): LET k=k+x*(((k+x)<=31) AND ((k+x)>=0))
 155 IF CODE w$=13 THEN  PRINT AT w,k; OVER 1; INVERSE 0;" ": LET k=0: LET w=(w<
21)*(w+1)
 160 IF w$=" " THEN  PRINT AT w,k; INVERSE 0; BRIGHT 1;" "
 165 IF CODE w$=12 OR w$="0" THEN  PRINT AT w,k; INVERSE 0; OVER 1;" ": LET w=w-
(w>0)*(k=0): LET k=(k>0)*(k-1)+(w>=0)*(k=0)*31: PRINT AT w,k; INVERSE 0;" "
 170 IF w$="x" THEN  LET x=(x=-1)-(x=0): GO SUB 1000
 175 IF w$="y" THEN  LET y=(y=-1)-(y=0): GO SUB 1000
 180 PRINT AT w,k; OVER 1; INVERSE 0; BRIGHT 1;" "
 185 GO TO 100
1000 REM KOMUNIKATY
1001 INVERSE i
1002 IF i=0 THEN  PRINT #0;AT 0,0;"VIDEO    "
1003 IF i=1 THEN  PRINT #0;AT 0,0;"VIDEO    "
1004 PRINT #0;AT 1,0;"X:";x;"  ";AT 1,5;"Y:";y;"    "
1005 IF g=0 THEN  PRINT #0;AT 0,8;"KLAWISZ: 1 2 3 4 5 6 7 8";AT 1,11;"ZNAK:
      A"
1006 IF g=1 THEN  PRINT #0;AT 0,8;"KLAWISZ: 1 2 3 4 5 6 7 8";AT 1,11;"ZNAK:
      "

1007 RETURN
1008
5000 REM TRYB EDYCJI
5002 POKE 23658,0
5004 PRINT #0;AT 0,0;"J - LOAD  S - SAVE  P - PRINT   Z - COPY  V - CLS   ENTER-
POWROT"
5006 LET w$=INKEY$: IF w$="" THEN  GO TO 5006
5008 IF w$="s" THEN  GO SUB 5200
5010 IF w$="j" THEN  GO SUB 5100
5012 IF w$="z" THEN  GO SUB 5300
5014 IF w$="v" THEN  GO SUB 5400
5016 IF w$="p" THEN  PAUSE 30: GO SUB 6000
5018 IF CODE w$=13 THEN  GO SUB 1000: RETURN
5020 GO TO 5000
5100 REM LOAD
5102 PRINT #0;AT 0,0;"CZY LADOWAC OBRAZ Z TASMY ?(T/N)
    "

5104 IF INKEY$="n" THEN  RETURN
5106 IF INKEY$<>"t" THEN  GO TO 5104
5108 INPUT "PODAJ NAZWE ZBIORU ";n$: IF n$="" OR LEN n$>10 THEN  GO TO 5108
5110 LOAD n$ SCREEN$
5112 POKE 23658,0: RETURN
5114
5200 REM SAVE

5202 PRINT #0;AT 0,0;"CZY NAGRYWASZ OBRAZ ? (T/N)
    "

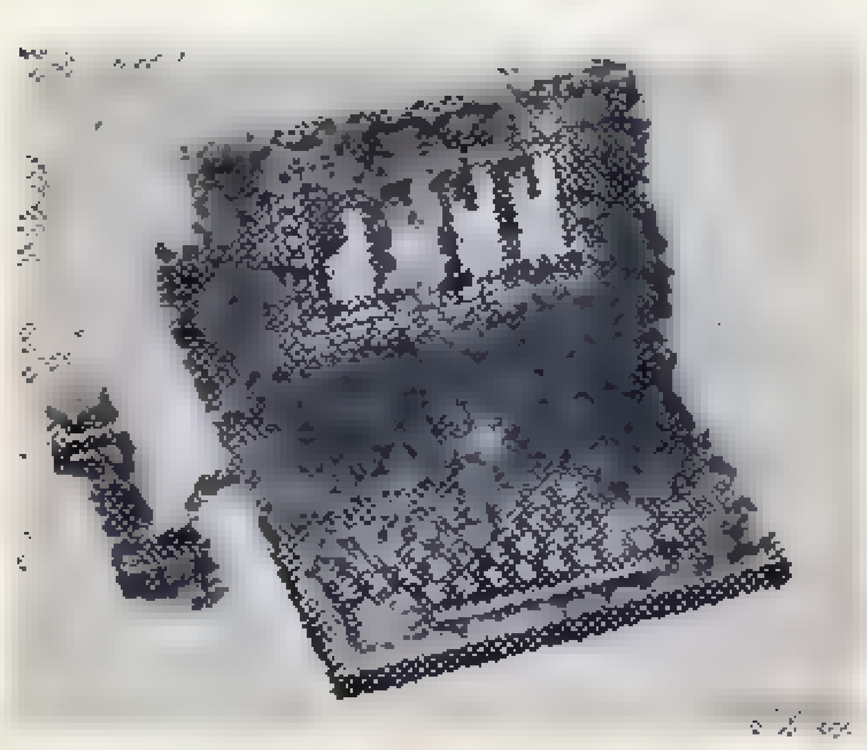
5204 IF INKEY$="n" THEN  RETURN
5206 IF INKEY$<>"t" THEN  GO TO 5204
5208 INPUT "PODAJ NAZWE ZBIORU ";n$: IF n$="" OR LEN n$>10 THEN  GO TO 5208
5210 SAVE n$ SCREEN$
5212 PRINT #0;AT 0,0;"GOTOWE - WCISNIJ  ENTER
    "

5214 IF CODE INKEY$<>13 THEN  GO TO 5214
5216 POKE 23658,0: RETURN
5218
5300 REM COPY
5302 PRINT #0;AT 0,0;"CZY DRUKOWAC OBRAZ ? (T/N)
    "

5304 IF INKEY$="t" THEN  COPY : RETURN
5306 IF INKEY$="n" THEN  RETURN
5308 GO TO 5304
```

```
5310
5400 REM CLS
5402 PRINT #0;AT 0,0;"CZY SKASOWAC EKTAN ? (T/N)
     "
5404 IF INKEY$="t" THEN  CLS : RETURN
5406 IF INKEY$="n" THEN  RETURN
5408 GO TO 5404
5410
6000 REM PRINT
6002 LET d=0
6004 IF p=0 THEN  POKE 23658,0: PRINT #0;AT 0,0;"TRYB TEKSTOWY L  kursor:
  ENTER  - POWROT DO GRAFIKI       ";AT 0,24;d
6006 IF p=1 THEN  POKE 23658,8: PRINT #0;AT 0,0;"TRYB TEKSTOWY C  kursor:
  ENTER  - POWROT DO GRAFIKI       ";AT 0,24;d
6008 LET w$=INKEY$: IF w$="" THEN  GO TO 6008
6010 IF CODE w$=6 THEN  LET p=1-p: PAUSE 50: GO TO 6004
6011 IF CODE w$=14 THEN  LET d=1-d: PAUSE 30: GO TO 6004
6012 IF CODE w$=15 THEN  GO SUB 6100: PRINT AT w,k; OVER 1; BRIGHT 1; INVERSE 0;
" ": GO TO 6004
6014 IF CODE w$=13 THEN  POKE 23658,0: RETURN
6016 IF CODE w$=12 THEN  PRINT AT w,k; OVER 1; INVERSE 0;" ": LET w=w-(d=0)*(w>0
)*(k=0)-(d=1)*(w>0): LET k=(k>0)*(d=0)*(k-1)+(d=0)*(w>=0)*(k=0)*31+(d=1)*k: PRIN
T AT w,k; INVERSE 0;" "
6018 IF w$>=" " AND w$<="♦" THEN  PRINT AT w,k;w$: LET w=w+d*((w+d)<=21): LET k=
k+(1-d)*((k+x)<=31)
6020 BEEP .05,12
6022 PRINT AT w,k; INVERSE 0; OVER 1;" "
6024 LET w=w+(w<21)*(CODE w$=10)-(w>0)*(CODE w$=11)
6026 LET k=k+(k<31)*(CODE w$=9)-(k>0)*(CODE w$=8)
6028 PRINT AT w,k; OVER 1; INVERSE 0; BRIGHT 1;" "
6030 GO TO 6008
6100 REM UDG
6102 PRINT #0;AT 0,0;"KLAWISZ:    ABCDEFGHIJKLMNOPQRSTUZNAK  G:               0 ZC
♦EABCD"
6104 LET w$=INKEY$: IF w$="" THEN  GO TO 6104
6106 IF CODE w$=13 THEN  LET w$="": PAUSE 30: RETURN
618 LET w$=CHR$ (((w$>="a") AND (w$<="u"))*47+((w$>="A") AND (w$<="U"))*79+CODE
w$): IF w$>=" " AND w$<="D" THEN  PRINT AT w,k;w$: LET w=w+d*(w<21): LET k=k+(1-
d)*(k<31): PAUSE 20: RETURN
6110 GO TO 6104
```

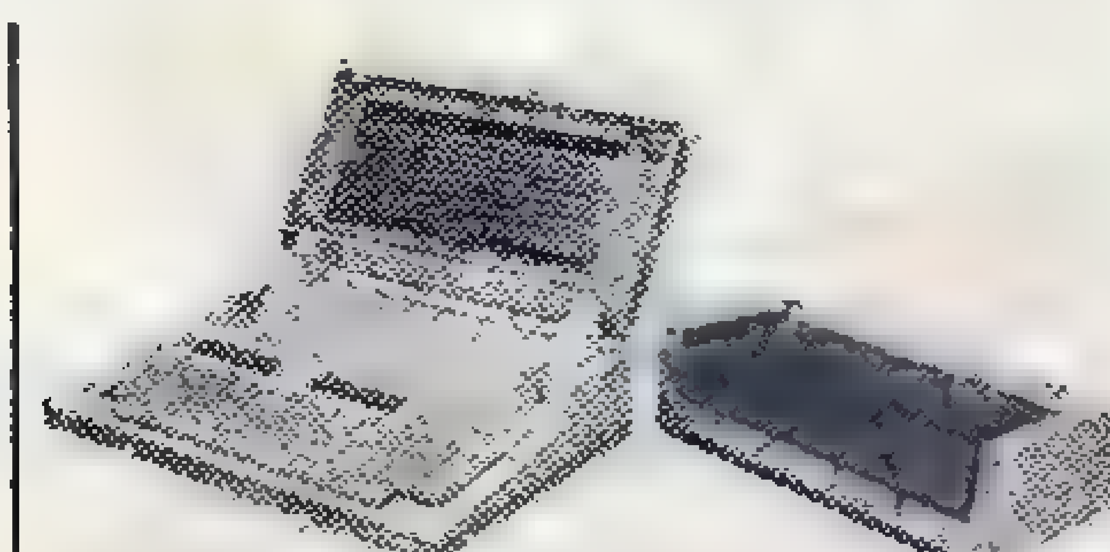
WIESŁAW RYCHLICKI



9. **Grid Compass**
Mikroprocesor: 8086 i 8087
System operacyjny: MS-DOS, Grid--OS
Pamięć RAM: 512 KB
Pamięć masowa: Floppy-Disc 360 KB, dysk twardy 10 MB
Grafika: 320x240, 512x256
Zasilanie: sieć lub akumulator
Rozmiary: 29x5x38 cm
Waga: 4.6 kg
Cena: 17 700 M

10. **HP 110** (Hewlett Packard)
Mikroprocesor: 80C86, 5.33MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 96 KB lub 272 KB
Pamięć masowa: Floppy-Disc (3.5") 710 KB, dysk twardy 10 MB
Grafika: 480x128
Zasilanie: sieć lub akumulator
Rozmiary: 32x7x25 cm
Waga: 3.9 kg

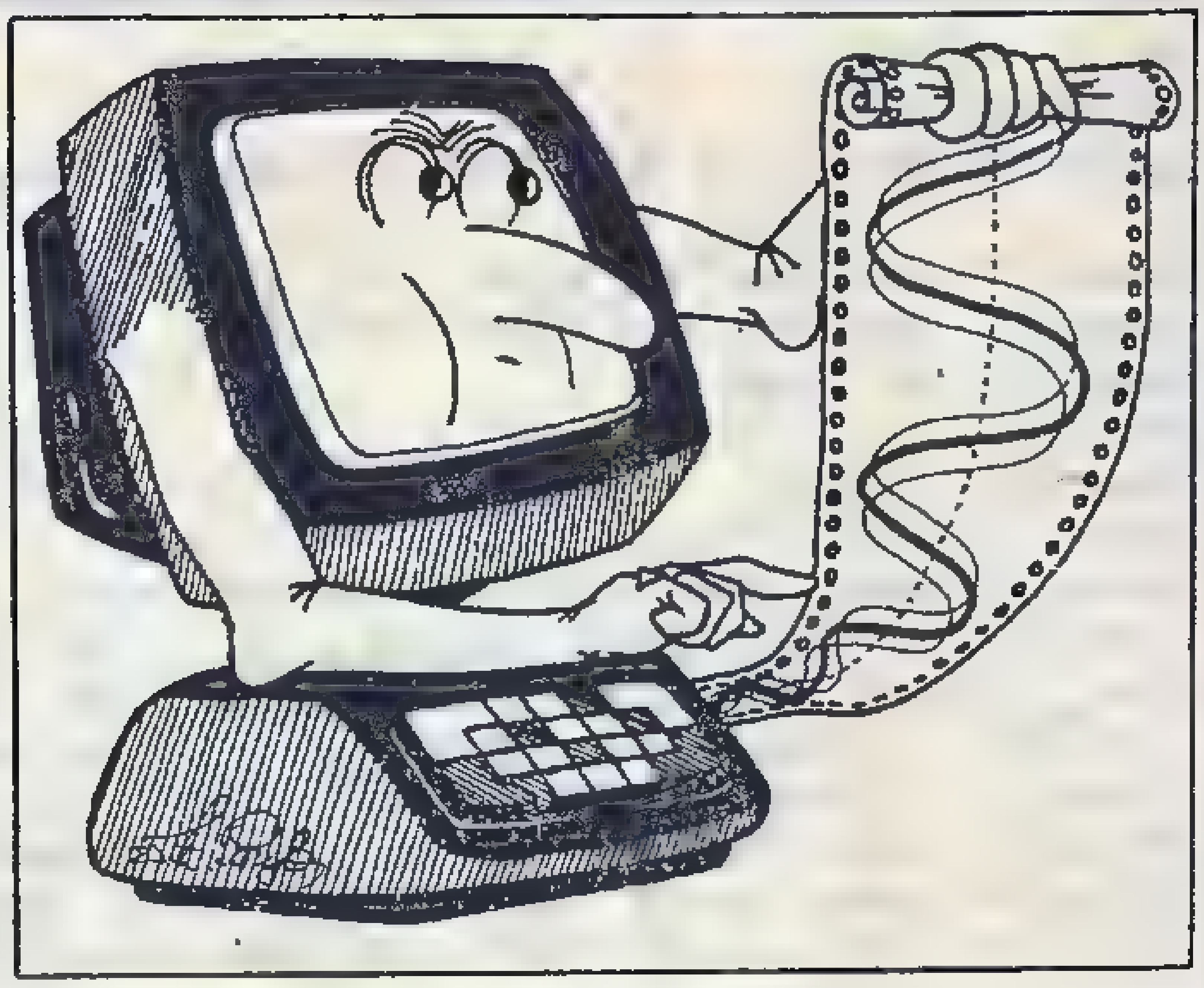




11. **HP Portable Plus** (Hewlett Packard)
Mikroprocesor: 80C86, 5.33MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 256 KB lub 1.2 MB
Pamięć masowa: RAM-Disc 32 KB lub 1.1 MB, Floppy--Disc 710 KB, dysk twardy 10 MB
Grafika: 480x200
Zasilanie: sieć lub akumulator
Rozmiary: 32x7.6x25 cm
Waga: 4.5 kg
Cena: 8500 M

AMSTRAD

PROGRAMY 5—8



PROSTA

Program kreśli prostą o równaniu y=ax+b w układzie współrzędnych OXY w zależności od podanych parametrów a i b; pozwala zaobserwować jak zmienia się rysowana prosta w zależności od zadanych parametrów. W przedstawionym układzie współrzędnych można narysować cztery proste.

Jeśli chcemy wykreślić linię (z założeń programowych n=2), to zmieniamy w programie wiersz 400 pisząc, zamiast l=3 l=n—1 i wiersz 410 pisząc zamiast l=4, l=n, gdzie za n wstawiamy liczbę naturalną, większą lub równą 2.

Aby rysowana prosta „zmieściła się" w oknie monitora, przyjęto ograniczenie na parametr b, a mianowicie $b \in \langle -18,18 \rangle$

```
10 REM P R O S T A
20 CLS
30 GOSUB 540
40 CLS:BORDER 3
50 WINDOW #2,10,31,25,25
60 PAPER #2,13:PEN #2,0:CLS #2
70 WINDOW #3,35,40,1,1
80 PAPER #3,13:PEN #3,0:CLS #3
90 PRINT #3,"  0"
100 WINDOW #1,1,8,1,3
110 BORDER 9
120 PAPER #1,13:PEN #1,0:CLS #1
130 PRINT #2,"       y= ax + b   "
140 PLOT 20,200
150 FOR s=20 TO 620
160 PLOT s,200:NEXT
170 PLOT 618,202:PLOT 618,199
180 PLOT 616,204:PLOT 616,197
190 PLOT 320,400
200 FOR s=20 TO 390
210 PLOT 320,s:NEXT
220 PLOT 318,388:PLOT 316,386
230 PLOT 322,388:PLOT 324,386
240 LOCATE 39,14:PRINT "x"
250 LOCATE 19,1:PRINT "y"
260 l=0
270 INPUT #1,"a=";a
280 GOTO 720
290 INPUT #1,"b=";b
300 IF b<-18 OR b>18 THEN 730
310 IF b>=0 THEN 430 ELSE 440
320 CLS #2
330 PRINT #2,"       y=";a;"x ";
    d$ABS(b)
340 FOR x=-320 TO 320
350 y=a*x*0.92+b*9.5
360 PLOT  320+x,200+y
370 NEXT
380 CLS #1:CLS #3:l=l+1
390 PRINT #3," ";l
400 IF l=3 THEN 450
410 IF l>=4 THEN 480
420 GOTO 270
430 d$="+":GOTO 320
440 d$="-":GOTO 320
450 ENV 3,=9,9000
460 SOUND 1,300,0,13,3
470 GOTO 270
480 LOCATE 9,24
490 PRINT " koniec -";
    " [DOWOLNY KLAWISZ]"
500 ENV 3,=9,9000
510 SOUND 1,100,0,13,3
520 IF INKEY$="" THEN 520
530 GOTO 40
540 t1$="P  R  O  S  T  A "
550 T2$=" "+CHR$(164)+
    "  "+" CJS "
560 MODE 0
570 ENV 1,=9,2000
580 ENT -1,6,3,1
590 ENV 2,127,0,0,127,0,
    0,127,0,0,127,0,0
600 ENV 3,=9,9000
610 FOR c=1 TO LEN(t1$)
620 LOCATE 2+c,10
630 FOR n2=LEN(t1$)
    TO c STEP -1
640 PRINT MID$(t1$,n2,1)
650 LOCATE 2+c,10
660 SOUND 135,20*n2,5,12,2,1
670 NEXT n2,c
```

```
680 SOUND 135,100,0,13,3,1,20
690 LOCATE 5,15:PRINT t2$
700 FOR c=1 TO 2000:NEXT
710 CLS:MODE 1:RETURN
720 LOCATE 10,23:GOTO 290
730 LOCATE 13,25
740 PRINT " PODAJ b";
    CHR$(180);"<-18,18>"
750 FOR w=1 TO 3500:NEXT
760 PRINT #2,"       y= ax + b  "
770 GOTO 290
```

PARABOLA, ELIPSA, HIPERBOLA

Program PARABOLA kreśli krzywą o równaniu $y=ax^2+bx+c$ w układzie współrzędnych OXY w zależności od podanych przez użytkownika parametrów a, b i c.

Program pozwala zaobserwować jak zmienia się rysowana parabola w zależności od zadanych parametrów a, b i c. W dowolnej części monitora w oknie przedstawione jest równanie $y=ax^2+bx+c$ aktualnie kreślonej paraboli.

W przedstawionym układzie współrzędnych można narysować l=4 parabole. Jeśli chcemy wykreślić $l=n\neq 4$ paraboli (ale z założeń programowych $l=n\geq 2$), to zmieniamy wiersz 460 pisząc zamiast *l=3* *l=n—1* i wiersz 470, pisząc zamiast *l=4* *l=n*, gdzie za *n* wstawiamy konkretną liczbę naturalną większą lub równą 2. Aby rysowana parabola zmieściła się w oknie przyjęto ograniczenia na parametr b i c, a mianowicie $b\in <-9,9>$, a $c\in <-9,9>$. Parametr a powinien być liczbą całkowitą z przedziału $<-9,9>$.

Konstrukcje pozostałych programów: „Elipsa" i „Hiperbola" oparte są dokładnie na tej samej zasadzie.

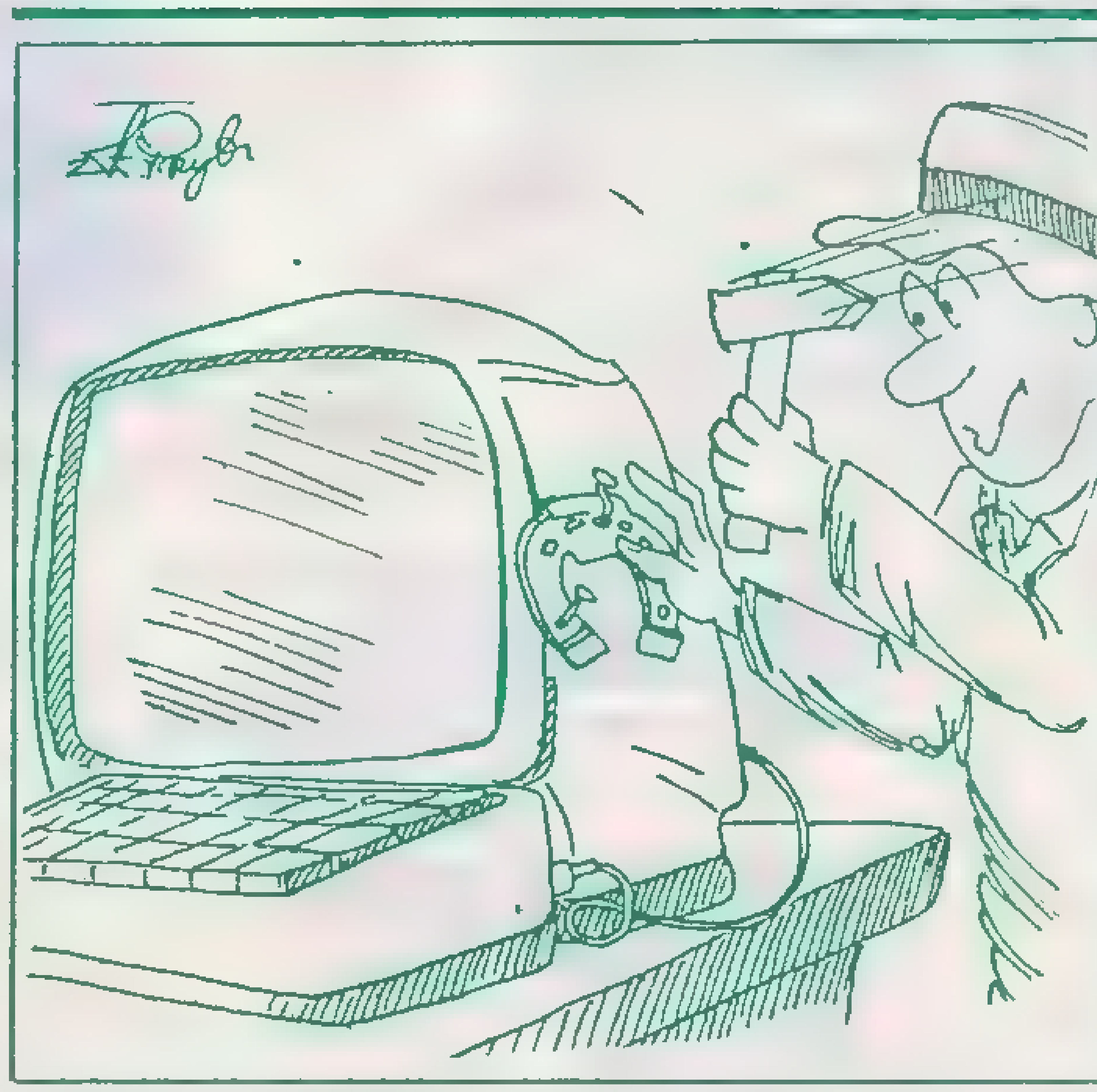
```
10 REM P A R A B O L A
20 CLS
30 GOSUB 640
40 CLS:BORDER 3
50 WINDOW #2,12,33,24,25
60 PAPER #2,13:PEN #2,0:CLS #2
70 WINDOW #3,35,40,1,1
80 PAPER #3,13:PEN #3,0:CLS #3
90 PRINT #3,"  0"
100 WINDOW #1,1,10,1,4
110 BORDER 9
120 PAPER #1,13:PEN #1,0:CLS #1
130 GOSUB 490
140 PLOT 100,220
150 FOR s=50 TO 620
160 PLOT s,220:NEXT
170 PLOT 618,222:PLOT 618,219
180 PLOT 616,224:PLOT 616,217
190 PLOT 320,400
200 FOR s=20 TO 390
210 PLOT 340,s:NEXT
220 PLOT 338,388:PLOT 336,386
230 PLOT 342,388:PLOT 344,386
240 LOCATE 39,13:PRINT "x"
250 LOCATE 20,1:PRINT "y"
260 l=0
270 INPUT #1,"a=";a
280 IF a=0 THEN 810
290 IF a>9 OR a<-9 THEN 850
300 INPUT #1,"b=";b
310 IF b>9 OR b<-9 THEN 890
320 INPUT #1,"c=";c
330 IF c>9 OR c<-9 THEN 930
340 IF b>=0 THEN 520 ELSE 540
350 IF c>=0 THEN 530 ELSE 550
360 CLS #2
370 PRINT #2, "         2   "
380 PRINT #2," y=";a;
    "x ";d$b;"x ";e$c
390 FOR x=-300 TO 300
400 y=(a*(x)^2)/100+(b*x)/4+c*18
410 IF y>180 OR y<-180 THEN 430
420 PLOT 340+x,220+y
430 NEXT
440 CLS #1:CLS #3:l=l+1
450 PRINT #3," ";l
460 IF l=3 THEN 560
470 IF l>=4 THEN 580
480 GOTO 270
490 PRINT #2,"         2   "
500 PRINT #2," y= ax + bx + c "
510 RETURN
520 d$="+":GOTO 350
530 e$="+":GOTO 360
540 d$=" ":GOTO 350
550 e$=" ":GOTO 360
560 ENV 3,=9,9000
570 SOUND 1,300,0,13,3:GOTO 270
580 LOCATE 9,23
590 PRINT " koniec -"
    " [DOWOLNY KLAWISZ]"
600 ENV 3,=9,9000
610 SOUND 1,100,0,13,3
620 IF INKEY$="" THEN 620
630 GOTO 40
640 t1$="P A R A B O L A"
650 T2$=CHR$(164)+" "+
    CHR$(32)+" CJS "
660 MODE 0
670 ENV 1,=9,2000:ENT -1,6,3,1
680 ENV 2,127,0,0,127,0,
    0,127,0,0,127,0,0
690 ENV 3,=9,9000
700 FOR c=1 TO LEN(t1$)
710 LOCATE 2+c,10
720 FOR n2=LEN(t1$) TO c STEP -1
730 PRINT MID$(t1$,n2,1)
```

```
740 LOCATE 2+c,10
750 SOUND 135,20*n2,5,12,2,1
760 NEXT n2,c
770 SOUND 135,100,0,13,3,1,20
780 LOCATE 5,15:PRINT t2$
790 FOR c=1 TO 2000:NEXT
800 CLS:MODE 1:RETURN
810 LOCATE 12,24
820 PRINT " WPISZ a"
    " ROZNE OD ZERA    "
830 FOR w=1 TO 2000:NEXT
840 GOSUB 490:GOTO 270
850 LOCATE 12,24
860 PRINT "     PODAJ a";
    CHR$(180);"<-9,9>     "
870 FOR w=1 TO 3500:NEXT
880 GOSUB 490:GOTO 270
890 LOCATE 12,24
900 PRINT "     PODAJ b";
    CHR$(180);"<-9,9>     "
910 FOR w=1 TO 3500:NEXT
920 GOSUB 490:GOTO 300
930 LOCATE 12,24
940 PRINT "     PODAJ c";
    CHR$(180);"<-9,9>     "
950 FOR w=1 TO 3500:NEXT
960 GOSUB 490:GOTO 320
```

```
10 REM E L I P S A
20 CLS
30 GOSUB 800
40 CLS:BORDER 3
50 WINDOW #2,11,31,21,25
60 PAPER #2,13:PEN #2,0:CLS #2
70 WINDOW #3,35,40,1,1
80 PAPER #3,13:PEN #3,0:CLS #3
90 PRINT #3,"  0"
100 WINDOW #1,1,9,1,4
110 BORDER 9
120 PAPER #1,13:PEN #1,0:CLS #1
130 GOSUB 660
140 PLOT 100,240
150 FOR s=100 TO 600
160 PLOT s,240:NEXT
170 PLOT 598,242:PLOT 598,239
180 PLOT 596,244:PLOT 596,237
190 PLOT 340,400
200 FOR s=80 TO 390
210 PLOT 340,s:NEXT
220 PLOT 338,388:PLOT 336,386
230 PLOT 342,388:PLOT 344,386
240 LOCATE 38,12:PRINT "x"
250 LOCATE 20,1:PRINT "y"
260 l=0
270 INPUT #1,"a=";a
280 IF a=0 THEN 970
290 IF a<-9 OR a>9 THEN 1030
300 a=a*15
310 INPUT #1,"b=";b
320 IF b=0 THEN 1000
330 IF b<-9 OR b>9 THEN 1070
340 b=b*15:CLS #2
350 PRINT #2,"      2      2 "
360 PRINT #2,"      x      y "
370 PRINT #2,"     --- + --- = 1"
380 PRINT #2,"      2      2 "
390 PRINT #2,"     ";
    a/15;"   ";b/15;"
400 n=200
410 FOR x=0 TO -n STEP -1
420 k=(a-x)*(a+x)
430 IF k<0 THEN 500
440 y=(b/a)*SQR(k)
450 IF y>158 THEN 470
460 PLOT 340+x,240+y
470 y1=-(b/a)*SQR(k)
480 IF y1<-158 THEN 510
490 PLOT 340+x,240+y1
500 NEXT
510 FOR z=0 TO n
520 k1=(a-z)*(a+z)
530 IF k1<0 THEN 600
540 y2=(b/a)*SQR(k1)
550 IF y2>158 THEN 570
560 PLOT 340+z,240+y2
570 y3=-(b/a)*SQR(k1)
580 IF y3<-158 THEN 610
590 PLOT 340+z,240+y3
600 NEXT
610 CLS #1:CLS #3:l=l+1
620 PRINT #3," ";l
630 IF l=2 THEN 720
640 IF l>=3 THEN 740
650 GOTO 270
660 PRINT #2,"      2      2 "
670 PRINT #2,"      x      y "
680 PRINT #2,"     --- + --- = 1"
690 PRINT #2,"      2      2 "
700 PRINT #2,"      a      b "
710 RETURN
720 ENV 3,=9,9000
730 SOUND 1,300,0,13,3:GOTO 270
740 LOCATE 9,21
750 PRINT " koniec -"
    " [DOWOLNY KLAWISZ]"
760 ENV 3,=9,9000
770 SOUND 1,100,0,13,3
780 IF INKEY$="" THEN 780
790 ORIGIN 0,0:GOTO 40
```

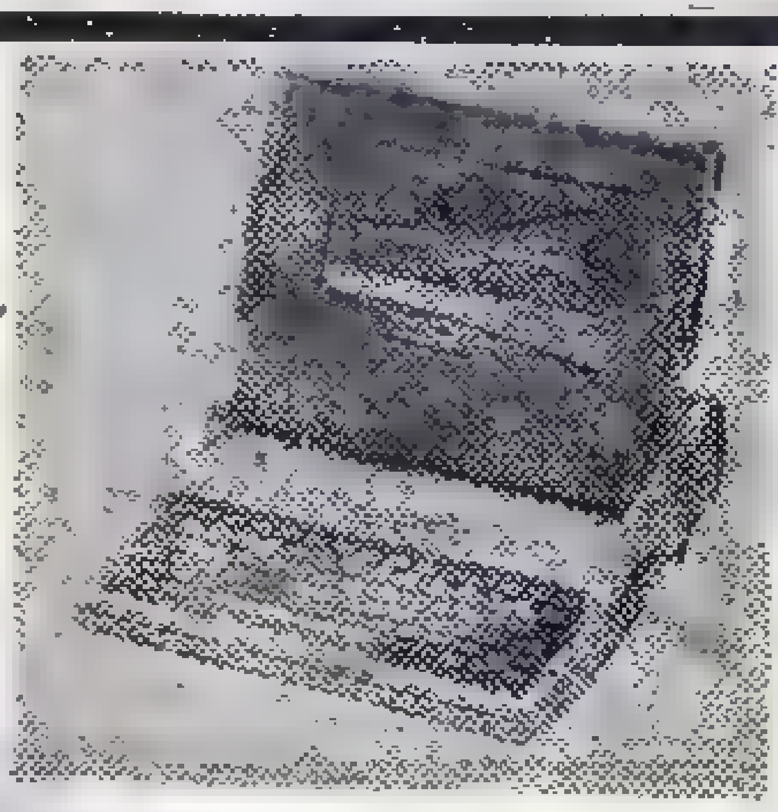
```
800 t1$="E  L  I  P  S  A"
810 T2$=" "+CHR$(164)+
    "  "+" CJS "
820 MODE 0
830 ENV 1,=9,2000:ENT -1,6,3,1
840 ENV 2,127,0,0,127,
    0,0,127,0,0,127,0,0
850 ENV 3,=9,9000
860 FOR c=1 TO LEN(t1$)
870 LOCATE 2+c,10
880 FOR n2=LEN(t1$) TO c STEP -1
890 PRINT MID$(t1$,n2,1)
900 LOCATE 2+c,10
910 SOUND 135,20*n2,5,12,2,1
920 NEXT n2,c
930 SOUND 135,100,0,13,3,1,20
940 LOCATE 5,15:PRINT t2$
950 FOR c=1 TO 2000:NEXT
960 CLS:MODE 1:RETURN
970 LOCATE 11,21:PRINT
    "WPISZ a ROZNE OD ZERA"
980 FOR w=1 TO 2000:NEXT
990 GOSUB 660:GOTO 270
1000 LOCATE 11,21:PRINT
     "WPISZ b ROZNE OD ZERA"
1010 FOR w=1 TO 2000:NEXT
1020 GOSUB 660:GOTO 310
1030 LOCATE 11,21
1040 PRINT "   PODAJ a";
     CHR$(180);"<-9,9>      "
1050 FOR w=1 TO 3500:NEXT
1060 GOSUB 660:GOTO 270
1070 LOCATE 11,21
1080 PRINT "   PODAJ b";
     CHR$(180);"<-9,9>      "
1090 FOR w=1 TO 3500:NEXT
1100 GOSUB 660:GOTO 310
```

```
10 REM H I P E R B O L A
20 CLS
30 GOSUB 640
40 CLS:BORDER 3
50 WINDOW #2,9,33,23,25
60 PAPER #2,13:PEN #2,0:CLS #2
70 WINDOW #3,35,40,1,1
80 PAPER #3,13:PEN #3,0:CLS #3
90 PRINT #3,"  0"
100 WINDOW #1,1,9,1,4
110 BORDER 9
120 PAPER #1,13:PEN #1,0:CLS #1
130 GOSUB 510
140 PLOT 100,220
150 FOR s=100 TO 600
160 PLOT s,220:NEXT
170 PLOT 598,222:PLOT 598,219
180 PLOT 596,224:PLOT 596,217
190 PLOT 340,400
200 FOR s=50 TO 390
210 PLOT 340,s:NEXT
220 PLOT 338,388:PLOT 336,386
230 PLOT 342,388:PLOT 344,386
240 LOCATE 38,13:PRINT "x"
250 LOCATE 20,1:PRINT "y"
260 l=0
270 INPUT #1,"a=";a
280 IF a=0 THEN 820
290 IF a<-9 OR a>9 THEN 860
300 IF a=-1 OR a=1 THEN 860
310 a=a*10:CLS #2
320 PRINT #2,"          1   2"
    "           1   2"
330 PRINT #2,"  xy = -";a/10;
    "  xy = - -";a/10;
340 PRINT #2,"          2"
    "              2   "
350 n=220
360 FOR x=-n TO 0 STEP 1
370 y=(a*a)/(2*x)
380 IF y<-170 THEN 410
390 PLOT 340+x,220+y
400 NEXT
410 FOR x=n TO 0 STEP -1
420 y=(a*a)/(2*x)
430 IF y>170 THEN 460
440 PLOT 340+x,220+y
450 NEXT
460 CLS #1:CLS #3:l=l+1
470 PRINT #3," ";l
480 IF l=2 THEN 550
490 IF l>=3 THEN 580
```



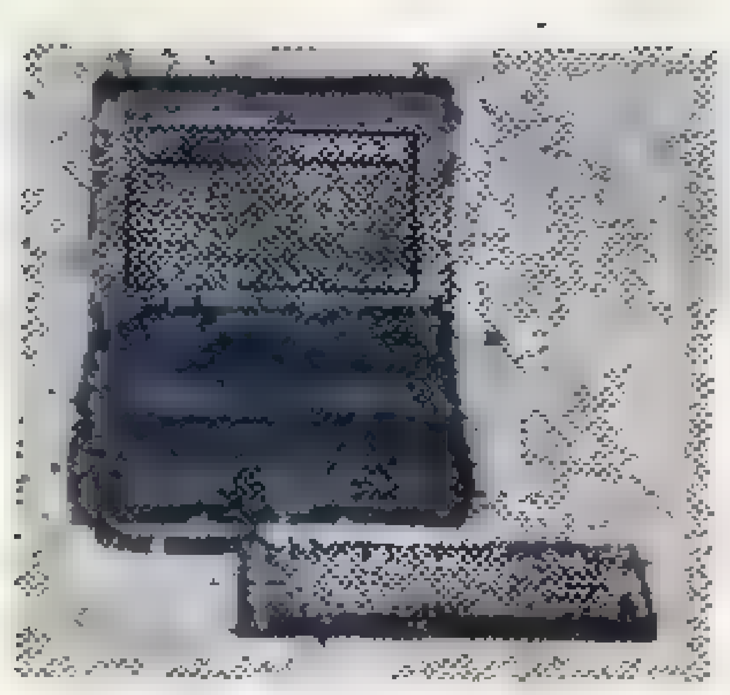
```
500 GOTO 270
510 PRINT #2,"      1  2";
    "          1  2 "
520 PRINT #2," xy = - a";
    "   xy = - - a  "
530 PRINT #2,"         2";
    "              2   "
540 RETURN
550 ENV 3,=9,9000
560 SOUND 1,300,0,13,3
570 GOTO 270
580 LOCATE 9,22
590 PRINT "koniec -";
    " [DOWOLNY KLAWISZ]"
600 ENV 3,=9,9000
610 SOUND 1,100,0,13,3
620 IF INKEY$="" THEN 620
630 ORIGIN 0,0:GOTO 40
640 t1$="H I P E R B O L A"
650 T2$=" "+CHR$(164)+
    "  "+" CJS "
660 MODE 0
670 ENV 1,=9,2000
680 ENT -1,6,3,1
690 ENV 2,127,0,0,127,
    0,0,127,0,0,127,0,0
700 ENV 3,=9,9000
710 FOR c=1 TO LEN(t1$)
720 LOCATE 2+c,10
730 FOR n2=LEN(t1$)
    TO c STEP -1
740 PRINT MID$(t1$,n2,1)
750 LOCATE 2+c,10
760 SOUND 135,20*n2,5,12,2,1
770 NEXT n2,c
780 SOUND 135,100,0,13,3,1,20
790 LOCATE 5,15:PRINT t2$
800 FOR c=1 TO 2000:NEXT
810 CLS:MODE 1:RETURN
820 LOCATE 9,23
830 PRINT "  WPISZ a";
    " ROZNE OD ZERA    "
840 FOR w=1 TO 2000:NEXT
850 GOSUB 510:GOTO 270
860 LOCATE 9,23
870 PRINT "PODAJ a";
    CHR$(180);"<-9,-2> ";
    CHR$(118);" a";
    CHR$(180);"<2,9>"
880 FOR w=1 TO 4000:NEXT
890 GOSUB 510:GOTO 270
```

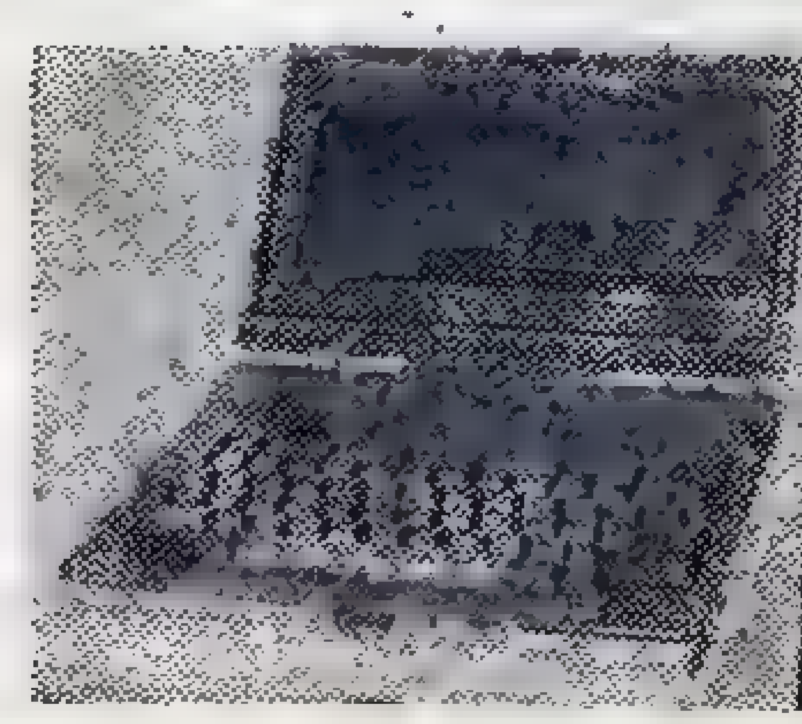
JACEK SZANIAWSKI
CEZARY SOBCZAK



12. **DG One, Modell 2** (Data General)
Mikroprocesor: 80C88, 4.77MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 256 KB lub 640 KB
Pamięć masowa: 1 lub 2 Floppy-Disc 720 KB, dysk twardy 10 MB
Grafika: 640x200 (EL), 640x256 (LCD)
Zasilanie: sieć lub akumulator
Rozmiary: 37x8x30 cm
Waga: 4.7 kg

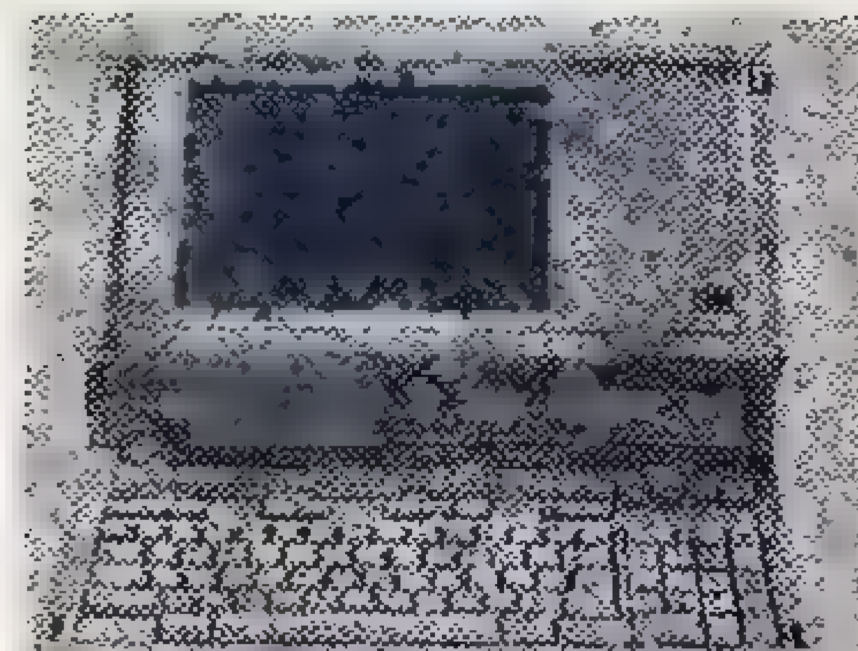
13. **Kaypro 2000**
Mikroprocesor: 8088, 4.77MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 768 KB
Pamięć masowa: Floppy-Disc 720 KB
Grafika: 640x200
Zasilanie: sieć lub akumulator
Rozmiary: 33x7x28.7 cm
Waga: 5 kg
Cena: 4350 M





14. **NEC 8401 BD**
Mikroprocesor: Z80A, 3.99MHz
System operacyjny: CP/M 2.2
Pamięć RAM: 64 KB lub 196 KB
Pamięć masowa: Floppy-Disc 320 KB
Grafika: 480x128
Zasilanie: bateria
Rozmiary: 30x7x21 cm
Waga: 2.7 kg
Cena: 1990 M

15. **Olivetti M 22**
Mikroprocesor: 2x80C88, 4.77MHz
System operacyjny: MS-DOS
Pamięć RAM: 256 KB lub 1 MB
Pamięć masowa: Floppy-Disc 360 KB, opcjonalnie dysk twardy 10 MB
Grafika: 640x200, 320x200
Zasilanie: sieć lub akumulator
Rozmiary: 40x9.1x34.4 cm
Waga: 7.5 kg
Cena: 7050 M



AMSTRAD

PROGRAM 9

KRZYŻÓWKA

Program KRZYŻÓWKA jest narzędziem do tworzenia najpopularniejszej formy rozrywek umysłowych. Głównym efektem tej zabawy są wydruki ułożonego diagramu, kontrolnego układu haseł oraz haseł wraz ze współrzędnymi do późniejszego ich zdefiniowania. Rozmiary krzyżówki są opcjonalne, lecz ograniczone do prostokąta 25x22 kratek, co w zasadzie powinno wystarczyć. W pola diagramu można wpisywać pełny alfabet polski. Litery kłopotliwe tzn. Ć, Ł, Ź... itp. wczytuje się przez CONTROL C, L, Z... itd. Przy wpisywaniu liter należy posługiwać się kursorem, który jest nieodłącznym atrybutem diagramu i spełnia rolę wskaźnika. Kursor porusza się w czterech kierunkach po naciśnięciu klawisza ze strzałką. Dodatkowe funkcje pozwalają na umieszczenie w diagramie wyrazów lub ich fragmentów w poziomie lub w pionie. Funkcje te umożliwią wpis jedynie wtedy, gdy wyraz będzie pasował do aktualnej konfiguracji liter. Kropka oznacza, że pole jest puste, zaś gwiazdka symuluje pole niewykorzystane przez autora (tzw. czarne).

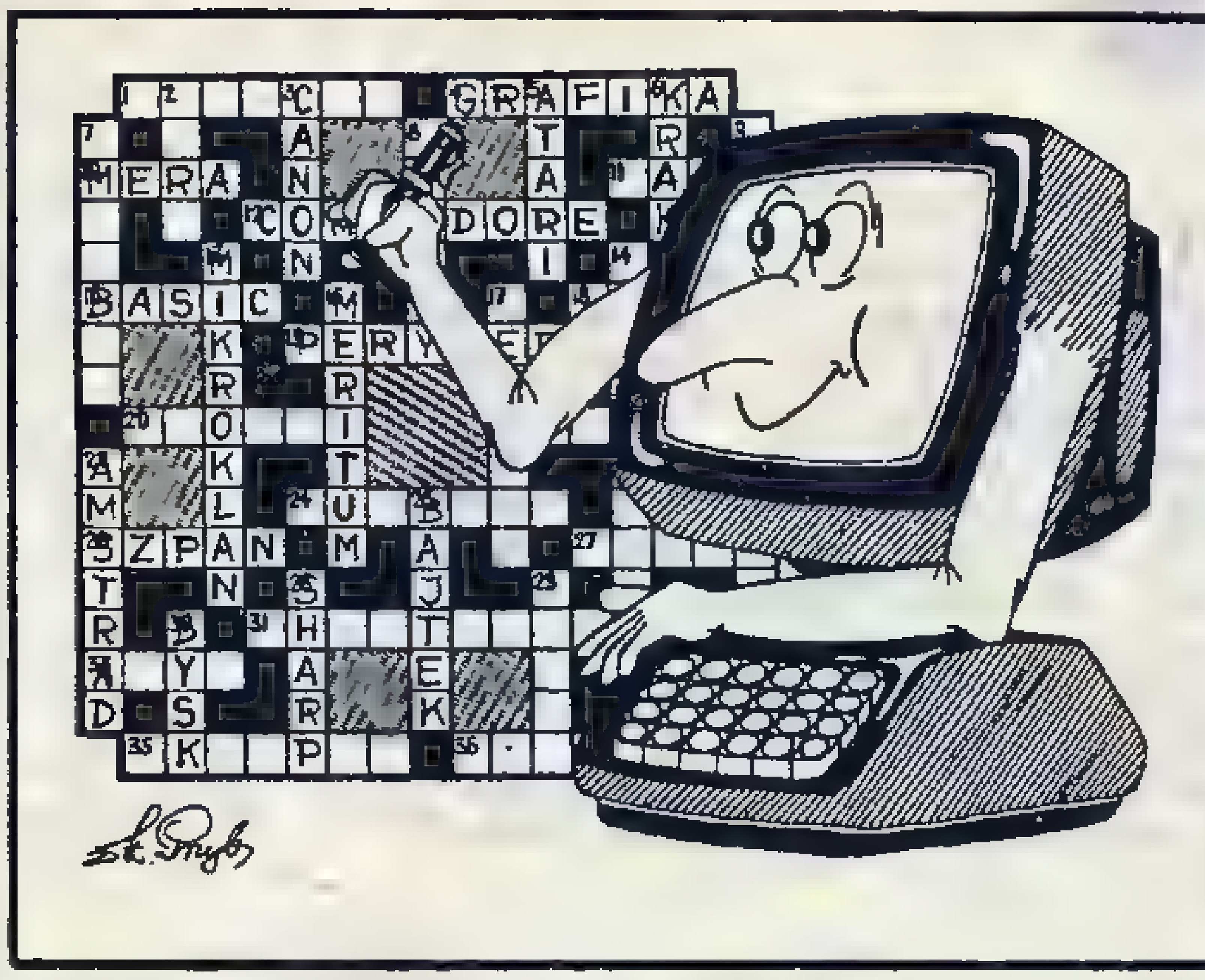
W przypadku przerwania pracy koncepcyjnej program pozwala na zapis dotychczasowego stanu na dyskietce, a potem na ewentualne powtórne wczytanie krzyżówki i kontynuowanie pracy. Należy podawać maksymalnie 8-literowy identyfikator krzyżówki bez uwzględniania typu zbioru, gdyż program automatycznie dodaje rozszerzenie KRZ. Jeśli krzyżówka posiada hasło będzie ono również zapamiętane na dyskietce.

Hasło wczytuje się na początku programu (RETURN ignoruje), lecz można je potem modyfikować w dowolnym momencie. Co kilka sekund hasło jest weryfikowane a na ekranie podawane są brakujące w diagramie litery. Jeśli w danym momencie wszystkie litery hasła występują już w diagramie, to ukazuje się komunikat „hasło kompletne".

Pełny zestaw możliwości programu wraz z klawiszami wywołującymi poszczególne moduły uzyskuje się po naciśnięciu klawisza TAB.

Trudno przedstawić wszystkie rozwiązania ze względu na brak miejsca, lecz krótko chciałbym wskazać na potężną pracę systemu przerwań, który regularnie weryfikuje hasło. Przerwania dostępne w Basic'u są bardzo wygodnym środkiem. Ich poznanie nie nastręcza większych trudności, a pozwala na niezwykle śmiałe posunięcia.

```
10 ' ** krzyzowka '86
20 DEFINT a-z:DIM krz$(25,22):ZONE 12
30 ON BREAK GOSUB 1520
40 diak$="6962583646346071"
50 nor$="ACELNOSZ":char=140
60 '
70 MODE 1:BORDER 0:PAPER 0:PEN 3
80 INK 0,0:INK 2,20:INK 1,21:INK 3,6
```



```
90 WINDOW #1,1,40,25,25
100 PAPER #1,2:PEN #1,0
110 WINDOW #3,29;40,2,23
120 PAPER #3,1: PEN #3,0:CLS #3
130 EVERY 50,3 GOSUB 8090
140 'definicje polskich liter
150 SYMBOL AFTER 140
160 SYMBOL 140,24,60,98,98,126,98,103,1
170 SYMBOL 141,62,104,200,192,192,102,60,0
180 SYMBOL 142,254,98,104,120,106,99,254,6
190 SYMBOL 143,240,100,104,112,98,102,254,0
200 SYMBOL 144,202,234,242,218,206,198,194,0
210 SYMBOL 145,56,84,214,198,198,68,56,0
220 SYMBOL 146,62,100,96,60,6,102,60,0
230 SYMBOL 147,254,38,76,24,50,102,254,0
240 'definicje klawiszy
250 FOR a=0 TO 7
260 KEY char+1,CHR$(char)
270 b=VAL(MID$(diak$,2*a+1,2))
280 n=ASC(MID$(nor$,a+1,1))
290 KEY DEF b,1,n,0,char+1
300 char=char+1:NEXT
310 'haslo i wymiary diagramu
320 GOSUB 10050:flag=0
330 WHILE flag>=0:CLS #1
340 IF flag=0 THEN INPUT #1,"szer.krzyzowki
(2..25)  ",sz ELSE INPUT #1,"wys.krzyzowki (
2..22) ",wys
350 IF sz<2 OR sz>26 THEN ELSE flag=1: PEN #
1,3
360 IF wys<2 OR wys>22 THEN ELSE flag=-1
370 WEND
380 FOR a=1 TO wys: FOR b=1 TO sz:krz$(b,a)=
".":NEXT b,a
390 krz$(0,1)=STR$(wys)
400 krz$(0,0)=STR$(sz)
410 WINDOW #2,3,sz+2,2,wys+1
420 PAPER #2,2:PEN #2,0:CLS#2:CLS #3
430 LOCATE 3,1:flag=1
440 WHILE flag<=sz
450 PRINT CHR$(flag+64);:flag=flag+1
460 WEND
470 LOCATE 1,2:flag=1
480 WHILE flag<=wys
490 IF flag<10 THEN PRINT" ";
500 PRINT MID$(STR$(flag),2)
510 flag=flag+1
520 WEND
530 GOSUB 1500:WINDOW SWAP 0,2
540 ON BREAK GOSUB 1510
550 FOR a=1 TO wys
560 FOR b=1 TO sz:PRINT krz$(b,a);
570 NEXT b,a
580 IF haslo$="" THEN GOSUB 9000 ELSE EVERY
500,2 GOSUB 8000
590 x=1:y=1:xp=1:yp=1:GOSUB 6000
```

```
600 'menu
1000 klaw$=UPPER$(INKEY$): IF klaw$="" THEN
1000 ELSE SOUND 1,10,4:item=0
1010 IF INKEY(0)=0 THEN item=1:'^
1020 IF INKEY(1)=0 THEN item=2:'prawo
1030 IF INKEY(2)=0 THEN item=3:'dol
1040 IF INKEY(8)=0 THEN item=4:'lewo
1050 IF INKEY(6)=0 THEN item=5:'enter
1060 IF INKEY(2)=128 THEN item=6:'pion
1070 IF INKEY(1)=128 THEN item=7:'poz
1080 IF INKEY(10)=0 THEN item=8:'zapis
1090 IF INKEY(11)=0 THEN item=9:'odczyt
1100 IF INKEY(9)=0 THEN item=10:'druk.
1110 IF INKEY(68)=0 AND haslo$<>"" THEN item
=11: 'help

1120 IF INKEY(5)=0 THEN item=12:'haslo
1130 IF item>0 THEN ON item GOSUB 2000,2030,
2050,2070,2090,3150,3000,5000,4000,7000,9040
,10000 : IF item=9 THEN 410 ELSE 1000
1140 IF klaw$>CHR$(139) AND klaw$<CHR$(148)
THEN item=32
1150 IF klaw$="." OR (klaw$>="A" AND klaw$<=
"Z") OR item=32 THEN krz$(yp,xp)=klaw$:GOSUB
 2030
1160 GOTO 1000
1500 PAPER #1,0:CLS#1:PAPER #1,2:RETURN
1510 WINDOW SWAP 0,2
1520 CALL &BB00:END
2000 'kierunki
2010 x=x-1:IF x=0 THEN x=wys
2020 GOSUB 6000:RETURN
2030 y=y+1:IF y=sz+1 THEN y=1
2040 GOSUB 6000:RETURN
2050 x=x+1:IF x=wys+1 THEN x=1
2060 GOSUB 6000:RETURN
2070 y=y-1:IF y=0 THEN y=sz
2080 GOSUB 6000:RETURN
2090 'enter
2100 krz$(yp,xp)="*":GOSUB 2030:RETURN
2110 ENT 1,50,10,30,20,5,40
2120 SOUND 1,30,50,15,,1:RETURN
3000 'poziom
3010 flag=0:dlug=sz:wsp=y:off=1
3020 PRINT #1,"wyraz ";:IF flag=0 THEN PRINT
 #1,"poziomo..."; ELSE PRINT #1,"pionowo..."
;
3030 INPUT #1,wyraz$:wyraz$=UPPER$(wyraz$)
3040 FOR a=1 TO LEN(wyraz$):s$=MID$(wyraz$,a
,1): IF s$<"A" OR (s$>"Z" AND s$<CHR$(140))
OR s$>CHR$(147) THEN PRINT#1,"wyraz zawiera
zabroniony znak...":AFTER 200 GOSUB 1500: RE
TURN
3050 NEXT
3060 IF wsp+LEN(wyraz$)>dlug+1 THEN PRINT#1,
"za dlugi wyraz...":AFTER 200 GOSUB 1500:RET
URN
3070 b=0:c=0:FOR a=1 TO LEN(wyraz$)
3080 IF krz$(y+b,x+c)="." OR krz$(y+b,x+c)=M
ID$(wyraz$,a,1) THEN c=c+flag:b=b+off:NEXT E
LSE IF krz$(y+b,x+c)="*" THEN PRINT#1,"czarn
e pole na drodze..":AFTER 200 GOSUB 1500:RET
URN ELSE PRINT#1,"konfuzja liter..":AFTER 20
0 GOSUB 1500:RETURN
3090 FOR a=1 TO LEN(wyraz$)
3100 krz$(yp,xp)=MID$(wyraz$,a,1)
3110 IF flag=0 THEN GOSUB 2030 ELSE GOSUB 20
50
3120 NEXT
3130 IF (yp>1 AND flag=0) OR (xp>1 AND flag=
1) THEN krz$(yp,xp)="*":IF flag=0 THEN GOSUB
 2030 ELSE GOSUB 2050
3140 GOSUB 1500:RETURN
3150 ' pion
3160 flag=1: dlug=wys: wsp=x :off=0
3170 GOTO 3020
4000 'odczyt tablicy
4010 a=REMAIN(2):b=REMAIN(3)
4020 INPUT #1,"ODCZYT- nazwa pliku...";plik$

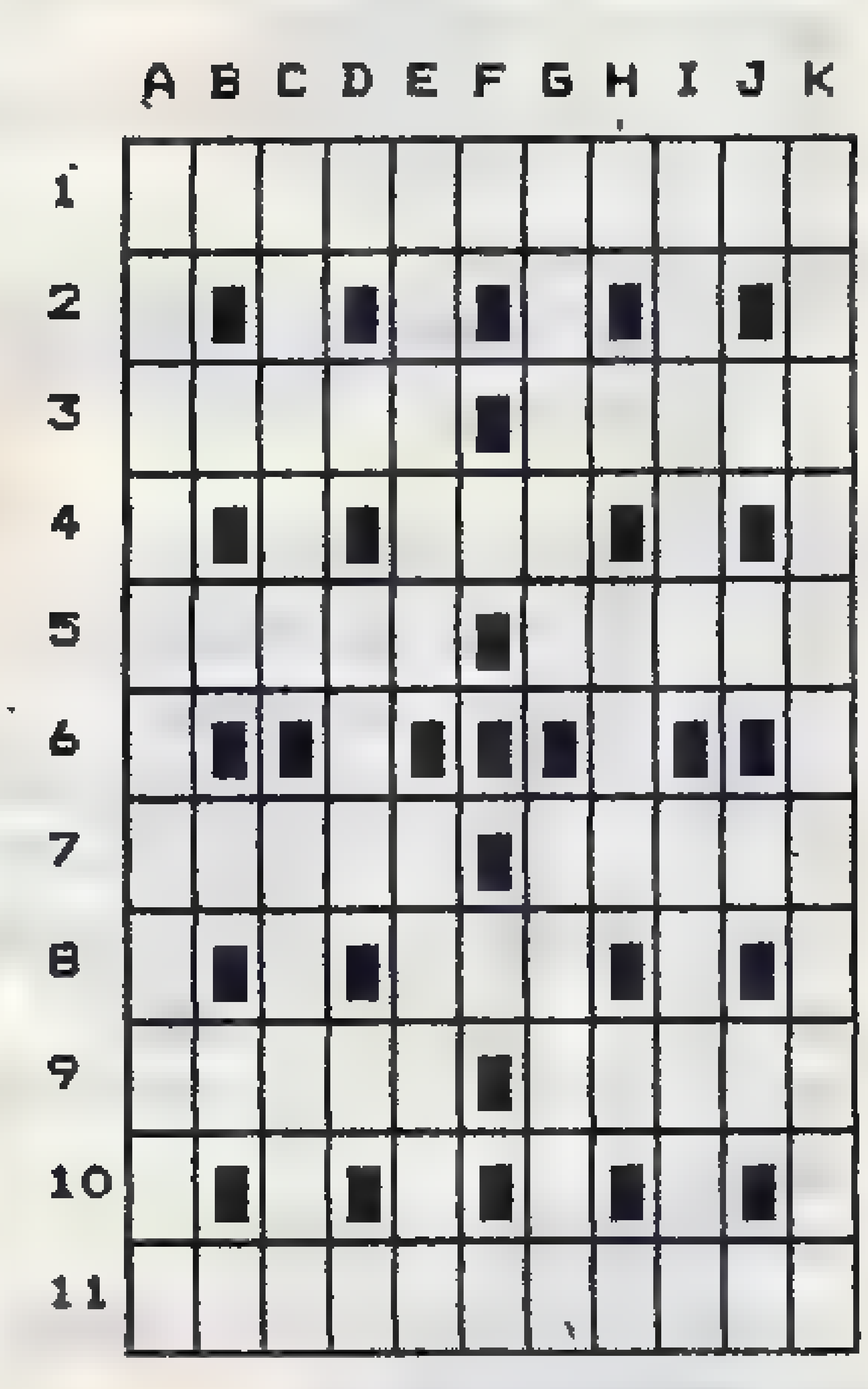
4030 IF plik$="" THEN GOSUB 1500:RETURN
4040 plik$=plik$+".krz"
4050 OPENIN plik$
4060 FOR a=0 TO 21
4070 FOR b=0 TO 25
4080 INPUT #9,krz$(b,a)
4090 NEXT b,a
4100 CLOSEIN
4110 wys=VAL(krz$(0,1)):sz=VAL(krz$(0,0)):ha
slo$=krz$(0,2)
4120 IF haslo$<>"" THEN EVERY 500,2 GOSUB 80
00
4130 EVERY 50,3 GOSUB 8090
4140 WINDOW SWAP 0,2:CLS:CLS#1
4150 GOSUB 2110:RETURN
5000 'zapis tablicy
5010 a=0:WHILE a=0:INPUT #1,"ZAPIS- nazwa pl
iku...";plik$
5020 IF LEN(plik$)<=8 THEN a=1
5030 WEND
5040 IF plik$="" THEN GOSUB 1500:RETURN
5050 plik$=plik$+".krz":krz$(0,2)=haslo$
5060 a=REMAIN(2):b=REMAIN(3)
5070 OPENOUT plik$
5080 FOR a=0 TO 21
5090 FOR b=0 TO 25
5100 WRITE #9,krz$(b,a)
5110 NEXT b,a
5120 CLOSEOUT:GOSUB 1500
5130 IF haslo$<>"" THEN EVERY 500,2 GOSUB 80
00
5140 EVERY 50,3 GOSUB 8090
5150 GOSUB 2110:RETURN
6000 'kursor
6010 LOCATE yp,xp:PRINT krz$(yp,xp);
6020 LOCATE y,x
6030 PRINT CHR$(22);CHR$(1);
6040 PEN 3:PRINT "_";
6050 PRINT CHR$(22);CHR$(0);
6060 PEN 0:yp=y:xp=x:RETURN
7000 'drukarka
7010 opc$="DKH"
7020 a=REMAIN(2):b=REMAIN(3)
7030 INPUT #1,"Diagram / Krzyzowka / Hasla";
a$:a$=UPPER$(a$)
7040 a$=UPPER$(a$)
7050 PRINT#8,CHR$(27);CHR$(64):'reset
7060 PRINT#8,CHR$(27);CHR$(80):'pica
7070 PRINT#8,CHR$(27);CHR$(71):'emfaza
7080 ON INSTR(opc$,a$) GOSUB 7120,7380,7490
7090 IF haslo$<>"" THEN EVERY 500,2 GOSUB 80
00
7100 EVERY 50,3 GOSUB 8090
7110 GOSUB 1500:GOSUB 2110:RETURN
7120 'Gemini Star diagram
7130 PRINT#8,SPC(4);
7140 FOR a=1 TO sz
7150 PRINT #8," ";UPPER$(CHR$(64+a));:NEXT
7160 PRINT#8,CHR$(27);CHR$(55);
7170 PRINT#8,CHR$(27);CHR$(62).
7180 FOR a=1 TO wys
7190 PRINT#8,STRING$(4,CHR$(255));
7200 FOR b=1 TO sz
7210 IF a=1 AND b=1 THEN PRINT#8,CHR$(218);C
HR$(196);
7220 IF a=1 AND b<>1 THEN PRINT#8,CHR$(203);
CHR$(196);
7230 IF a<>1 AND b<>1 THEN PRINT#8,CHR$(215)
;CHR$(196);
7240 IF a<>1 AND b=1 THEN PRINT#8,CHR$(198);
CHR$(196);
7250 NEXT b
7260 IF a=1 THEN PRINT#8,CHR$(183) ELSE PRIN
T#8,CHR$(181)
```

```
7270 PRINT#8, CHR$(27);CHR$(61);a;:IF a<10 T
HEN PRINT#8," ";
7280 PRINT#8,CHR$(27);CHR$(62);
7290 FOR b=1 TO sz
7300 PRINT#8, CHR$(179);
7310 IF krz$(b,a)="*" THEN PRINT#8,CHR$(219)
; ELSE PRINT#8,CHR$(255);
7320 NEXT b:PRINT#8,CHR$(179):NEXT a
7330 PRINT#8,STRING$(4,CHR$(255));
7340 FOR a=1 TO sz
7350 IF a=1 THEN PRINT#8,CHR$(211);CHR$(205)
; ELSE PRINT#8,CHR$(207);CHR$(205);
7360 NEXT:PRINT#8,CHR$(188)
7370 PRINT#8,CHR$(27);CHR$(61):RETURN
7380 FOR a=1 TO wys
7390 FOR b=1 TO sz
7400 IF krz$(b,a)<="Z" THEN PRINT#8,krz$(b,a
);:GOTO 7420
7410 c=ASC(krz$(b,a)):GOSUB 7440
7420 PRINT#8," ";:NEXT b:PRINT#8:NEXT a
7430 RETURN
7440 PRINT#8,MID$(nor$,c-139,1);
7450 IF c=140 OR c=142 THEN PRINT#8,CHR$(8);
",";
7460 IF c=141 OR c>143 THEN PRINT#8,CHR$(8);
"'";
7470 IF c=143 THEN PRINT#8,CHR$(8);"/";
7480 RETURN
7490 'hasla
7500 PRINT#8,"POZIOMO:"
7510 FOR a=1 TO wys:FOR b=1 TO sz
7520 IF krz$(b,a)<>"." AND krz$(b,a)<>"*" TH
EN b$=krz$(b,a):c$=CHR$(64+b):dlug=sz:off=1:
GOSUB 7590
7530 NEXT b,a
7540 PRINT#8,"PIONOWO:"
7550 FOR b=1 TO sz:FOR a=1 TO wys
7570 IF krz$(b,a)<>"." AND krz$(b,a)<>"*" TH
EN d=a:b$=krz$(b,a):dlug=wys:off=0:GOSUB 759
0
7580 NEXT a,b:RETURN
7590 IF off=1 THEN b=b+1: IF b>dlug THEN RET
URN
7600 IF off=0 THEN a=a+1: IF a>dlug THEN RET
URN
7610 WHILE krz$(b,a)<>"." AND krz$(b,a)<>"*"
 AND ((b<=dlug AND off=1) OR (a<=dlug AND of
f=0))
7620 b$=b$+krz$(b,a)
7630 IF off=1 THEN b=b+1 ELSE a=a+1
7640 WEND
7650 IF LEN(b$)<2 THEN RETURN
7660 IF off=1 THEN PRINT#8,a;"-";c$;" "; ELS
E PRINT#8,CHR$(64+b);"-";d;" ";
7670 FOR e=1 TO LEN(b$)
7680 c=ASC(MID$(b$,e,1))
7690 IF c<140 THEN PRINT#8,MID$(b$,e,1); ELS
E GOSUB 7440
7700 NEXT e
7710 PRINT#8:RETURN
8000 'przerwania
8010 CLS#3:PEN #3,3:PRINT#3,"HASLO:":PEN #3,
0:PRINT #3,haslo$
8020 tekst$=haslo$
8030 por$=SPACE$(LEN(tekst$))
8040 FOR a=1 TO wys:FOR b=1 TO sz
8050 IF krz$(b,a)="." OR krz$(b,a)="*" THEN
ELSE c=INSTR(tekst$,krz$(b,a)):IF c>0 THEN M
ID$(tekst$,c,1)=" "
8060 NEXT b,a
8070 IF tekst$<>por$ THEN PRINT#3:PEN #3,3:P
RINT#3,"BRAKI:":PEN #3,0:PRINT#3,tekst$ ELSE
 PRINT#3,,"haslo","kompletne"
8080 RETURN
8090 POKE &B632,255:RETURN
9000 'help
9010 CLS#3:PRINT#3,"    HELP",,:FOR a=8 TO 11
:PRINT#3,CHR$(1);CHR$(a);:NEXT:PRINT#3,"  ku
rsor",,"ENTER    *":PRINT#3
9020 PRINT#3,"wyraz:","poziomo CTL";CHR$(1);
CHR$(9);"pionowo CTL";CHR$(1);CHR$(10)
9030 PRINT#3,"F7     zapis","F8    odczyt","
F3  zm.hasla","COPY   druk.","TAB    help":R
ETURN
9040 'help
9050 a=REMAIN(2):GOSUB 9000
9060 WHILE INKEY$="":WEND:CLS#3
9070 EVERY 500,2 GOSUB 8000:RETURN
10000 'zmiana hasla
10010 CLS#3:IF haslo$<>"" THEN a=REMAIN(2)
10020 PRINT#3:PRINT#3,haslo$:GOSUB 10050
10030 CLS#3: IF haslo$<>"" THEN EVERY 500,2
GOSUB 8000
10040 GOSUB 2110:RETURN
10050 'haslo
10060 flag=1
10070 WHILE flag=1: PRINT #3,"haslo...":PRIN
T:INPUT #3,haslo$
10080 a=1:flag=0:WHILE a<=LEN(haslo$)
10090 a$=MID$(haslo$,a,1)
10100 IF a$=" " OR (a$>="A" AND a$<="Z") OR
(a$>=CHR$(140) AND a$<=CHR$(147)) THEN a=a+1
 ELSE a=LEN(haslo$)+1:flag=1
10110 WEND
10130 WEND:haslo$=UPPER$(haslo$):RETURN
```

POZIOMO:
1 -A KONSERWACJA
3 -A NUBIA
3 -G ELAND
4 -E NIZ
5 -A EUBEA
5 -G ARTEL
7 -A TRIAS
7 -G MARSZ
8 -E YMA
9 -A TOPAZ
9 -G KOLEJ
11 -A RATYFIKACJA

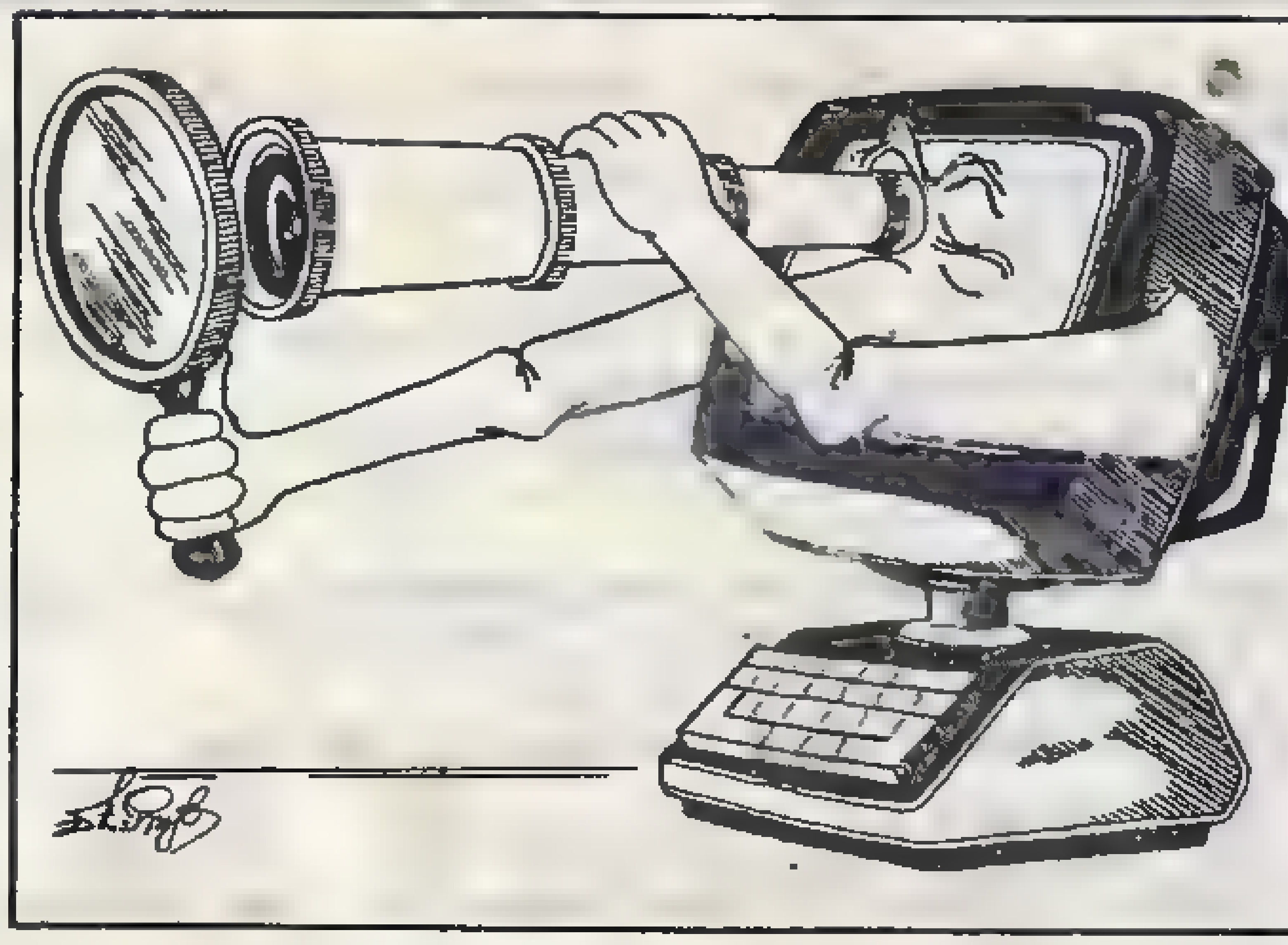
PIONOWO:
A- 1 KONTESTATOR
C- 1 NABAB
C- 7 IMPET
D- 5 ERA
E- 1 ELANA
E- 7 SYZYF
G- 1 WIEZA
G- 7 MAKAK
H- 5 REA
I- 1 CHART
I- 7 RYLEC
K- 1 ANDALUZYJKA

```
K O N S E R W A C J A
O * A * L * I * H * N
N U B I A * E L A N D
T * A * N I Z * R * A
E U B E A * A R T E L
S * * R * * * E * * U
T R I A S * M A R S Z
A * M * Y M A * Y * Y
T O P A Z * K O L E J
O * E * Y * A * E * K
R A T Y F I K A C J A
```



Krzysztof MAMCARZ

MERITUM
PROGRAMY 10—15



PROGRAMY EDUKACYJNE Z FIZYKI

```
10 REM TEST Z OPTYKI, AUTOR JAN GOLLA
20 CLEAR 500:CLS
30 PRINT@468,CHR$(23)"TEST Z OPTYKI"
40 PRINT@910,"W CELU KONTYNUACJI"
50 PRINT@974,"NACISNIJ KLAWISZ SPACJA"
60 IF INKEY$<>" " GOTO 60
70 FOR L=1 TO 5:'ILOSC PYTAN
80 READ P$
90 FOR K=1 TO 4:'ILOSC MOZLIWYCH ODPO.
100 READ A$(K)
110 NEXT:P=4:'NR. POPRAWNEJ ODPOWIEDZI
120 REM LOSOWA ZMIANA KOLEJNOSC ODPO.
130 FOR K=1 TO 5
140 RANDOM:W1=RND(4):W2=RND(4)
150 IF W2=P THEN P=W1:GOTO 170
160 IF W1=P THEN P=W2
170 B$=A$(W1):A$(W1)=A$(W2):A$(W2)=B$
180 NEXT K
190 CLS
200 PRINT@18,"PYTANIE  NR";L
210 PRINT:PRINT P$:PRINT
220 FOR K=1 TO 4
230 PRINT K;A$(K)
240 NEXT K:PRINT
250 INPUT"PODAJ NUMER ODPOWIEDZI";X
260 IF X<>P GOTO 300
270   PRINT"ODPOWIEDZIALES DOBRZE"
280   T=T+1:GOTO 320
300   PRINT"ZLE!!!!
310   PRINT"ODPOWIEDZ DOBRA MA NUMER";P
320 IF INKEY$<>" " GOTO 320
330 NEXT L
340 REM OCENA WYNIKOW TESTU
350 IF T=5 THEN O$="BARDZO DOBRA"
360 IF T=4 THEN O$="DOBRA"
370 IF T=3 THEN O$="DOSTATECZNA"
380 IF T<3 THEN O$="NIEDOSTATECZNA"
390 IF T=0 OR T>4 THEN I$="PYTAN"
400 IF T=1 THEN I$="PYTANIE"
410 IF T=2 OR T=3 THEN I$="PYTANIA"
420 CLS
430 PRINT@262,CHR$(23);
440 PRINT"ODPOWIEDZIALES DOBRZE NA";T
450 PRINT@340,I$
460 PRINT
470 PRINT"OTRZYMUJESZ OCENE ";O$
480 IF INKEY$<>" " GOTO 480 ELSE RUN
490 REM PYTANIE NR 1
500 DATA "ZJAWISKO PODWOJNEGO ZALAMANIA
SWIATLA ZACHODZI:"
510 REM ODPOWIEDZI
520 DATA WE WSZYSTKICH CIALACH PRZE
   ZROCZYSTYCH
530 DATA WE WSZYSTKICH PRZEZROCZYSTYCH
   CIALACH STALYCH
540 DATA WE WSZYSTKICH KRYSZTALACH
550 DATA W CIALACH ANIZOTROPOWYCH
560 REM PYTANIE NR 2
570 DATA "O WSPOLCZ. ZALAMANIA SWIATLA
DECYDUJE:"
580 DATA GESTOSC OBU OSRODKOW
590 DATA PREDKOSC ROZCHODZENIA SIE
   SWIATLA W OSRODKACH
600 DATA SPREZYSTOSC OBU OSRODKOW
610 DATA STOSUNEK SIN. KATOW PADANIA
   I ZALAMANIA
620 REM PYTANIE NR 3
630 DATA "FOTOMETR TO PRZYRZAD
SLUZACY DO:"
640 DATA USTALANIA CZASU NASWIETLANIA
   BLONY FOTOGRAFICZNEJ
650 DATA POMIARU PREDKOSCI SWIATLA
660 DATA POMIARU STRUMIENIA SWIETLNEGO
670 DATA POROWNYWANIA NATEZEN DWOCH
   ZRODEL SWIATLA
680 REM PYTANIE NR 4
690 DATA "W SZKLE PREDKOSC SWIATLA
JEST:"
710 DATA DLA WSZYSTKICH BARW
   JEDNAKOWA
720 DATA NAJWIEKSZA DLA SWIATLA
   FIOLETOWEGO
740 DATA NAJWIEKSZA DLA ROZNYCH BARW
   ZALEZNIE OD GATUNKU SZKLA
```

```
750 DATA NAJWIEKSZA DLA SWIATLA
   CZERWONEGO
760 REM PYTANIE NR 5
770 DATA "OBRAZ OTRZYMANY W SOCZEWCE
ROZPRASZAJACEJ JEST:"
780 DATA ZAWSZE POWIEKSZONY I
   RZECZYWISTY
790 DATA POWIEKSZONY LUB POMNIEJSZONY
   ZALEZNIE OD OGNISKOWEJ
800 DATA ZAWSZE POMNIEJSZONY I
   RZECZYWISTY
810 DATA ZAWSZE POMNIEJSZONY I
   POZORNY
```

```
10 REM RUCH JEDNOSTAJNIE PRZYSPIESZONY
20 REM AUTOR JAN GOLLA
30 CLS
40 DIM V(10):DIM S(10)
50 FOR X=1 TO 7:PRINT:NEXT
60 PRINT CHR$(23);"RUCH JEDNOSTAJNIE PRZYSPIESZONY"
70 PRINT@846,"NACISNIJ KLAWISZ SPACJA"
80 IF INKEY$()" " GOTO 80
90 CLS
100 INPUT"PODAJ PRZYSPIESZENIE 'A'";A
110 CLS
120 PRINT TAB(12)"PRZYSPIESZENIE 'A' WYNOSI ";A
130 PRINT" T=";" V=";" S=";" DS="
140 PRINT
150 FOR T=1 TO 10
160 V(T)=A*T:S(T)=(A*T*T)/2:DS=S(T)-(A*(T-1)*(T-1))/2
170 PRINT T,V(T),S(T),DS
180 NEXT T
190 PRINT@902,"W CELU KONTYNUACJI NACISNIJ KLAWISZ SPACJA"
200 IF INKEY$()" " GOTO 200
210 CLS:PRINT@0,"V":GOSUB 400
220 FOR T=1 TO 10
230 ON ERROR GOTO 500
240 SET(T*9,44-V(T)/10)
250 NEXT T
260 IF INKEY$()" " GOTO 260
270 CLS:PRINT@0,"S":GOSUB 400
280 FOR T=1 TO 10
290 ON ERROR GOTO 520
300 SET(T*9,44-S(T)/10)
310 NEXT T
320 IF INKEY$()" " GOTO 320
330 RUN
400 FOR X=0 TO 41
410 SET(X,44):SET(0,44-X)
420 NEXT X
430 FOR X=42 TO 100
440 SET(X,44)
450 NEXT X
460 PRINT@945,"T"
470 RETURN
500 IF INKEY$()" " GOTO 500
510 RESUME 270
520 IF INKEY$()" " GOTO 520
530 RUN
```

```
10 REM RZUT POZIOMY
20 REM AUTOR  JAN  GOLLA
30 CLS
40 FOR X=1 TO 7:PRINT:NEXT
50 PRINT CHR$(23);TAB(9)"RZUT POZIOMY"
60 PRINT@896,TAB(7)"NACISNIJ KLAWISZ SPACJA"
70 IF INKEY$()" " GOTO 70
80 CLS
90 INPUT"PODAJ WYSOKOSC Z KTOREJ RZUCAMY CIALO(H (48)";H
100 IF H(=0 OR H)=48 GOTO 90
110 INPUT"PODAJ PREDKOSC POCZATKOWA V0(OKOLO 100)";V0
120 CLS:PRINT TAB(49)"H=";INT(H*10)/10
130 PRINT TAB(48)"V0=";INT(V0*10)/10
140 T=SQR(2*H/9.81):D=V0*T:VK=SQR(V0*V0+2*9.81*H)
150 PRINT:PRINT:PRINT TAB(45)"T=SQR(2*H/9.81)",
160 PRINT TAB(49)"T=";INT(T*10)/10
170 PRINT:PRINT TAB(49)"D=V0*T"
180 PRINT TAB(49)"D=";INT(D*10)/10
190 PRINT:PRINT TAB(42)"VK=SQR(V0*V0+2*9.81*H)",
200 PRINT TAB(48)"VK=";INT(VK*10)/10
210 PRINT:PRINT TAB(48)"NACISNIJ"
220 PRINT TAB(48)"SPACJE"
230 FOR X=0 TO 47:SET(X,47):SET(0,47-X):NEXT
240 FOR X=48 TO 84:SET(X,47):NEXT
250 ON ERROR GOTO 370
260 XS=0:YS=0:FOR T=0 TO 84 STEP .5
270 X=V0/10*T
280 Y=9.81*T*T/20
290 IF X)84 OR Y)H GOTO 310
300 RESET(XS,47-H+YS):SET(X,47-H+Y):XS=X:YS=Y:NEXT
310 FOR T=0 TO 84 STEP 0.5
320 X=V0/10*T:Y=9.81*T*T/20
330 IF X)84 OR Y)H GOTO 350
340 SET(X,47-H+Y):NEXT
350 IF INKEY$()" " GOTO 350
360 GOTO 80
370 IF INKEY$()" " GOTO 370
380 RESUME 60
```

```
10 REM RZUT UKOSNY
20 REM AUTOR  JAN GOLLA
25 CLS
30 FOR X=1 TO 7:PRINT:NEXT
40 PRINT CHR$(23)TAB(10)"RZUT UKOSNY"
50 PRINT@910,"NACISNIJ KLAWISZ SPACJA"
60 IF INKEY$()" " GOTO 60
```

```
70 CLS
80 INPUT"PODAJ PREDKOSC POCZATKOWA V0(OKOLO 100)";V0
90 PRINT"PODAJ KAT(A) NACHYLENIA WEKTORA PREDKOSCI POCZ."
100 INPUT"DO POZIOMU W STOPNIACH(0 < A < 90)";A
110 CLS
120 PRINT:PRINT TAB(5)"V0=";V0:PRINT TAB(5)"A=";A
130 A=A*3.14/180
140 T=2*V0*SIN(A)/9.81
150 D=V0*V0*SIN(2*A)/9.81
160 H=V0*V0*SIN(A)*SIN(A)/(2*9.81)
170 PRINT:PRINT
180 PRINT "T=2*V0*SIN(A)/9.81",TAB(40)"T=";T
190 PRINT
200 PRINT "D=V0*V0*SIN(2*A)/9.81",TAB(40)"D=";D
210 PRINT
220 PRINT "H=V0*V0*SIN(A)*SIN(A)/(2*9.81)",TAB(40)"H=";H
230 PRINT@840,"W CELU KONTYNUACJI NACISNIJ KLAWISZ SPACJA"
240 IF INKEY$<>" " GOTO 240
250 CLS:FOR X=0 TO 47:SET(X,47):SET(0,47-X):NEXT
260 FOR X=48 TO 127:SET(X,47):NEXT
270 ON ERROR GOTO 420
280 XS=0:YS=0
290 FOR T=0 TO 40 STEP .5
300 X=V0*COS(A)*T/4
310 Y=(V0*SIN(A)*T-9.81*T*T)/8
320 RESET(XS,47-YS):SET(X,47-Y):XS=X:YS=Y:NEXT
330 ON ERROR GOTO 400
340 FOR T=0 TO 127 STEP .5
350 X=V0*COS(A)*T/4
360 Y=(V0*SIN(A)*T-9.81*T*T)/8
370 SET(X,47-Y):NEXT
380 IF INKEY$<>" " GOTO 380
390 GOTO 70
400 IF INKEY$<>" " GOTO 400
410 RESUME 70
420 RESUME 330
```

```
10 REM FIGURY LISSAJOUS
20 REM AUTOR JAN GOLLA
25 CLS
30 FOR X=1 TO 7:PRINT:NEXT
40 PRINT CHR$(23)
50 PRINT TAB(8)"FIGURY LISSAJOUS"
60 PRINT@896,TAB(8)"NACISNIJ SPACJE"
70 IF INKEY$<>" " GOTO 70
80 CLS
90 INPUT"PODAJ STOSUNEK CZESTOTLIWOSCI
DRGAN K";K
100 INPUT"PODAJ PRZESUNIECIE FAZOWE
DRGAN F(W RADIANACH)";F
110 PRINT@774,"JESLI CHCESZ ROZPOCZAC
      RYSOWANIE TO:"
120 PRINT TAB(20)"NACISNIJ SPACJE"
130 IF INKEY$<>" " GOTO 130
140 CLS
150 PRINT "K=";K:PRINT"F=";F
160 FOR T=1 TO 600
170 X=63+38*SIN(.05*T)
180 Y=23+21*SIN(K*.05*T+F)
190 RESET(XS,47):RESET(18,YS)
200 SET(18,Y):SET(X,47):SET(X,Y)
210 XS=Y:YS=Y
220 IF INKEY$=" " GOTO 80
230 NEXT
240 IF INKEY$<>" " GOTO 240
250 GOTO 80
```

```
10 REM PROGRAM W TRYBIE KONWERSACYJNYM DO TWORZENIA
20 REM TESTU EGZAMINACYJNEGO
30 REM AUTOR   JAN GOLLA
35 REM PRZYGOTOWANIE PYTAN EGZAMINACYJNYCH
40 CLEAR 10000
50 CLS
60 INPUT"Z JAKIEGO MATERIALU JEST TEN TEST";Z$
70 INPUT"PODAJ ILOSC PYTAN";N
80 INPUT"PODAJ ILOSC ODPOWIEDZI";M
85 IF M<2 GOTO 80
90 INPUT"ZA ILE DOBRYCH ODPOWIEDZI BEDZIE OCENA DOSTATECZNA";A
91 INPUT"ZA ILE DOBRYCH ODPOWIEDZI BEDZIE OCENA DOBRA";B
92 INPUT"ZA ILE DOBRYCH ODPOWIEDZI BEDZIE OCENA BARDZO DOBRA";C
120 CLS:DIM P$(N),D$(N,M),P(N)
130 FOR X=1 TO N
140 PRINT@66,"PODAJ";X;"PYTANIE":INPUT P$(X):P(X)=M
150 FOR Y=1 TO M-1
160 PRINT@66,"PODAJ ZLA ODPOWIEDZ":INPUT D$(X,Y)
170 CLS:NEXT
180 PRINT@66,"PODAJ DOBRA ODPOWIEDZ":INPUT D$(X,Y)
190 CLS:NEXT
200 REM URUCHOMIENIE PRZYGOTOWANEGO PROGRAMU
210 CLS:T=0
220 PRINT@460,CHR$(23)"TEST Z ";Z$
230 PRINT@910,"W CELU KONTYNUACJI"
240 PRINT@976,"NACISNIJ KLAWISZ SPACJA"
250 IF INKEY$<>" " GOTO 250
260 CLS:FOR L=1 TO N
270 FOR K=1 TO 5
280 RANDOM:W1=RND(M):W2=RND(M)
290 IF W1=P(L) THEN P(L)=W2:GOTO 310
300 IF W2=P(L) THEN P(L)=W1
310 B$=D$(L,W1):D$(L,W1)=D$(L,W2):D$(L,W2)=B$
320 NEXT:CLS
330 PRINT@18,CHR$(23)"PYTANIE NR";L
340 PRINT:PRINT P$(L):PRINT
350 FOR K=1 TO M
360 PRINT K;D$(L,K):NEXT:PRINT
370 INPUT"PODAJ NUMER POPRAWNEJ ODPOWIEDZI";X
380 IF X<>P(L) GOTO 400
390 PRINT"ODPOWIEDZIALES DOBRZE":T=T+1:GOTO 410
400 PRINT"ZLE!!ODPOWIEDZ DOBRA MA NR";P(L)
410 IF INKEY$<>" " GOTO 410 ELSE NEXT
420 IF T<A THEN O$="NIEDOSTATECZNA"
430 IF T>=A AND T<B THEN O$="DOSTATECZNA"
440 IF T>=B AND T<C THEN O$="DOBRA"
450 IF T>=C THEN O$="BARDZO DOBRA"
460 CLS:PRINT CHR$(23)
470 PRINT@264,"ODPOWIEDZIALES DOBRZE NA"
480 PRINT@330,T;"SPOSROD";N;"PYTAN"
490 PRINT:PRINT"OTRZYMUJESZ OCENE ";O$
500 IF INKEY$<>" " GOTO 500 ELSE 200
```

JAN GOLLA

AMSTRAD
PROGRAM 16

WIEŻA HANOI

Mamy trzy pręty i *n* krążków o różnych średnicach. Krążki mogą być nakładane na pręty: tworząc w ten sposób wieże. Załóżmy, że *n* krążków zostało umieszczonych początkowo na pręcie A w kolejności malejących średnic (na rysunku zilustrowano tę sytuację dla *n = 3*).

Naszym zadaniem jest przesunięcie *n* krążków z prętu A na pręt C z zachowaniem pierwotnego porządku. Zasady przesuwania są następujące:

(1) w każdym kroku przesuwa się dokładnie jeden krążek z jednego pręta na inny,

(2) krążek nie może być nigdy nałożony na krążek o mniejszej średnicy,

(3) pręt B może być użyty jako „magazyn pomocniczy".

Dane do programu:

— ilość krążków (1—5) maksymalną ilość krążków można zwiększyć zmieniając w programie wartość stałej MAXILKR,

— żądanie wyświetlenia stanu wież po przełożeniu krążka (1 — tak, 0 — nie).

Kontynuacja wyświetlania wyników (po wyświetleniu 16 wierszy), wciśnięcie dowolnego klawisza. Program napisany jest w Pascalu.

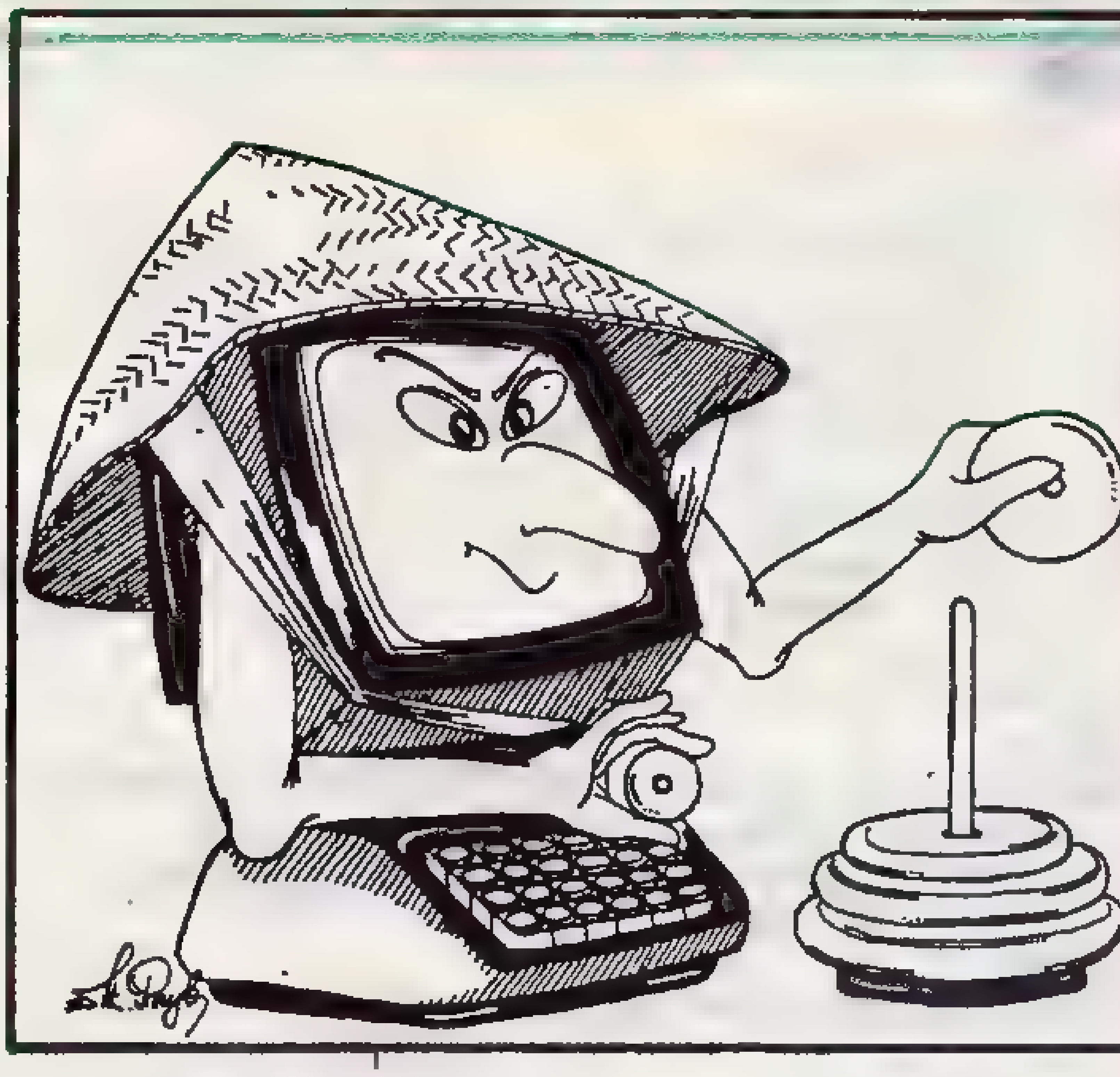
S. R.

N. Wirth: „Algorytmy + struktury danych = programy", WNT, Warszawa 1980, s. 172, ćwiczenie 3.1.

```
PROGRAM wiezahanoi;
 {$a-}
  CONST
    maxilkr=5; {maksymalna ilosc krazkow}
    maxilw=16; {maksymalna ilosc wyswietlanych
               wierszy}
  TYPE
    znakp='a'..'c'; {nazwa preta}
    wyswiez=1..maxilkr; {maksymalna wysokosc wiezy
                        mierzona iloscia krazkow}
    ilw=1..maxilw; {ilosc wyswietlonych wierszy}
    opiswiez=ARRAY [znakp] OF  {opis wiez}
        RECORD
          w:wyswiez; {wskaznik wierzcholka wiezy}
          wieza:ARRAY [wyswiez] OF wyswiez
        END;
  VAR
    ilkr:integer; {zadana ilosc krazkow}
    start,  {pret, z ktorego przesuwamy krazki}
    cel,    {pret, na ktory przesuwamy krazki }
    posr:znakp; {pret - magazyn pomocniczy   }
    opw:opiswiez;
    druk:integer;
    ilwierszy:ilw;
  PROCEDURE ustawstpocz;
    {procedura ustawiajaca poczatkowy stan wiez}
   VAR
     s:znakp; {zmienne robocze}
     i:wyswiez;
    BEGIN
     WITH opw['a'] DO
      BEGIN
      w:=maxilkr-ilkr+1;
      FOR i:=w TO maxilkr DO
       wieza[i]:=i-w+1
     END;
    opw['b'].w:=0;
    opw['c'].w:=0
   END;
PROCEDURE czyscekran;
   {procadura kasujaca na zadanie (wcisniecie
    dowolnego klawisza) poprzedni obraz po
    wyswietleniu maksymalnej ilosci wierszy}
  VAR
    x:char; {zmienna robocza}
  BEGIN
    IF ilwierszy>=maxilw THEN
     BEGIN
      REPEAT

    UNTIL keypressed;
    READ(kbd,x);
    clrscr;
    ilwierszy:=0
   END
END;

PROCEDURE drukopw;
   {procedura drukujaca numery krazkow
    znajdujacych sie na poszczegolnych pretach}
  VAR
    s:znakp;  {zmienne robocze}
    i:wyswiez;
  BEGIN
    FOR s:='a' TO 'c' DO
     WITH opw[s] DO
      BEGIN
       WRITE(' ':10,'wieza ',s,' :');
       IF w>0 THEN
        FOR i:=w TO maxilkr DO
         WRITE(' ':2,wieza[i]:2);
       WRITELN
      END;
    ilwierszy:=ilwierszy+3
  END;
PROCEDURE przen(start,cel:znakp);
  {procedura przenoszaca pojedynczy krazek
   z preta "start" na pret "cel"}
  BEGIN
   czyscekran;
   WRITELN('przenies krazek z preta ',
            start,' na pret ',cel);
   ilwierszy:=ilwierszy+1;
```



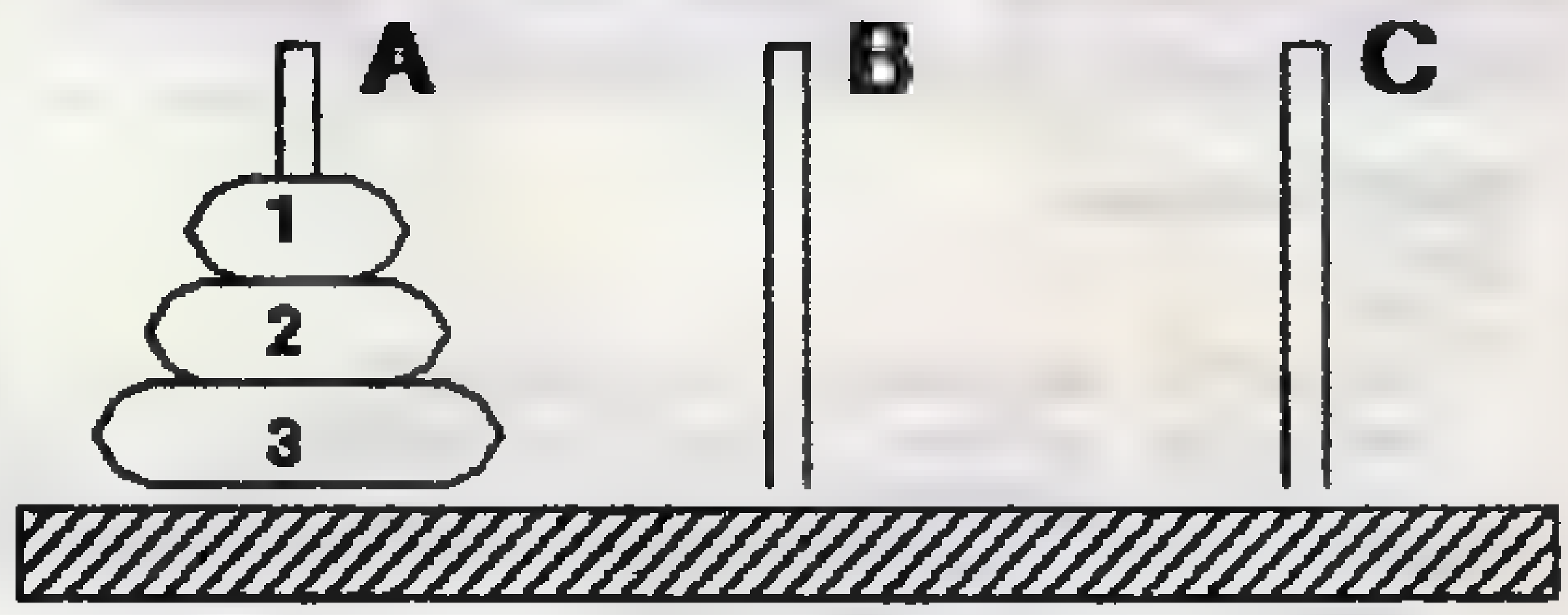
```
  IF druk=1 THEN
   BEGIN
    WITH opw[cel] DO
     BEGIN
      IF w=0 THEN  w:=maxilkr
      ELSE  w:=w-1;
      wieza[w]:=opw[start].wieza[opw[start].w]
     END;
    WITH opw[start] DO
     IF w=maxilkr THEN  w:=0
     ELSE  w:=w+1;
    drukopw
   END
 END;
PROCEDURE przel(n:wyswiez; VAR start,posr,cel:znakp);
  {procedura przesuwajaca "n" krazkow z preta
   "start" na pret "cel" z wykorzystaniem preta
   "posr" jako magazynu}
 BEGIN
  IF n=1 THEN przen(start,cel)

    ELSE
     BEGIN
      przel(n-1,start,cel,posr);
      przen(start,cel);
      przel(n-1,posr,start,cel)
     END
  END;


BEGIN
  REPEAT

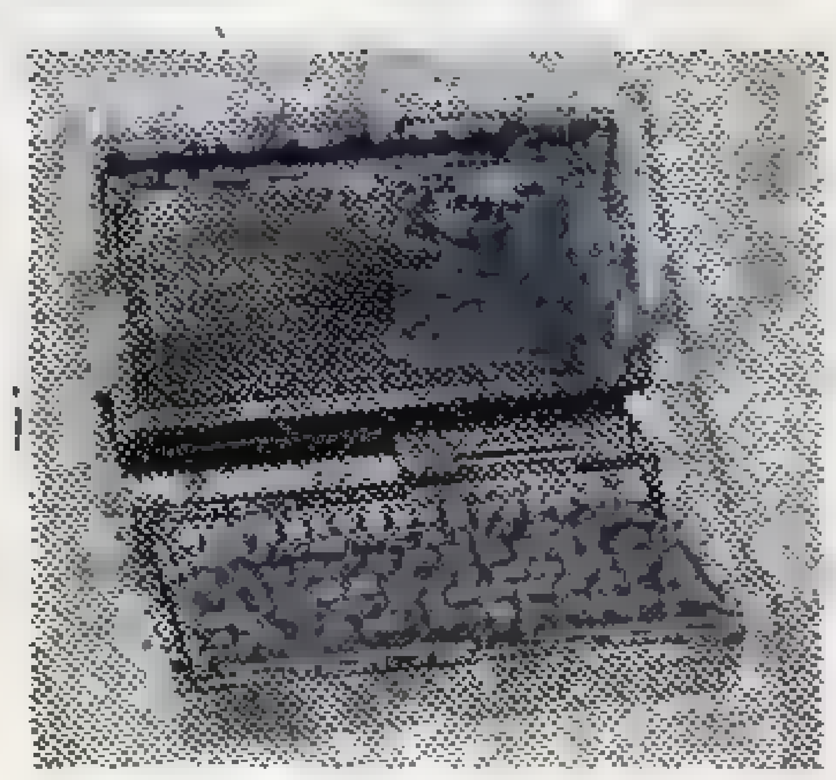
  WRITELN('podaj ilosc krazkow <= ',maxilkr);
  READLN(ilkr);
  WRITELN('czy chcesz wyswietlac stanywiez po');
  WRITELN(' przelozeniu kazdego krazka? 1-tak,0-nie');
  READLN(druk)
 UNTIL (ilkr>0) AND (ilkr<=maxilkr);
 clrscr;
 ilwierszy:=0;
 IF druk=1 THEN
  BEGIN
   ustawstpocz;
   WRITELN('stan poczatkowy :');
   ilwierszy:=ilwierszy+1;
   drukopw
  END;
  start:='a';
  cel:='c';
  posr:='b';
  przel(ilkr,start,posr,cel)
END.
```

```
stan poczatkowy :
          wieza a :   1   2   3
          wieza b :
          wieza c :
przenies krazek z preta a na pret c
          wieza a :   2   3
          wieza b :
          wieza c :   1
przenies krazek z preta a na pret b
          wieza a :   3
          wieza b :   2
          wieza c :   1
przenies krazek z preta c na pret b
          wieza a :   3
          wieza b :   1   2
          wieza c :
przenies krazek z preta a na pret c
          wieza a :
          wieza b :   1   2
          wieza c :   3
przenies krazek z preta b na pret a
          wieza a :   1
          wieza b :   2
          wieza c :   3
przenies krazek z preta b na pret c
          wieza a :   1
          wieza b :
          wieza c :   2   3
przenies krazek z preta a na pret c
          wieza a :
          wieza b :
          wieza c :   1   2   3
```



STEFAN ROZMUS

16. **Pro Lite** (Texas Instruments)
Mikroprocesor: 80C88, 4.77MHz
System operacyjny: MS-DOS 2.12, CP/M 86
Pamięć RAM: 256 KB lub 768 KB
Pamięć masowa: Floppy-Disc 720 KB
Grafika: 640x200
Zasilanie: akumulator
Rozmiary: 29x7.5x33 cm
Waga: 4.8 kg
Cena: 11 300 M



ATARI

PROGRAM 17

MINIASEMBLER

Prezentowany program nazwany MINIASEMBLER realizuje dwie zasadnicze funkcje: asemblacji i disasemblacji. Proces asemblacji polega na tłumaczeniu programu napisanego za pomocą nazw mnemonicznych rozkazów mikroprocesora 6502 na kody cyfrowe tych rozkazów, które mogą być w takiej postaci wykonywane przez mikroprocesor. Natomiast disasemblacja jest procesem odwrotnym. Należy zaznaczyć, że program MINIASEMBLER nie może i nie powinien zastępować prawdziwego (firmowego) Asemblera i Disasemblera oraz nie jest narzędziem przeznaczonym do tworzenia dużych programów w języku maszynowym. Uwzględniając jednak fakt, że przeciętny użytkownik mikrokomputera ma duże trudności ze „zdobyciem" firmowego asemblera, program ten jest szczególnie przydatny dla początkujących programistów w języku maszynowym. Program posiada jeszcze inne funkcje, które zostały dalej dokładnie opisane. Możliwe obszary zastosowań tego programu rozciągają się od pisania i testowania małych programów w języku maszynowym, poprzez „sklejania" programów w BASIC-u z procedurami w języku maszynowym aż do modyfikacji posiadanego oprogramowania (np. bardzo łatwo za pomocą tego programu „spolszczyć" firmowe programy, zmieniać zakodowane w nich stałe itp.

Po wprowadzeniu programu do pamięci należy go uruchomić instrukcją RUN. Program wyświetli MENU, które informuje o następujących opcjach programu:

M — drukuj pamięć (heksadecymalnie).

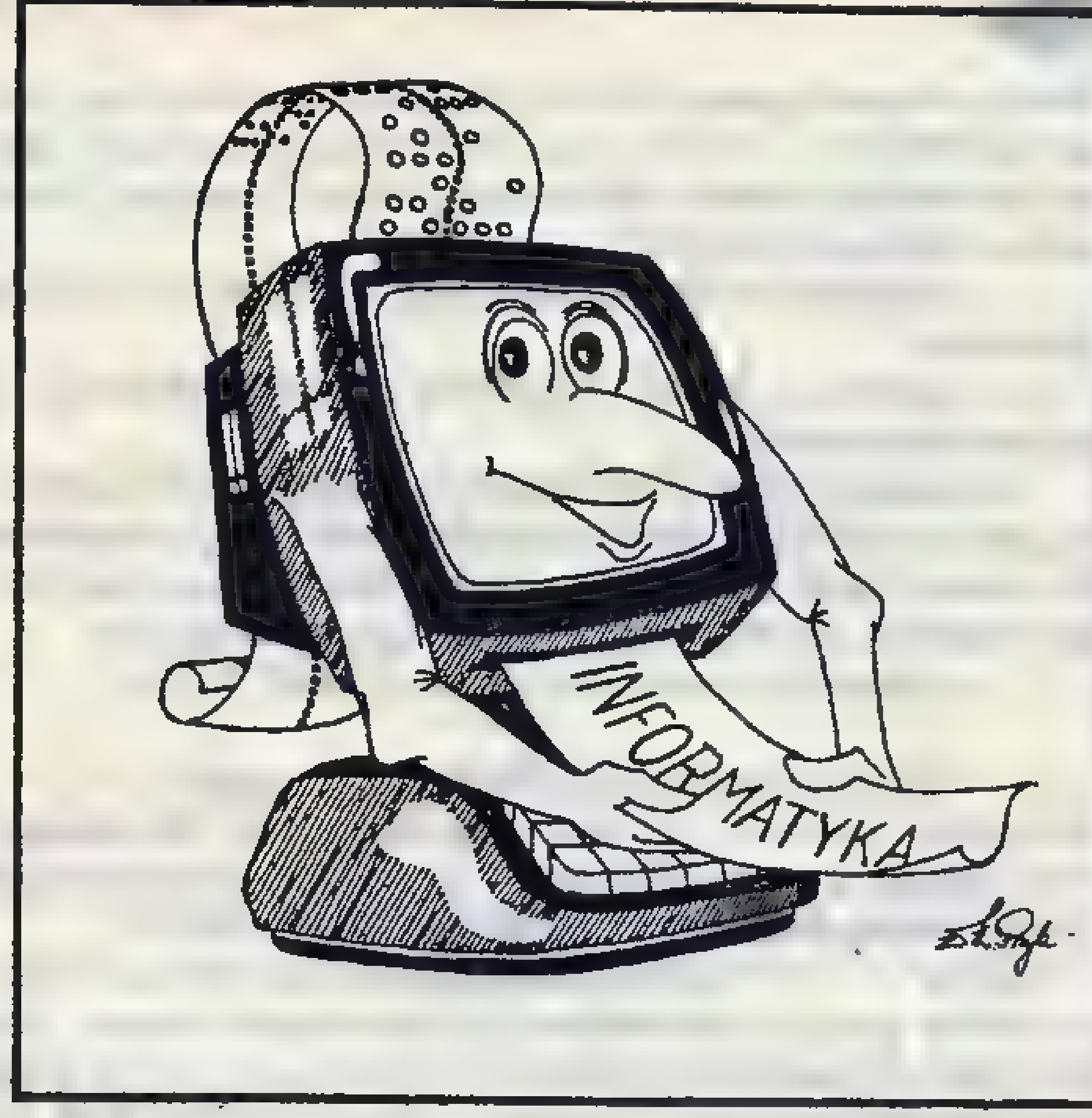
Począwszy od określonego przez użytkownika adresu z zakresu \$0000—\$FFFF wyświetlana jest wierszami zawartość pamięci. W jednym wierszu wyprowadzana jest zawartość 8 bajtów. Każdy wiersz zawiera: adres (hexadecymalnie) pierwszego z 8 bajtów, 8 liczb heksadecymalnych będących zawartością każego z 8 bajtów i 8 odpowiadających tym zawartościom znaków ATASCII ujętych w (")

B — drukuj pamięć (binarnie)

każdy bajt pamięci jest wyprowadzany w postaci liczby heksadecymalnej i odpowiadającej jej liczby binarnej.

D — wykonaj disasemblację określonego obszaru pamięci.

Wykonywana jest disasemblacja obszaru RAM, którego adres początkowy określa użytkownik. Dla każdego rozkazu podawany jest jego adres (heksadecymalnie), nazwa mnemoniczna rozkazu mikroprocesora, zawartość operandu oraz używane rejestry. Dla każdego rozkazu będącego skokiem względnym podawany jest adres bezwzględny pamięci, do którego przekazywane jest sterowanie programu przy wykonaniu tego skoku. Dla bajtów, których zawartość nie może być zamieniona na nazwę mnemoniczną rozkazu podawany jest znak ATASCII ujęty w apostrofy, natomiast w miejsce nazwy mnemonicznej wyprowadzane są znaki: ???.



P — wpisz do pamięci zawartość heksadecymalną; wyświetlony jest adres bajtu pamięci i jego zawartości w postaci dwuznakowej liczby heksadecymalnej. Następnie użytkownik ma możliwość zmiany tej zawartości poprzez podanie dwuznakowej wartości heksadecymalnej, która zostanie wpisana w miejsce wyświetlonej. Starą wartość można pozostawić wciskając tylko klawisz RETURN. Wówczas program „pokaże" zawartość następnego bajtu pamięci.

T — wprowadź tekst do pamięci.

Pod wskazany adres pamięci wprowadzony jest tekst, który użytkownik wpisuje z klawiatury. Koniec tekstu sygnalizuje wciśnięcie klawisza RETURN.

L — Ładuj do pamięci (Load).

Program podaje wielkość wolnego obszaru RAM w postaci dwóch adresów — dolnego i górnego. Następnie użytkownik ma możliwość wprowadzenia w wolny obszar RAM informacji z dowolnego urządzenia podając specyfikacje zbioru np.: D: nazwa zbioru. ext, C lub E. Za pomocą tej opcji można wprowadzić do pamięci komputera zbiory danych, programy w BASIC-u, jak i w kodzie maszynowym. Po zakończeniu ładowania program wyprowadza (w postaci heksadecymalnej) liczbę załadowanych bajtów.

S — wyprowadź z pamięci (SAVE). Użytkownik wprowadza do komputera adres pocz. obszaru RAM, którego zawartość będzie wyprowadzona na określone urządzenie oraz liczbę bajtów do wyprowadzenia (heksadecymalnie). Możliwości jak w opcji L.

A — uruchamia miniasembler.

Użytkownik określa adres RAM, do którego mają być wprowadzone rozkazy w kodzie maszynowym, powstałe w wyniku asemblacji. Asemblacja dokonywana jest wierszami (liniami) tzn. użytkownik wprowadza instrukcję programu w języku asemblera, która jest natychmiast tłumaczona na kod maszynowy, który następnie wraz z operandem jest wyświetlony na monitorze. Składnia tworzonych instrukcji jest analogiczna do obowiązującej w opracowanych dla ATARI asemblerach jak ASEMBLER EDITOR lub MAC65. Ponieważ asemblacja dokonywana jest na pojedynczych wierszach programu. Używanie adresów symbolicznych (etykiet) jest niestety niemożliwe. Po zakończeniu

wprowadzania programu (END) może on być od razu uruchomiony, a efekt jego działania sprawdzony poprzez wydrukowanie obszarów (lokacji) pamięci przez niego wykorzystanych.

> — zamiana liczby dziesiętnej na postać heksadecymalną.

< — zamiana liczby w postaci heksadecymalnej na postać dziesiętną.

przełącza wyjście programu z monitora na drukarkę lub odwrotnie. Jeżeli drukarka nie jest podłączona do komputera automatycznie wybierane jest wyjście na monitor, a użycie tej opcji będzie ignorowane.

} ! — przekazuje sterowanie do ATARI, BASIC

␣ — (spacja) wyświetla MENU — program.

Wszystkie adresy pamięci wprowadzone do programu muszą być podawane w postaci czteroznakowej liczby heksadecymalnej z zakresu $0000-$FFFF.

Opcje ">" i "<" umożliwiają szybkie przeliczanie adresów w postaci dziesiętnej na heksadecymalną i odwrotnie. Wykonywanie każdej funkcji (oprócz L, S) może być przerwane po wciśnięciu klawisza ESCAPE (w trakcie jej realizacji) lub RETURN przed rozpoczęciem realizacji.

Konstrukcja programu umożliwia łatwe rozbudowywanie go o dowolne fukcje potrzebne użytkownikowi. W tym celu w linie 830—990 należy wpisać instrukcję wywołującą podprogram opracowany przez użytkownika, a w linie 11000—29000 tenże podprogram. Przy konstruowaniu własnych podprogramów przydatne mogą być następujące podprogramy z prezentowanego programu. Oto one:

linia 100 — podprogram ADRES

— pobiera z klawiatury podany przez użytkownika czteroznakowy adres obszaru pamięci (w post. heksadecymalnej), dokonuje jego zamiany na liczbę dziesiętną, którą umieszcza w zmiennej AD,

linia 200 — podprogram HEXDEC

— zamienia czteroznakową liczbę heksadecymalną zawartą w zmiennej HEX na liczbę dziesiętną, którą umieszcza w zmiennej N,

linia 300 — podprogram DECHEX

— zamienia liczbę dziesiętną zawartą w zmiennej DEC na liczbę heksadecymalną, którą umieszcza w zmiennej HEX$. Długość łańcucha HEX$ (wartość 2 lub 4) znajduje się po wykonaniu podprogramu w zmiennej L,

linia 400 — podprogram POBZ

— pobiera z klawiatury znak, którego kod umieszcza w zmiennej A,

linia 500 — podprogram DECBIN

— zamienia liczbę dziesiętną zawartą w zmiennej DEC na ośmioznakową liczbę binarną, którą umieszcza w BINS,

linia 600 — podprogram BINDEC

— zamienia liczbę binarną zawartą w BIN$ na liczbę dziesiętną, którą umieszcza w DEC,

linia 700 — podprogram BREAK

— bada czy został wciśnięty klawisz ESCAPE i jeżeli tak, to przerywa realizowaną w chwili naciśnięcia funkcję.

Przy wprowadzaniu programu do pamięci komputera proponuję zwrócić szczególną uwagę na cztery problemy:

1. Niektóre linie programu są bardzo długie (taki sposób zapisywania zmniejsza pamięć zajmowaną przez program), aby mogły być przyjęte przez komputer należy przed rozpoczęciem wprowadzenia programu wykonać instrukcję POKE 82,0

2. Znak „strzałka\ w inwersie wideo" linia 840, 850... itp. uzyskuje się po wciśnięciu klawiszy ESC/CTRL-2.

3. Wartość zmiennej tekstowej ML$ bardzo istotna dla poprawnego działania opcji L i S w linii 30030 składa się z następujących sześciu znaków: {h, klawisz ATARI-SHIFT-2, CTRL-P, L, V, klawisz ATARI — małe d} stała ML$ zawiera trzyrozkazowy podprogram w kodzie maszynowym, który przekazuje sterowanie do procedury systemu operacyjnego obsługującej urządzenia zewnętrzne.

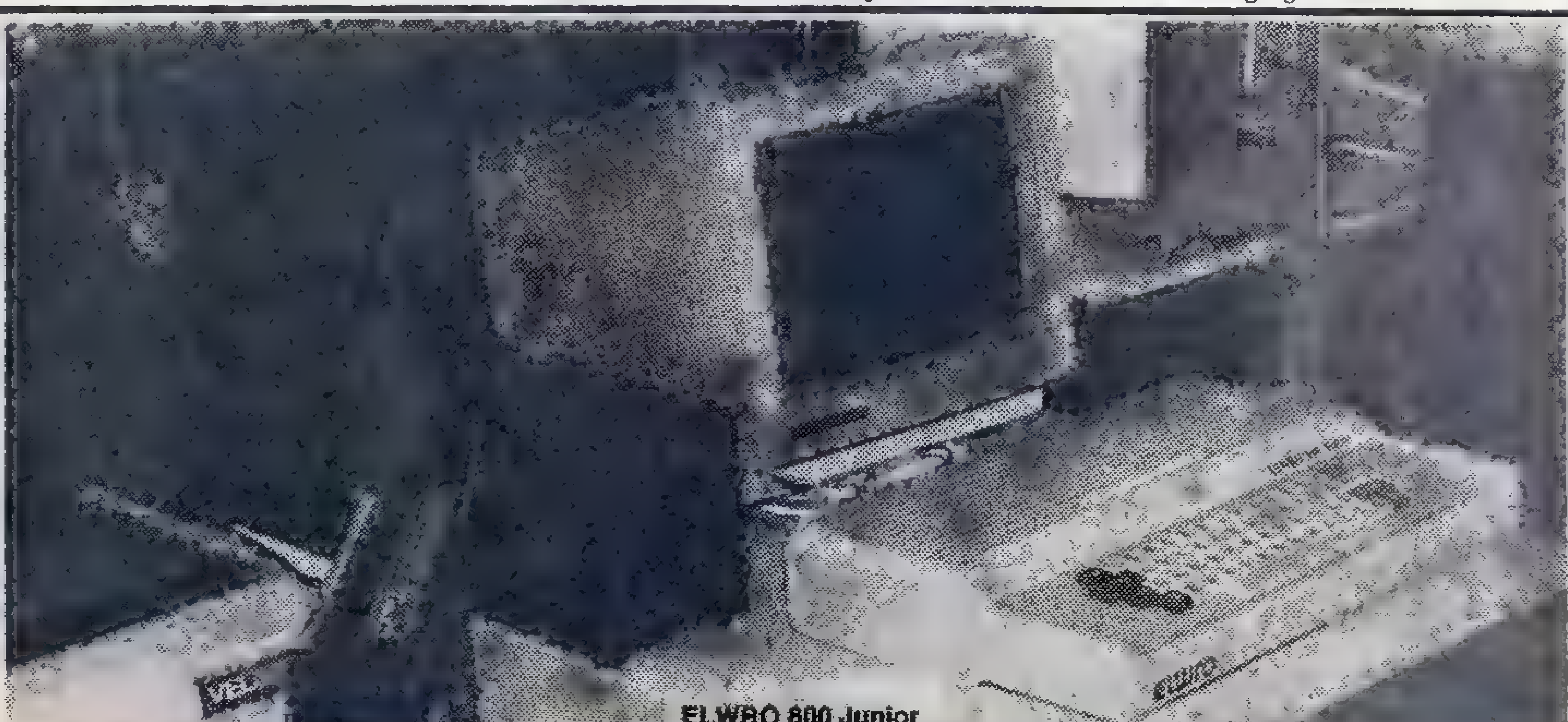
4. Problem komentarzy — proponuje nie wprowadzać linii komentarzy ponieważ stanowią one jedynie informacje dla czytelnika, a im większa ich ilość w programie, tym większa pracochłonność związana z wprowadzeniem programu do pamięci komputera i wolniejsze działania programu.

Program zajmuje ok. 10 KB pamięci RAM.

```
1 REM *************************
*******
2 REM PROGRAM ASEMBLER
3 REM *************************
*******
10 GOSUB 30000
20 GOTO 30150
95 REM *************************
*******
96 REM PODPROGRAM "POBIERZ ADRES
"
97 REM *************************
******
100 POKE KEY,STR-N1:? "Adres poc
z. obszaru";:INPUT HEX$:IF LEN(H
EX$)<>N4 THEN ? :? "Blad w danyc
h wej.":POP :GOTO
110 GOSUB HEXDEC:IF EF=N1 THEN G
OTO ADRES
120 AD=N:RETURN
195 REM *************************
*******
196 REM PODPROGRAM "ZAMIANA HEX
-> DEC"
197 REM *************************
*******
200 N=N0:EF=N0:FOR I=N1 TO LEN(H
EX$):NIB$=HEX$(I,I):IF NIB$<"0"
THEN EF=N1
210 IF NIB$<="9" THEN N=N*16+VAL
(NIB$):GOTO 240
220 IF NIB$<"A" OR NIB$>"F" THEN
 EF=N1
230 N=N*16+ASC(NIB$)-55
240 NEXT I:RETURN
295 REM *************************
*******
296 REM PODPROGRAM "ZAMIANA DEC
-> HEX"
297 REM *************************
*******
300 HEX$=""
```

```
310 TEMP=DEC:DEC=INT(DEC/16):TEM
P=TEMP-DEC*16:IF TEMP<N10 THEN H
EX$(L,L)=STR$(TEMP):GOTO 330
320 HEX$(L,L)=CHR$(TEMP+55)
330 IF L<>N1 THEN L=L-N1:GOTO 31
0
340 RETURN
395 REM ***********************
*******
396 REM PODPROGRAM "POBIERZ ZNAK
"
397 REM ***********************
*******
400 OPEN #N5,N4,N0,"K:":GET #N5,
A:CLOSE #N5:RETURN
495 REM ***********************
*******
496 REM PODPROGRAM "ZAMIANA DEC
-> BIN"
497 REM ***********************
*******
500 BIN$="00000000":NIB$="1":N=S
TR/N2:FOR I=N1 TO N8:IF DEC>N-1
THEN BIN$(I,I)=NIB$:DEC=DEC-N
510 N=N/N2:NEXT I:RETURN
595 REM ***********************
*******
596 REM PODPROGRAM "ZAMIANA BIN
-> DEC"
597 REM ***********************
*******
600 DEC=N0:L=STR/N2:FOR J=N1 TO
N8:DEC=DEC+L*VAL(BIN$(J,J)):L=L/
N2:NEXT J:RETURN
695 REM ***********************
*******
696 REM PODPROGRAM "BADANIE KL.
ESCAPE"
697 REM ***********************
*******
700 IF PEEK(KEY)=28 THEN ? :? :?
 BREAK$:POP :GOTO START
710 RETURN
795 REM ***********************
*******
796 REM BADANIE WYBRANEJ OPCJI
797 REM ***********************
*******
800 ? :? ">";
820 GOSUB POBZ
821 IF CHR$(A)=" " THEN GOTO 301
80
830 IF CHR$(A)="M" THEN ? "Druko
wanie pamieci":GOTO 1000
840 IF CHR$(A)="B" THEN ? CHR$(2
53);"Drukowanie pamieci/binarnie
/":GOTO 2000
850 IF CHR$(A)="!" THEN ? "!";CH
R$(253):END
860 IF CHR$(A)="D" THEN ? CHR$(2
53);"Disasemblacja":GOTO 3000
870 IF CHR$(A)="P" THEN ? CHR$(2
53);"Pisz do pamieci":GOTO 4000
680 IF CHR$(A)="T" THEN ? CHR$(2
53);"Pisz tekst do pamieci":GOTO
 5000
890 IF CHR$(A)="A" THEN ? CHR$(2
53);"Asemblacja":GOTO 6000
900 IF CHR$(A)="L" THEN ? CHR$(2
53);"Laduj do pamieci":GOTO 7000
910 IF CHR$(A)="S" THEN ? CHR$(2
53);"Wyprowadz z pamieci":GOTO 8
000
920 IF CHR$(A)="#" THEN ? CHR$(2
53);"#":GOTO 9000
930 IF CHR$(A)=">" THEN ? CHR$(2
53);"Zamiana dec -> hex":GOTO 10
000
940 IF CHR$(A)="<" THEN ? CHR$(2
53);"Zamiana hex -> dec":GOTO 11
000
950 IF CHR$(A)=CHR$(125) THEN 30
160
990 GOTO 820
395 REM ***********************
*******
996 REM PODPROGRAM
997 REM DRUKUJE ZAWARTOSC PAMIEC
I
998 REM ***********************
*******
1000 GOSUB ADRES
1010 ? #V:? #V;HEX$;", ";:FOR I=A
D TO AD+N7:DEC=PEEK(I):L=N2:GOSU
B DECHEX:? #V;HEX$;" ";:NEXT I:?
 #V;":";
1020 FOR I=AD TO AD+N7:DEC=PEEK(
I):IF V=N4 THEN 1040
1030 ? #V;CHR$(27);:IF DEC=EOL T
HEN DEC=46
1035 GOTO 1050
1040 DEC=DEC-(DEC>127)*128:IF DE
C<32 THEN DEC=46
1050 ? #V;CHR$(DEC);
1060 NEXT I:? #V;":";:AD=AD+N8:D
EC=AD:L=N4:GOSUB DECHEX:GOSUB BR
EAK
1070 GOTO 1010
1995 REM ***********************
*******
1996 REM PODPROGRAM
1997 REM DRUKUJE ZAWARTOSC PAMIE
CI
1998 REM W POSTACI BINARNEJ
1999 REM ***********************
*******
2000 GOSUB ADRES
2010 ? #V:? #V;HEX$;"  ";:TRAP S
TART:DEC=PEEK(AD):L=N2:GOSUB DEC
HEX:DEC=PEEK(AD):GOSUB DECBIN
2020 ? #V;HEX$;" ";BIN$;:AD=AD+N
1:DEC=AD:L=N4:GOSUB DECHEX:GOSUB
 BREAK
2030 GOTO 2010
2995 REM ***********************
*******
2396 REM PODPROGRAM DISASEMBLACJ
I
2997 REM ***********************
*******
3000 GOSUB ADRES:TRAP N0:D1=-STR
3010 DEC=AD:L=N4:GOSUB DECHEX:?
#V;HEX$;"  ";:DEC=PEEK(AD):D0=DE
C:L=N2:GOSUB DECHEX
3011 ? #V;HEX$;" ";:C=A(D0):M=IN
T(C/1000)
3070 IF M>N0 THEN 3090
3080 ? #V;"'";:IF V=4 THEN 3084
3082 ? #V;CHR$(27);:IF D0=EOL TH
EN D0=46
3083 GOTO 3085
3084 D0=D0-(D0>127)*128:IF D0<32
 THEN D0=46
3085 ? #V;CHR$(D0);"'         ???":
AD=AD+N1:GOTO 3230
3090 B=INT(C/100)-M*N10:P=INT(C/
N10)-M*100-B*N10:O=C-M*1000-B*10
0-P*N10:IF B=N1 THEN ? #V;"
  ";:GOTO 3150
3110 D1=PEEK(AD+N1):DEC=D1:L=N2:
GOSUB DECHEX:? #V;HEX$;" ";:IF B
=N2 THEN ? #V;"   ";:GOTO 3150
3130 D2=PEEK(AD+N2):DEC=N2:L=N2:
GOSUB DECHEX:? #V;HEX$;" ";:D1=D
1+D2*STR
3150 IF M<N4 OR M>13 OR M=11 OR
M=N7 THEN 3170
3160 IF D1>127 THEN D1=-(STR-D1)
3165 D1=AD+D1+N2
3170 ? #V;"   ";N$(M*N3-N2,M*N3)
." ";L$(P*N3+N1,P*N3+N3);
3180 DEC=D1:L=N2:IF DEC>STR-N1 T
HEN L=N4
3190 GOSUB DECHEX:IF D1=-STR THE
N ? #V:GOTO 3205
3200 ? #V;"$";HEX$;T$(O*N3+N1,O*
N3+N3)
```

Podstawowe wiadomości o programowaniu w języku maszynowym można znaleźć w książce pod red. W. M. Miguta „ATARI BASIC KAW". Kraków 1985 oraz w numerach 2, 3, 5 miesięcznika „KOMPUTER" z ubiegłego roku



ELWRO 800 Junior

```
3205 D1=-STR:AD=AD+B
3230 GOSUB BREAK:GOTO 3010
3995 REM *******************************
3996 REM PODPROGRAM WPISUJE DO PAMIECI
3997 REM LICZBY HEKSADECYMALNE
3998 REM *******************************
4000 GOSUB ADRES
4010 TRAP N0:DEC=AD:L=N4:GOSUB DECHEX:? HEX$;"  ";:DEC=PEEK(AD):L=N2:GOSUB DECHEX:? HEX$;" ";:K=N0
4030 GOSUB POBZ
4047 IF A=ESC THEN ? :? :? BREAK$:GOTO START
4050 IF A=EOL THEN GOSUB HEXDEC:POKE AD,N:AD=AD+N1:? :GOTO 4010
4060 IF A>126 THEN IF K>0 THEN ? CHR$(A);:K=K-1
4070 IF (A>47 AND A<58) OR (A>64 AND A<71) THEN IF K<N2 THEN ? CHR$(A);:K=K+N1:HEX$(K,K)=CHR$(A)
4080 GOTO 4030
4995 REM *******************************
4996 REM PODPROGRAM WPISUJE DO PAMIECI
4997 REM TEXT
4998 REM *******************************
5000 GOSUB ADRES
5010 TRAP N0:DEC=AD:L=N4:GOSUB DECHEX:? :? HEX$;"  ";:K=N0
5030 GOSUB POBZ
5040 IF A=EOL THEN POKE AD,EOL:? :GOTO START
5050 IF A=ESC THEN ? :? :? BREAK$:GOTO START
5060 IF A=126 THEN AD=AD-N1:GOTO 5010
5070 ? CHR$(27);CHR$(A);:POKE AD,A:AD=AD+1:K=K+1:IF K>15 THEN 5010
5080 GOTO 5030
5995 REM *******************************
5996 REM PODPROGRAM ASEMBLACJI
5997 REM *******************************
6000 GOSUB ADRES:TRAP N0:?
6005 I=N1:DEC=N0:HEX$="":EDIT$=HEX$:OP$=HEX$:L=N1
6010 GET #N2,A:EDIT$(I,I)=CHR$(A):IF A<>EOL THEN I=I+N1:GOTO 6010
6020 B=LEN(EDIT$):A=N0:IF B<N3 THEN 6230
6030 FOR I=N1 TO B:IF EDIT$(I,I)<>"" THEN A=I:I=B
6040 NEXT I:EDIT$=EDIT$(A):OP$(N1,N3)=EDIT$(A,A+N2):C=N0:IF OP$="END" THEN GOTO START
6050 FOR I=N1 TO 168 STEP N3:IF N$(I,I+N2)=OP$ THEN C=(I+N2)/N3:I=STR
6060 NEXT I:IF C=N0 THEN 6230
6070 TRAP 6210:X=N0:Y=N0:D=N0:H=N0:Q=N0:FOR I=A+N3 TO B:IF EDIT$(I,I)<>" " THEN D=I:I=B
6080 NEXT I:OP$(N1,B-D)=EDIT$(D,B):D=N1
6090 IF OP$(D,D)="A" THEN X=N3:D=D+N1:L=N1:GOTO 6210
6100 IF OP$(D,D)="#" THEN X=N1:D=D+N1
6105 IF OP$(D,D)="(" THEN X=N2:D=D+N1
6110 IF OP$(D,D)="$" THEN H=N1:D=D+N1
6120 IF OP$(D,D)="%" THEN Q=N1:D=D+N1
6130 OP$=OP$(D):D=LEN(OP$):IF D<N3 THEN 6160
6140 IF OP$(D-N2,D)=",X)" THEN Y=N1:D=D-N3:GOTO 6160
6145 IF OP$(D-N2,D)="),Y" THEN Y=N2:D=D-N3:GOTO 6160
6150 IF OP$(D-N1,D)=",X" THEN Y=N3:D=D-N2:GOTO 6160
6155 IF OP$(D-N1,D)=",Y" THEN Y=N4:D=D-N2
6160 IF OP$(D,D)=")" THEN Y=N5:D=D-N1
6170 OP$=OP$(N1,D)
6180 IF H THEN HEX$=OP$:GOSUB HEXDEC:DEC=N:GOTO 6200
6190 IF Q THEN BIN$=OP$:GOSUB BINDEC:GOTO 6200
6195 DEC=VAL(OP$)
6200 L=N2+((DEC>=STR) OR (LEN(HEX$)=N4))
6210 K=C*1000+L*100+X*N10+Y:FOR I=N0 TO STR-N1:IF A(I)=K THEN B=I:I=STR:NEXT I:GOTO 6300
6220 NEXT I
6230 ? CHR$(253);CHR$(28);"Syntax Error!";:FOR I=N0 TO EOL:NEXT I:? CHR$(156);:GOTO 6005
6300 POKE AD,B:IF L=N2 THEN POKE AD+N1,DEC
6310 IF L=3 THEN POKE AD+N1,DEC-INT(DEC/STR)*STR:POKE AD+N2,INT(DEC/STR)
6315 ? CHR$(28);CHR$(156);:J=L:DEC=AD:L=N4:GOSUB DECHEX:? HEX$;"  ";
6320 FOR I=N0 TO J-N1:DEC=PEEK(AD+I):L=N2:GOSUB DECHEX:? HEX$;" ";:NEXT I
6330 AD=AD+J:POKE 85,18:? EDIT$;:GOTO 6005
6995 REM *******************************
6996 REM PODPROGRAM WPROWADZA DO PAMIECI
6997 REM DANE Z URZADZENIA ZEW. (LOAD)
6998 REM *******************************
7000 ? :? "Wolna pamiec od.adr: $";:DEC=PEEK(144)+STR*PEEK(145):L=N4:GOSUB DECHEX:? HEX$;" - $";
7010 DEC=PEEK(741)+STR*PEEK(742):L=N4:GOSUB DECHEX:? HEX$
7020 ? "Adres obszaru do ladowania ? $";:INPUT HEX$:IF LEN(HEX$)<>4 THEN ? :? "Blad w danych wej.":GOTO START
7030 GOSUB HEXDEC:BH=INT(N/STR):BL=N-BH*STR
7040 ? "Podaj specyfikacje zbioru ";:INPUT SPEC$:TRAP START:GS=N0:IF SPEC$(1,1)="C" THEN GS=STR/N2
7050 OPEN #N1,N4,GS,SPEC$:POKE IOBASE+N2,N7:POKE IOBASE+N8,N0:POKE IOBASE+N9,PEEK(742)-PEEK(145)
7060 POKE IOBASE+N4,BL:POKE IOBASE+N5,BH:I=USR(ADR(ML$)):CLOSE #N1
7070 SL=PEEK(IOBASE+N8):SH=PEEK(IOBASE+N9):DEC=SL+STR*SH:L=4:GOSUB DECHEX:? "Zaladowano bajtow: $";HEX$:GOTO START
7995 REM *******************************
7996 REM PODPROGRAM WYPROWADZA Z PAMIECI
7997 REM DANE NA URZADZENIE ZEW. (SAVE)
7998 REM *******************************
8000 ? "Adres - skad wyprowadzac ? ";:INPUT HEX$:IF LEN(HEX$)<>4 THEN ? :? "Blad w danych wej. ":GOTO START
8010 GOSUB HEXDEC:BH=INT(N/STR):BL=N-BH*STR
8020 ? "Liczba bajtow do wyprowadzenia ? ";:INPUT HEX$:IF LEN(HEX$)<>4 THEN ? :? "Blad w danych wej. ":GOTO START
8030 GOSUB HEXDEC:SH=INT(N/STR):SL=N-SH*STR
8040 ? "Podaj specyfikacje zbioru ";:INPUT SPEC$:TRAP START:GS=N0:IF SPEC$(1,1)="C" THEN GS=STR/N2
8050 OPEN #N1,N8,GS,SPEC$:POKE IOBASE+N2,N10+N1:POKE IOBASE+N8,SL:POKE IOBASE+N9,SH
8060 POKE IOBASE+N4,BL:POKE IOBASE+N5,BH:I=USR(ADR(ML$)):CLOSE #N1:GOTO START
8995 REM *******************************
8996 REM PODPROGRAM "WYJSCIE NA DRUKARKE
8997 REM *******************************
9000 IF V=N3 THEN V=N4:? "Wydruk na drukarke":GOTO START
9010 IF V=N4 THEN V=N3:? "Wyjscie na monitor":GOTO START
9995 REM Zamiana dec -> hex
10000 TRAP START:? "Liczba dziesietna ";:INPUT DEC:L=N4:GOSUB DECHEX:? "Heksadecymalna :";HEX$:GOTO START
10995 REM Zamiana hex -> dec
11000 ? "Liczba heksadecymalna ";:INPUT HEX$:GOSUB HEXDEC:IF EF=N1 THEN GOTO START
11010 ? "Dziesietna :";N:GOTO START
29995 REM *******************************
29996 REM PODPROGRAM INICJUJACY
29997 REM DZIALANIE PROGRAMU
29998 REM *******************************
30000 RESTORE 30000:POKE 559,0
30010 READ N0,N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N12,ESC,ADRES,EOL,HEXDEC,STR,DECHEX,POBZ,DECBIN,BINDEC,BREAK,KEY,STAR
30020 DATA 0,1,2,3,4,5,6,7,8,9,10,12,27,100,155,200,256,300,400,500,600,700,764,800
30030 DIM ML$(N6),SPEC$(N10+N5):ML$="h  LV":ML$(2,2)=CHR$(162):ML$(3,3)=CHR$(16):ML$(6,6)=CHR$(228)
30031 IOBASE=848:V=N3
30100 DIM N$(171),A(STR-N1),L$(N12),T$(N10+N8),HEX$(4),NIB$(N1),BIN$(N8),BREAK$(N10+N1),EDIT$(N10*N4),OP$(N12)
30110 N$(N1,99)="ADCANDASLBCCBCSBEQBITBMIBNEBPLBRKBVCBVSCLCCLDCLICLVCMPCPXCPYDECDEXDEYEORINCINXINYJMPJSRLDALDXLDYL"
30120 N$(100,171)="NOPORAPHAPHPPLAPLPROLRTIRTSSBCSECSEDSEISTASTXSTYTAXTAYTSXTXATXSTYAROR???"
30140 L$="   #  (   A  ":T$="   ,X)),Y,X ,Y )  ":L$(2,2)=CHR$(31):L$(3,3)=CHR$(30):L$(5,5)=CHR$(30)
30141 L$(6,6)=CHR$(31):L$(8,8)=CHR$(31):L$(9,9)=CHR$(30):T$(2,2)=CHR$(31):T$(3,3)=CHR$(30)
30142 BREAK$="** BREAK **":FOR I=N0 TO STR-N1:READ A:A(I)=A:NEXT I:RETURN
30150 OPEN #N2,N12,N0,"E:":OPEN #N3,N8,N0,"E:":TRAP 30160:OPEN #N4,N8,N0,"P:":GOTO 30170
30160 CLOSE #N4:OPEN #N4,N8,N0,"E:"
```

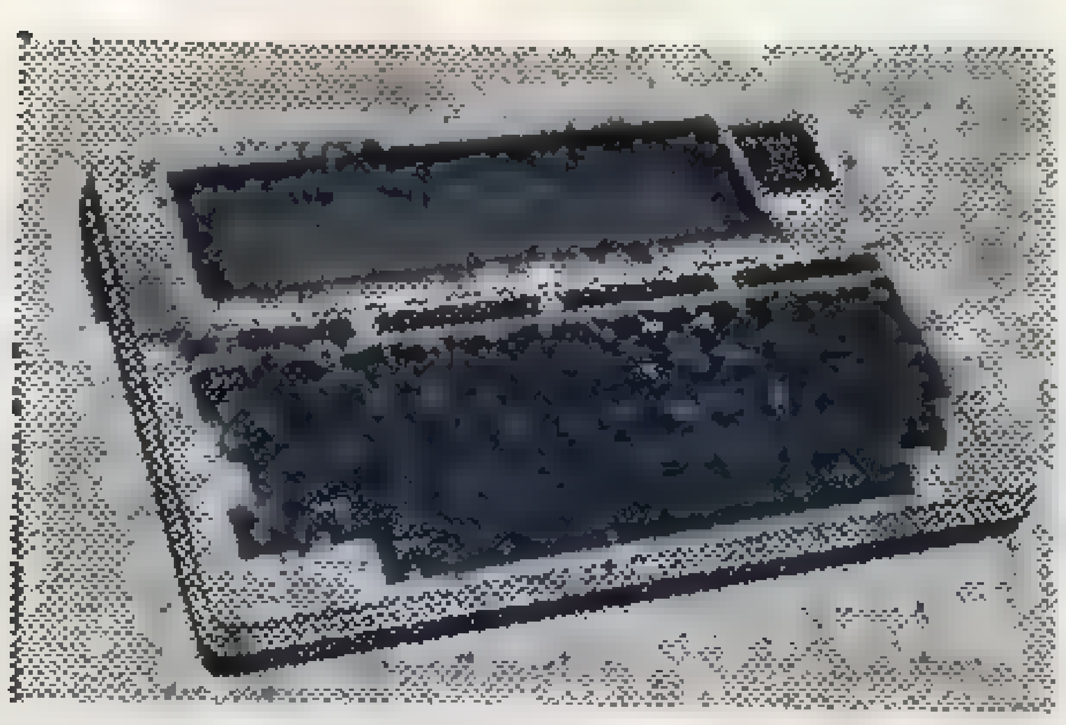
```
30170 GRAPHICS N0:SETCOLOR N1,N0
,N10:SETCOLOR N2,N12+N2,N2:SETCO
LOR N4,N12+N2,N2
30180 PRINT CHR$(125):PRINT :PRI
NT "              ASEMBLER WIERSZO
WY      ":PRINT :PRINT "
        M E N U":P
30182 ? "M - Drukuj zawartosc pa
mieci"
30183 ? "B - Drukuj zawartosc pa
mieci/binarnie/"
30184 ? "D - Disasemblacja"
30185 ? "P - Pisz do pamieci (he
ksadecymalnie)"
30186 ? "T - Pisz tekst do pamie
ci"
30187 ? "A - Asemblacja"
30188 ? "L - Laduj do pamieci -
LOAD"
30189 ? "S - Wyprowadz z pamieci
 - SAVE"
30190 ? "> - Zamiana dec -> hex"
30191 ? "< - Zamiana hex -> dec"
30192 ? "# - Wyjscie na drukarke
"
30193 ? "! - Sterowanie do ATARI
 BASIC"
30194 ? "  - Powrot do M E N U"
30195 ? :? :? "                  W
YBIERASZ ?  ":POKE 82,N0:GOTO ST
ART
30200 DATA 11100,35221,0,0,0,352
00,3200,0,37100,35210,3130,0,0,3
5300,3300,0,10200,35222,0,0,0,35
203,3203,0,14100
30210 DATA 35304,0,0,0,35303,330
3,0,29300,2221,0,0,7200,2200,402
00,0,39100,2210,40130,0,7300,230
0,40300,0,8200
30220 DATA 2222,0,0,0,2203,40203
,0,44100,2304,0,0,0,2303,40303,0
,41100,24221,0,0,0,24200,33200,0
,36100,24210
30230 DATA 33130,0,28300,24300,3
3300,0,12200,24222,0,0,0,24203,3
3203,0,16100,24304,0,0,0,24303,3
3303,0,42100,1221
30240 DATA 0,0,0,1200,56200,0,38
100,1210,56130,0,28325,1300,5630
0,0,13200,1222,0,0,0,1203,56203,
0,46100,1304
30250 DATA 0,0,0,1303,56303,0,0,
47221,0,0,49200,47200,48200,0,23
100,0,53100,0,49300,47300,48300,
0,4200,47222,0,0
30260 DATA 49203,47203,48204,0,5
5100,47304,54100,0,0,47303,0,0,3
2210,30221,31210,0,32200,30200,3
1200,0,51100,30210
30270 DATA 50100,0,32300,30300,3
1300,0,5200,30222,0,0,32203,3020
3,31204,0,17100,30304,52100,0,32
303,30303,31304
30280 DATA 0,20210,18221,0,0,202
00,18200,21200,0,27100,18210,221
00,0,20300,18300,21300,0,9200,18
222,0,0,0,18203
30290 DATA 21203,0,15100,18304,0
,0,0,18303,21303,0,19210,43221,0
,0,19200,43200,25200,0,26100
30300 DATA 43210,34100,0,19300,4
3300,25300,0,6200,43222,0,0,0,43
203,25203,0,45100,43304,0,0,0,43
303,25303,0
```

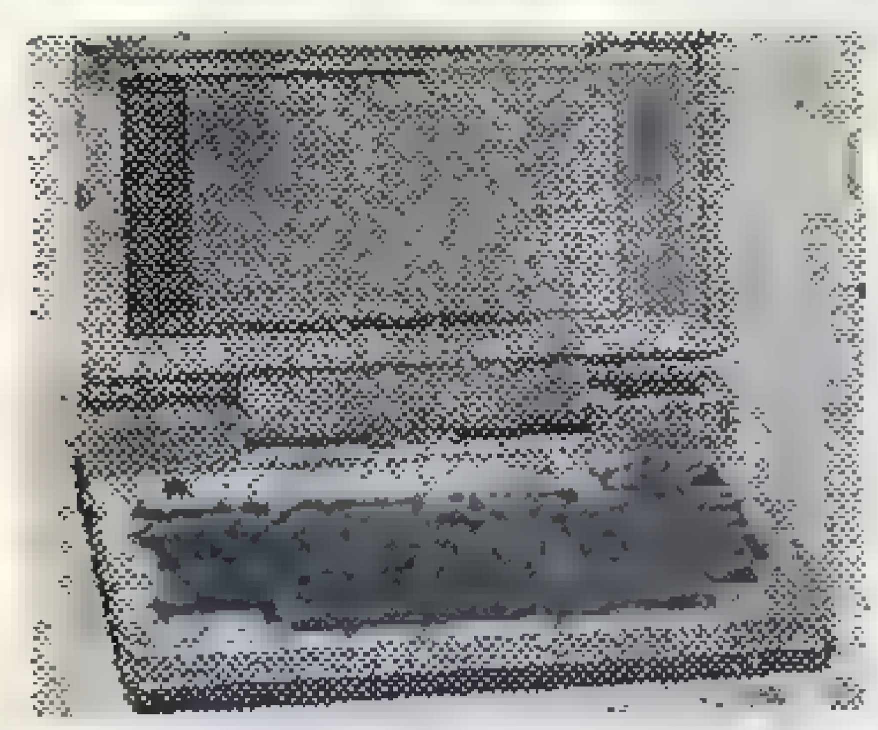
Oprac. H. KRASUSKI



17. **Sharp PC 5000**
Mikroprocesor: 8088, 4.77MHz
System operacyjny: MS-DOS 2.0
Pamięć RAM: 128 KB lub 256 KB
Pamięć masowa: Floppy-Disc 360 KB
Grafika: 640x80
Zasilanie: sieć lub akumulator
Rozmiary: 32.6x8.7x30.5 cm
Waga: 4.4 kg
Cena: 5000 M

18. **Tandy 102**
Mikroprocesor: 80C85, 2.4MHz
System operacyjny: ROM-based
Pamięć RAM: 24 KB lub 32 KB
Pamięć masowa: Mikrofloppy 100 KB
Grafika: 240x64
Zasilanie: sieć lub bateria
Rozmiary: 30x4.2x21.5 cm
Waga: 1.6 kg
Cena: 1400 M





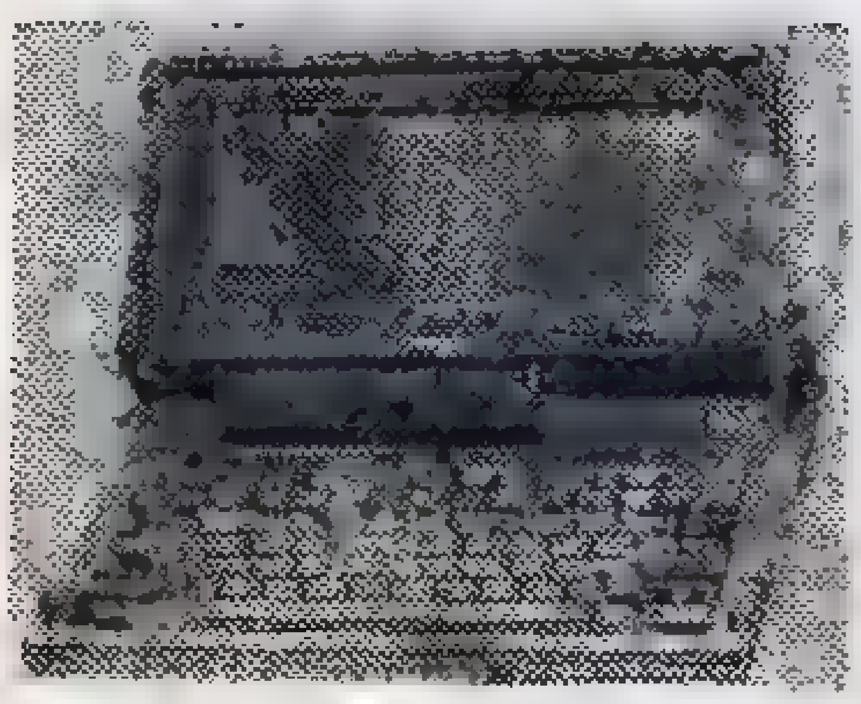
19. **Tandy 200**
Mikroprocesor: 80C85, 2.4MHz
System operacyjny: ROM-based
Pamięć RAM: 24 KB lub 72 KB
Pamięć masowa: Mikrofloppy 100 KB
Grafika: 240x128
Zasilanie: sieć, bateria lub akumulator
Rozmiary: 29.9x5.6x21.6 cm
Waga: 2.1 kg
Cena: 2500 M



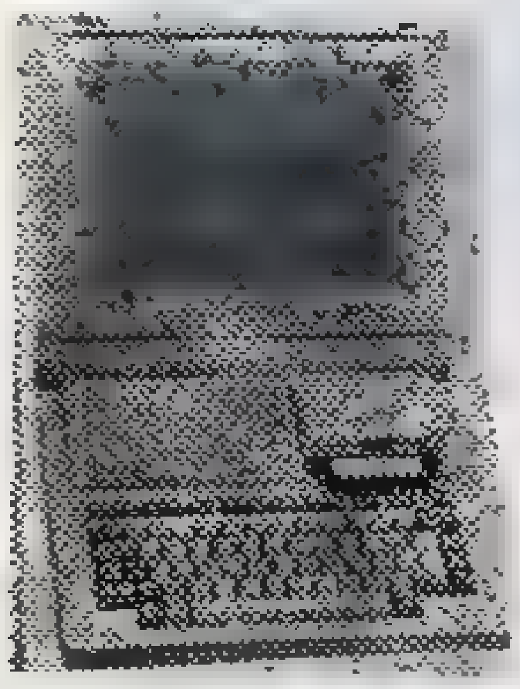
20. **Toshiba T 1100**
Mikroprocesor: 80C88, 4.77MHz
System operacyjny: MS-DOS 2.11
Pamięć RAM: 256 KB lub 512 KB
Pamięć masowa: Floppy-Disc 20 MB
Grafika: 640x200, 320x200
Zasilanie: sieć lub akumulator
Rozmiary: 31x6.6x30.5 cm
Waga: 4.2 kg
Cena: 4700 M



22. **Zenith Z 181 Portable PC**
Mikroprocesor: 80C88, 4.77MHz
System operacyjny: MS-DOS 3.1
Pamięć RAM: 640 KB lub 1.2 MB
Pamięć masowa: 2 Floppy-Disc po 720 KB, dysk twardy 20 MB
Grafika: 640x200
Zasilanie: sieć lub bateria
Rozmiary: 29x8x34 cm
Waga: 6 kg



21. **Zenith Z 171**
Mikroprocesor: 80C88, 4.77MHz
System operacyjny: MS-DOS 3.1
Pamięć RAM: 256 KB lub 640 KB
Grafika: 640x200
Zasilanie: sieć lub akumulator
Rozmiary: 33x24.1x16.8 cm
Waga: 6 kg
Cena: 6780 M



„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Michał Przybyłowski, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 813664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8570. Nr. ind. 361682. K-81

PROGRAMY 2

INFORMATYKA KOMPUTERY SYSTEMY

Dodatek „Żołnierza Wolności" – zeszyty programów komputerowych Cena 80 zł



W numerze:

AMSTRAD

- Grafika „Wieza Hanoi" — str. 12
- Technologia przenoszenia programów ze Spectrum na Amstrada — str. 21
- Rezystor - str. 24
- Hiperbole równoosiowe i sprzężone — str. 27
- Gra Hamiltona — str. 29

ATARI

- Automatyczny start programów napisanych w Basic-u na Atari 800 XL — str. 6
- Dysk — Menu — str. 7
- Rozszczepienie światła w pryzmacie — str. 8
- Regresja i korelacja — str. 18

COMMODORE

- „Życie" — str. 14

SHARP

- Dżungla — str 16

SPECTRUM

- Kod maszynowy — str. 3
- Rubik — str. 10

Okładka i rysunki przy programach: Michał Przybyłowski

Oto drugi numer zeszytu programów komputerowych. Sądzimy, że każdy użytkownik komputera znajdzie w nim coś dla siebie. **Ale uwaga** — nasze programy proponujemy traktować jedynie jako propozycje. Każdy z nich można nie tylko poprawić zaopatrując na przykład w atrakcyjną grafikę, ale także przetłumaczyć na dialekt zrozumiały przez inny komputer. Zabieg ten choć wymaga nieco praktyki i wiedzy opłaci się sowicie.

I jeszcze przypominamy: niektóre z programów będą emitowane przez „Radiokomputer" — program rozgłośni harcerskiej. Słuchajcie go zawsze we wtorki o godzinie 22.50 w programie IV i w piątki o 16.00 również w programie IV.

REDAKCJA

WARSZTATY KOMPUTEROWE'87

Pod patronatem Ogólnopolskiej Federacji Klubów Komputerowych Młodych Mistrzów Techniki ZSMP w dn. 19–21 marca br. w Pałacu Kultury i Nauki odbyły się warsztaty komputerowe INFOKLUB'87. Z ogólnej liczby 255 klubów, do Warszawy zaproszono dwadzieścia trzy. Stoisko Centralnej Składnicy Harcerskiej, punkt naprawy komputerów oraz wystawa firmy „LASER", stanowiły dodatkowe atrakcje dla zwiedzających.

Jaki cel miała ta impreza?

Organizatorzy nie ukrywali, że tego typu spotkania mają służyć przede wszystkim edukacji informatycznej młodzieży oraz wspieraniu procesu komputeryzacji społeczeństwa i gospodarki narodowej. Wynikało to również z celów, jakie stawiają sobie same kluby.

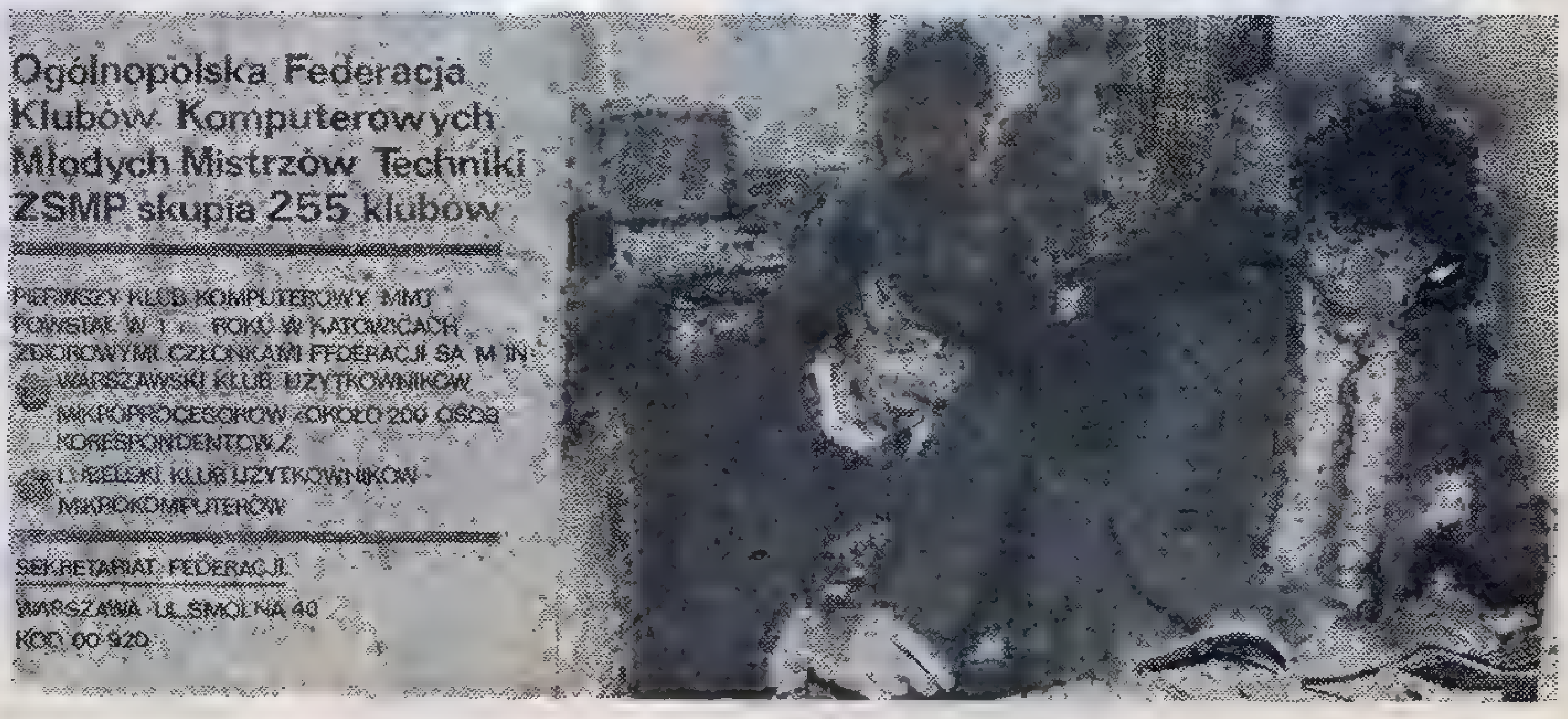
Podczas trwania „warsztatów" można było wymienić programy, skopiować dyskietki z ciekawymi grami, kupić specjalistyczną prasę, a nawet oddać do naprawy komputer i być świadkiem usunięcia usterki.

Przedstawiciele federacji wykazali dużo pomysłowości i inicjatywy w zorganizowaniu INFOKLUBU'87, ale jeszcze lepiej zaprezentowały się poszczególne kluby. Niektóre z nich przywiozły prawie wszystkie typy komputerów spotykanych w Polsce. Uzupełnieniem do tego był dodatkowy osprzęt z pełnym oprogramowaniem służącym zarówno do szkolenia, jak i zabawy.

Jakie mają osiągnięcia? Jakie plany? Na te pytania szukaliśmy odpowiedzi w bezpośrednich rozmowach z przedstawicielami klubów.

Tekst i foto: Rogoń Ryszard

▶ 5



Asembler

Napisanie programu w języku wewnętrznym procesora jest zajęciem żmudnym i powinno się ograniczać do tych przypadków, gdzie nie można z różnych względów zastosować języka wysokiego poziomu. Niemniej zwolenników kodu maszynowego nie brakuje i stąd prezentowany program.

Pozwala on poznać strukturę obszaru Basic'a (opcja P), może służyć jako narzędzie do kontroli pracy w tym języku. Opcja Z wyświetla użyte w Basic'u zmienne, a po wyborze klawisza W otrzymamy podstawowe dane o programie. Inne używane klawisze: D — kasowanie ostatniego znaku podczas wpisywania numeru linii, BREAK — powrót do menu.

Z poziomu interpretera Basic'a uruchamiamy program przez np. RANDOMIZE USR 64200. a opuszczamy go opcją Q. **Nie zapominajmy o obniżeniu RAMTOP komendą CLEAR 64199.**

Struktura programu w asemblerze powinna być szczególnie uporządkowana, gdyż tylko wtedy możliwy jest końcowy sukces. Spójrzmy na wydruk. Na początku wprowadzamy adres, od którego asembler będzie składował kod wynikowy (dyrektywa ORG). Rozkaz ENT nie jest konieczny, ale pozwala na uruchamianie programu z poziomu asemblera, co staje się wygodne przy testowaniu poszczególnych procedur lub fragmentów programu. Teraz wczytujemy skok do miejsca inicjalizacji kodu (START). W ten sposób tworzy się miejsce na „stałe fragmenty gry". W tym przypadku są to definicje tekstów, etykiety, zmienne programu (numer linii i licznik przesunięć ekranu) oraz procedury. Cały ten blok nie tylko poprawia czytelność struktury kodu, lecz przede wszystkim znacznie ułatwia programowanie. Oczywiście blok operacyjny rozpoczyna etykieta START.

Użyte zmienne systemowe ROM są lub będą opisane w miesięczniku „IKS", pozostaje więc krótko opisać procedury ROM, z których program korzysta. A zatem:

- 5633 — ustala kanał do transmisji danych (numer kanału w A),
- 3082 — wyprowadza teksty uporządkowane jak na wydruku,
- 6683 — wyprowadza na używany kanał liczbę z zakresu 0...9999,
- 8202 — ustala pozycję dla wydruku tekstu (linia w C, kolumna w B),
- 8220 — ustala pozycję kolumny dla wydruku (w C),
- 8020 — sprawdza, czy przyciśnięto klawisz BREAK (lub CAPS + SPACE),
- 6584 — przeszukuje obszar programu lub zmiennych BASIC,
- 3582 — skroluje ekran o jedną linię w górę,
- 6654 — wydobywa liczbę umieszczoną cyfra po cyfrze w pamięci (do BC),
- 6510 — wyprowadza adres linii o podanym numerze w HL,
- 11547 — testuje, czy bajt w akumulatorze jest cyfrą,
- 11563 — umieszcza parę BC na stosie kalkulatora,
- 11747 — wyprowadza zawartość stosu kalkulatora na używany kanał.

Wszystkie adresy podane są w programie dziesiętnie, co może przeszkadzać „heksadecymalnym". W takim wypadku trzeba po prostu podzielić adres przez 16 lub skorzystać z programu MONITOR.

Krzysztof MAMCARZ

```
  10 ;** BASICDUMP '85
  20 ;GENS3M2 assembler
  30                ORG  64200
  40                ENT  64200
  50                JP   START
  60 ;teksty
  70 NAPISY         DEFB 128
  80                DEFM "1985 BASI"
  90                DEFM "C DUMP"
 100                DEFB " "+128
 110                DEFM "obszar prog"
 120                DEFM "ramu pusty"
 130                DEFB " "+128
 140                DEFM "obszar zmie"
 150                DEFM "nnych pusty"
 160                DEFB " "+128
 170                DEFM "koniec obsza"
 180                DEFM "ru programu"
 190                DEFB " "+128
 200                DEFM "P budowa "
 210                DEFM "programu"
 220                DEFB " "+128
 230                DEFM "Z lista "
 240                DEFM "zmiennych"
 250                DEFB " "+128
 260                DEFM "W wskazniki"
 270                DEFB " "+128
 280                DEFM "Q basic"
 290                DEFB " "+128
 300                DEFM "od numeru "
 310                DEFM " linii"
 320                DEFB " "+128
 330                DEFM "program"
 340                DEFB " "+128
 350                DEFM "zmienne"
 360                DEFB " "+128
 370                DEFM "wolne"
 380                DEFB " "+128
 390                DEFM "ramtop"
 400                DEFB " "+128
 410                DEFM "liczba"
 420                DEFB " "+128
 430                DEFM "(5 bajtow)"
 440                DEFB " "+128
 450                DEFM "linia nr."
 460                DEFB " "+128
 470                DEFM "dlug.linii"
 480                DEFB " "+128
 490                DEFM "koniec linii"
 500                DEFB " "+128
 510                DEFM "ktrl"
 520                DEFB " "+128
 530 ;etykiety
 540 ; zmienne syst.
 550 BORDCR         EQU  23624
 560 KSTATE         EQU  23556
 570 ATTR_T         EQU  23695
 580 VARS           EQU  23627
 590 PROG           EQU  23635
 600 ELINE          EQU  23641
 610 STKEND         EQU  23653
 620 RAMTOP         EQU  23730
 630 ; procedury ROM
 640 OPENCH         EQU  5633
 650 POMSG          EQU  3082
 660 OUTNUM         EQU  6683
 670 NUMBER         EQU  11547
 680 AT             EQU  8202
 690 TAB            EQU  8220
 700 NEXT           EQU  6584
 710 SCROLL         EQU  3582
 720 STK_BC         EQU  11563
 730 PR_FP          EQU  11747
 740 BRTEST         EQU  8020
 750 GETLN          EQU  6654
 760 ADDRLN         EQU  6510
 770 ;zmienne programu
 780 BUFOR          DEFS 5
 790 LICZ           DEFB 10
 800 ;procedura CLS
 810 CLS            LD   A,2
 820                CALL OPENCH
 830                LD   HL,16384
 840                LD   (HL),0
 850                PUSH HL
 860                POP  DE
 870                INC  DE
 880                LD   BC,6143
 890                LDIR
 900                RET
 910 ;procedura PRTSTR(A)
 920 ;we: A-nr. napisu
 930 PRTSTR         LD   DE,NAPISY
 940                JP   POMSG
 950 ;procedura BORDER(A)
 960 ;we: A-kolor ramki
 970 BORDER         OUT  (254),A
 980                RLCA
 990                RLCA
1000                RLCA
1010                LD   (BORDCR),A
1020                RET
1030 ;procedura KOLORY(A,C)
1040 ;we: A-paper
1050 ;     C-ink
1060 KOLORY         RLCA
1070                RLCA
1080                RLCA
```

```
1090                OR    C
1100                LD    HL,22528
1110                LD    (HL),A
1120                PUSH  HL
1130                POP   DE
1140                INC   DE
1150                LD    BC,767
1160                LDIR
1170                LD    (ATTR_T),A
1180                RET
1190 ;procedura PREDIT(A)
1200 ;we: A-nr. napisu
1210 PREDIT         PUSH  AF
1220                LD    A,1
1230                CALL  OPENCH
1240                LD    BC,256
1250                CALL  AT
1260                POP   AF
1270                CALL  PRTSTR
1280                LD    A,2
1290                JP    OPENCH
1300 ;procedura KLAWISZ
1310 ;wy A-kod ASCII
1320 KLAW           RES   5,(IY+1)
1330                BIT   5,(IY+1)
1340                JR    Z,KLAW
1350                LD    A,(KSTATE)
1360                RET
1370 ;procedura BREAK
1380 BREAK          CALL  BRTEST
1390                JR    C,BREAK
1400                RET
1410 ;procedura DLPROG
1420 DLPROG         LD    HL,(VARS)
1430                LD    BC,(PROG)
1440                AND   A
1450 MINUS          SBC   HL,BC
1460                LD    B,H
1470                LD    C,L
1480 DRUK           CALL  STK_BC
1490                CALL  PR_FP
1500                LD    A,13
1510                RST   16
1520                LD    A,13
1530                RST   16
1540                RET
1550 ;procedura DLZMIE
1560 DLZMIE         LD    HL,(ELINE)
1570                LD    BC,(VARS)
1580                SCF
1590                JR    MINUS
1600 ;procedura WOLNE
1610 WOLNE          LD    HL,0
1620                ADD   HL,SP
1630                LD    BC,(STKEND)
1640                AND   A
1650                JR    MINUS

1660 ;procedura TOPRAM
1670 TOPRAM         LD    BC,(RAMTOP)
1680                JR    DRUK
1690 ;procedura KURSOR
1700 KURSOR         LD    A,18
1710                RST   16
1720                LD    A,1
1730                RST   16
1740                XOR   A
1750                RST   16
1760                LD    A,18
1770                RST   16
1780                XOR   A
1790                RST   16
1800                LD    A,8
1810                RST   16
1820                RET
1830 ;procedura DELETE
1840 DELETE         LD    HL,BUFOR
1850                XOR   A
1860                SBC   HL,DE
1870                RET   Z
1880                LD    A,32
1890                RST   16
1900                LD    A,8
1910                RST   16
1920                LD    A,8
1930                RST   16
1940                DEC   DE
1950                RET
1960 ;procedura CYFRA
1970 ;we: A-kod ASCII
1980 ;    DE-wskaz. bufora
1990 CYFRA          CALL  NUMBER
2000                RET   C
2010                LD    HL,BUFOR+4
2020                AND   A
2030                SBC   HL,DE
2040                RET   Z
2050                LD    B,A
2060                RST   16
2070                LD    A,B
2080                LD    (DE),A
2090                INC   DE
2100                RET
2110 ;procedura PR_ADR
2120 ;we: HL-adres linii
2130 PR_ADR         PUSH  HL
2140                LD    BC,21
2150                CALL  AT
2160                PUSH  HL
2170                POP   BC
2180                CALL  STK_BC
2190                CALL  PR_FP
2200                LD    BC,6
2210                CALL  TAB
2220                POP   HL
2230                LD    B,0
2240                LD    C,(HL)
2250                CALL  OUTNUM

2260                LD    A,6
2270                RST   16
2280                RET
2290 ;procedura PRZESUW
2300 PRZES          LD    A,(LICZ)
2310                DEC   A
2320                LD    (LICZ),A
2330                AND   A
2340                JR    NZ,UP
2350                LD    A,10
2360                LD    (LICZ),A
2370                CALL  KLAW
2380 UP             PUSH  HL
2390                CALL  SCROLL
2400                POP   HL
2410                RET
2420 ;******************
2430 ;PROGRAM GLOWNY
2440 START          CALL  CLS
2450                LD    A,4
2460                CALL  BORDER
2470                LD    A,4
2480                LD    C,0
2490                CALL  KOLORY
2500                LD    BC,10
2510                CALL  AT
2520                LD    A,4
2530                CALL  PRTSTR
2540                LD    A,13
2550                RST   16
2560                LD    A,5
2570                CALL  PRTSTR
2580                LD    A,13
2590                RST   16
2600                LD    A,6
2610                CALL  PRTSTR
2620                LD    A,13
2630                RST   16
2640                LD    A,7
2650                CALL  PRTSTR
2660 MENU           CALL  KLAW
2670                CP    "Q"
2680                RET   Z
2690                CP    "P"
2700                JP    Z,DUMP
2710                CP    "W"
2720                JP    Z,WSK
2730                CP    "Z"
2740                JR    Z,ZMIEN
2750                JR    MENU
2760 ;**********************
2770 ZMIEN          CALL  CLS
2780                LD    BC,0
2790                CALL  AT
2800                LD    A,5
2810                CALL  BORDER
2820                LD    A,5
2830                LD    C,2
2840                CALL  KOLORY
2850                LD    HL,(VARS)
2860                LD    A,(HL)
2870                CP    128
2880                JR    NZ,PETLA
2890                LD    A,2
2900                CALL  PREDIT
2910 EXIT           CALL  BREAK
2920                JP    START
2930 PETLA          CALL  NEXT
2940                LD    A,(HL)
2950                CP    128
2960                JR    Z,EXIT
2970                AND   31
2980                SET   5,A
2990                SET   6,A
3000                RST   16
3010                LD    A,(HL)
3020                AND   224
3030                RLCA
3040                RLCA
3050                RLCA
3060                CP    2
3070                JR    NZ,DAL1
3080                LD    A,"x"
3090                RST   16
3100                JR    NAST
3110 DAL1           CP    6
3120                JR    NZ,DAL2
3130                LD    A,"x"
3140                RST   16
3150 DIM            LD    A,233
3160                RST   16
3170                JR    NAST
3180 DAL2           CP    7
3190                JR    NZ,DAL3
3200                LD    A,235
3210                RST   16
3220                JR    NAST
3230 DAL3           CP    4
3240                JR    Z,DIM
3250                CP    3
3260                JR    Z,NAST
3270                PUSH  DE
3280                EX    DE,HL
3290                XOR   A
3300                CALL  POMSG
3310                POP   DE
3320 NAST           LD    A,6
3330                RST   16
3340                EX    DE,HL
3350                JR    PETLA
3360 ;*********************
3370 WSK            CALL  CLS
3380                XOR   A
3390                LD    C,7
3400                CALL  KOLORY

3410                LD    BC,5
3420                CALL  AT
3430                LD    A,9
3440                CALL  PRTSTR
3450                CALL  DLPROG
3460                LD    A,10
3470                CALL  PRTSTR
3480                CALL  DLZMIE
3490                LD    A,11
3500                CALL  PRTSTR
3510                CALL  WOLNE
3520                LD    A,12
3530                CALL  PRTSTR
3540                CALL  TOPRAM
3550                CALL  BREAK
3560                JP    START
3570 ;*********************
```

```
3580 DUMP      CALL CLS
3590           LD   A,7
3600           LD   C,1
3610           CALL KOLORY
3620           LD   HL,(VARS)
3630           LD   DE,(PROG)
3640           AND  A
3650           SBC  HL,DE
3660           JR   NZ,NRLIN
3670           LD   A,1
3680           CALL PREDIT
3690           CALL BREAK
3700           JP   START
3710 NRLIN     LD   BC,10
3720           CALL AT
3730           LD   A,8
3740           CALL PRTSTR
3750           LD   DE,BUFOR
3760 LINIA     CALL KURSOR
3770           CALL KLAW
3780           CP   13
3790           JR   Z,KOMPL
3800           CALL CYFRA
3810           CP   "D"
3820           CALL Z,DELETE
3830           JR   LINIA
3840 KOMPL     LD   A,255
3850           LD   (DE),A
3860           CALL CLS
3870           LD   HL,BUFOR
3880           CALL GETLN
3890           LD   H,B
3900           LD   L,C
3910           CALL ADDRLN
3920 LIN1      PUSH HL
3930           LD   DE,(VARS)
3940           AND  A
3950           SBC  HL,DE
3960           POP  HL
3970           JR   NZ,CONT
3980           LD   A,3
3990           CALL PREDIT
4000           CALL BREAK
4010           JP   START
4020 CONT      CALL PR_ADR
4030           PUSH HL
4040           LD   A,15
4050           CALL PRTSTR
4060           POP  HL
4070           LD   B,(HL)
4080           INC  HL
4090           LD   C,(HL)
4100           CALL OUTNUM
4110           CALL PRZES
4120           CALL PR_ADR
4130           CALL PRZES
4140           INC  HL
4150           CALL PR_ADR
4160           PUSH HL
4170           LD   A,16
4180           CALL PRTSTR
4190           POP  HL
4200           LD   C,(HL)
4210           INC  HL
4220           LD   B,(HL)
4230           PUSH HL
4240           CALL STK_BC
4250           CALL PR_FP
4260           POP  HL
4270           CALL PRZES
4280           CALL PR_ADR
4290           CALL PRZES
4300           INC  HL
4310 INLINE    CALL PR_ADR
4320           LD   A,(HL)
4330           CP   13
4340           JR   NZ,KOD14
4350           PUSH HL
4360           LD   A,17
4370           CALL PRTSTR
4380           POP  HL
4390           CALL PRZES
4400           INC  HL
4410           JR   LIN1
4420 KOD14     CP   14
4430           JR   NZ,CTRL
4440           PUSH HL
4450           LD   A,13
4460           CALL PRTSTR
4470           POP  HL
4480           CALL PRZES
4490           INC  HL
4500           CALL PR_ADR
4510           PUSH HL
4520           LD   A,14
4530           CALL PRTSTR
4540           POP  HL
4550           CALL PRZES
4560           INC  HL
4570           LD   B,4
4580 FLOAT     PUSH BC
4590           CALL PR_ADR
4600           CALL PRZES
4610           INC  HL
4620           POP  BC
4630           DJNZ FLOAT
4640           JR   INLINE
4650 CTRL      CP   16
4660           JR   C,NORMAL
4670           CP   21
4680           JR   NC,NORMAL
4690           PUSH HL
4700           LD   A,18
4710           CALL PRTSTR
4720           POP  HL
4730           LD   A,(HL)
4740           ADD  A,201
4750           RST  16
4760           CALL PRZES
4770           INC  HL
4780           CALL PR_ADR
4790 WYK       CALL PRZES
4800           INC  HL
4810           JR   INLINE
4820 NORMAL    RST  16
4830           JR   WYK
```

Wojewódzki Klub Komputerowy MMT **„Sobótka"** działa pod patronatem Uniwersytetu Mikołaja Kopernika w Toruniu. Siedzibą klubu jest Młodzieżowy Dom Kultury „Kameleon". Może właśnie dlatego członkowie „Sobótki" mają tak zmienne szczęście. Dostając 31 grudnia 1986 roku 300 tys. złotych musieli wydać je właśnie w tym samym dniu. Jak kupić potrzebny sprzęt w tak krótkim czasie? Tej tajemnicy nikt nie chciał nam zdradzić.

Oprócz typowej działalności szkoleniowej dla młodzieży i dorosłych, klub prowadzi akcję „alfabetyzacji komputerowej" dla dzieci od lat 6 do 10



7

ATARI

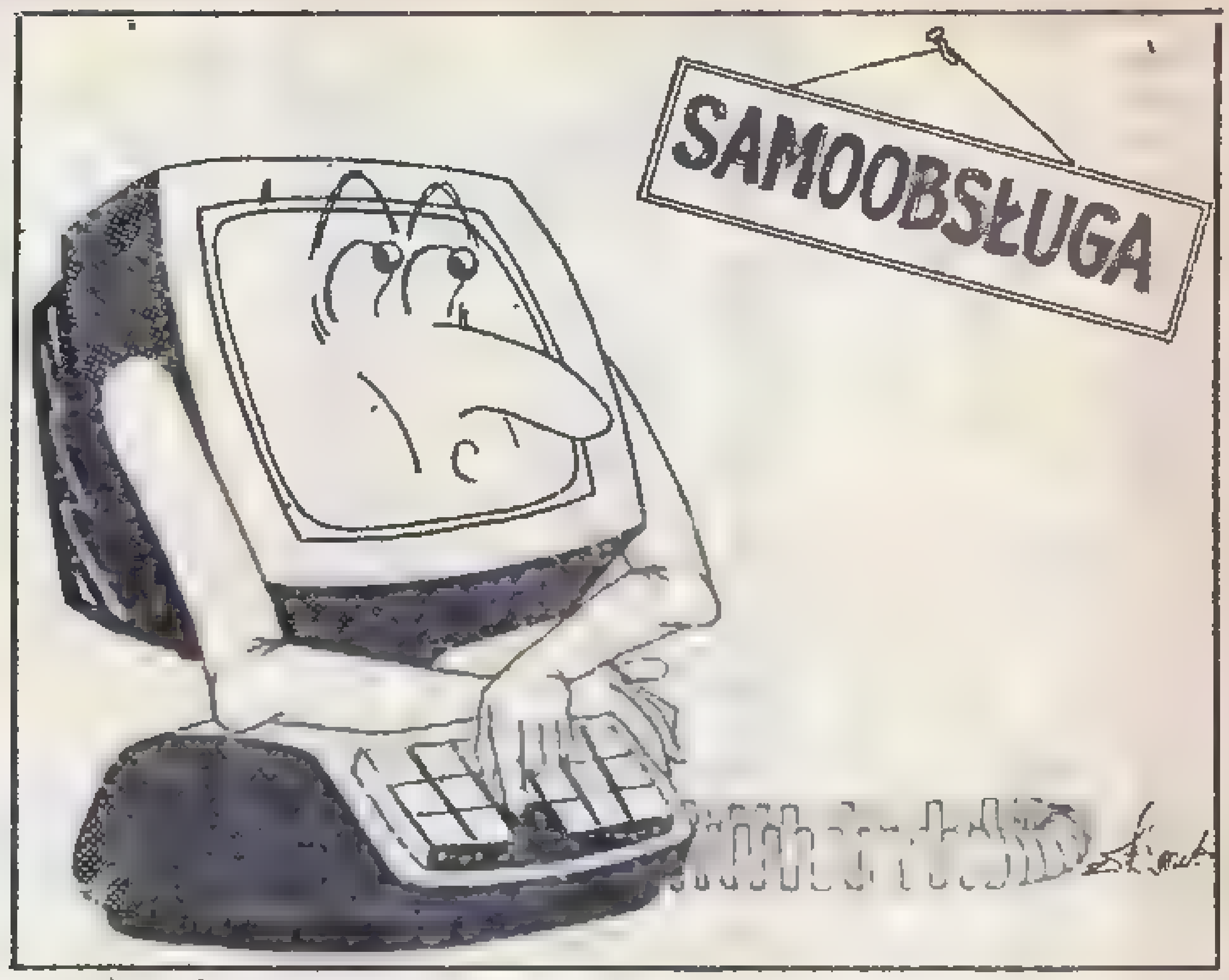
PROGRAM 2—3

AUTOMATYCZNY START PROGRAMÓW NAPISANYCH W BASICU NA ATARI 800XL

Programy firmowe dla mikrokomputerów ATARI 800XL napisane w kodzie maszynowym i dostępne na dyskach, np. gry, programy użytkowe, posiadają bardzo wygodny sposób uruchamiania. Po prostu: wkłada się dyskietkę do stacji dysków i włącza komputer. Natomiast uruchomienie programu napisanego w języku BASIC wymaga od użytkownika napisania komendy **LOAD** z odpowiednią nazwą zbioru i dopiero po wykonaniu jej przez mikrokomputer, komendą RUN można uruchomić załadowany program.

Przedstawiony poniżej program **AUTOSTART** umożliwia uruchamianie programów w **BASIC-u** w sposób tak wygodny jak uruchamianie programów firmowych. Istota jego działania polega na tym, że zapisuje on na dyskietce zbiór o nazwie **AUTORUN.SYS** zawierający program w kodzie maszynowym (linie 2000—2036). Dla dyskowego systemu operacyjnego (DOS) mikrokomputera ATARI zbiór o nazwie **AUTORUN.SYS** ma specjalne znaczenie. Zawarty w zbiorze program binarny (zapisany w kodzie maszynowym) jest ładowany i uruchamiany automatycznie po włączeniu mikrokomputera.

Po wprowadzeniu linii programu do pamięci należy go zapisać na dyskietkę, a następnie uruchomić komendą **RUN.** Program zatrzyma się i zażąda podania nazwy zbioru dyskowego zawierającego program w **BASIC-u,** który ma być automatycznie uruchamiany. Wówczas należy podać nazwę zbioru np.: **PROGRAM1.BAS** i wcisnąć klawisz **RETURN**. Następnie do stacji dysków należy włożyć dyskietkę zawierającą zbiór z podaną wcześniej nazwą i wcisnąć **START.**. Program uzupełni kod maszynowy zawarty w liniach 2000—2036 o podaną nazwę zbioru i zapisze zaktualizowany kod do zbioru **AUTORUN.SYS,** po czym uruchomi się od początku. Analogiczne operacje program **AUTOSTART** będzie powtarzał z kolejnymi dyskietkami. Jeżeli chcemy przerwać działanie programu, należy w momencie, gdy żąda on podania nazwy zbioru napisać * (gwiazdkę) i wcisnąć **RETURN**. Gdy mamy już zapisany na dyskietce zbiór **AUTORUN.SYS** możemy automatycznie uruchomić program. W tym celu wyłączamy na kilka sekund mikrokomputer i po chwili włączamy go ponownie. Przy włączaniu komputera nie należy wciskać klawisza ØPTION. Po kilku sekundach program „zamelduje się" na ekranie. W tym miejscu należy wymienić trzy warunki które muszą być spełnione, aby program **AUTOSTART** poprawnie działał:



1. W opisany wyżej sposób można uruchamiać programy w **BASIC-u** zapisane na dysk komendą **SAVE** (zapisane komendą **LIST** generować będą przy próbie uruchomienia błąd nr 21).
2. Na dyskietce oprócz zbioru zawierającego uruchamiany program, zbioru **AUTORUN.SYS,** musi znajdować się **DOS** wraz z pakietem **DUP.SYS.**
3. Na dyskietce można zapisać TYLKO JEDEN zbiór o nazwie AUTORUN.SYS

Z warunku trzeciego wynika, że program AUTOSTART jest pożyteczny, gdy na dysku mamy jeden program w BASIC-u. Co zrobić w sytuacji, gdy na dyskietce zapisanych jest np.: 20 lub więcej programów. Też można je wszystkie automatycznie uruchamiać. W tym celu wprowadzamy do pamięci mikrokomputera drugi program o nazwie DYSK—MENU (linie 1—1950). Wyświetla on wykaz zbiorów na dysku przyporządkowując jednocześnie każdemu zbiorowi pojedynczy znak — poczynając od znaku A np.:

PROGRAM1 PROGRAM2BAS
PROSTA BAS PROGRAM4

itd. Chcąc uruchomić program zawarty w zbiorze **PROSTA.BAS** wystarczy tylko nacisnąć klawisz C. Wciśnięcie klawisza **RETURN** przekazuje sterowanie do translatora BASIC-a. Program **DYSK—MENU** musi być zapisany na dyskietkę zawierającą programy, które chcemy automatycznie uruchamiać komendą **SAVE.** Następnie programem AUTOSTART wpisujemy na tę samą dyskietkę zbiór AUTORUN.SYS. Gdy program AUTOSTART zażąda nazwy zbioru z programem do automatycznego startu należy podać nazwę zbioru, do którego wpisany został program **DYSK—MENU.** Od tej chwili możemy w komfortowy sposób uruchamiać dowolny z wielu programów zapisanych na dysku. Ma to jeszcze jedną zaletę. Nie musimy pamiętać, ani skrzętnie ewidencjonować zbiorów na dyskietce. Aby „zobaczyć" co na niej jest, wkładamy ją do stacji dysków, włączamy komputer i po chwili wszystko już wiemy.

```
1 REM *****************************
2 REM PROGRAM AUTOSTART
3 REM *****************************
100 DIM D(168),F$(16),L$(16)
110 ? CHR$(125):POKE 752,1
120 POSITION 0,0:? CHR$(17);:FOR
 LI=1 TO 38:? CHR$(18);:NEXT LI
? CHR$(5);
122 POSITION 0,1:? "|  PROGRAM Z
APISUJE ZBIOR AUTORUN.SYS  |";
124 POSITION 0,2:? "|  UMOZLINIA
JACY AUTOMATYCZNY START    |";
125 POSITION 0,3:? "|   PROGRAMO
W NAPISANYCH W BASICU      |";
126 POSITION 0,4:? CHR$(26);:FOR
 LI=1 TO 38:? CHR$(18);:NEXT LI
? CHR$(3);
130 POSITION 3,6:? "Zbior AUTORU
N.SYS zajmuje 2 sektory.";
150 POSITION 3,7:? "Podaj nazwe
zbioru zawierajacego"
151 POSITION 3,8:? "program, kto
ry ma miec automat.START"
152 INPUT F$
155 IF F$(1,1)="*" THEN GRAPHICS
 0:END
160 L=LEN(F$)
162 FOR I=1 TO L:IF F$(I,I)=" "
THEN L=I-1:GOTO 170
165 NEXT I
170 FOR N=1 TO L
172 IF N=1 AND F$(N,N)<CHR$(65)
OR F$(N,N)>CHR$(90) THEN GOTO 11
00
174 IF N<>1 AND F$(N,N)<CHR$(48)
 AND F$(N,N)<>"." THEN GOTO 1100
```

```
176 IF N<>1 AND (F$(N,N)>CHR$(57
) AND F$(N,N)<CHR$(65) OR F$(N,N
)>CHR$(90)) THEN 1100
178 IF F$(N,N)="." AND (L>N+3 OR
 P=1) THEN 1100
180 IF F$(N,N)="." THEN P=1
190 NEXT N
200 POSITION 3,10:? "Wloz dyskie
tke do stacji dysk0w"
220 FOR I=1 TO 168:READ A:D(I)=A
:NEXT I
240 POSITION 3,12:?."Nacisnij ST
ART "
242 IF PEEK(53279)=6 THEN GOTO 2
50
245 GOTO 242
250 FOR I=148 TO 137 STEP -1
260 J=J+1:IF J>L THEN D(I)=32:GO
TO 270
265 D(I)=ASC(F$(J,J))
270 NEXT I
280 GOSUB 1000
290 POSITION 3,14:? "Prosze chwi
le poczekac..."
300 GOTO 1510
1000 TRAP 1500:CLOSE #1:OPEN #1,
8,0,"D:AUTORUN.SYS"
1010 FOR I=1 TO 168:PUT #1,D(I)
1020 NEXT I:CLOSE #1:RETURN
1100 ? CHR$(253);CHR$(254);:RUN
1500 POSITION 3,20:? "Wystapil b
lad numer ";PEEK(195)
1510 FOR T=1 TO 1100:NEXT T:? CH
R$(253):RUN
2000 DATA 255,255,106,6,255,6,16
9,80
2002 DATA 141,0,3,169,1,141,1,3
2004 DATA 169,63,141,2,3,169,64,
141
2006 DATA 3,3,169,5,141,6,3,141
2007 DATA 5,3,169,0,141,4,3,141
2008 DATA 9,3,141,10,3,141,11,3
2010 DATA 169,12,141,8,3,32,89,2
28
2012 DATA 16,1,96,162,11,189,0,5
2014 DATA 157,0,3,202,16,247,32,
89
2016 DATA 228,48,6,32,6,5,108,12
2018 DATA 0,96,169,193,141,33,3,
169
2020 DATA 6,141,34,3,96,251,243,
51
2022 DATA 246,207,6,163,246,51,2
46,60
2024 DATA 246,76,228,243,172,255
,6,240
2026 DATA 9,185,234,6,206,255,6,
160
2028 DATA 1,96,140,33,3,169,228,
141
2030 DATA 34,3,169,155,160,1,96
2032 DATA 34,67,88,69,46,71,78,7
3,84,82,65,84,83,58,68,34,78,85,
82
2034 DATA 32,20,226,2,227,2
2036 DATA 106,6,224,2,225,2,160,
6
```

Program **DYSK—MENU** umożliwia wybór jednego z 26 programów, które mogą być automatycznie uruchamiane (litery A—Z). Jeżeli mamy na dysku więcej zbiorów (maksymalnie 64) należy w linii 1820 zmienić liczbę 91 na inną. Na przykład: liczba 101 zwiększa liczbę programów z 26 do 36. Należy jednak pamiętać, że w tej sytuacji kolejnym programom będą przyporządkowane kolejne znaki (kody ATASCII), „większe" od liter alfabetu. Wybór programów do automatycznego startu będzie odbywał się poprzez „wybieranie" z klawiatury tych znaków, co nie zawsze jest wygodne gdyż np.: wybranie małej litery „a" wymaga użycia dwóch klawiszy (CAPS i A).

```
1 REM ***************************
2 REM PROGRAM DYSK-MENU
3 REM ***************************
1600 DIM A$(20),NZ$(12),TP$(768)
,P$(14)
1605 POKE 752,1
1610 CLOSE #1:OPEN #1,6,0,"D:*.*
"
1615 Y=65:J=1
1620 FOR X=1 TO 64:INPUT #1,A$
1630 IF A$(3,3)<"A" OR A$(3,3)>"
Z" THEN 1670
1640 TP$(J,J)=CHR$(Y):TP$(J+1,J+
11)=A$(3,13)
1650 J=J+12:Y=Y+1
1660 IF A$(1,1)=" " OR A$(1,1)="
*" THEN NEXT X
1670 Y=155:TP$(J,J)=CHR$(155)
1700 ? CHR$(125)
1710 POSITION 8,1:? "WYKAZ ZBIOR
OW NA DYSKIETCE"
1720 J=1:Y=3
1730 IF TP$(J,J)=CHR$(155) THEN
GOTO 1800
1740 POSITION 4,Y:? TP$(J,J);" "
;TP$(J+1,J+11):J=J+12
1750 IF TP$(J,J)=CHR$(155) THEN
GOTO 1800
1760 POSITION 22,Y:? TP$(J,J);"
";TP$(J+1,J+11):J=J+12:Y=Y+1:GOT
O 1730
1800 CLOSE #1:OPEN #1,4,0,"K:"
1810 POSITION 15,22:? "WYBIERASZ
? "
1820 GET #1,B:IF B=155 THEN END
:IF B<=64 OR B>=91 THEN GOTO 182
0
1830 J=1
1840 IF TP$(J,J)=CHR$(B) THEN GO
TO 1900
1850 IF TP$(J,J)=CHR$(155) THEN
GOTO 1950
1860 J=J+12:GOTO 1840
1900 P$(1,2)="D:":Y=2
1901 FOR X=0 TO 7
1902 IF TP$(J+1+X,J+1+X)=" " THE
N GOTO 1904
1903 Y=Y+1:P$(Y,Y)=TP$(J+1+X,J+1
+X):NEXT X
1904 Y=Y+1:P$(Y,Y)="."
1905 IF TP$(J+9,J+11)="   " THEN
P$(Y,Y)=" ":GOTO 1910
1906 P$(Y+1,Y+3)=TP$(J+9,J+11)
1910 POSITION 6,20:? "URUCHAMIAM
PROGRAM ";P$:RUN P$
1950 POSITION 0,0:? "zle wybrale
s program !!! ":GOTO 1800
```

Opracował H. KRASUSKI

Stoisko Mikołaja Przychody było nie mniej oblegane od innych. „Złote rączki" mistrza potrafiły w ciągu kilku minut usunąć różnorodne usterki z komputera. Za dodatkową zapłatą można było również wymienić rodzime układy w Meritum na podobne z inicjałami renomowanych firm elektronicznych. W odczuciu większości uczestników utworzenie punktu naprawy komputerów było znakomitym pomysłem organizatorów.

9



ATARI

PROGRAM 4



Rozszczepienie światła w pryzmacie

Program ten, umożliwia zaobserwowanie zjawiska rozszczepienia światła w pryzmacie. Jako dane, należy podać rodzaj materiału, z którego wykonany jest pryzmat, kąt łamiący pryzmatu oraz kąt padania wiązki światła białego na pryzmat. Zakończenie prezentacji rozszczepienia światła jest sygnalizowane krótkim sygnałem dźwiękowym.

W tym momencie, aby przejść do następnej części programu, nalezy wcisnąć klawisz SELECT. W tej części programu, mozna uzyskać dane dotyczące rozszczepienia i załamania poszczególnych promieni w pryzmacie. Dalsze informacje dotyczące obsługi programu będą ukazywane podczas wykonywania programu. Program ten, działa także w TURBO BASIC'u XL, co powoduje znaczne przyspieszenie wykonywanych obliczeń.

Posiadaczom odbiorników czarno-białych polecilbym dokonanie w linii 200 zmiany trybu graficznego z 11 na 9. Można dokonać też wielu zmian dotyczących ilości rozpatrywanych kolorów, lecz pozostawiam to jedynie dociekliwym użytkownikom komputerów ATARI.

Olgierd NIEMYJSKI

```
0 CLR ·DEG
1 MX=0:DIM V$(40),PR(5,3),LL(5)
5 GRAPHICS 0:POKE 710,12:POKE 709,0
6 POL=0
7 FOR I=1 TO 5:FOR II=1 TO 3
8 PR(I,II)=0:NEXT II:NEXT I
10 H=170 Z=16 Q=0.25:N1=1:N2=1.5
15 POSITION 5,3:POKE 752,6
16 ? "                              "
20 POSITION 5,4
21 ? "     ROZCZEPIENIE SWIATLA     "
23 POSITION 5,5
24 ? "         W PRYZMACIE          "
25 POSITION 5.6
26 ? "                              "
30 POSITION 10,10:? "1- Crown lekki"
32 POSITION 10,12:? "2- Flint lekki"
34 POSITION 10,14:? "3- Flint ciezki
"
36 POSITION 8,19
38 ? "PODAJ RODZAJ PRYZMATU"
39 GOSUB 9000
40 OPEN #1,4,0,"K":GET #1,ZZ
41 CLOSE #1 SK=ZZ-49
42 IF SK=0 THEN 50
43 IF SK=1 THEN 50
44 IF SK=2 THEN 50
45 GOTO 40
50 RESTORE 5000+100*SK
60 GRAPHICS 0:POKE 710,12:POKE 709,0
65 POSITION 6,3:TRAP 65
66 ? "Podaj kat lamiacy pryzmatu"
68 POSITION 18,5:INPUT GAMA
69 IF GAMA>110 OR GAMA<1 THEN 65
70 POSITION 10,9:TRAP 70
71 ? "Podaj kat padania"
72 POSITION 18,11:INPUT AL
73 IF AL>90 OR AL<0 THEN 70
74 IF MX=1 THEN 110
75 POSITION 12,20:POKE 752;6
76 ? " DANE PRYZMATU":TRAP 40000
85 FOR M=1 TO 6:POKE 755,2
87 POSITION 15,18:? " SELECT "
89 SOUND 1,50,10,15
90 FOR M1=1 TO 130:NEXT M1
95 SOUND 1,0,0,0:POKE 755,0
100 FOR M1=1 TO 70:NEXT M1
105 NEXT M
110 GA=GAMA/2
200 GRAPHICS 11:COLOR 15
500 REM RYSOWANIE PRYZMATU
510 APO=H*SIN(GA)/COS(GA)
515 IF APO>160 THEN H=H*0.8 GOTO 510
520 APX=(160-APO)*Q:APY=191-Z
530 BPX=APX+APO*Q:BPY=191-Z-H
540 CPX=BPX+APO*Q:CPY=APY
550 PLOT APX,APY:DRAWTO BPX,BPY:
DRAWTO CPX,CPY
560 FOR R=Z TO 1 STEP -1:K=191-R
565 COLOR R:PLOT 0,K:DRAWTO 78,K
570 NEXT R
590 COLOR 15
600 REM *****  *****  ***** ****
*
610 A=SIN(AL-GA)/COS(AL-GA)
620 BX1=APX/Q+APO/2:BY1=H/2+Z
630 B=BY1-A*BX1
640 FOR F=0 TO BX1
650 Y=A*F+B
660 TRAP 680
665 IF Y<=16 THEN 680
670 PLOT F*Q,-Y+191
680 NEXT F
690 TRAP 40000
780 FOR PROMIEN=1 TO 5
790 READ COL,N2
792 LL(PROMIEN)=N2
795 COLOR COL
1000 REM === PROMIEN W PRYZMACIE
===
1010 O1=N1*SIN(AL)/N2
1020 BETA=ATN(O1/SQR(1-O1^2))
1025 PR(PROMIEN,1)=BETA
1030 A=SIN(BETA-GA)/COS(BETA-GA)
1040 B=BY1-A*BX1
1050 A1=SIN(90+GA)/COS(90+GA)
1060 B1=Z-(A1*(CPX/Q))
1070 CY1=(A1*B-B1*A)/(A1-A)
1080 CX1=(CY1-B1)/A1
1110 A11=(CY1-BY1)/(CX1-BX1)
1120 B11=BY1-A11*BX1
1130 FOR F=BX1 TO CX1
1145 Y=A*F+B
1147 IF Y<=16 THEN POL=10:GOTO 3
000
1150 TRAP 1170
1160 PLOT F*Q,-Y+191
1170 NEXT F
1180 TRAP 40000
1300 REM === PROMIEN ZA PRYZMATE
M ===
1310 BETAB=ABS(BETA-GAMA)
1315 PR(PROMIEN,2)=BETAB
1320 O3=N1/N2
1330 GR=ATN(O3/SQR(1-O3^2))
1340 IF BETAB>=GR THEN 1500
1350 O2=N2*SIN(BETAB)/N1
1360 DEL=ATN(O2/SQR(1-O2^2))
1365 PR(PROMIEN,3)=DEL
1370 IF BETA-GAMA>=0 THEN GOSUB
2240
1375 IF BETA-GAMA<0 THEN GOSUB 2
230
1380 B=CY1-A*CX1
1390 FOR F=CX1 TO 79/Q
1400 Y=A*F+B
1405 IF Y<=Z THEN 1430
1410 TRAP 1430
1420 PLOT F*Q,-Y+191
1430 NEXT F
1440 TRAP 40000
1499 GOTO 2000
1500 REM ** ODBICIE W PRYZMACIE
**
1510 A=SIN(GA+BETAB)/COS(GA+BETA
B)
1520 B=CY1-A*CX1
1521 PR(PROMIEN,3)=-10
1522 IF BETAB<90-GA THEN GOSUB 2
100
1524 IF BETAB>90-GA THEN GOSUB 2
200
1526 IF BETAB=90-GA THEN 2300
1530 FOR F=CX1 TO T STEP TT
1540 Y=A*F+B
1545 IF Y<=16 THEN POP :GOTO 240
0
1550 PLOT F*Q,-Y+191
1560 NEXT F
2000 GOTO 3000
2100 T=APX/Q:TT=-0.5:RETURN
2200 T=CPX/Q:TT=0.2:RETURN
2230 A=SIN(GA-DEL)/COS(GA-DEL)
2235 RETURN
2240 A=SIN(GA+DEL)/COS(GA+DEL)
2245 RETURN
2300 DRAWTO CX1*Q,191-Z:GOTO 300
0
2400 PLOT CX1*Q,-CY1+191
2410 DRAWTO F*Q,191-Z
3000 NEXT PROMIEN
3010 SOUND 1,80,10,15
3020 FOR M=1 TO 50:NEXT M
3030 SOUND 1,0,0,0
3040 IF PEEK(53279)=5 THEN MX=1
GOTO 6000
3050 GOTO 3040
4990 REM ** CROWN LEKKI **
5000 DATA 4,1.5118
5010 DATA 15,1.5153
5020 DATA 12,1.5186
5030 DATA 8, 1.5267
5040 DATA 10,1.5312
5090 REM ** FLINT LEKKI **
5100 DATA 4,1.6020
5110 DATA 15,1.6085
5120 DATA 12,1.6145
5130 DATA 8,1.6307
5140 DATA 10,1.6404
```

```
5190 REM ** FLINT CIEZKI **
5200 DATA 4,1.7406
5210 DATA 15,1.7515
5220 DATA 12,1.7685
5230 DATA 8,1.7922
5240 DATA 10,1.8045
6000 REM ** ** ** ** ** ** ** **
 **
6010 GRAPHICS 8:COLOR 1:POKE 710
,12
6015 POKE 709,0:POKE 752,6
6020 XL=58:HL=100:ZL=20
6030 PLOT ZL,ZL+HL:DRAWTO ZL+XL,
ZL
6040 DRAWTO ZL+2*XL,ZL+HL
6050 DRAWTO ZL,ZL+HL
6060 PLOT 0,85:DRAWTO ZL+XL/2,ZL
+HL/2
6070 DRAWTO 49+55,70-4
6080 DRAWTO 49+55+50+ZL,70-4+14
6090 COL=-1
6100 AT=-0.577:QT=130:WT=1:ET=78
6110 GOSUB 8000
6120 AT=0.5:QT=150:WT=25:ET=2
6130 GOSUB 8000
6140 TX=ZL+XL:TY=ZL:I1=332:I2=39
0
6150 GOSUB 8500
6155 PX=8:PY=10:V$="GAMMA":GOSUB
 8700
6160 TX=ZL+XL/2:TY=ZL+HL/2
6170 I1=245:I2=292:GOSUB 8500
6180 PX=0:PY=65:V$="ALFA":GOSUB
8700
6190 I1=57:I2=95:GOSUB 8500
6200 PX=5:PY=80:V$="BETA":GOSUB
8700
6210 TX=49+55:TY=70-4
6220 I1=272:I2=300:GOSUB 8500
6230 PX=10:PY=80:V$="BETAB":GOSU
B 8700
6240 I1=82:I2=117:GOSUB 8500
6250 PX=16:PY=59:V$="DELTA":GOSU
B 8700
6255 PX=9:PY=100:V$="N2":GOSUB 8
700
6260 PX=3:PY=30:V$="N1=1":GOSUB
8700
6270 PX=25:PY=5
6272 IF SK=0 THEN V$="CROWN LEKK
I"
6274 IF SK=1 THEN V$="FLINT LEKK
I"
6276 IF SK=2 THEN V$="FLINT CIEZ
KI"
6278 FOR I=3 TO 14
6280 PLOT 197,I:DRAWTO 202+LEN(V
$)*8,I
6282 NEXT I
6284 GOSUB 8700
6295 REM +*=-+*=-+*=-+*=-+*=-+*=
-+*=-
6300 PX=24:PY=20
6305 V$="WYBIERZ KOLOR":GOSUB 87
00
6310 PX=25:PY=30
6315 V$="DLA KTOREGO":GOSUB 8700
6320 PX=24:PY=40
6325 V$="CHCESZ UZYSKAC":GOSUB 8
700
6330 PX=28:PY=50
6335 V$="DANE":GOSUB 8700
6340 PX=25:PY=70
6345 V$="1-CZERWONY":GOSUB 8700
6350 PX=25:PY=80
6355 V$="2-ZOLTY":GOSUB 8700
6360 PX=25:PY=90
6365 V$="3-ZIELONY":GOSUB 8700
6370 PX=25:PY=100
6375 V$="4-NIEBIESKI":GOSUB 8700
6380 PX=25:PY=110
6385 V$="5-FIOLETOWY":GOSUB 8700
6389 POKE 764,255:GOSUB 9000
6390 OPEN #1,4,0,"K":GET #1,I
6395 CLOSE #1:II=I-48:COLOR 0
6396 IF II>5 OR II<1 THEN 6390
6400 FOR I=20 TO 118
6405 PLOT 190,I:DRAWTO 305,I
6410 NEXT I
6415 PX=23:PY=30
6420 V$="GAMMA=":GOSUB 8700
6425 PX=29:PY=30:V$=STR$(GAMA)
6430 GOSUB 8700
6435 PX=23:PY=40:V$="ALFA ="
6440 GOSUB 8700
6445 PX=29:PY=40:V$=STR$(AL)
6450 GOSUB 8700
6455 PX=23:PY=50:V$="BETA ="
6460 GOSUB 8700
6465 PX=29:V$=STR$(PR(II,1))
6470 GOSUB 8700:IF POL=10 THEN 6
520
6475 PX=23:PY=60:V$="BETAB="
6480 GOSUB 8700
6485 PX=29:V$=STR$(PR(II,2))
6490 GOSUB 8700
6491 IF PR(II,3)=-10 THEN 6520
6495 PX=23:PY=70:V$="DELTA="
6500 GOSUB 8700
6505 PX=29:V$=STR$(PR(II,3))
6510 GOSUB 8700
6520 PX=23:PY=85:V$="WSP. ZALAMA
NIA"
6525 GOSUB 8700
6530 PX=25:PY=95:V$="PRYZMATU :"
6535 GOSUB 8700
6540 PX=25:PY=105:V$="N2="
6545 GOSUB 8700
6550 PX=28:V$=STR$(LL(II))
6555 GOSUB 8700:GOSUB 9000
6560 IF PR(II,3)=-10 THEN GOSUB
8900
6615 ? " SELECT  - Powrot do MEN
U":?
6620 ? " START   - Powrot do PRO
GRAMU"
6625 SA=PEEK(53279)
6630 IF SA=6 THEN 5
6635 IF SA=5 THEN 8800
6640 GOTO 6625
7000 END
8000 REM ===== ==== ====== ====
==
8010 FOR F=WT TO QT STEP 10
8013 COLOR 1
8020 FOR F1=1 TO 5
8030 YT=AT*(F+F1)+ET
8040 PLOT F+F1,ZL+HL-YT
8050 NEXT F1
8060 NEXT F
8070 RETURN
8500 REM ++ ++ ++ ++ ++ ++ ++ ++
++
8505 DEG
8510 FOR I=I1 TO I2 STEP 5
8520 PLOT TX+15*SIN(I),TY+15*COS
(I)
8530 NEXT I
8540 RETURN
8700 REM == == == == == == == ==
==
8710 DL=PEEK(560)+PEEK(561)*256
8720 D1=PEEK(DL+4)+PEEK(DL+5)*25
6
8730 FOR UU=1 TO LEN(V$)
8740 D2=57344+((ASC(V$(UU,UU))-3
2)*8)
8750 D3=D1+PY*40+PX+UU-1
8760 FOR UUU=0 TO 7
8770 POKE D3+UUU*40,PEEK(D2+UUU)
8780 NEXT UUU:NEXT UU
8790 RETURN
8800 REM ***********************
***
8810 COLOR 0:? :? :? :?
8815 FOR I=30 TO 115
8820 PLOT 185,I:DRAWTO 319,I
8825 NEXT I
8830 IF PR(II,3)=-10 THEN GOSUB
8950
8840 GOTO 6300
8899 REM ====================
=
8900 PX=4:PY=135:COLOR 1:GOSUB 8
950
8905 V$="CALKOWITE ODBICIE W PRY
ZMACIE !!"
8910 GOSUB 8700
8930 RETURN
8950 FOR I=130 TO 147
8960 PLOT 26,I:DRAWTO 293,I
8970 NEXT I
8980 RETURN
9000 FOR I=1 TO 7
9005 SOUND 0,20,10,15:NEXT I
9010 SOUND 0,0,0,0:RETURN
```

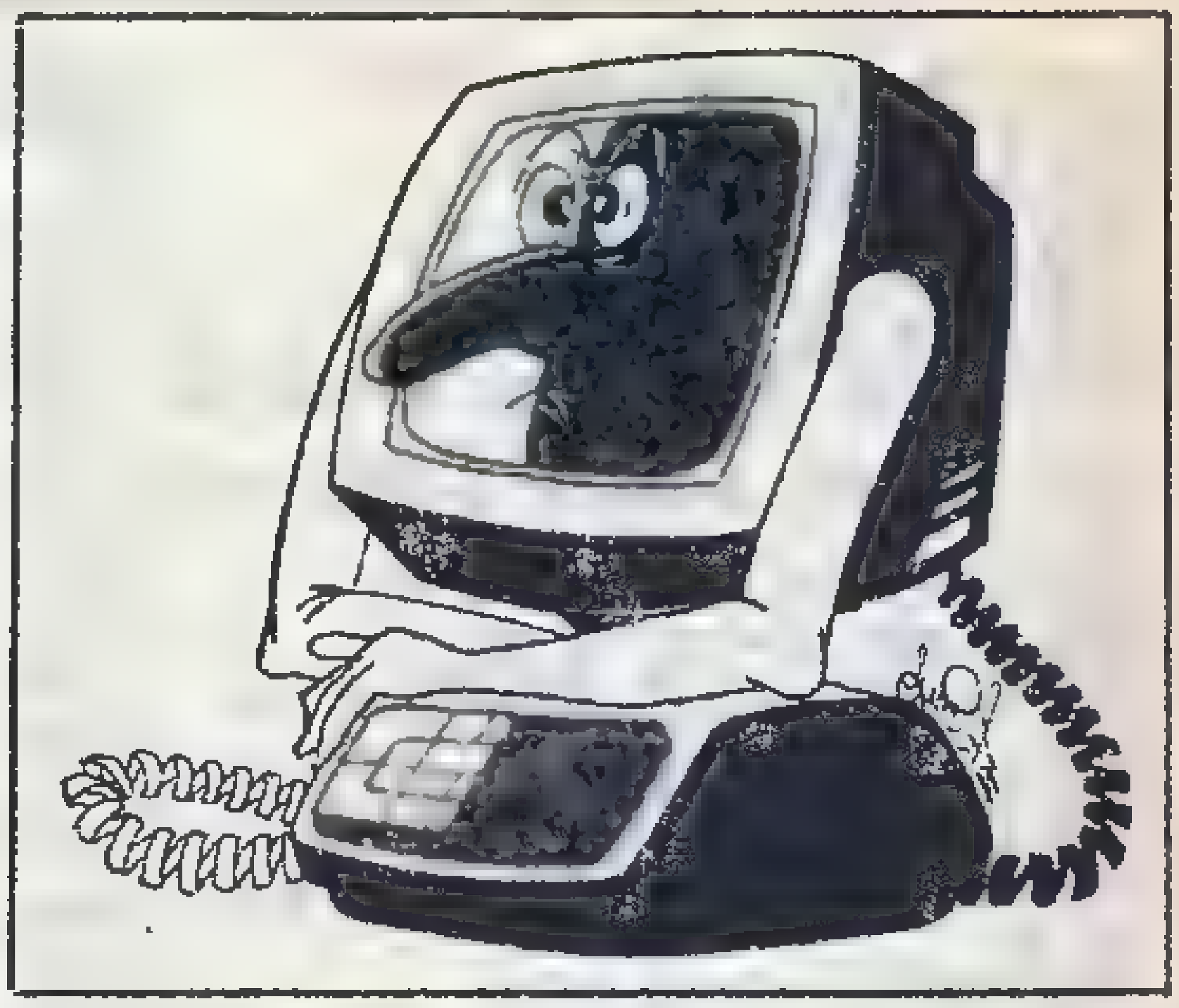
Siedzibą klubu **„MERIZAP"** jest pracownia techniki cyfrowej w Zespole Szkół Technicznych w Ostrowie Wielkopolskim. W rozmowie z Mariuszem Rudowiczem dowiedzieliśmy się, że klub posiada bogatą bibliotekę programów edukacyjnych, narzędziowych i gier.

Członkowie sami opracowują programy dydaktyczne z fizyki i elektroniki. W najbliższej przyszłości za pomocą „Meritum" mają zamiar skomputeryzować również szkolną bibliotekę.

15



SPECTRUM

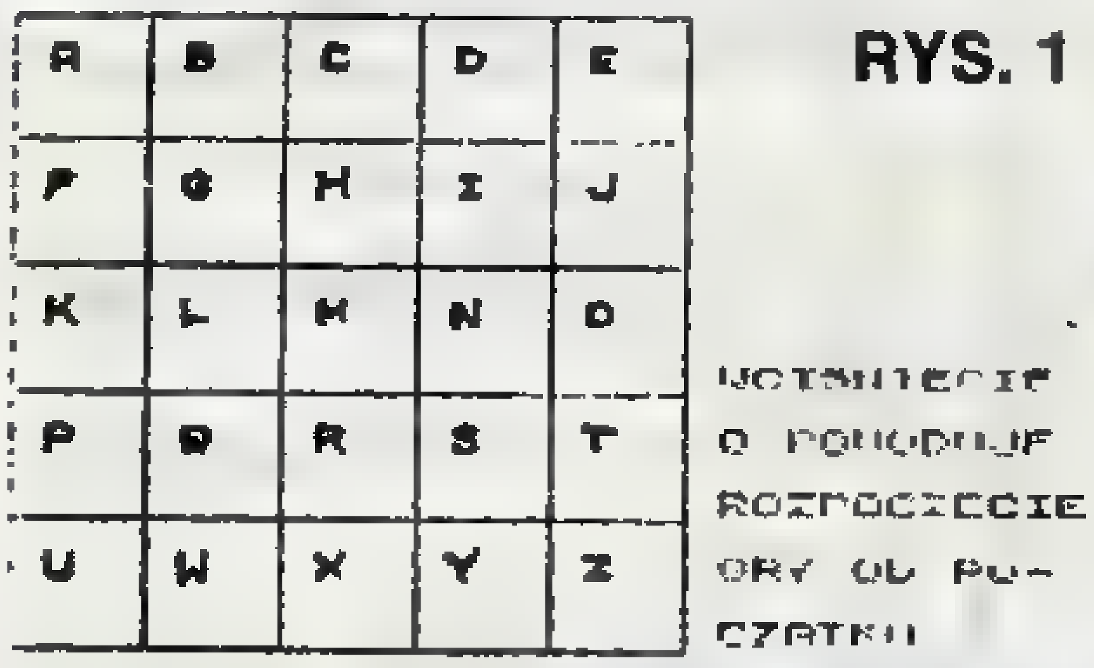


PROGRAM 5

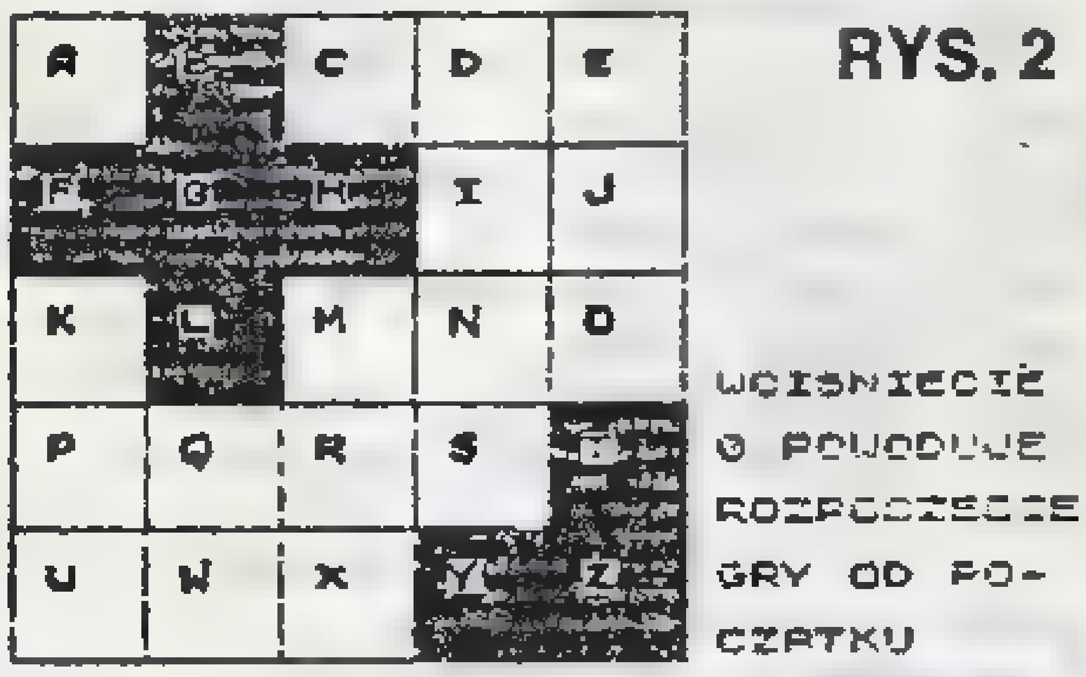
RUBIK

Prezentowana gra polega na zamianie koloru przedstawionej na rys. 1 planszy na kolor przeciwny (negatyw). Po wczytaniu i uruchomieniu gry, na ekranie ukazuje się plansza (rys. 1). Wciśnięcie dowolnego klawisza, odpowiadającego jednej z widocznych na planszy liter, powoduje zaczernienie pola z wybraną literą i pól sąsiednich „na krzyż". Np. wciśnięcie klawisza „G" i „Z" sprawia, że plansza przybiera wygląd jak na rys. 2. Jeżeli teraz zostanie naciśnięty klawisz „H", a potem „O", to plansza ma wygląd, jak na rys. 3. Widać stąd, że odpowiednie pola dotychczas białe, zostaną zaczernione, a pola czarne staną się białe.

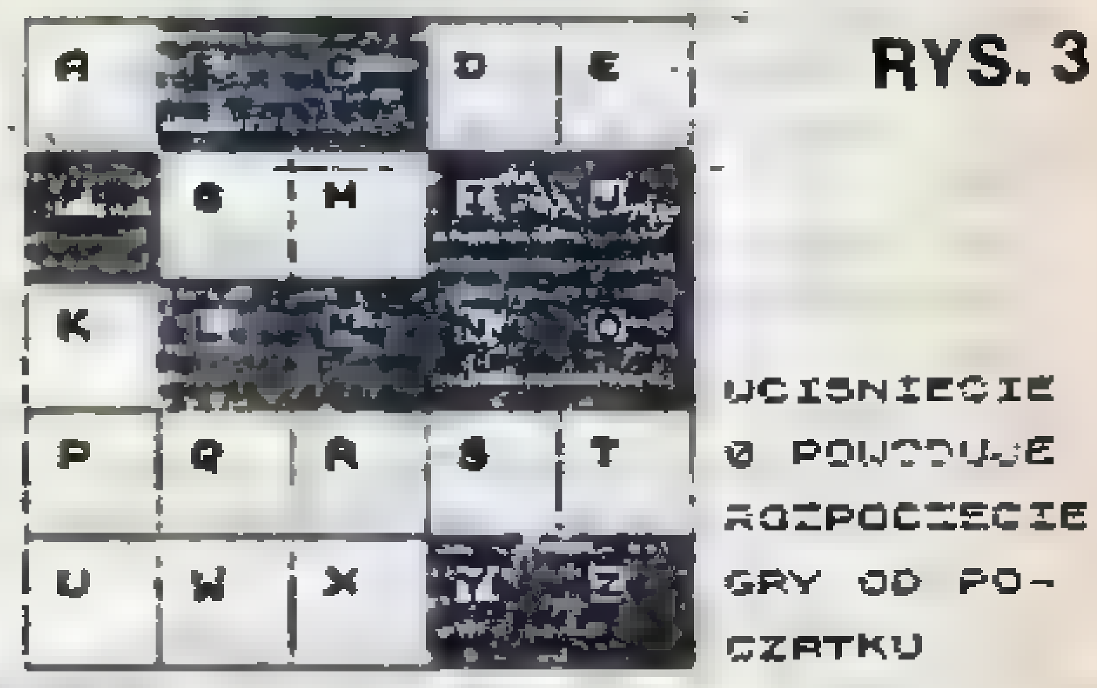
Celem gry jest zamiana koloru całej planszy — jak na rys. 4. Oczywiście można dążyć do innych efektów, np. do otrzymania szachownicy, jak na rys. 5.



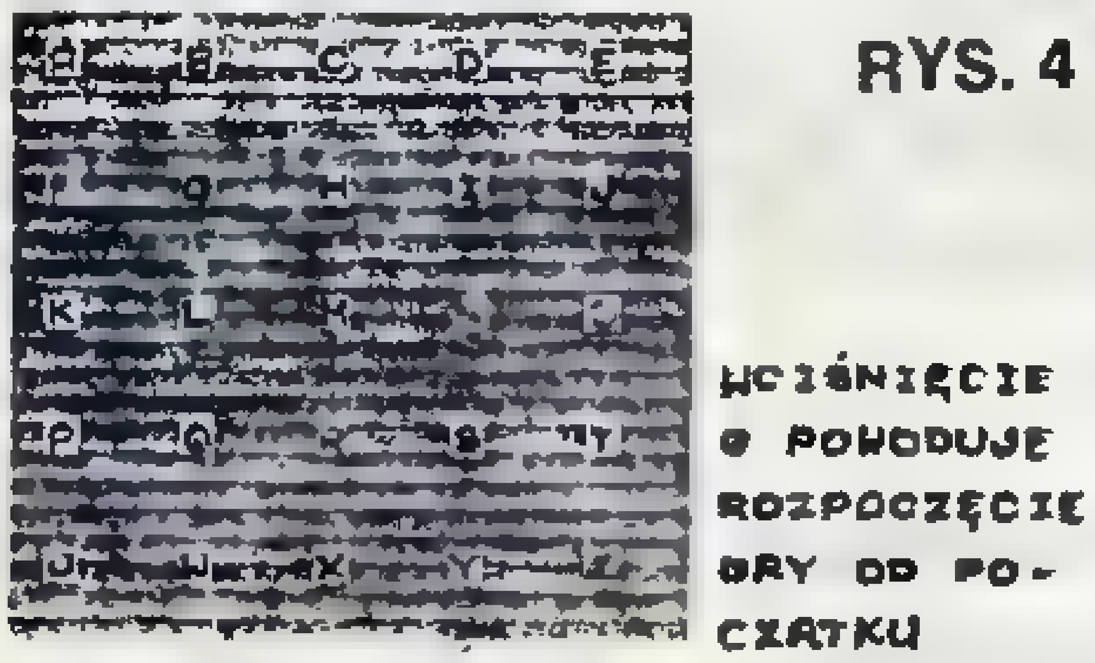
RYS. 1



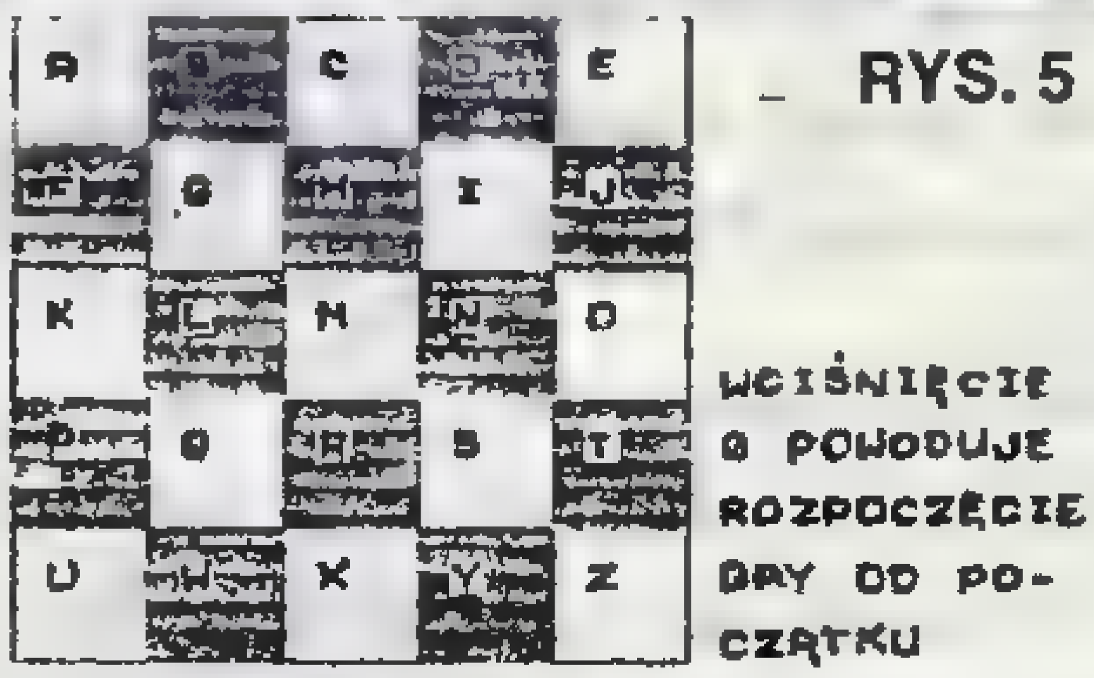
RYS. 2



RYS. 3



RYS. 4



RYS. 5

```
  1 REM Gra "SPECTRUM-RUBIK"          PIOTR ININSKI 1986.11.14
 10 BORDER 0: PAPER 0: INK 6: CLS
100 GO SUB 1000
200 IF INKEY$="a" OR INKEY$="A" THEN  GO SUB 3000: GO SUB 3010: GO SUB 3050: GO
TO 100
210 IF INKEY$="b" OR INKEY$="B" THEN  GO SUB 3000: GO SUB 3010: GO SUB 3020: GO
SUB 3060: GO TO 100
220 IF INKEY$="c" OR INKEY$="C" THEN  GO SUB 3010: GO SUB 3020: GO SUB 3030: GO
SUB 3070: GO TO 100
230 IF INKEY$="d" OR INKEY$="D" THEN  GO SUB 3020: GO SUB 3030: GO SUB 3040: GO
SUB 3080: GO TO 100
240 IF INKEY$="e" OR INKEY$="E" THEN  GO SUB 3030: GO SUB 3040: GO SUB 3090: GO
TO 100
250 IF INKEY$="f" OR INKEY$="F" THEN  GO SUB 3000: GO SUB 3050: GO SUB 3060: GO
SUB 3100: GO TO 100
260 IF INKEY$="g" OR INKEY$="G" THEN  GO SUB 3010: GO SUB 3050: GO SUB 3060: GO
SUB 3070: GO SUB 3110: GO TO 100
270 IF INKEY$="h" OR INKEY$="H" THEN  GO SUB 3020: GO SUB 3060: GO SUB 3070: GO
SUB 3080: GO SUB 3120: GO TO 100
280 IF INKEY$="i" OR INKEY$="I" THEN  GO SUB 3030: GO SUB 3070: GO SUB 3080: GO
SUB 3090: GO SUB 3130: GO TO 100
290 IF INKEY$="j" OR INKEY$="J" THEN  GO SUB 3040: GO SUB 3080: GO SUB 3090: GO
SUB 3140: GO TO 100
300 IF INKEY$="k" OR INKEY$="K" THEN  GO SUB 3050: GO SUB 3100: GO SUB 3110: GO
```

```
SUB 3150: GO TO 100
 310 IF INKEY$="l" OR INKEY$="L" THEN  GO SUB 3060: GO SUB 3100: GO SUB 3110: GO
SUB 3120: GO SUB 3160: GO TO 100
 320 IF INKEY$="m" OR INKEY$="M" THEN  GO SUB 3070: GO SUB 3110: GO SUB 3120: GO
SUB 3130: GO SUB 3170: GO TO 100
 330 IF INKEY$="n" OR INKEY$="N" THEN  GO SUB 3080: GO SUB 3120: GO SUB 3130: GO
SUB 3140: GO SUB 3180: GO TO 100
 340 IF INKEY$="o" OR INKEY$="O" THEN  GO SUB 3090: GO SUB 3130: GO SUB 3140: GO
SUB 3190: GO TO 100
 350 IF INKEY$="p" OR INKEY$="P" THEN  GO SUB 3100: GO SUB 3150: GO SUB 3160: GO
SUB 3200: GO TO 100
 360 IF INKEY$="q" OR INKEY$="Q" THEN  GO SUB 3110: GO SUB 3150: GO SUB 3160: GO
SUB 3170: GO SUB 3210: GO TO 100
 370 IF INKEY$="r" OR INKEY$="R" THEN  GO SUB 3120: GO SUB 3160: GO SUB 3170: GO
SUB 3180: GO SUB 3220: GO TO 100
 380 IF INKEY$="s" OR INKEY$="S" THEN  GO SUB 3130: GO SUB 3170: GO SUB 3180: GO
SUB 3190: GO SUB 3230: GO TO 100
 390 IF INKEY$="t" OR INKEY$="T" THEN  GO SUB 3140: GO SUB 3180: GO SUB 3190: GO
SUB 3240: GO TO 100
 400 IF INKEY$="u" OR INKEY$="U" THEN  GO SUB 3150: GO SUB 3200: GO SUB 3210: GO
 TO 100
 410 IF INKEY$="w" OR INKEY$="W" THEN  GO SUB 3160: GO SUB 3200: GO SUB 3210: GO
SUB 3220: GO TO 100
 420 IF INKEY$="x" OR INKEY$="X" THEN  GO SUB 3170: GO SUB 3210: GO SUB 3220: GO
SUB 3230: GO TO 100
 430 IF INKEY$="y" OR INKEY$="Y" THEN  GO SUB 3180: GO SUB 3220: GO SUB 3230: GO
SUB 3240: GO TO 100
 440 IF INKEY$="z" OR INKEY$="Z" THEN  GO SUB 3190: GO SUB 3230: GO SUB 3240: GO
 TO 100
 450 IF INKEY$="0" THEN  RUN
 460 GO TO 200
1000 RESTORE : FOR i=0 TO 5: PLOT i*32,175: DRAW 0,-160: NEXT i
1010 FOR i=0 TO 5: PLOT 0,i*32+15: DRAW 160,0: NEXT i
1020 FOR i=1 TO 17 STEP 4: FOR j=1 TO 17 STEP 4: READ a$: PRINT AT i,j;a$: NEXT
j: NEXT i
1030 PRINT AT 11,21;"WCISNIECIE ";AT 13,21;"0 POWODUJE";AT 15,21;"ROZPOCZECIE";A
T 17,21;"GRY OD PO-";AT 19,21;"CZATKU"
1200 RETURN
3000 LET a=0: LET b=3: LET c=0: LET d=3: GO SUB 6000: RETURN
3010 LET a=4: LET b=7: LET c=0: LET d=3: GO SUB 6000: RETURN
3020 LET a=8: LET b=11: LET c=0: LET d=3: GO SUB 6000: RETURN
3030 LET a=12: LET b=15: LET c=0: LET d=3: GO SUB 6000: RETURN
3040 LET a=16: LET b=19: LET c=0: LET d=3: GO SUB 6000: RETURN
3050 LET a=0: LET b=3: LET c=4: LET d=7: GO SUB 6000: RETURN
3060 LET a=4: LET b=7: LET c=4: LET d=7: GO SUB 6000: RETURN
3070 LET a=8: LET b=11: LET c=4: LET d=7: GO SUB 6000: RETURN
3080 LET a=12: LET b=15: LET c=4: LET d=7: GO SUB 6000: RETURN
3090 LET a=16: LET b=19: LET c=4: LET d=7: GO SUB 6000: RETURN
3100 LET a=0: LET b=3: LET c=8: LET d=11: GO SUB 6000: RETURN
3110 LET a=4: LET b=7: LET c=8: LET d=11: GO SUB 6000: RETURN
3120 LET a=8: LET b=11: LET c=8: LET d=11: GO SUB 6000: RETURN
3130 LET a=12: LET b=15: LET c=8: LET d=11: GO SUB 6000: RETURN
3140 LET a=16: LET b=19: LET c=8: LET d=11: GO SUB 6000: RETURN
3150 LET a=0: LET b=3: LET c=12: LET d=15: GO SUB 6000: RETURN
3160 LET a=4: LET b=7: LET c=12: LET d=15: GO SUB 6000: RETURN
3170 LET a=8: LET b=11: LET c=12: LET d=15: GO SUB 6000: RETURN
3180 LET a=12: LET b=15: LET c=12: LET d=15: GO SUB 6000: RETURN
3190 LET a=16: LET b=19: LET c=12: LET d=15: GO SUB 6000: RETURN
3200 LET a=0: LET b=3: LET c=16: LET d=19: GO SUB 6000: RETURN
3210 LET a=4: LET b=7: LET c=16: LET d=19: GO SUB 6000: RETURN
3220 LET a=8: LET b=11: LET c=16: LET d=19: GO SUB 6000: RETURN
3230 LET a=12: LET b=15: LET c=16: LET d=19: GO SUB 6000: RETURN
3240 LET a=16: LET b=19: LET c=16: LET d=19: GO SUB 6000: RETURN
6000 FOR j=a TO b: FOR i=c TO d: PRINT  OVER 1;AT i,j;"■": NEXT i: NEXT j: RETU
RN
9000 DATA "A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q";"R
","S","T","U","W","X","Y","Z"
```

AMSTRAD

PROGRAM 6

Grafika „Wieża Hanoi"

Zamieszczony w poprzednim zeszycie programów komputerowych program WIEŻA HANOI wyświetlał wyniki w postaci tekstu na ekranie monitora. Wykorzystując podstawowe procedury tego programu realizujące przełożenie krążków na prętach oraz przeniesienie pojedynczego krążka można rozbudować program, który w sposób graficzny wyświetla wyniki.

W programie zastosowano elementy pakietu graficznego systemu TURBO PASCAL zaimplementowanego na komputerze AMSTRAD CPC6128.

Warto je przechować w bibliotece programów PASCALA. Komentarze zawierające znaki „+" ograniczają niezmienione fragmenty programu WIEŻA HANOI.

Istotne jest to, dla tych, którzy zechcą zmodyfikować posiadany program. Dodatkowo w programie wprowadzono podkład muzyczny dla grafiki kolorowej.



Andrzej

```
PROGRAM graficznawiezahanoi;
{$a-}
{ ********************************* }
{     - GRAFIKA TURBO PASCALA       }
{ ********************************* }

const CPM3 = true;

const tagflag:boolean = false;
      tagaddr:integer = 0;
      indscr :integer = 0;
      fct             = 0.017453292520;
      xerror :real    = 0.0;
      yerror :real    = 0.0;
      getmode:byte    = 2;
      getgrafmode:byte= 0;

var   itg:integer;
      btg:byte;
      rd:array[1..13] of byte;
      re:real;

procedure InitGrafik;
begin
   write(#$1b,'0',#$1b,'y');
end;

procedure leaveGrafik;
begin
   write(#$1b,'1',#$1b,'x');
end;

procedure gotoxy(x,y:integer);
begin
   write(#$1b,'Y',chr(y+31),chr(x+31));
end;

procedure sound(g:byte;h,i:integer;j,k,l,m:byte);
var sd:array[0..8] of byte;
begin
  sd[0] := g;
  move(h,sd[3],2);
  move(i,sd[7],2);
  sd[1] := k;
  sd[2] := l;
  sd[5] := m;
  sd[6] := j;
  inline($21/sd/$cd/$5a/$fc/$aa/$bc);
end;

procedure soundreset;
begin
  inline($cd/$5a/$fc/$a7/$bc);
end;

procedure drawr(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/$F9/$BB);
end;

procedure grafmove(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/$C0/$BB);
end;

procedure grafmover(x,y:integer):
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/$c3/$BB);
end;

procedure plot(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/$EA/$BB);
end;

procedure grafwindow(x1,y1,x2,y2:integer);
begin
  inline($2A/x2/$EB/$2A/x1/$cd/$5a/$fc/$CF/$BB/
         $2A/y2/$EB/$2A/y1/$cd/$5a/$fc/$D2/$BB);
end;

procedure grafpen(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$DE/$BB);
end;

procedure grafpaper(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$E4/$BB);
end;

procedure clg(fs:byte);
begin
  inline($cd/$5a/$fc/$DB/$BB);
  grafpaper(fs);
end;

procedure ink(fs.f1.f2:byte);
begin
  inline($3a/f1/$47/$3a/f2/$4f/$3a/fs/$cd/$5a/$fc/
         $32/$bc);
end;

procedure paper(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$96/$BB);
end;

procedure pen(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$90/$BB);
end;

procedure mode(md:byte);
begin
  if md in [0..2] then
  begin
    getmode := md;
    inline($3a/md/$cd/$5a/$fc/$0e/$bc);
  end;
end;

procedure border(v1,v2:byte);
begin
  inline($3a/v1/$47/$3a/v2/$4f/$cd/$5a/$fc/$38/$bc);
end;

procedure grafout(ch:char);
begin
  inline($3A/ch/$cd/$5a/$fc/$FC/$BB);
end;

procedure tag;
begin
  if not tagflag then
  begin
    tagaddr := conoutptr;
    tagflag := true;
    conoutptr := addr(grafout);
  end;
end;

procedure tagoff;
begin
  if tagflag then
  begin
    conoutptr := tagaddr;
    tagflag := false;
  end;
end;

{ ******************************************** }
{  DEKLARACJA TYPOW                            }
{  DEFINICJA STALYCH I ZMIENNYCH               }
{ ******************************************** }

  CONST
    {++++++++++++++++++++++++++++++++++++}
    maxilkr=5; (maksymalna ilosc krazkow)
    {++++++++++++++++++++++++++++++++++++}

    wysk=50; (wysokosc krazka)
    kolw=26; (kolor wiezy-bialy)
    kolt=0;  (kolor tla-czarny)
    dpozw=150; (dlugosc pozioma wiezy)
    wysw=-270;  (wysokosc wiezy)
    kolzn=16;  (kolor znaku-rozowy)
    pxzn=-16;  (pozycjonowarie znaku)
    pyzn=-25;
    kolob=0; (kolor obrzeza)

  TYPE
    {+++++++++++++++++++++++++++++++++++++++++}
    znakp='a'..'c'; (nazwa preta)
    wyswiez=1..maxilkr; (maksymalna wysokosc wiezy
                         mierzona iloscia krazkow)
    opiswiez=ARRAY [znakp] OF  (opis wiez)
        RECORD
          w:wyswiez; (wskaznik wierzcholka wiezy)
          wieza:ARRAY (wyswiez] OF wyswiez
        END;
    {+++++++++++++++++++++++++++++++++++++++++}

    gopiswiez=ARRAY [znakp] OF
            RECORD
              wspx:integer; (wspolrzedne poczatku)
              wspy:integer; (wiezy)
              wwx:integer; (wspolrzedne wierzcholka)
              wwy:integer;
              parkr:ARRAY [wyswiez] OF   (poziom)
                   RECORD
                     wspx:ARRAY[wyswiez] OF integer;
                       (wspolrzedne poczatku krazka)
                     wspy:integer;
                   END
            END;
VAR
```

```
   {++++++++++++++++++++++++++++++++++++++++++++}
   ilkr:integer; {zadana ilosc krazkow}
   start,  {pret, z ktorego przesuwamy krazki}
   cel,    {pret, na ktory przesuwamy krazki }
   posr:znakp; {pret - magazyn pomocniczy    }
   opw:opiswiez;
   {++++++++++++++++++++++++++++++++++++++++++++}

   x,obraz:char;
   dlkraz:ARRAY [wyswiez] OF integer; {dlugosci krazka}
   kolkr:ARRAY [wyswiez] OF integer;  {kolory krazka}
   ogw:gopiswiez; {graficzny opis wiez}

{ ******************************************** }
{  PROCEDURY INICJOWANIA DANYCH                 }
{ ******************************************** }

  {++++++++++++++++++++++++++++++++++++++++++++}
  PROCEDURE ustawstpocz;
    {procedura ustawiajaca poczatkowy stan wiez}
   VAR
     s:znakp; {zmienne robocze}
     i:wyswiez;
   BEGIN
     WITH opw['a'] DO
      BEGIN
       w:=maxilkr-ilkr+1;
       FOR i:=w TO maxilkr DO
        wieza[i]:=i-w+1
      END;
     opw['b'].w:=0;
     opw['c'].w:=0
   END;
  {++++++++++++++++++++++++++++++++++++++++++++}

  PROCEDURE ustawdgraf;
  {procedura ustawiajaca poczatkowy stan}
  {danych graficznych}
    VAR
     s:znakp; {zmienne robocze}
     i:wyswiez;
     j:integar;
    BEGIN
     soundreset;
     grafwindow(0,10,640,400);
     ink(8,0,0);  {tekst}
     ink(9,26,26);  { tlo tekstu}

      FOR i:=1 TO maxilkr DO
          dlkraz[i]:=70+((i-1)*20);
      kolkr[1]:=15;   {pomarancz}
      kolkr[2]:=17;   {magneta}
      kolkr[3]:=6;    {czerwony}
      kolkr[4]:=1;    {niebieski}
      kolkr[5]:=21;   {zielony}

     WITH ogw['a'] DO
      BEGIN
       wspx:=31;
       wspy:=75;
       wwx:=106;
       wwy:=345;
       FOR i:=1 TO maxilkr DO  {poziom}
         BEGIN
           FOR j:=1 TO maxilkr DO    {numer krazka}
             parkr[i].wspx[j]:=71-((j-1)*10);
          parkr[i].wspy:=275-((i-1)*50)
         END;
      END;

     WITH ogw['b'] DO
      BEGIN
       wspx:=244;
       wspy:=75;
       wwx:=319;
       wwy:=345;
       FOR i:=1 TO maxilkr DO
         BEGIN
           FOR j:=1 TO maxilkr DO
             parkr[i].wspx[j]:=284-((j-1)*10);
           parkr[i].wspy:=275-((i-1)*50)
         END;
      END;
     WITH ogw['c'] DO
      BEGIN
       wspx:=457;
       wspy:=75;
       wwx:=532;
       wwy:=345;
       FOR i:=1 TO maxilkr DO
         BEGIN
          FOR j:=1 TO maxilkr DO
             parkr[i].wspx[j]:=497-((j-1)*10);
           parkr[i].wspy:=275-((i-1)*50)
         END;
      END;
   END;

{ ********************************** }
{ PROCEDURY  KRESLENIA   WIEZ        }
{ ********************************** }

PROCEDURE pusta(o:znakp);
{kreslenie wiezy bez krazkow}
 VAR zn:char;
   BEGIN
     ink(6,kolw,kolw);
     grafpen(6);  {pioro wiezy}
     WITH ogw[o] DO
      BEGIN
       sound(120,wspx,50,15,3,0,0);
       sound(120,wspy,50,15,3,0,0);
       sound(40,kolw*(1+random(150)),25,15,0,0,0);
       plot(wspx,wspy);
       drawr(dpozw,0);
       grafmove(wwx,wwy);
      END;
     drawr(0,wysw);

     CASE o OF
     'a':zn:='A';
     'b':zn:='B';
     'c':zn:='C';
     END;
     ink(7,kolzn,kolzn);
     tag;  grafpen(7);
     grafmover(pxzn,pyzn);
     grafout(zn);
     tagoff;
   END;

PROCEDURE niepusta(o:znakp);
{kreslenie krazkow jednej wiezy }
 VAR nrkr,i:wyswiez;
     j:integer;
   BEGIN
     WITH opw[o] DO
       WITH ogw[o] DO
         FOR i:=maxilkr DOWNTO w DO
           BEGIN
            nrkr:=wieza[i];
            ink(nrkr,kolkr[nrkr],kolkr[nrkr]);
                                  {pioro krazka}
            grafpen(nrkr);
            sound(100,parkr[i].wspy+200*i,7,12,0,0,0);
            sound(200,parkr[i].wspy+200*i,7,12,0,0,0);
            sound(240,nrkr*(1+random(150))+200*i,7,12,
                  0,0,0);
            FOR j:=0 TO 48 DO
             BEGIN
              sound(1,parkr[i].wspx[nrkr],6,10,0,0,0);
              sound(2,parkr[i].wspy+j,6,10,0,0,0);
              sound(4,j*nrkr*(1+random(150)),6,10,0,0,0);
              plot(parkr[i].wspx[nrkr],parkr[i].wspy+j);
              drawr(dlkraz[nrkr],0);
             END;
           END;
  END;

  PROCEDURE kreslenie;
  {procedura kreslaca stany wiez}
   VAR
    i:integer;
    s:znakp;  {zmienne robocze}
    BEGIN
      clg(0);
      FOR s:='a' TO 'c' DO
       WITH opw[s] DO
        BEGIN
         pusta(s);  {kreslenie wiezy bez krazkow}
         IF w>0 THEN niepusta(s);
        END;
      IF obraz='n' THEN
      FOR i:=1 TO 31000 DO
      s:=s; {opoznienie}
    END;

{ ********************************** }
{ PROCEDURY PRZENOSZENIA KRAZKOW     }
{ ********************************** }

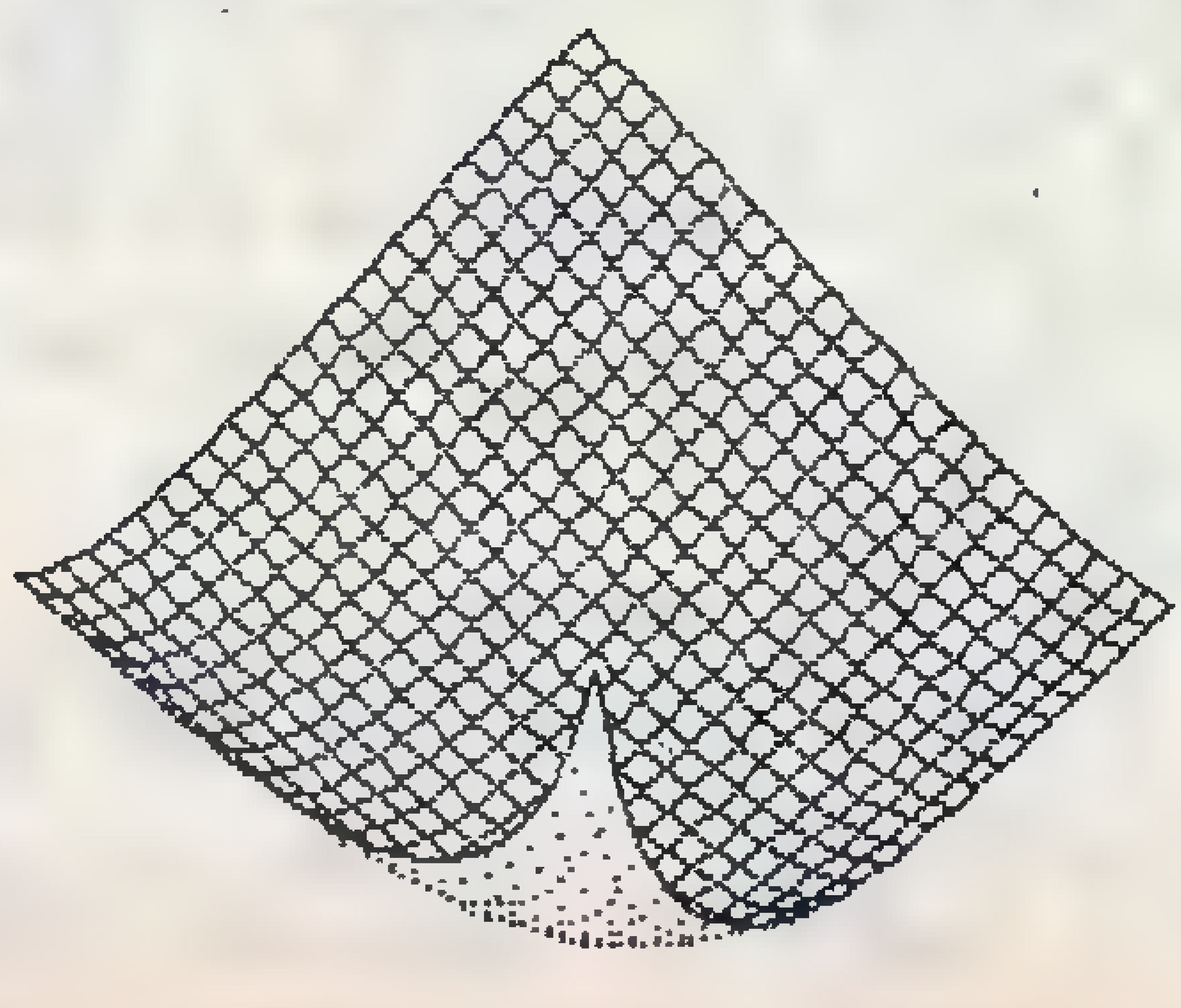
  PROCEDURE przen(start,cel:znakp);
  {zmodyfikowana procedura przenoszaca pojedynczy
   krazek z preta "start" na pret "cel"}
    BEGIN
        WITH opw[cel] DO
         BEGIN
          IF w=0 THEN  w:=maxilkr
          ELSE  w:=w-1;
          wieza[w]:=opw[start].wieza[opw[start].w]
         END;
        WITH opw[start] DO
         IF w=maxilkr THEN  w:=0
         ELSE  w:=w+1;
        kreslenie;
        IF obraz='t'  THEN
         BEGIN
           gotoxy(1,25);
           WRITE('Wcisnij klawisz');
           REPEAT UNTIL keypressed;
           READ(kbd,x);
         END;
    END;

  {++++++++++++++++++++++++++++++++++++++++++++++++}
  PROCEDURE przel(n:wyswiez; VAR start,posr,cel:znakp);
  {procedura przesuwajaca "n" krazkow z preta
      "start" na pret "cel" z wykorzystaniem preta
      "posr" jako magazynu}
    BEGIN
      IF n=1
       THEN
           przen(start,cel)
      ELSE
       BEGIN
        przel(n-1,start,cel,posr);
        przen(start,cel);
        przel(n-1,posr,start,cel)
       END
    END;
  {++++++++++++++++++++++++++++++++++++++++++++++++}

{ ********************************** }
{      MODUL STERUJACY               }
{ ********************************** }


 BEGIN
   REPEAT
    WRITELN('podaj ilosc krazkow <= ',maxilkr);
    READLN(ilkr);
    WRITELN('Czy chcesz akceptowac-wciskajac');
    WRITELN('dowolny klawisz-przelorzenie');
    WRITELN('kazdego krazka  (t/n)');
    READLN(obraz);
   UNTIL (ilkr>0) AND (ilkr<=maxilkr) AND
         ((obraz='t') OR (obraz='n'));
   clrscr;
    BEGIN
     ustawstpocz;
     ustawdgraf;
     initgrafik;
     mode(0);
     border(kolob,kolob);
     ink(0,kolt,kolt);
     grafpaper(0);
     paper(9);
     pen(8);
     kreslenie;
     gotoxy(1,25);
     WRITE('Poczatek-wcisnij kl.');
     REPEAT UNTIL keypressed;
     READ(kbd,x);
    END;
    start:='a';
    cel:='c';
    posr:='b';
    przel(ilkr,start,posr,cel);
    gotoxy(1,25);
    WRITE('Koniec-wcisn.klawisz');
    REPEAT UNTIL keypressed;
    READ(kbd,x);
    leavegrafik;
 END.
```

GRAFIKA



COMMODORE

PROGRAM 7

„Życie"

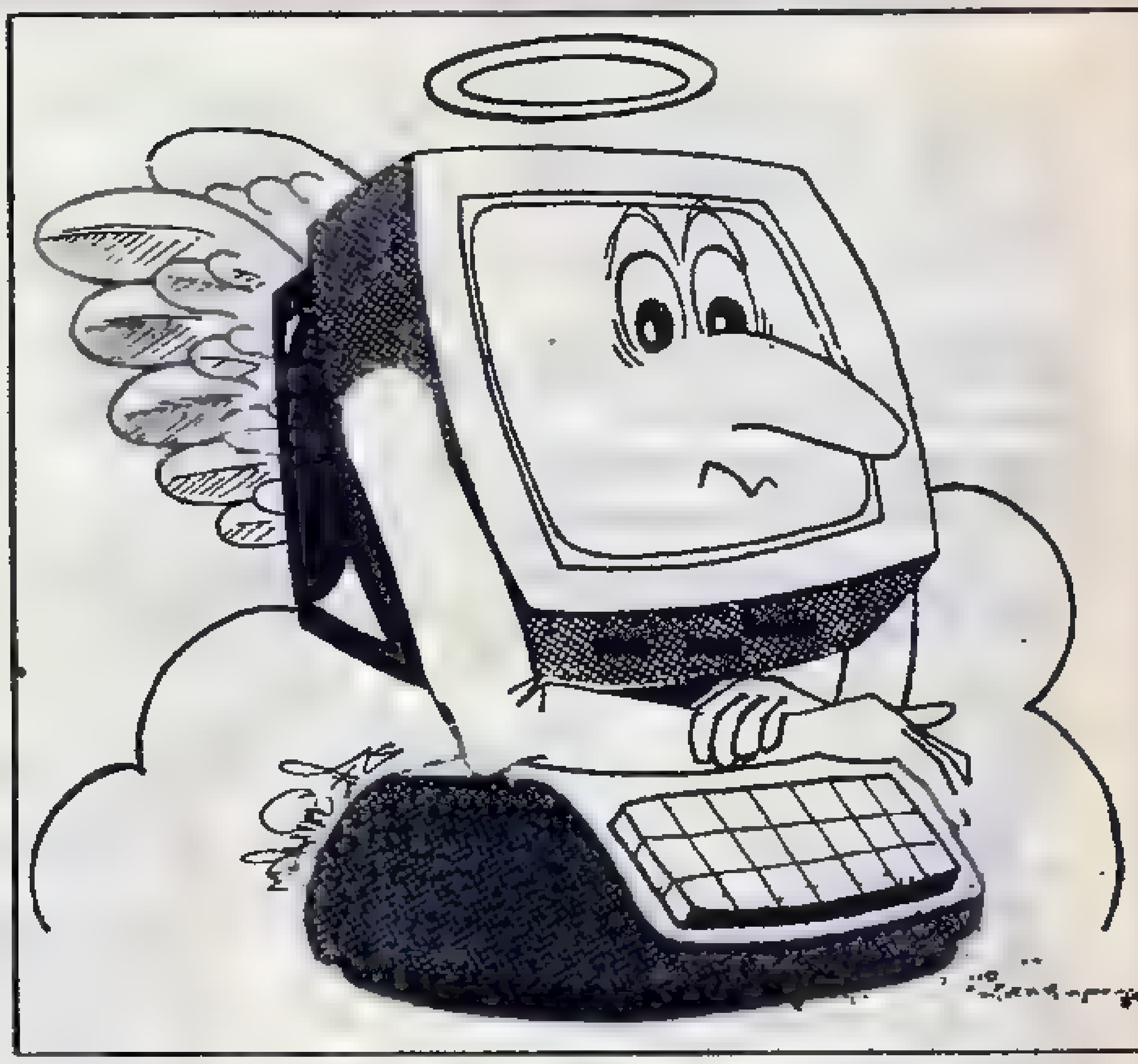
Gra symuluje powstawanie i zanikanie pokoleń organizmów w populacji. Oparta jest na trzech zasadach:

a) każdy organizm otoczony przez dwa lub trzy inne ocaleje,

b) cztery lub więcej organizmów sąsiednich lub ich brak powoduje zniszczenie organizmu na skutek przerostu populacji lub izolacji organizmu,

c) w pustej komórce otoczonej dokładnie trzema sąsiednimi organizmami powstaje nowe życie (nowy organizm).

Program został pierwotnie opracowany w języku BASIC, natomiast w niżej przedstawionym zastosowano w kilku miejscach język maszynowy. Dzięki temu gra przebiega około 100 razy szybciej. Prędkość przebiegu programu można zmienić przez zmianę końcowej wartości dla pętli określonej w linii 240. Działanie programu można zatrzymać w dowolnej chwili klawiszem SPACJA i ponownie uruchomić jego działanie naciskając RETURN. Usunięcie linii 240—260 powoduje zwiększenie szybkości działania programu, bez możliwości zatrzymania i ponownego uruchomienia w sposób wyżej opisany.



Dodanie linii: 230 SYS 49452, powoduje, że gra odbywa się na ekranie bez określonych granic — ekran jak gdyby zawija się z góry do dołu i z lewej strony na prawą. Aby grać w „Życie" należy program po prostu wpisać do pamięci komputera i uruchomić. Na czarnym ekranie pojawi się świecący jasno kursor w postaci piłki (kuli). Do zbudowania początkowego wzoru służą klawisze kursora (ustawianie piłki w żądanym miejscu) i klawisz mnożenia (załączenie komórki — powołanie do życia). Gdy wzór jest gotowy, naciśnięcie RETURN zapoczątkuje jego ewolucję. Naciśnięcie RUN/STOP przerywa działanie programu i kończy ewolucję wzoru. Gra może być zabawą graficzną, jak również ciekawą grą matematyczną.

Opracował: **Tadeusz CISEK**

```
READY.

5 PRINT"ZYCIE"
10 W=51968:C=55296:POKE 53280,0:POKE53281,0
20 IF PEEK (49152)<>169 OR PEEK (49580)<>96 THEN GOSUB 300
30 SYS 49397:K=0:S=0
40 GET A$:IF A$="" THEN A=PEEK(C+K):POKEC+K,NOT(A AND 1)AND (A OR 1
50 IF A$=CHR$(42)THEN S=1:D=1:GOSUB 150
60 IF A$=CHR$(13)THEN POKEC+K,S:GOTO210
70 IF A$=CHR$(29) THEN D=1:GOSUB  150
80 IF A$=CHR$(157) THEN D=-1:GOSUB 150
90 IF A$=CHR$(17) THEN D=40:GOSUB 150
100 IF A$=CHR$(145)THEN D=-40:GOSUB 150
110 IF A$=CHR$(32)THEN S=0:D=1:GOSUB150
120 GOTO 40
150 POKE C+K,S:K=K+D:IF K>999 OR K<0 THEN K=K-D
160 S=PEEK(C+K):RETURN
210 SYS 49250
220 SYS 49152
240 FOR I=0 TO 100:NEXT
250 GET A$:IF A$<>CHR$(32) THEN 270
260 GET A$:IF A$<>CHR$(13) THEN 260
270 SYS 49298:GOTO 210
300 PRINT"(CLEAR, DOWN5,RIGHT8)LADOWANIE JEZYKA MASZYNOWEGO"
310 FOR K=49152 TO 49580:READ M:POKE K,M:NEXT:RETURN
```

```
1000 DATA 169,39,133,253,169,0,133,254
1010 DATA 169,216,133,255,169,0,141,36
1020 DATA 192,169,203,141,37,192,160,0
1030 DATA 177,254,41,15,240,11,160,7
1040 DATA 190,90,192,254,26,207,136,16
1050 DATA 247,238,36,192,208,3,238,37
1060 DATA 192,198,253,16,18,169,39,133
1070 DATA 253,24,173,36,192,105,2,141
1080 DATA 36,192,144,3,238,37,192,230
1090 DATA 254,208,2,230,255,165,254,201
1100 DATA232,208,195,165,255,201,219,208
1110 DATA 189,96,0,1,2,42,44,84
1120 DATA 85,86,169,0,141,111,192,169
1130 DATA 203,141,112,192,162,0,142,110
1140 DATA 207,238,111,192,208,3,238,112
1150 DATA192,173,111,192,201,110,208,238
1160 DATA173,112,192,201,207,208,231,165
1170 DATA 251,201,15,208,2,134,251,230
1180 DATA 251,96,169,39,133,253,169,0
1190 DATA133,254,169,216,133,255,169,43
1200 DATA141,171,192,169,203,141,172,192
1210 DATA 160,0,173,69,207,201,2,240
1220 DATA 19,201,3,208,12,177,254,41
1230 DATA 15,208,9,165,251,145,254,208
1240 DATA 3,152,145,254,238,171,192,208
1250 DATA 3,238,172,192,198,253,16,18
1260 DATA 169,39,133,253,24,173,171,192
1270 DATA 105,2,141,171,192,144,3,238
1280 DATA 172,192,230,254,208,2,230,255
1290 DATA165,254,201,232,208,188,165,255
1300 DATA 201,219,208,182,96,169,0,133
1310 DATA 254,169,216,133,255,169,0,133
1320 DATA 252,169,4,133,253,160,0,169
1330 DATA 0,145,254,169,81,145,252,230
1340 DATA 252,208,2,230,253,230,254,208
1350 DATA 2,230,255,165,254,201,232,208
1360 DATA230,165,255,201,219,208,224,169
1370 DATA 1,133,251,96,160,1,24,185
1380 DATA 0,203,121,26,207,153,26,207
1390 DATA 24,185,69,207,121,42,203,153
1400 DATA 42,203,200,192,41,208,231,169
1410 DATA 25,133,253,169,42,133,254,169
1420 DATA 203,133,255,160,0,24,177,254
1430 DATA 160,40,113,254,145,254,160,41
1440 DATA 24,177,254,160,1,113,254,145
1450 DATA 254,24,165,254,105,42,133,254
1460 DATA 144,2,230,255,198,253,208,219
1470 DATA 24,173,0,203,160,40,121,26
1480 DATA 207,153,26,207,160,41,24,185
1490 DATA 0,203,160,41,121,26,207,153
1500 DATA 26,207,24,173,68,207,160,40
1510 DATA 121,42,203,153,42,203,160,41
1520 DATA 24,185,68,207,160,1,121,42
1530 DATA 203,153,42,203,96
1540 END

READY.
```

„ANDROMEDA" (Planetarium i Obserwatorium Astronomiczne w Łodzi) wyposażona w mikrokomputery Spectrum, Meritum, Commodore oraz Amstrad zajmuje się przede wszystkim tworzeniem biblioteki programów astronomicznych, popularyzacją informatyki oraz nauczaniem języków programowania.

Plany na przyszłość obejmują nie tylko „komputerowe loty" poza naszą galaktykę, ale znaczne rozszerzenie działalności poprzez organizację masowych imprez dla młodzieży.

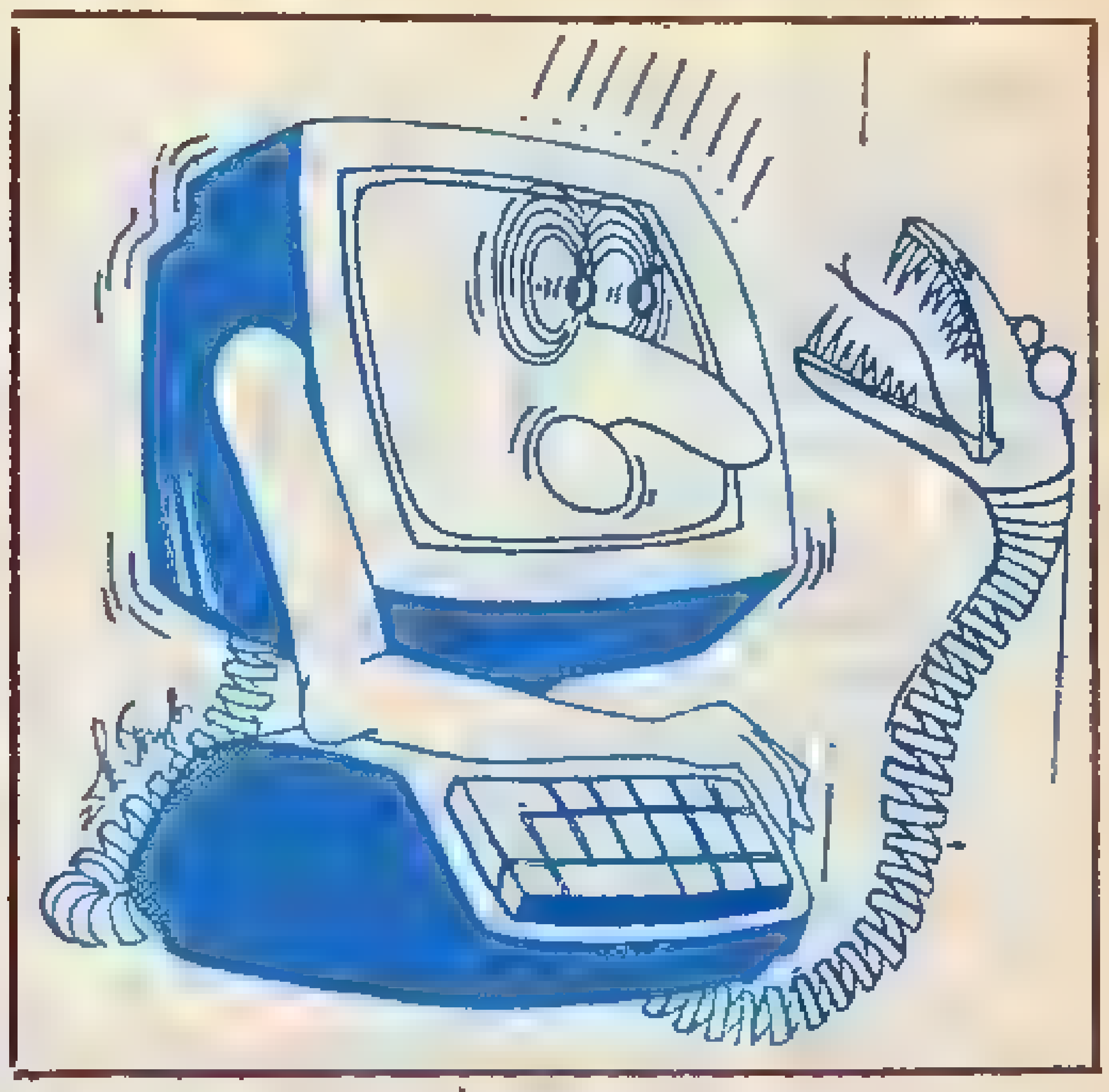


Tworząc programy astronomiczne zgodnie z narzuconymi algorytmami można również wykazać dużo własnej inicjatywy.

17

SHARP

PROGRAM 8



DŻUNGLA

Ten program dedykujemy Czytelnikom z fantazją: proponujemy... wędrówkę przez dżunglę!

Oczywiście „wędrownik" jest uzbrojonym po zęby strzelcem wyborowym. Jako wytrawny podróżnik ma kompas, tyle tylko że... nie zna drogi!

A wokół rządzą *prawa dżungli!* Co krok jakaś *niespodzianka:* raz urwisko, innym razem pyton, udający gałąź lub... sprężona do ataku pantera!

Wędrownik ma tylko jeden cel: *wyjść z dżungli.* I to jak najszybciej!

Może on wybrać kierunek północny, południowy, wschodni lub zachodni. Tyle tylko że każdy kierunek jest jednakowo dobry (a raczej jednakowo zły), gdyż startuje on ze „środka" dżungli.

Teoretycznie można wyjść po 20 manewrach. Praktycznie: po 30, 40...

Gdy wędrownik zbyt długo błądzi (po 60 manewrach), wtedy zwycięża dżungla: pojawia się meldunek *„zaginiony"* i... komputer staje.

Rysunek podaje przykład udanego wyjścia z dżungli. *Uwaga:* nie znamy „sposobu na wyjście", gdyż niespodzianki modelują dwa generatory liczb losowych: jeden (R=RND4) lokalizuje, zaś drugi (T=RND 8) definiuje zdarzenie.

Program wywołuje etykieta „X" (RUN 10)

Życzymy szybkiego wyjścia!

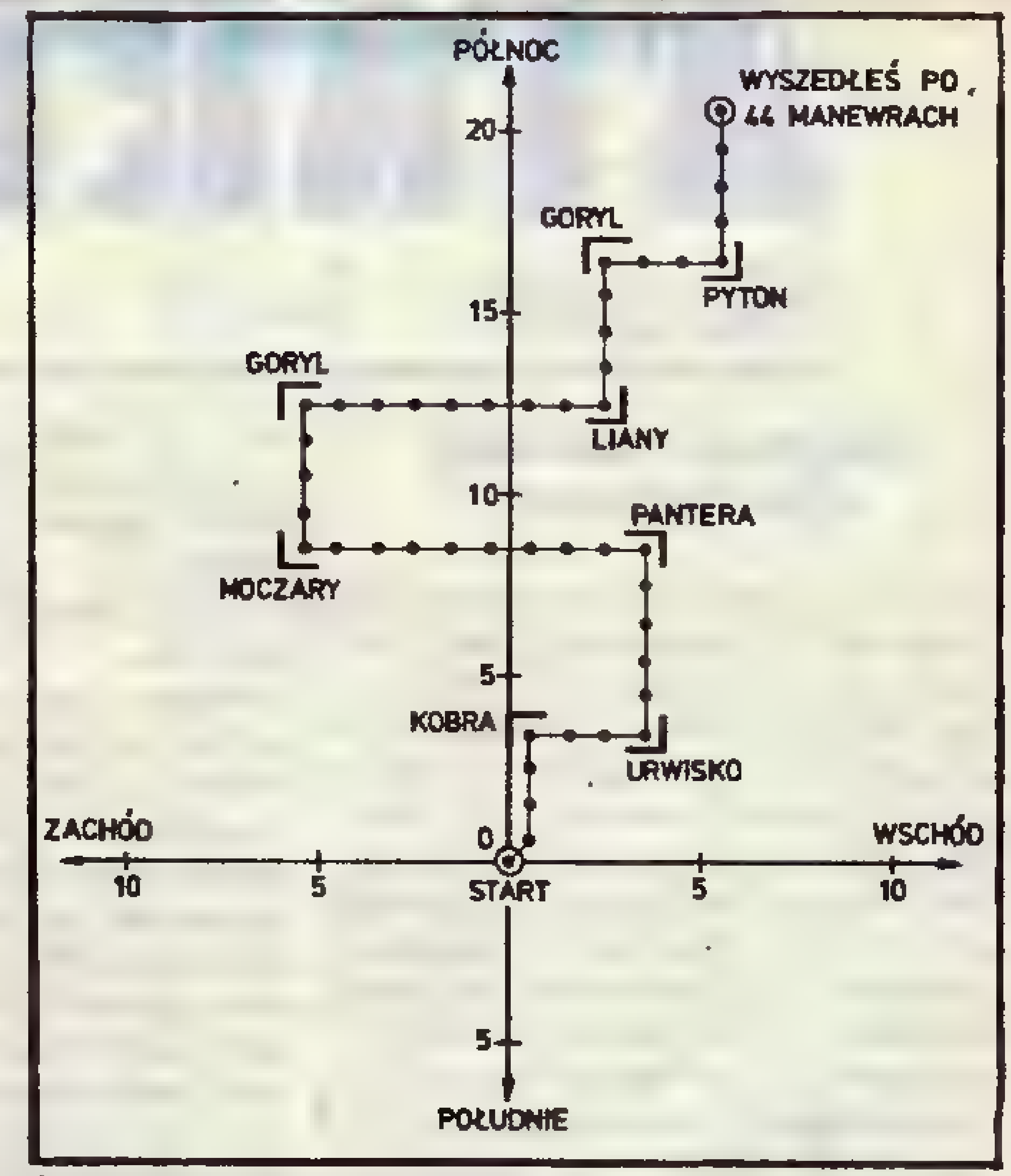
Janusz MILLER

```
10:"X":CLEAR :
   RANDOM :USING
   :WAIT 0:DIM P$
   (8)
20:DATA "U R W I
   S K O","L I A
   N Y","M O C Z
   A R Y","P O T
   O K"
30:DATA "P A N T
   E R A","K O B
   R A","G O R Y
   L","P Y T O N"
40:RESTORE 20:FOR
   K=1TO 4:READ P
   $(K):NEXT K
50:RESTORE 30:FOR
   K=5TO 8:READ P
   $(K):NEXT K
60:I$=" ? T=1 N=0
   ":R$="U C I E
   K A J !!"
70:CLS :R=RND 4:T
   =RND 8:GOTO 90
80:N=0:S=0:W=0:E=
   0:PRINT P$(T):
   BEEP 3,25,500
90:WAIT 0:PRINT "
   POLNOC";I$;:
   INPUT "";N
100:CLS :IF N=0
   GOTO 130
110:IF R*N=1OR R*N
   =2GOTO 80
120:Y=Y+1:Z=Z+1:
   GOTO 170
130:PRINT "POLUDNI
   E";I$;:INPUT "
   ";S
140:CLS :IF S=0
   GOTO 170
150:IF R*S=3OR R*S
   =4GOTO 80
160:Y=Y-1:Z=Z+1
170:PRINT "WSCHOD"
   ;I$;:INPUT "";
   E
180:CLS :IF E=0
   GOTO 210
190:IF R*E=2OR R*E
   =3GOTO 80
200:X=X+1:Z=Z+1:
   GOTO 240
210:PRINT "ZACHOD"
   ;I$;:INPUT "";
   W
220:CLS :IF W=0
   GOTO 240
230:X=X-1:Z=Z+1
240:IF (N+S+W+E)=0
   THEN PRINT R$:
   BEEP 10,25,50:
   GOTO 90
250:IF Z>=60THEN
   WAIT :PRINT "Z
    A G I N I O N
    Y":GOTO 290
260:N=0:S=0:W=0:E=
   0
270:IF ABS X<20AND
   ABS Y<20GOTO 7
   0
280:WAIT :PRINT "W
   YSZEDLES PO";Z
   ;" MANEWRACH"
290:WAIT 0:PRINT "
   OD NOWA";I$;:
   INPUT "";P
300:CLS :IF P=0
   THEN END
310:X=0:Y=0:Z=0:
   GOTO 70
```

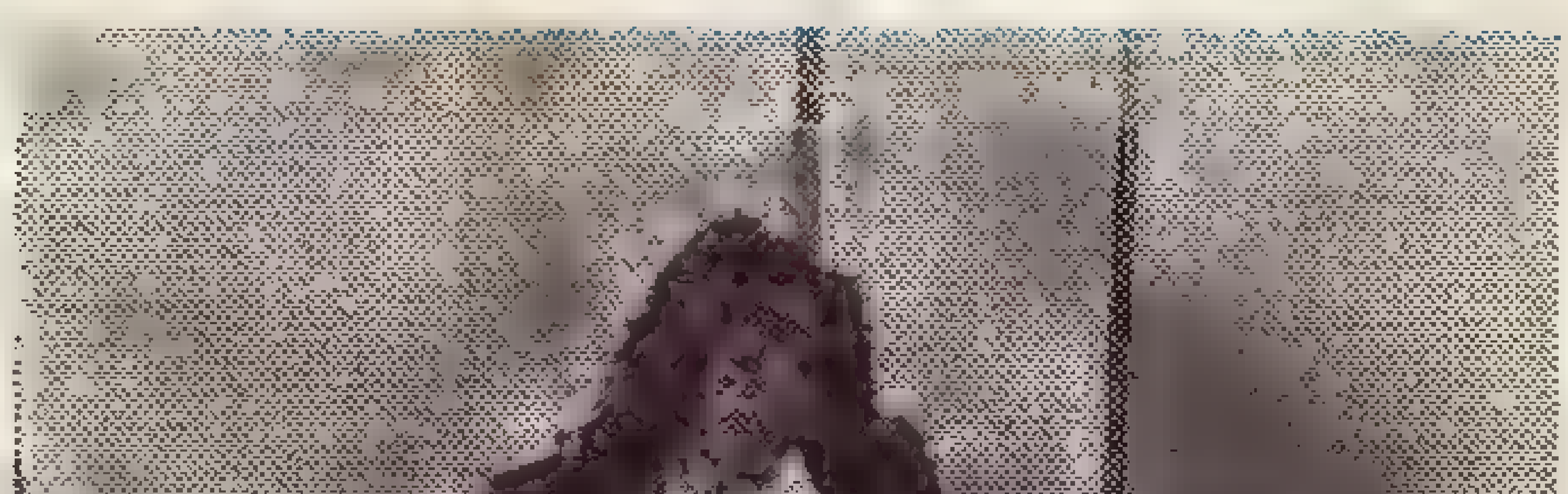
SPECTRUM

```
   5 REM Dzungla
   6 LET y=0: LET z=0: LET x=0
   8 LET n=0: LET s=0: LET w=0
   9 LET e=0: LET p=0: LET pn=0
  10 CLS : DIM p$(8,17)
  11 CIRCLE 125,85,2: PLOT 125,6
5
  20 DATA "U R W I S K O"
  21 DATA "L I A N Y"
  22 DATA "M O C Z A R Y"
  23 DATA "P O T O K"
  30 DATA "P A N T E R A"
  31 DATA "K O B R A"
  32 DATA "G O R Y L"
```

```
 33 DATA "P Y T O N"
 35 RESTORE 20
 40 FOR k=1 TO 8
 42 READ p$(k,1 TO 17)
 44 NEXT k
 60 LET i$=" ? Tak = 1  Nie = 0
"
 62 LET r$=" U C I E K A J !"
 75 LET r=INT (RND*4)
 77 LET t=INT (RND*7)+1
 79 GO TO 90
 80 LET n=0: LET s=0
 82 LET e=0: LET w=0
 84 PRINT AT 20,1;p$(t,1 TO 17)
 86 BEEP .5,1: BEEP .5,1
 87 BEEP 1,1: BEEP 1,1
 90 PRINT AT 1,1;"POLNOC   ";i$
 94 INPUT "";n
102 IF n=0 THEN GO TO 130
105 LET u=r*n
110 IF u=1 OR u=2 THEN GO TO 80
111 DRAW 0,3
120 LET y=y+1: LET z=z+1
122 GO TO 170
130 PRINT AT 1,1;"POLUDNIE";i$
134 INPUT "";s
142 IF s=0 THEN GO TO 170
150 LET u=r*s
152 IF u=3 OR u=4 THEN GO TO 80
153 DRAW 0,-3
160 LET y=y-1: LET z=z+1
170 PRINT AT 1,1;"WSCHOD   ";i$
172 INPUT "";e
182 IF e=0 THEN GO TO 210
190 LET u=r*e
192 IF u=2 OR u=3 THEN GO TO 80
193 DRAW 3,0
200 LET x=x+1: LET z=z+1
202 GO TO 240
210 PRINT AT 1,1;"ZACHOD  ";i$
212 INPUT "";w
222 IF w=0 THEN GO TO 240
224 DRAW -3,0
230 LET x=x-1: LET z=z+1
240 LET u=n+s+w+e
 242 IF u=0 THEN PRINT AT 2,10;r
$: BEEP 2,1: PRINT AT 2,10;"
             ": GO TO 90
 250 IF z<=62 THEN GO TO 260
 252 PRINT AT 10,1; FLASH 1;"Z A
G I N I O N Y"
 253 BEEP 3,0
 254 GO TO 290
 260 LET n=0: LET s=0
 262 LET w=0: LET e=0
 270 LET u=ABS x: LET v=ABS y
 272 IF u<20 AND v<20 THEN GO TO
 70
 280 PRINT AT 10,1; FLASH 1;"Wys
zedles po ";z;" manewrach"
 281 FOR j=1 TO 8: BEEP .1,j: NE
XT j
 290 INPUT "czy grasz dalej (tak
/nie) ";d$
 292 IF d$="nie" THEN STOP
 294 LET x=0: LET y=0: LET z=0
 296 CLS : GO TO 70
```



Rys. 1. Przykład podróży w dżungli



Firma „LASER" przywędrowała aż z USA. W rozmowie z Jadwigą Bieńkowską dowiedzieliśmy się, że jej bossowie nastawiają się na sprzedaż profesjonalnych komputerów dla zakładów pracy i instytucji. Oczywiście z pełnym oprogramowaniem. Niskie ceny, to główny atut, mający najbardziej przemawiać do wyobraźni ewentualnych nabywców Tylko że... zakupy można realizować za twardą walutę poprzez wpłaty na konto w Berlinie Zachodnim.

26

ATARI

PROGRAM 9–10

Regresja i korelacja dwu zmiennych

Program „Regresja i korelacja dwu zmiennych" (napisany w języku Atari-Basic) pozwala obliczyć kowariancję i współczynnik korelacji zmiennych losowych X i Y oraz parametry (wyraz wolny i współczynnik regresji) prostej regresji według najmniejszych kwadratów (regresja zmiennej Y względem zmiennej X i zmiennej X względem zmiennej Y), wartości regresyjne, błędy szacunku i współczynnik zbieżności. Program oblicza ponadto i wyprowadza na ekran szereg wyników pośrednich w postaci różnych sum zmiennych X i Y. Tym samym program daje użytkownikowi pełen zestaw charakterystyk liczbowych potrzebnych do analizy współzależności dwóch zmiennych losowych. Może to okazać się szczególnie przydatne w procesie dydaktycznym, w trakcie demonstracji różnych wzorów (metod) służących do obliczania ww. parametrów (w tym również szacowania parametrów prostej regresji uproszczoną metodą najmniejszych kwadratów).

Aby podnieść walory dydaktyczne programu wyposażono go w generator polskich liter (wg W. Zientara, Komputer nr 8, 1986), który jest wczytywany po uruchomieniu programu instrukcją **RUN**. Jeżeli nie był używany klawisz **RESET**, to następnych uruchomień można dokonać instrukcją **GOTO 80** (bez linii).

Program pozwala na wprowadzenie wartości zmiennych X i Y dwoma sposobami, a mianowicie „ręcznym" (tj. z klawiatury) i za pomocą instrukcji **DATA**. Ten ostatni sposób wprowadzania danych jest szczególnie wart polecenia albowiem ułatwia sprawdzanie i poprawianie danych wejściowych (gdy mamy wątpliwość co do poprawności obliczeń możemy sprawdzić poprawność wprowadzonych danych i wnieść poprawki bez konieczności ponownego wprowadzania wszystkich wartości, co przy dużej liczbie prób jest bardzo uciążliwe), a także pozwala na wielokrotne powtarzanie obliczeń i wprowadzanie danych z taśmy (zapisanych instrukcją LIST). Wartość zmiennych X i Y wprowadza się instrukcjami DATA (w kolejności: X1, Y1, X2, Y2, ..., Xn, Yn) począwszy od linii 730 (program zajmuje linie od 20 do 720, a generator polskich liter od 29000 do 29058).

Przy wprowadzaniu danych z klawiatury program zabezpiecza proces wprowadzania przed:

a) zbyt szybkim naciśnięciem klawisza **RETURN** (przed wprowadzeniem wartości X lub Y),
b) błędnym wprowadzeniem współrzędnych aktualnie zapisywanego punktu.

Po stwierdzeniu poprawności wpisania współrzędnych kolejnego punktu należy to potwierdzić wpisaniem liczby 2 i naciśnięciem klawisza **RETURN**. Dopiero wówczas wartości współrzędnych zostaną zaakceptowane przez program i można przejść do zapisu współrzędnych następnego punktu. Po stwierdzeniu błędu należy wpisać 1 i nacisnąć klawisz **RETURN**. Wówczas program umożliwi powtórne wpisanie wartości współrzędnych danego punktu.

Liczność prób N nie może przekraczać 100. Można ją zmienić przez wprowadzenie zmian w deklaracji zmiennych X, Y, YR w linii 80.

W programie użyto określeń zaczerpniętych z książek: Oktaba W., Niedokos E.: Matematyka i podstawy statystyki matematycznej, PWN, Warszawa 1974 i Zając K.: Zarys metod statystycznych, PWE, Warszawa 1971.

Prezentowany program różni się od innych programów tego typu przede wszystkim zakresem dostarczanej informacji, co czyni zeń narzędzie mogące zaspokoić wymagania również profesjonalnego użytkownika. Walory użytkowe programu znakomicie podnosi możność wprowadzania danych za pomocą instrukcji DATA a jego walory dydaktyczne zastosowanie polskiego alfabetu.

Andrzej GRZESIAK

```
20 ? CHR$(125):POKE 752,1
30 POSITION 3,10:? "REGRESJA I K
ORELACJA DWU ZMIENNYCH"
40 FOR F=1 TO 1500:NEXT F
50 ? CHR$(125):POSITION 1,10:? "
Poczekaj 21 sekund.Wprowadzam ge
nerator"
60 POSITION 13,12:? "polskich li
ter":GOSUB 29000
70 POKE 752,0
80 CLR :DIM X(100),Y(100),X$(10)
,Y$(10),YR(100),Q$(1)
90 ? CHR$(125):? "Wprowadzanie d
anych r";CHR$(4);"czne czy za po
moc";CHR$(17);" instrukcji DATA?
 R/D"
91 INPUT Q$
100 IF Q$(1)="D" THEN 210
110 N=1:? CHR$(125):? "Podaj war
to";CHR$(23);"ci zmiennych X i Y
"
120 ? :? "Zako";CHR$(13);"czenie
 wprowadzania danych:wpisz END i
 naci";CHR$(23);"nij RETURN":?
130 ? "Je";CHR$(24);"eli ";CHR$(
7);"le wprowadzi";CHR$(123);"e";
CHR$(23);
134 ? " wsp";CHR$(16);CHR$(123);
"RZ";CHR$(4);"dne danego punktu"
;
135 ? "naci";CHR$(23);"nij 1 i R
ETURN,je";CHR$(24);"eli dobrze -
 naci";CHR$(23);"nij 2 i RETURN"
140 ? :? "Punkt";" ";N:INPUT X$:
TRAP 140:IF X$(1,1)="E" THEN N=N
-1:GOTO 300
150 INPUT Y$:TRAP 150:X(N)=VAL(X
$):Y(N)=VAL(Y$)
160 ? "X(";N;")=";X(N)
170 ? "Y(";N;")=";Y(N)
180 INPUT U:IF U=1 THEN 140
190 N=N+1:GOTO 140
200 REM Wprowadzenie danych inst
rukcjami DATA
210 ? CHR$(125):? "Liczba pr";CH
R$(16);"b N=";:INPUT N:FOR F=1 T
O N:X(F)=0:Y(F)=0:NEXT F:K=0
220 FOR F=1 TO N
230 FOR G=1 TO 2
240 READ L
250 IF G=1 THEN X(F)=L:? "X(";F;
")=";X(F):GOTO 270
260 Y(F)=L:? "Y(";F;")=";Y(F):K=
K+1:IF K=9 THEN K=0:? :? "c.d. p
o naci";CHR$(23);"ni";CHR$(4);"c
iu RETURN"
261 INPUT Q$
270 NEXT G
280 NEXT F
290 ? :? "c.d. po naci";CHR$(23)
;"ni";CHR$(4);"ciu RETURN":INPUT
 Q$
300 S1=0:S2=0:S3=0:S4=0:S5=0
310 FOR F=1 TO N
320 S1=S1+X(F):S2=S2+Y(F):S3=S3+
X(F)*X(F):S4=S4+X(F)*Y(F):S5=S5+
Y(F)*Y(F)
330 NEXT F
340 XS=S1/N:YS=S2/N
350 ? CHR$(125)
360 ? "SIGMA X=";S1,"SIGMA Y=";S
2:? "SIGMA X^2=";S3,"SIGMA Y^2="
;S5:? "SIGMA X*Y=";S4
370 ? "X";CHR$(23);"red.=";XS,"Y
";CHR$(23);"red.=";YS
380 SXY=S4/N-XS*YS:? :? "Kowaria
ncja=";SXY:NSXY=N*SXY
390 NS2X=S3-(S1*S1)/N:NS2Y=S5-(S
2*S2)/N:RXY=NSXY/SQR(NS2X*NS2Y)
400 ? :? "SIGMA (X-X";CHR$(23);"
red)^2=";NS2X:? "SIGMA (Y-Y";CHR
$(23);"red)^2=";NS2Y
410 ? "SIGMA (X-X";CHR$(23);"red
)(Y-Y";CHR$(23);"red)=";NSXY
411 ? :? "Wsp";CHR$(16);CHR$(123
);"czynnik korelacji=";RXY
420 BYX=NSXY/NS2X:WW=YS-BYX*XS
430 ? :? "R";CHR$(15);"WNANIE PR
OSTEJ REGRESJI"
440 ? :? CHR$(217);"=";WW;"+";BY
X;CHR$(216)
450 SYWX=SQR((NS2Y/N)*(1-RXY^2))
460 ? :? "Odchylenie standardowe
=";SYWX
```

```
470 VYWX=N*(SYWX^2)/(N-2)
480 ? "B";CHR$(123);CHR$(17);"d
";CHR$(23);"redni=";SQR(VYWX)
490 BSWR=SQR(VYWX/NS2X)
500 POSITION 1,21:? "B";CHR$(123
);CHR$(17);"d standardowy wsp";C
HR$(16);CHR$(123);"czynnika regr
esji=";BSWR
510 ? "Ci";CHR$(17);"g dalszy po
 naci";CHR$(23);"ni";CHR$(4);"ci
u [RETURN]":INPUT Q$
520 ? CHR$(125):? "Warto";CHR$(2
3);"ci regresyjne":?
530 K=0
540 FOR F=1 TO N
550 Y=WW+BYX*X(F):YR(F)=Y
560 ? "Y";F;"=";YR(F)
570 K=K+1:IF K=15 THEN K=0:? "c.
d. wydruku po naci";CHR$(23);"ni
";CHR$(4);"ciu RETURN":INPUT Q$
580 NEXT F
590 REM Obliczanie wspolczynnika
 zbieznosci
600 FOR F=1 TO N
610 SUMA1=SUMA1+(Y(F)-YR(F))^2
620 SUMA2=SUMA2+(Y(F)-YS)^2
630 NEXT F
640 SUMA3=SUMA1/SUMA2
650 ? :? "Wsp";CHR$(16);CHR$(123
);"czynnik zbie";CHR$(24);"no";C
HR$(23);"ci=";SUMA3
660 ? :? "Czy chcesz obliczy";CH
R$(22);" regresj";CHR$(4);" odwr
otn";CHR$(17);"? T/N":INPUT Q$
670 IF Q$="N" THEN 700
680 FOR F=1 TO N:Z1=Y(F):Z2=X(F)
:X(F)=Z1:Y(F)=Z2:NEXT F
690 GOTO 300
700 ? :? "Czy pracujemy dalej? T
/N":INPUT Q$
710 IF Q$="T" THEN 80
720 END
29000 CHB=(PEEK(106)-8)*256
29010 FOR I=0 TO 1023:POKE CHB+I
,PEEK(57344+I):NEXT I:RESTORE 29
040
29020 READ A:IF A=+1 THEN POKE 7
56,CHB/256:RETURN
29030 FOR I=0 TO 7:READ B:POKE C
HB+A*8+I,B:NEXT I:GOTO 29020
29040 DATA 65,0,24,60,102,102,12
6,102,12
29041 DATA 67,12,60,102,96,96,10
2,60,0
29042 DATA 68,0,0,60,102,126,96,
60,6
29043 DATA 69,0,126,96,124,96,96
,126,12
29044 DATA 70,12,24,126,12,24,48
,126,0
29045 DATA 71,12,24,0,126,12,48,
126,0
29046 DATA 76,0,96,120,112,224,9
6,126,0
29047 DATA 77,12,24,0,124,102,10
2,102,0
29048 DATA 78,24,102,118,126,126
,110,102,0
29049 DATA 79,12,60,102,102,102,
102,60,0
29050 DATA 80,12,24,0,60,102,102
,60,0
29051 DATA 81,0,0,60,6,62,102,62
,12
29052 DATA 83,12,60,96,60,6,6,60
,0
29053 DATA 86,12,24,0,60,96,96,6
0,0
29054 DATA 87,12,24,62,96,60,6,1
24,0
29055 DATA 88,0,24,0,126,12,48,1
26,0
29056 DATA 90,24,0,126,12,24,48,
126,0
29057 DATA 123,0,56,24,28,56,24,
60,0
29058 DATA 1,0,0,0,0,0,0,0,0
```

Korelacja wielu zmiennych

Program pozwala obliczyć współczynniki równania regresji, średni błąd szacunku (tzw. standardowy błąd oceny), wartości regresyjne, współczynniki korelacji prostej (wyrażającej związek danej pary cech obliczony bez względu na ewentualny związek z innymi cechami) oraz współczynnik korelacji wielokrotnej, dla dowolnej liczby P zmiennych i dowolnej liczby N prób (ograniczenia wynikają tylko z pojemności pamięci). Oprócz tego obliczane są i wyświetlane na monitorze wyniki pośrednie:

SIGMAX(P) — suma (po N) wartości zmiennej X z indeksem P,

SIGMAX2(P) — suma kwadratów zmiennej X,

XSRED(P) — wartość średnia zmiennej X,

NS2X(P) — suma kwadratów odchyleń po jedynczych wyników od średniej,

SIGMAXX(P,P) — suma iloczynów mieszanych,

NSXX(P,P) — suma iloczynów mieszanych odchyleń od ich średnich.

Wartości sum SIGMAX(P), SIGMAXX(P,P), NS2X(P) oraz NSXX(P,P) przechowywane są w pamięci (zadeklarowane jako zmienne tablicowe w linii 120), stąd przy ewentualnych przeróbkach programu mogą być bezpośrednio wykorzystane do obliczeń. Pozostałe sumy i wartości średnie należy w takim przypadku zadeklarować jako zmienne tablicowe i nadać im wartości (np. w linii 300)

Znajomość współczynników korelacji prostej pozwala na obliczenie współczynników korelacji cząstkowej.

Aby uczynić program jak najbardziej czytelnym, a tym samym ułatwić użytkownikom dokonywanie przeróbek, nadano zmiennym, wykorzystywanym w trakcie obliczeń, nazwy odpowiadające typowemu sposobowi zapisu stosowanemu w podręcznikach z zakresu statystyki (W. Oktaba, E. Niedokos: Matematyka i podstawy statystyki matematycznej, PWN, Warszawa 1974).

Dane, X(P,N), wprowadza się instrukcjami DATA (poczynając od linii 1400), wg narastającej wartości P (przy ustalonym N). Ilustruje to poniższy przykład, dla P = 2:
1400 DATA X(1,1), X(2,1),..., X(1,N), X(2,N).
Na pierwszym miejscu (X(1,1),..., X(1,N), ...) wprowadza się zmienną niezależną. Wprowadzanie danych można przerwać (bez uszczerbku w procesie wprowadzania) w dowolnym momencie, naciskając jednocześnie klawisze **CONTROL** i **1**; wznowienie procesu wczytywania następuje po kolejnym naciśnięciu klawiszy. Operacja ta wymaga pewnej wprawy.

Program można wpisać bez linii 1110—1300, w których umieszczono informacje dotyczące programu (w tym przypadku również bez linii 60 i 70), jednak doświadczenie autora wskazuje, że trud związany z wprowadzeniem tej informacji opłaca się sowicie przy dłuższej eksploatacji programu.

Metodę rozwijania równań liniowych, z których oblicza się w programie współczynniki równania regresji oraz wzory według których oblicza się błąd standardowy oceny współczynnik korelacji wielokrotnej można znaleźć w książce: F. C. Mills: Statistical Methods..., Columbia University, third edition, N. York 1955 (wyd. w jęz. rosyjskim z 1958 r.).

Błędem standardowym nazywa się w programie błąd uzyskany w wyniku dzielenia, w którym mianownik równy jest N, a błędem standardowym poprawionym — błąd uzyskany w wyniku dzielenia, w którym mianownik równy jest ((N — P).

```
10 REM Strona tytulowa
20 DIM Q$(3)
30 ? CHR$(125):GRAPHICS 2:SETCOL
OR 2,0,0:POSITION 6,2:? #6;"KORE
LACJA "
40 POSITION 8,4:? #6;"WIELU":POS
ITION 6,6:? #6;"ZMIENNYCH"
50 FOR ZMP=1 TO 1000:NEXT ZMP
60 ? "Czy chcesz informacje o pr
ogramie? T/N":INPUT Q$
70 IF Q$="T" THEN GRAPHICS 0:GOS
UB 1120
80 ? CHR$(125)
90 ? :? "Podaj liczbe zmiennych
P=";:INPUT P
100 ? :? "Podaj liczbe prob N=";
:INPUT N
110 GRAPHICS 0
120 DIM X(P,N),A(P,P),B(P),SIGMA
X(P),SIGMAXX(P,P),NS2X(P),NSXX(P
,P),Z(P)
130 REM Wprowadzanie danych
140 FOR F=1 TO N:FOR G=1 TO P
150 READ L:X(G,F)=L
160 ? "X(";G;",";F;")=";X(G,F)
170 NEXT G:NEXT F
180 ? :? "Koniec wprowadzania da
nych":? :? "c.d. po nac. ";CHR$(
210);CHR$(197);CHR$(212):INPUT Q
$
190 ? CHR$(125):? "Czy odczytuje
sz wartosci posrednie?   TAK-1;N
IE-2":INPUT OWP
200 REM Podprogram: obliczanie w
artosci posrednich
210 REM Obliczanie SIGMAX, SIGMA
X2, NS2X i XSRED
220 FOR G=1 TO P:SIGMAX=0:SIGMAX
2=0
230 FOR F=1 TO N
240 SIGMAX=SIGMAX+X(G,F)
```

```
250 SIGMAX2=SIGMAX2+X(G,F)*X(G,F
)
260 NEXT F
270 NS2X=SIGMAX2-(SIGMAX^2)/N
280 XSRED=SIGMAX/N
290 IF G=1 THEN B(G)=SIGMAX:ZD2=
SIGMAX2
300 A(G,G)=SIGMAX2:SIGMAX(G)=SIG
MAX:NS2X(G)=NS2X
310 IF G>1 THEN A(1,G)=SIGMAX:A(
G,1)=SIGMAX
320 ? CHR$(125)
330 ? :? "SIGMAX(";G;")=";SIGMAX
:? :? "SIGMAX2(";G;")=";SIGMAX2
340 ? :? "NS2X(";G;")=";NS2X:?
? "XSRED(";G;")=";XSRED
350 IF OWP=2 THEN 370
360 ? :? :? :? "c.d. po nac. ";C
HR$(210);CHR$(197);CHR$(212):INP
UT Q$
370 NEXT G
380 REM Obliczanie SIGMAXX
390 FOR G=1 TO P-1
400 FOR J=G+1 TO P:SIGMAXX=0
410 FOR F=1 TO N
420 SIGMAXX=SIGMAXX+X(G,F)*X(J,F
)
430 NEXT F
440 NSXX=SIGMAXX-(SIGMAX(G)*SIGM
AX(J))/N
450 SIGMAXX(G,J)=SIGMAXX:NSXX(G,
J)=NSXX:IF G=1 THEN B(J)=SIGMAXX
460 IF G>1 THEN A(G,J)=SIGMAXX:A
(J,G)=SIGMAXX
470 ? CHR$(125)
480 ? :? "SIGMAXX(";G;",";J;")="
;SIGMAXX
490 ? :? "NSXX(";G;",";J;")=";NS
XX
500 IF OWP=2 THEN 520
510 ? :? :? :? :? "c.d. po nac.
";CHR$(210);CHR$(197);CHR$(212):
INPUT Q$
520 NEXT J
530 NEXT G
540 A(1,1)=N
550 REM Podprogram:obliczanie ws
polczynnikow rownania regresji (
uklad rownan liniowych)
560 ZMP=1.0E-07
570 FOR F=1 TO P:ZMD=ZMP
580 FOR I=F TO P:G=A(I,F):IF ABS
(G)>ABS(ZMD) THEN ZMD=G:J=I:NEXT
 I
590 IF ABS(ZMD)<=ZMP THEN ? "Ukl
ad rownan nie ma jednoznacznego
rozwiazania":END
600 FOR I=F TO P:G=A(J,I):A(J,I)
=A(F,I):A(F,I)=G:NEXT I
610 G=B(J):B(J)=B(F):B(F)=G:IF F
=P THEN GOTO 640
620 FOR I=F+1 TO P:G=A(I,F)/ZMD:
FOR J=F+1 TO P:A(I,J)=A(I,J)-G*A
(F,J):NEXT J:B(I)=B(I)-G*B(F):NE
XT I
630 NEXT F
640 Z(P)=B(P)/A(P,P)
650 FOR F=P-1 TO 1 STEP -1:Z(F)=
B(F)/A(F,F):FOR J=F+1 TO P:Z(F)=
Z(F)-A(F,J)/A(F,F)*Z(J):NEXT J:N
EXT F
660 REM Wydruk
670 ? CHR$(125)
680 POSITION 13,1:? "ROWNANIE RE
GRESJI"
690 POSITION 3,3:? "X1 = B11 + B
12*X2 + B13*X3 + B14*X4 +"
700 POSITION 12,4:? "+ B15*X5 +
...... + B1P*XP"
710 ? :? "gdzie dla P=";P:?
720 LI=0
730 FOR I=1 TO P
740 PRINT " B1";I;"=";Z(I)
750 LI=LI+1:IF LI=8 THEN LI=0:?
:? "c.d. po nac. ";CHR$(210);CHR
$(197);CHR$(212):INPUT Q$
760 NEXT I
770 REM Podprogram: obliczanie w
spolczynnikow korelacyjnych
780 REM Obliczanie bledu stand.
i wsp. kor. wielokrotnej
790 ZMP=0
800 FOR G=2 TO P:ZMP=ZMP+Z(G)*SI
GMAXX(1,G):NEXT G
810 ZMP=(ZD2-Z(1)*SIGMAX(1)-ZMP)
/N
820 ? :? "Blad standardowy=";SQR
(ZMP)
830 ZMP=ZMP*(N/(N-P))
840 ? :? "Blad stand. poprawiony
=";SQR(ZMP)
850 ZMP=0
860 FOR G=2 TO P:ZMP=ZMP+Z(G)*SI
GMAXX(1,G):NEXT G
870 K=N*((SIGMAX(1)/N))^2
880 ZMP=(Z(1)*SIGMAX(1)+ZMP-K)/(
ZD2-K)
890 ? :? "Wspolczynnik korelacji
=";SQR(ZMP)
900 ? :? "c.d. po nac. ";CHR$(21
0);CHR$(197);CHR$(212):INPUT Q$
910 REM Obliczanie wartosci regr
esyjnych
920 ? CHR$(125):POSITION 12,2:?
"WARTOSCI REGRESYJNE":? :FOR I=1
 TO 500:NEXT I
930 LI=0
940 FOR F=1 TO N:ZMP=0
950 FOR G=2 TO P
960 ZMP=ZMP+Z(G)*X(G,F)
970 NEXT G
980 ZMP=ZMP+Z(1)
990 ? "Y(";F;")=";ZMP
1000 LI=LI+1:IF LI=15 THEN LI=0:
? "c.d. po nac. ";CHR$(210);CHR$
(197);CHR$(212):INPUT Q$
1010 NEXT F
1020 ? :? "c.d. po nac. ";CHR$(2
10);CHR$(197);CHR$(212):INPUT Q$
1030 REM Obliczanie wsp. kor. pr
ostej
1040 ? CHR$(125):? "WSPOLCZYNNIK
I KORELACJI PROSTEJ":? :FOR ZMP=
1 TO 1000:NEXT ZMP:LI=1
1050 FOR F=1 TO P-1:FOR G=F+1 TO
 P
1060 ZMP=NSXX(F,G)/SQR(NS2X(F)*N
S2X(G))
1070 ? "R";F;G;"=";ZMP
1080 LI=LI+1:IF LI=15 THEN LI=0:
? :? "c.d. po nac. ";CHR$(210);C
HR$(197);CHR$(212):INPUT Q$
1090 NEXT G:NEXT F
1100 END
1110 REM Blok informacji o progr
amie
1120 ? CHR$(125):GRAPHICS 0
1130 ? "Program oblicza wspolczy
nniki rownania regresji, blad st
andardowy, wspolczyn nik korelac
ji ";
1140 ? "wielokrotnej,wartosci  r
egresyjne oraz wspolczynniki kor
ela- cji prostej";
1150 ? " ";:? "dla dowolnej licz
by P z-  miennych (w tym zm. nie
zalezna)";
1160 ? "i dowolnej liczby N prob
 (wielkosci P i N o- graniczone
sa tylko pojemnoscia pamie-ci).
"
1170 ? "Dane X(P,N) wprowadza si
e instrukcjamiDATA (poczynajac o
d linii 1400)wg narastajacej war
tosci indeksu P."
1180 ? "Przyklad wpisywania dany
ch (dla P=2):"
1190 POSITION 1,14:? "1400 DATA
";CHR$(216);CHR$(168);CHR$(177);
CHR$(172);CHR$(177);CHR$(169);
1191 ? ",X(2,1),.,";CHR$(216);CH
R$(168);CHR$(177);CHR$(172);CHR$
(206);CHR$(169);",X(2,N)"
1200 ? "W inverse video zapisana
 jest zmienna niezalezna."
1210 ? :? "c.d. po nac. ";CHR$(2
10);CHR$(197);CHR$(212):INPUT Q$
1220 ? CHR$(125):POSITION 14,1:?
 "OZNACZENIA"
1230 ? :? "SIGMAX(P) - suma (po
N) wartosci zmien            nej
 X z indeksem P"
1240 ? :? "SIGMAX2(P) - suma kwa
dratow zmiennej X"
1250 ? :? "XSRED(P) -wartosc sre
dnia zmiennej X"
1260 ? :? "NS2X(P) - suma kwadra
tow odchylen po-          jedyn
czych wynikow od sred-
niej"
1270 ? :? "SIGMAXX(P,P) -suma il
oczynow miesza-               n
ych"
1280 ? :? "NSXX(P,P) - suma iloc
zynow mieszanych            odc
hylen"
1290 ? :? "c.d. po nac. ";CHR$(2
10);CHR$(197);CHR$(212):INPUT Q$
1300 RETURN
```

SPROSTOWANIE

W opublikowanym zeszycie nr 1 programów komputerowych „IKS-a" złośliwy chochlik drukarski „poobcinał" niektóre linie listingu programu MINI-ASEMBLER. Oto ich wykaz:

linia 100 — jest ...POP:GOTO, powinno być ...POP:GOTO START.

linia 3130 — jest ...:DEC = N2:, a powinno być ...:DEC = D2:...

linia 30010 — ostatnią zmienną w instrukcji READ jest START a nie STAR jak na listingu,

linia 30110 — w linii brakuje trzech znaków, oto one SR"

linia 30180 — na końcu linii powinno być ...MENU".PRINT a nie ...MENU":P.

Za powstałe z tego powodu kłopoty z uruchomieniem programu wszystkich czytelników serdecznie przepraszamy.

REDAKCJA



AMSTRAD

PROGRAM 11



Technologia przenoszenia programów ze Spectrum na Amstrada

Prezentowane rozwiązanie pozwala na przenoszenie programów czy zbiorów danych napisanych w języku Basic z mikrokomputera **Spectrum** na mikrokomputer **Amstrad.** Kazdy program (zbiór), który mozna wylistować na Spectrum komendami **LLIST** lub **LPRINT** może być przesłany, przy użyciu taśmy kasetowej, do mikrokomputera Amstrad. Uzyta może być dowolna wersja Spectrum (najstarsza 48 k, Spectrum + Spectrum 128 lub najnowszy Spectrum 128+2)

Z wersji mikrokomputerów Amstrad można zastosować CPC 464, CPC 664 lub CPC 6128 . Bardziej dogodna w użyciu jest dyskowa wersja Amstrada (CPC 664 lub CPC 6128), nie mniej w ostateczności mozna wykorzystać wersję taśmową CPC 464 Całość operacji realizują trzy z prezentowanych czterech programów. Pierwszy z nich (wydruk 1) rezyduje w górnej pamięci. Spectrum i dostarcza kody dla komendy SAVE, dzięki której poprzez instrukcje LLIST lub LPRINT następuje zapis przesyłanego programu (zbioru) danych na taśmę kasetową Jednocześnie przesyłany program (zbiór) jest zapisywany w taki sposób, że może być z łatwością odczytany na Amstradzie. Pozostałe trzy programy są związane z mikrokomputerem Amstrad. Pierwszy z nich (wydruk 2), to program w kodzie maszynowym, którego działanie wzorowane jest na instrukcji **LOAD** używanej w mikrokomputerze Spectrum. Postać binarną, **ZX.BIN** pierwszego programu (wydruk 2) wykorzystuje napisany w Basic'u program kontrolny (wydruk 3 lub wydruk 4).

Powyższe programy służą do przesyłania innych programów lub zbiorów danych, zapisanych w języku **Basic** — całej operacji nie należy jednak utozsamiać z translacją. nie mamy tu także do czynienia z kompatybilnością mikrokomputerów. Mówiąc prościej, chociaz omawiane cztery programy zapewniają przeniesienie innych programów (zbiorów danych) to jedynie najprostsze ich wersje nadają się natychmiast do uruchomienia na mikrokomputerze Amstrad. Języka Basic na mikrokomputer Amstrad nie nalezy utozsamiać z językiem Basic na Spectrum. W Basic'u na Spectrum jest wiele elementów, które będą sygnalizowane na Amstradzie jako błędy składniowe.

Przed operacją przesłania nalezy program przejrzeć i dokonać ewentualnych poprawek. Basic na Amstradzie dopuszcza w jednej linii programowej maksimum 255 znaków i dlatego zbyt długie linie programowe na Spectrum nalezy skrócić, wstawiając nowe linie. Wychwycenie zbyt długich linii programowych jest bardzo wazne, gdyż w innym razie ładowanie na Amstrada programu przeniesionego ze Spectrum, zakończy się błędem, a całość operacji trzeba będzie powtórzyć.

Poprawiania sygnalizowanych błędów składniowych można dokonywać po przeniesieniu programu na Amstrada, wykorzystując metodę kursora kopiującego, czy korzystając ze specjalizowanego oprogramowania mikrokomputerowego. Programy, posiadające bogatą grafikę będą przesyłane dłużej niż pozostałe. Programy, zawierające większą ilość tekstów i danych w kodzie ASCII, np. gry przygodowe powinny się dać łatwo przenieść.

Programy, w których występuje duża ilość instrukcji **PEEK** i **POKE** właściwie nie nadają się do przeniesienia. W celu przesłania programu ze Spectrum na Amstrada uzywa się taśmy kasetowej. Zapis programu na kasetę umozliwia program pokazany na wydruku pierwszym oraz instrukcja **LLIST** (przy przesyłaniu danych lepiej wykorzystać instrukcję **LPRINT**). Aby oznaczyć koniec przesyłanego programu należy jako ostatnią linię programową, wprowadzić instrukcję **REM © X,** gdzie symbol © oznacza **COPYRIGHT**

Przesyłany program wprowadzany jest najpierw do buforu RAM, a potem wysyłany na taśmę. Bezpośrednie przesłanie programu na kasetę instrukcją **LLIST** bajt po bajcie nie jest mozliwe. Program w kodzie maszynowym (wydruk 1) zajmuje górne adresy pamięci od F000H do F1DFH, natomiast przesyłany program w Basic'u może zająć pozostałą jej część. Program przesyłający (wydruk 1) nie jest uruchamiany automatycznie i dlatego przed użyciem instrukcji **LLIST** lub **LPRINT** należy podać:

RANDOMIZE USR 61440.

W tym momencie program binarny pozostaje w stanie oczekiwania na instrukcję **LLIST** lub **LPRINT**. Przesyłany program jest zapisywany na taśmie kasetowej z bufora pamięci, służącego do przechowywania obrazu ekranu tzw. video RAM (VRAM), zajmujący obszar od adresu 4000H do adresu 5B00H. Omawiany bufor służy do przechowywania danych wyjściowych skierowanych tam zamiast na drukarkę. Zapis na taśmę w postaci zbioru binarnego odbywa się, kiedy bufor VRAM będzie zapełniony lub, gdy wcześniej zakończy się wykonywanie instrukcji **LLIST**. Należy zauważyć, ze chociaż zapisywany jest zbiór binarny to przenoszona treść jest w swojej naturze czysto tekstowa.

Większość programów na Spectrum może być przesłana w jednym cyklu, ale niektóre szczególnie długie programy będą musiały być podzielone i przesłane w dwóch lub więcej częściach. Wydaje się to dziwne tym bardziej że oba mikrokomputery mogą mieć taką samą wielkość pamięci użytkowej RAM. Powyższa rozbieżność wynika stąd, ze program w Basic'u na Spectrum jest zwarty, gdy rezyduje w pamięci RAM, ale rozszerza się w trakcie listowania. Na przykład słowo kluczowe RANDOMIZE jest jednobajtowym symbolem w pamięci RAM, ale w trakcie listowania wyrazone będzie znakami zajmującymi 9 bajtów. Z drugiej strony liczby na Spectrum zajmują w czasie listowania mniej miejsca. O ile dany program się rozszerzy decyduje jego rodzaj. „Symbol graficzny", „Kształt" zdefiniowany przez uzytkownika zajmuje jeden bajt pamięci RAM, ale na Amstrad jest przesyłany jako trzycyfrowy kod w klamrach np. [133], zajmując aż cztery bajty. Przechodząc do praktycznych rozważań w pierwszej kolejności powinniśmy się dowiedzieć w ilu częściach będziemy musieli przesłać program i z którego programu kontrolnego należy skorzystać. W tym celu po załadowaniu na Spectrum, określamy górny adres, zajmowany przez interesujący nas program, podając:

PRINT PEEK (23628) * 256 + PEEK (23627).

- Jeśli uzyskana wartość jest mniejsza od 39 000 to przesłanie programu powinno się zrealizować w jednym cyklu. Na Amstradzie wykorzystujemy program kontrolny z wydruku trzeciego
- Jeśli uzyskana wartość jest zawarta pomiędzy 39 000 i 50 000 to przesłanie programu będzie zrealizowane także w jednym cyklu ale na Amstradzie powinien być wykorzystany program kontrolny z wydruku czwartego
- Jeśli uzyskana wartość jest większa od 50 000 to oznacza, że program powinien zostać podzielony na dwie części i przesłany w dwóch cyklach. Na Amstradzie należy wykorzystać program kontrolny z wydruku czwartego.

Po uruchomieniu jednego z dwóch programów kontrolnych pojawi się na monitorze pytanie

PODAJ ILOŚĆ ŁADOWANYCH SEGMENTÓW

Aby wpisać prawidłową ilość segmentów należy pamiętać, aby po zakończeniu instrukcji LLIST wprowadzić:

PRINT PEEK (61506) — 64

która spowoduje wypisanie liczby, określającej ilość segmentów. W celu przywrócenia użytym instrukcjom LLIST lub LPRINT ich normalnego działania należy podać:

RANDOMIZE USR 61490

Po zapisaniu na taśmę kasetową przesyłanego programu nie mozna stwierdzić czy treść programu na taśmie jest poprawna. Aby to sprawdzić wgrywamy program z taśmy do pamięci Spectrum. Kazdy błąd będzie sygnalizowany jako błąd ładowania taśmy.

Nadszedł wreszcie moment, aby wziąć taśmę z poprawnie nagranym programem i włożyć ją do odtwarzacza kasetowego, współpracującego z Amstradem. W pamięci Amstrada powinny rezydować dwa programy związane z przesyłaniem programu ze Spectrum. Pierwszy o nazwie ZX. BIN zapisany w kodzie maszynowym (wydruk 2) rezyduje w pamięci od adresu 8000. W wersjach dyskowych Amstrada adres ten zaliczany jest do obszaru pamięci górnej. Drugi program (wydruk 3 lub wydruk 4) jest programem kontrolnym i mieści się w pamięci dolnej. Po załadowaniu z taśmy kasetowej przesyłanego programu do pamięci RAM Amstrada, program kontrolny przepisze ten załadowany program na dyskietkę lub taśmę w postaci zbioru ASCII. Cała operacja jest realizowana w Basic'u i dlatego użytkownicy, wykorzystujący pamięć dyskową, zauważą nietypowe opóźnienia, przy realizowanym etapami zapisie programu na dyskietkę. W wyniku działania programu kontrolnego na dyskietce (taśmie) zostanie utworzony zbiór X.ASC, gdzie X jest podaną przez użytkownika nazwą przesyłanego programu, a ASC określa typ zbioru i jest dopisywany automatycznie przez program kontrolny.

Po zakończeniu pracy program kontrolny pozostawia MEMORY ustawione na wartość 3FFF i dlatego należy podać:
MEMORY & 7FFF
gdyż z chwilą ładowania programu do pamięci może pojawić się komunikat błędu przekroczenia pamięci. Po poprawnym załadowaniu do pamięci można rozpocząć edycję (poprawienie) przesłanego programu. Ładowanie zakończy się komunikatem błędu, jeśli linia programu zawiera więcej niż 255 znaków

ABY PRZESŁAĆ PROGRAM W BASIC'U Z MIKROKOMPUTERA SPECTRUM NA MIKROKOMPUTER AMSTRAD NALEŻY POSTĘPOWAĆ WEDŁUG NASTĘPUJĄCEGO ALGORYTMU:

1. Ładujemy do Spectrum program z wydruku „1" komendą **LOAD.**

2. Uruchamiamy program z wydruku „1" komendą **RUN.** W wyniku działania, program będzie chciał zapisać swoją postać binarną na taśmie (włączanie magnetofonu nie jest konieczne).

3. Ładujemy do Spectrum komendą **LOAD** program w BASIC'u, który ma być przesłany na mikrokomputer Amstrad.

4. Należy sprawdzić czy program nie zawiera zbyt długich linii. Jeśli ilość znaków w liniach programowych przekracza liczbę 255, należy utworzyć nowe linie.

5. Wprowadzamy do programu jako ostatnią linię wyrażenie: **REM © X**

6. Określamy górny adres pamięci przesyłanego programu podając:
PRINT PEEK (23628) * 256+-**PEEK** (23627)

— Jeśli uzyskana wartość jest mniejsza od 39 000, to przesłanie programu realizujemy w jednym cyklu. Na Amstradzie powinien być wykorzystany program kontrolny z wydruku trzeciego.

— Jeśli uzyskana wartość jest zawarta pomiędzy 39 000 i 50 000, to przesłanie programu będzie zrealizowane także w jednym cyklu, ale na Amstradzie powinien być wykorzystany program kontrolny z wydruku czwartego.

— Jeśli uzyskana wartość wynosi powyżej 50 000 to oznacza, że program powinien zostać podzielony na dwie części i przesłany w dwóch cyklach. Na Amstradzie powinien być wykorzystany program kontrolny z wydruku czwartego.

7. Wprowadzamy rozkaz:
RANDOMIZE USR 61440

8. Wkładamy do magnetofonu kasetę z taśmą w celu nagrania przesyłanego programu i włączamy magnetofon na ZAPIS.

9. Wprowadzamy rozkaz:
LLIST

10. Po zakończeniu wykonywania instrukcji **LLIST** zatrzymujemy magnetofon. Sprawdzamy z ilu segmentów składa się przesłany przez nas program podając:
PRINT PEEK (61506) — 64

11. Ładujemy do Amstrada komendą **LOAD** program z wydruku drugiego.

12. Uruchamiamy załadowany program komendą **RUN**, w wyniku czego na dyskietce (taśmie) zostanie zapisany program binarny **ZX. BIN.**

13. W zależności od wartości liczbowej otrzymanej w p. 6 ładujemy do Amstrada komendą **LOAD** program kontrolny z wydruku trzeciego lub wydruku czwartego.

14. Podłączamy do Amstrada magnetofon z zapisanym na taśmie kasetowej programem ze Spectrum.

15. Uruchamiamy na Amstradzie komendą **RUN** program kontrolny, podając w trakcie działanie: ilość segmentów przesyłanego programu (patrz punkt 10) i jego nazwę (maksimum 6 znaków). Z chwilą pojawienia się komunikatu związanego z ładowaniem segmentu, włączamy magnetofon, umożliwiając tym samym zapis przesyłanego programu do pamięci RAM Amstrada.

16. Po załadowaniu do pamięci RAM przygotowujemy dyskietkę (taśmę) do zapisu przesłanego ze Spectrum programu i naciskamy dowolny klawisz.

17. Po zapisaniu programu na dyskietkę (taśmę) ściągamy go do pamięci i dokonujemy niezbędnych poprawek po czym uruchamiamy komendą RUN poprawiony program.

UWAGA:

Jeśli przesyłamy program w dwóch częściach, to łączymy obie na Amstradzie instrukcją MERGE.

Życzymy owocnej pracy!

Na podstawie „YUOR COMPUTER" nr 1/87, 3/87 opracował:
Jacek SZANIAWSKI
Marian MARCINIAK

Wydruk nr 1

```
10 REM PROGRAM PRZENOSZACY ZBIORY Z M/K SPECT
RUM NA M/K AMSTRAD
20 REM PROGRAM NA M/K SPECTRUM
30 CLEAR 61439: RESTORE 160
40 LET a=61440: LET d=499
50 LET s=0: LET d=d+1
60 FOR r=1 TO 8
70 READ v
80 LET s=s+v: POKE a,v: LET a=a+1
90 NEXT r
100 READ chk: IF chk<>s THEN PRINT "WIERSZ O
NR ";d;" JEST NIEPRAWIDLOWY":STOP
110 IF a>61919 THEN GOTO 130
120 GOTO 50
130 PRINT "LADOWANIE"
140 PRINT "ZACHOWANIE ZBIORU NA TASMIE..."
150 SAVE "transfer" code 61440,488: STOP
160 DATA 42,79,92,17,15,0,25,94,364
170 DATA 35,86,0,0,0,0,17,37,175
180 DATA 241,43,115,35,114,221,33,82,884
190 DATA 240,221,54,0,0,221,54,1,791
200 DATA 0,221,54,2,0,221,54,3,555
210 DATA 255,221,54,4,63,221,54,240,1112
220 DATA 64,201,42,79,92,17,15,0,510
230 DATA 25,237,91,87,240,115,35,114,944
240 DATA 201,3,64,32,32,32,32,32,428
250 DATA 32,32,32,32,210,1,0,64,403
260 DATA 0,0,1,0,0,147,65,244,457
270 DATA 9,254,1,18,0,174,16,39,511
280 DATA 232,3,100,0,10,0,1,0,346
290 DATA 0,0,0,0,0,197,58,66,321
300 DATA 240,60,50,66,240,221,33,82,992
```

```
310 DATA 240,221,115,250,221,114,251,221,1633
320 DATA 33,65,240,213,17,17,0,175,760
330 DATA 205,194,4.209,6,50,118,16,802
340 DATA 253,221,33,0,64,62,255,205,1093
350 DATA 194.4,193,201,197,33,0,64,886
360 DATA 54,13,17,1,64,1,255,26,431
370 DATA 237,176,193,201,229,213,197,79,1525
380 DATA 58,82,240,254,0,32,8,205,879
390 DATA 156,240,62,1,50,82,240,58,889
400 DATA 83,240,254,1,40,54,42,85,799
410 DATA 240,35,34,85,240,124,254,91,1103
420 DATA 32,20,125,254,0,32,15,17,495
430 DATA 0,27,205,109,240,205,156,240,1182
440 DATA 33,0,64,34,85,240,121,254,831
450 DATA 127,40,54,119,58,84,240,254,976
460 DATA 1,40,20,121,254,127,40,8,611
470 DATA 175,50,84,240,193,209,225,201,1377
480 DATA 62,1,50,84,240,24,245,121,827
490 DATA 254,88,32,231,17,255,63,237,1177
500 DATA 82,235,205,109,240,62,1,50,984
510 DATA 83,240,6,24,205,68,14,24,664
520 DATA 219,62,99,24,198,245,254,13,1114
530 DATA 40,86,254,32,56,8,254,128,858
540 DATA 56,78,254,165,48,74,71,58,804
550 DATA 92,240,183,120,32.61,38,0,766
560 DATA 111,17,104,240,221,33,94,240,1060
570 DATA 62,47,221,78,0,221,70,1,700
580 DATA 198,1,237,66,48,250,9,18,827
590 DATA 221,35,221,35,19,13,32,232,808
600 DATA 62,91,205,152,241,58,106,240,1155
610 DATA 205,152,241,58,107,240,205,152,1360
620 DATA 241,58,108,240,205,152,241,62,1307
630 DATA 93,24,24,58,93,240,24,19,575
640 DATA 254,255,40,8,254,165,56,11,1043
650 DATA 254,255,48,7,214,165,205.95,1243
660 DATA 11,241,201,205,152,241,241,201,1493
670 DATA 229,111,58,89,240,60,103,58,948
680 DATA 91,240,60,50,91,240,188,32,992
690 DATA 14,205,198,241,175,50,91,240,1214
700 DATA 125,254,13,40,15,24,227,125,823
710 DATA 254,13,32,5,175,50,91,240,860
720 DATA 125,205,204,241,225,201,62,13,1276
730 DATA 205,204,241,201,245,205,172,240,1713
740 DATA 254,13,32,10,58,90,240,32.729
750 DATA 5,62,10,205,172,240,241,201,1136
760 DATA 0,0,0,0,0,0,0,0,0
```

Wydruk nr 2

```
10 REM PROGRAM LADUJACY NA M/K AMSTRAD
20 MEMORY &3FFF:RESTORE 150
30 A=&8000:D=499
40 S=0:D=D+1
50 FOR R=1 TO 8
60 READ V$:V=VAL("&"+V$)
70 S=S+V:POKE A,V:A=A+1
80 NEXT R
90 READ V$:CHK=VAL("&"+V$)
100 IF CHK<>S THEN GOTO 130
110 IF A>&812F THEN SAVE "ZX.BIN",B,&8000,
304:STOP
120 GOTO 40
130 PRINT "WIERSZ O NR ";D;" JEST BLEDNY"
140 STOP
150 DATA cd,09,80,c9,00,40,00,00,025f
160 DATA 00,cd,a7,bc,cd,6e,bc,f3,051a
170 DATA af,32,06,80,08,d9,f5,d5,0412
180 DATA 16,10,d9,01,0e,f4,ed,49,0338
190 DATA 01,d0,f6,ed,49,0e,10,ed,0408
200 DATA 49,01,92,f7,ed,49,01,58,0362
210 DATA f6,ed,49,dd,21,2b,81,dd,04b3
220 DATA e5,11,11,00,af,cd,7f,80,0382
230 DATA dd,e1,30,ef,3a,2b,81,fe,04c1
240 DATA 03,20,e8,3e,ff,ed,5b,36,03c6
250 DATA 81,ed,53,07,80,dd,2a,04,0353
260 DATA 80,cd,7f,80,08,d9,d1,f1,04ef
270 DATA d9,08,01,82,f7,ed,49,01,0392
280 DATA 10,f6,ed,49,dd,22,04,80,03bf
290 DATA fb,f5,cd,71,bc,f1,38,01,0514
300 DATA c9,3e,01,32,06,80,c9,14,029d
310 DATA 08,15,3e,f5,db,00,e6,80,0391
320 DATA d9,1e,4c,d9,4f,bf,c0,cd,04b7
330 DATA 02;81,30,fa,21,15,04,10,01f7
340 DATA fe,2b,7c,b5,20,f9,cd,fe,053e
350 DATA 80,30,eb,06,9c,cd,fe,80,0488
360 DATA 30,e4,3e,c0,b8,30,e0,24,03fe
370 DATA 20,f1,06,c9,cd,02,81,30,0360
380 DATA d5,78,fe,d4,30,f4,cd,02,0512
390 DATA 81,d0,d9,1e,4a,d9,26,00,0391
400 DATA 06,b0,18,18,08,20,05,dd,01f0
410 DATA 75,00,18,0a,cb,19,ad,c0,02e8
420 DATA 79,17,4f,13,18,02,dd,23,020c
430 DATA 1b,08,06,b2,2e,01,cd,fe,02d5
440 DATA 80,d0,3e,bf,b8,cb,15,06,03eb
450 DATA b0,30,f3,7c,ad,67,7a,b3,0490
460 DATA 20,d2,7c,fe,01,c9,cd,02,0405
470 DATA 81,d0,3e,16,3d,20,fd,a7,03a6
480 DATA 04,c8,3e,f4,db,00,e6,04,03c3
490 DATA ee,04,c0,3e,f5,db,00,a9,0469
500 DATA e6,80,28,ec,79,2f,4f,d9,044a
510 DATA 7b,ee,1f,5f,ed,51,ed,59,046b
520 DATA d9,37,c9,00,00,00,00,00,01d9
```

Wydruk nr 3

```
10 REM PROGRAM KONTROLNY NR 3 (BASIC-AMSTRAD)
20 |DISC:::BORDER 13:INK 0,23:INK 1,0:INK 2,0:INK 3,11
30 KEY 9,CHR$(174):MODE 2:MEMORY &3FFF:LOAD "zx.bin":|TAPE.IN
40 POKE &8004,0:POKE &8005,&40 : 'PRZEZNACZA SIE ADRES &4000
50 TL=0:S=1:INPUT "PODAJ ILOSC LADOWANYCH SEGMENTOW ";N
60 PRINT "PODAJ NAZWE PRZESYLANEGO ZBIORU (MAX 6 ZNAKOW)"
70 PRINT "PODANA NAZWA NIE MOZE ZAWIERAC ROZSZERZENIA"
80 INPUT "NAZWA ZBIORU ";A$:A$=UPPER$(A$):A$=A$+".ASC":PRINT
90 PRINT "WLACZ KASETE ZE ZBIOREM DO PRZEGRANIA":PRINT
100 :
110 FOR R=1 TO N
120 PRINT "LADOWANIE SEGMENTU O NR ";S;" ZBIORU ZE SPEKTRUM":CA
LL &8000
130 F=PEEK(&8006):IF F=0 THEN PRINT "BLEDNE LADOWANIE ZBIORU":S
TOP
140 L=((PEEK(&8008))*256)+PEEK(&8007):TL=TL+L
150 PRINT "WIELKOSC ZALADOWANEGO DO PAMIECI SEGMENTU O NR ";S;"
 WYNOSI ";HEX$(L,4);"H"
160 S=S+1:PRINT:NEXT R
170 :
180 PRINT "WCISNIJ DOWOLNY KLAWISZ GDY DYSKIETKA(TASMA) JEST PR
ZYGOTOWANA DO ZAPISU"
190 PRINT "ZBIOR O NAZWIE ";A$
200 IF INKEY$="" THEN GOTO 200
210 PRINT:PRINT "ZAPIS ZBIORU - PROSZE CZEKAC":OPENOUT A$
220 FOR A=&4000 TO &4000+TL-3
230 V$=CHR$(PEEK(A)):PRINT #9,V$;:NEXT A
240 CLOSEOUT:|DISC:STOP
```

Wydruk nr 4

```
10 REM PROGRAM KONTROLNY NR 4 (BASIC-AMSTRAD)
20 |DISC:::BORDER 13:INK 0,23:INK 1,0:INK 2,0:INK 3,11
30 KEY 9,CHR$(174):MODE 2:MEMORY &3FFF:LOAD "zx.bin":|TAPE.IN
40 INPUT "PODAJ ILOSC LADOWANYCH SEGMENTOW ";N:T=64:S=1
50 PRINT "PODAJ NAZWE PRZESYLANEGO ZBIORU (MAX 6 ZNAKOW)"
60 PRINT "PODANA NAZWA NIE MOZE ZAWIERAC ROZSZERZENIA"
70 INPUT "NAZWA ZBIORU ";A$:A$=UPPER$(A$):CLS
80 IF LEN(A$)>6 THEN A$=LEFT$(A$,6)
90 :
100 FOR R=1 TO N:CLS
110 PRINT "ZALOZ TASME ZE ZBIOREM DO PRZEGRANIA"
120 PRINT "WCISNIJ DOWOLNY KLAWISZ"
130 IF INKEY$="" THEN GOTO 130
140 POKE &8004,0:POKE &8005,&40:T=T+1
150 PRINT "LADOWANIE SEGMENTU O NR ";S;"Z TASMY ZE SPEKTRUM"
160 CALL &8000:F=PEEK(&8006):IF F=0 THEN PRINT "BLEDNE LADOWANI
E":STOP
170 L=((PEEK(&8008))*256)+PEEK(&8007):B$=A$+"-"+CHR$(T)+".ASC"
180 S=S+1:PRINT "WLOZ DYSKIETKE(TASME) DO ZAPISU ZBIORU"
190 PRINT "ZAPIS ZBIORU <<";B$;">> - WCISNIJ DOWOLNY KLAWISZ"
200 IF INKEY$="" THEN GOTO 200
210 PRINT:PRINT "ZAPIS ZBIORU - PROSZE CZEKAC":OPENOUT B$
220 FOR A=&4000 TO &4000+L-2
230 V$=CHR$(PEEK(A)):PRINT #9,V$;:NEXT A
240 CLOSEOUT:NEXT R
250 |DISC:STOP
```

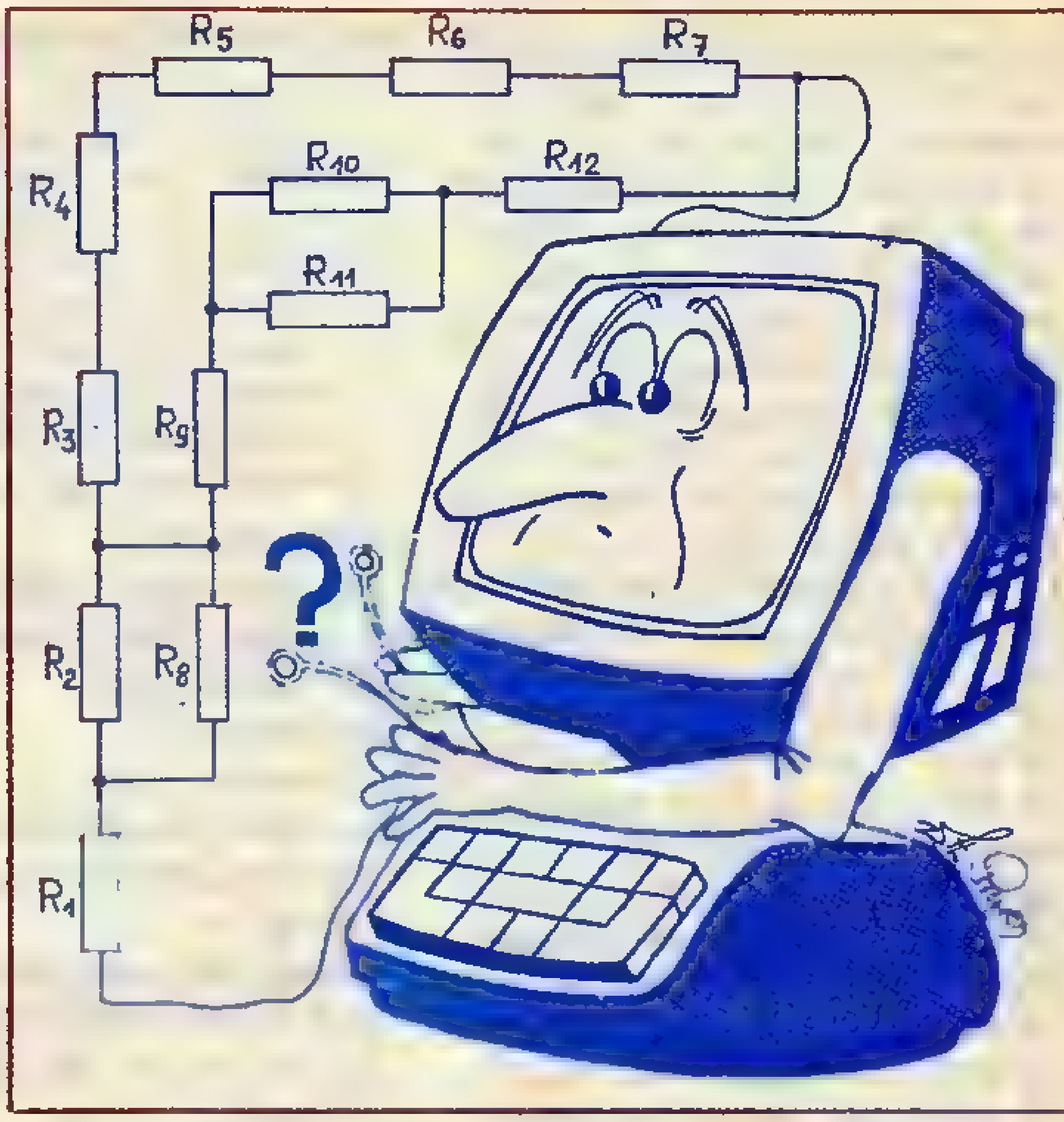
AMSTRAD

PROGRAM 12

„REZYSTOR”

Po załadowaniu i uruchomieniu programu komendą RUN na ekranie monitora pojawi się opis czterech czynności. Chcąc wybrać jedną z czynności przyciskamy klawisz 1,2,3 lub 4. Pierwsze dwie czynności przedstawiają szeregowe i równoległe połączenie rezystorów. Obok rysunków typowych schematów pokazane są wzory na rezystancję zastępczą szeregowo lub równolegle połączonych rezystorów. Kolejne dwie czynności przedstawiają rysunki oraz umożliwiają obliczenie rezystancji obwodu przy przejściu z połączenia rezystorów w „trójkąt” w połączenie w „gwiazdę” i odwrotnie.

Program ten okaże się pomocny w nauce „Teorii obwodów”.




```
10 MODE 1
20 WINDOW #1,10,27,23,25
30 PRINT:PRINT
40 PRINT "1. POLACZENIE SZEREGOWE REZYSTOROW"
50 PRINT:PRINT
60 PRINT "2. POLACZENIE ROWNOLEGLE REZYSTOROW"
70 PRINT:PRINT
80 PRINT "3. OBLICZANIE REZYSTANCJI ZASTEPCZEGO"
90 PRINT "   POLACZENIA TYPU 'TROJKAT' PRZY"
100 PRINT "   PRZEJSCIU Z POLACZENIA TYPU 'GWIAZDA'   W POLACZENI
E TYPU 'TROJKAT'"
110 PRINT:PRINT
120 PRINT "4. OBLICZANIE REZYSTANCJI ZASTEPCZEGO"
130 PRINT "   POLACZENIA TYPU 'GWIADZDA' PRZY"
140 PRINT "   PRZEJSCIU Z POLACZENIA TYPU 'TROJKAT'   W POLACZENI
E TYPU 'GWIAZDA'"
150 INK 1,24:PEN #1,0,0:PAPER #1,13
160 LOCATE #1,11,24
170 PRINT #1," WYBIERZ CZYNNOSC"
180 a$=INKEY$
190 IF a$="" THEN 180
200 IF a$="1" THEN 250
210 IF a$="2" THEN 630
220 IF a$="3" THEN 1090
230 IF a$="4" THEN 1810
240 GOTO 180
250 REM ########################################
260 REM #                                      #
270 REM #   POLACZENIE SZEREGOWE REZYSTOROW    #
280 REM #                                      #
290 REM ########################################
300 MODE 1
310 PRINT SPC(4);"POLACZENIE SZEPEGOWE REZYSTOROW"
320 LOCATE 7,7:PRINT CHR$(79)
330 LOCATE 5,9:PRINT "U"
340 MOVE 103,233:DRAWR 0,50:DRAWR -5,-10:DRAWR 5,10:DRAWR 5,-10
350 MOVE 114,297:DRAWR 26,0:DRAWR 0,8:DRAWR 34,0:DRAWR 0,-17:DRAW
R -34,0:DRAWR 0,8:DRAWR 0,-8:DRAWR 34,0:DRAWR 0,8:DRAWR 34,0
360 DRAWR 0,-22:DRAWR 8,0:DRAWR 0,-34:DRAWR -17,0:DRAWR 0,34:DRAW
R 8,0:DRAWR -8,0:DRAWR 0,-34:DRAWR 9,0
370 DRAWR 0,-22:DRAWR -34,0:DRAWR 0,8:DRAWR 0,-17:DRAWR -34,0:DRA
WR 0,17:DRAWR 34,0:DRAWR -34,0:DRAWR 0,-9:DRAWR -28,0
380 LOCATE 7,12:PRINT CHR$(79)
390 LOCATE 10,4:PRINT "R":LOCATE 11,5:PRINT "1"
400 LOCATE 15,9:PRINT "R":LOCATE 16,10:PRINT "2"
410 LOCATE 10,14:PRINT "R":LOCATE 11,15:PRINT "3"
420 MOVE 441,297:DRAWR 56,0:DRAWR 0,-22:DRAWR 8,0:DRAWR 0,-34:DRA
WR -17,0:DRAWR 0,34:DRAWR 8,0:DRAWR -8,0:DRAWR 0,-34:DRAWR 9,0
430 DRAWR 0,-22:DRAWR -56,0
440 LOCATE 33,9:PRINT "R"
450 LOCATE 34,10:PRINT "Z"
460 LOCATE 28,7:PRINT CHR$(79)
470 LOCATE 28,12:PRINT CHR$(79)
480 LOCATE 26,9:PRINT "U"
490 MOVE 438,233:DRAWR 0,50:DRAWR -5,-10:DRAWR 5,10:DRAWR 5,-10
500 MOVE 297,250:DRAWR 40,0
510 MOVE 297,270:DRAWR 40,0
520 MOVE 357,260:DRAWR -20,20:DRAWR 20,-20:DRAWR -20,-20
530 LOCATE 6,18:PRINT "REZYSTANCJE ZASTEPCZA OBWODU"
540 LOCATE 6,19:PRINT "WYZNACZAMY ZE WZORU:"
550 LOCATE 10,21:PRINT "R  = R  + R  + R"
560 LOCATE 11,22:PRINT "Z   1   2   3"
570 WINDOW #2,2,39,24,25
580 INK 1,24:PEN #2,0,0:PAPER #2,13
590 LOCATE #2,3,25
600 PRINT #2,"POWROT DO OPISU - [DOWOLNY KLAWISZ]"
610 IF INKEY$="" THEN 610
620 GOTO 10
630 REM ########################################
640 REM #                                      #
650 REM #   POLACZENIE ROWNOLEGLE REZYSTOROW   #
660 REM #                                      #
670 REM ########################################
680 MODE 1
690 PRINT SPC(4);"POLACZENIE ROWNOLEGLE REZYSTOROW"
700 MOVE 90,326:DRAWR 154,0:DRAWR 0,-22:DRAWR 8,0:DRAWR 0,-34:DRA
WR -17,0:DRAWR 0,34:DRAWR 8,0:DRAWR -8,0:DRAWR 0,-34:DRAWR 9,0
710 DRAWR 0,-22:DRAWR -164,0
720 MOVE 180,326:DRAWR 0,-22:DRAWR 8,0:DRAWR 0,-34:DRAWR -17,0:DR
AWR 0,34:DRAWR 8,0:DRAWR -8,0:DRAWR 0,-34:DRAWR 9,0:DRAWR 0,-22
730 MOVE 116,326:DRAWR 0,-22:DRAWR 8,0:DRAWR 0,-34:DRAWR -17,0:DR
AWR 0,34:DRAWR 8,0:DRAWR -8,0:DRAWR 0,-34:DRAWR 9,0:DRAWR 0,-22
740 LOCATE 5,5:PRINT CHR$(79)
750 LOCATE 5,10:PRINT CHR$(79)
760 LOCATE 3,7:PRINT "U"
```

```
770 MOVE 70, 262:DRAWR 0,50:DRAWR -5,-10:DRAWR 5,10:DRAWR 5,-10
780 MOVE 80,326:DRAWR 10,0
790 LOCATE 9,7:PRINT "R"
800 LOCATE 10,8:PRINT "1"
810 LOCATE 13,7:PRINT "R"
820 LOCATE 14,8:PRINT "2"
830 LOCATE 17,7:PRINT "R"
840 LOCATE 18,8:PRINT "3"
850 MOVE 461,326:DRAWR 56,0:DRAWR 0,-22:DRAWR 8,0:DRAWR 0,-34:DRA
WR -17,0:DRAWR 0,34:DRAWR 8.0:DRAWR -8,0:DRAWR 0,-34:DRAWR 9,0:DR
AWR 0,-22:DRAWR -56,0
860 LOCATE 29,5:PRINT CHR$(79)
870 LOCATE 29,10:PRINT CHR$(79)
880 LOCATE 34,7:PRINT "R"
890 LOCATE 35,8:PRINT "Z"
900 MOVE 455,262:DRAWR 0,50:DRAWR -5,-10:DRAWR 5,10:DRAWR 5,-10
910 LOCATE 27,7:PRINT "U"
920 MOVE 320,280:DRAWR 40,0
930 MOVE 320,300:DRAWR 40,0
940 MOVE 380,290:DRAWR -20,20:DRAWR 20,-20:DRAWR -20,-20
950 LOCATE 5.14:PRINT "REZYSTANCJE ZASTEPCZA OBWODU"
960 LOCATE 5,15:PRINT "WYZNACZAMY ZE WZORU:"
970 LOCATE 27,17:PRINT "R R R"
980 LOCATE 2,18:PRINT "1   1.  1   1":LOCATE 28,18:PRINT "1 2 3"
990 LOCATE 2,19:PRINT "- = - + - + -     R ="
1000 LOCATE 23,19:PRINT "----------------"
1010 LOCATE 2,20:PRINT "R   R   R   R";SPC(5);"Z  R R + R R + R R
"
1020 LOCATE 3,21:PRINT "Z   1   2   3";SPC(8);"1 2   1 3   2 3"
1030 WINDOW #2,2,39,24,25
1040 INK 1,24:PEN #2,0,0:PAPER #2,13
1050 LOCATE #2.3.25
1060 PRINT #2,"POWROT DO OPISU - [DOWOLNY KLAWISZ]"
1070 IF INKEY$="" THEN 1070
1080 GOTO 10
1090 REM ***********************************************
1100 REM *                                             *
1110 REM *  PRZEJSCIE Z POLACZENIA REZYSTOROW W 'TROJKAT'  *
1120 REM *  W POLACZENIE REZYSTOROW W 'GWIAZDE'           *
1130 REM *                                             *
1140 REM ***********************************************
1150 MODE 2
1160 PRINT SPC(4);"OBLICZANIE REZYSTANCJI ZASTEPCZEGO POLACZENIA
TYPU 'TROJKAT'"
1170 PRINT SPC(4);"PRZY PRZEJSCIU Z POLACZENIA TYPU 'GWIAZDA' W P
OLACZENIE TYPU 'TROJKAT'"
1180 LOCATE 76,5:PRINT "R R"
1190 LOCATE 77,6:PRINT "A C"
1200 LOCATE 63,7:PRINT "R  = R + R + ----"
1210 LOCATE 64,8:PRINT "1    A   C   R"
1220 LOCATE 78,9:PRINT "B"
1230 LOCATE 76,12:PRINT "R R"
1240 LOCATE 77,13:PRINT "A B"
1250 LOCATE 63,14:PRINT "R  = R + R + ----"
1260 LOCATE 64,15:PRINT "2    A   B   R"
1270 LOCATE 78,16:PRINT "C"
1280 LOCATE 76,19:PRINT "R R"
1290 LOCATE 77,20:PRINT "B C"
1300 LOCATE 63,21:PRINT "R  = R + R + ----"
1310 LOCATE 64,22:PRINT "3    B   C   R"
1320 LOCATE 78,23:PRINT "A"
1330 MOVE 20,220:DRAWR 10,10:DRAWR -3,3:DRAWR 20,20:DRAWR 7,-7:DR
AWR -20,-20:DRAWR -4,4
1340 MOVE 51,250:DRAWR 10,10:DRAWR 0,16:DRAWR -4,0:DRAWR 0,28:DRA
WR 9,0:DRAWR 0,-28:DRAWR -6,0:MOVE 61,304:DRAWR 0,16
1350 MOVE 61,260:DRAWR 10,-10:DRAWR -3,-3:DRAWR 20,-20:DRAWR 7,7:
DRAWR -20,20:DRAWR -4,-4:MOVE 92,230:DRAWR 10,-10
1360 LOCATE 1,12:PRINT "A"
1370 LOCATE 8,4:PRINT "C"
1380 LOCATE 16,12:PRINT "B"
1390 LOCATE 3,9:PRINT "R":LOCATE 4,10:PRINT "A"
1400 LOCATE 5,6:PRINT "R":LOCATE 6,7:PRINT "C"
1410 LOCATE 12,9:PRINT "R":LOCATE 13,10:PRINT "B"
1420 MOVE 240,220:DRAWR 40,40:DRAWR -4,4:DRAWR 20,20:DRAWR 9,-9:D
RAWR -20,-20:DRAWR -5,5:MOVE 300,280:DRAWR 40,40
1430 MOVE 340,320:DRAWR 40,-40:DRAWR 4,4:DRAWR 20,-20:DRAWR -9,-9
:DRAWR -20,20:DRAWR 5,5:MOVE 440,220:DRAWR -40,40
1440 MOVE 440,220:DRAWR -84,0:DRAWR 0,6:DRAWR -32,0:DRAWR 0,-13:D
RAWR 32,0:DRAWR 0,7:MOVE 240,220:DRAWR 84,0
1450 LOCATE 29,12:PRINT "A":LOCATE 58,12:PRINT "B":LOCATE 43,4:PR
INT "C"
1460 LOCATE 34,7:PRINT "R":LOCATE 35,8:PRINT "1"
1470 LOCATE 50,7:PRINT "R":LOCATE 51,8:PRINT "3"
1480 LOCATE 43,9:PRINT "R":LOCATE 44,10:PRINT "2"
1490 MOVE 150,270:DRAWR 30,0:MOVE 150,286:DRAWR 30,0:MOVE 192,278
:DRAWR -15.15:MOVE 192,278:DRAWR -15,-15
1500 LOCATE 2,15:PRINT " PODAJ REZYSTANCJE "
1510 LOCATE 24,15:PRINT " REZYSTANCJE UKLADU W TROJKAT :"
1520 LOCATE 28,17:PRINT "R  =":LOCATE 29,18:PRINT "1"
1530 LOCATE 28,19:PRINT "R  =":LOCATE 29,20:PRINT "2"
1540 LOCATE 28,21:PRINT "R  =":LOCATE 29,22:PRINT "3"
1550 LOCATE 3,17:PRINT "R  =":LOCATE 4,18:PRINT "A"
1560 LOCATE 8,17:INPUT ra
1570 IF ra<=0 OR ra>10000 THEN GOTO 1740
1580 LOCATE 3,19:PRINT "R  =":LOCATE 4.20:PRINT "B"
1590 LOCATE 8,19:INPUT rb
1600 IF rb<=0 OR rb>10000 THEN GOTO 1760
1610 LOCATE 3,21:PRINT "R  =":LOCATE 4,22:PRINT "C"
1620 LOCATE 8,21:INPUT rc
1630 IF rc<=0 OR rc>10000 THEN GOTO 1780
1640 r1=ra+rc+(ra*rc)/rb
1650 r2=ra+rb+(ra*rb)/rc
1660 r3=rb+rc+(rb*rc)/ra
1670 LOCATE 33.17:PRINT r1:LOCATE 33,19:PRINT r2:LOCATE 33,21:PRI
NT r3
1680 GOTO 570
1690 WINDOW #5,14.65,24,25
1700 INK 1,24:PEN #5,0,0:PAPER #5,13
1710 LOCATE #5.16,25:PRINT #5,"  PODAJ WARTOSC REZYSTANCJI Z PRZE
DZIALU <1,10000>"
1720 FOR i=1 TO 3000:NEXT:LOCATE 14,24:PRINT SPC(55):LOCATE 14,25
:PRINT SPC(55)
1730 RETURN
1740 GOSUB 1690
1750 LOCATE 6,17:PRINT SPC(12):GOTO 1560
1760 GOSUB 1690
1770 LOCATE 6,19:PRINT SPC(12):GOTO 1590
1780 GOSUB 1690
1790 LOCATE 6,21:PRINT SPC(12):GOTO 1620
1800 GOTO 580
1810 REM ***********************************************
1820 REM *                                             *
1830 REM *  PRZEJSCIE Z POLACZENIA REZYSTOROW W 'GWIAZDE'  *
1840 REM *  W POLACZENIE REZYSTOROW W 'TROJKAT'           *
1850 REM *                                             *
1860 REM ***********************************************
1870 MODE 2
1880 PRINT SPC(4);"OBLICZANIE REZYSTANCJI ZASTEPCZEGO POLACZENIA
TYPU 'GWIAZDA'"
1890 PRINT SPC(4);"PRZY PRZEJSCIU Z POLACZENIA TYPU 'TROJKAT' W P
OLACZENIE TYPU 'GWIAZDA'"
```

```
1900 LOCATE 70,5:PRINT "R R"
1910 LOCATE 71,6:PRINT "1 2"
1920 LOCATE 62,7:PRINT "R = ----------"
1930 LOCATE 63,8:PRINT "A   R + R + R"
1940 LOCATE 68,9:PRINT "1   2   3"
1950 LOCATE 70,12:PRINT "R R"
1960 LOCATE 71,13:PRINT "2 3"
1970 LOCATE 62,14:PRINT "R = ----------"
1980 LOCATE 63,15:PRINT "B   R + R + R"
1990 LOCATE 68,16:PRINT "1   2   3"
2000 LOCATE 70,19:PRINT "R R"
2010 LOCATE 71,20:PRINT "1 3"
2020 LOCATE 62,21:PRINT "R = ----------"
2030 LOCATE 63,22:PRINT "C   R + R + R"
2040 LOCATE 68,23:PRINT "1   2   3"
2050 MOVE 20,210:DRAWR 40,40:DRAWR -4,4:DRAWR 20,20:DRAWR 9,-9:DR
AWR -20,-20:DRAWR -5,5:MOVE 80,270:DRAWR 40,40
2060 MOVE 120,310:DRAWR 40,-40:DRAWR 4,4:DRAWR 20,-20:DRAWR -9,-9
:DRAWR -20,20:DRAWR 5,5:MOVE 220,210:DRAWR -40,40
2070 MOVE 220,210:DRAWR -84,0:DRAWR 0,6:DRAWR -32,0:DRAWR 0,-13:D
RAWR 32,0:DRAWR 0,7:MOVE 20,210:DRAWR 84,0
2080 LOCATE 2,13:PRINT "A":LOCATE 29,13:PRINT "B":LOCATE 15,5:PRI
NT "C"
2090 LOCATE 7,7:PRINT "R":LOCATE 8,8:PRINT "1":LOCATE 22,7:PRINT
"R":LOCATE 23,8:PRINT "3":LOCATE 15,10:PRINT "R":LOCATE 16,11:PRI
NT "2"
2100 MOVE 340,210:DRAWR 10,10:DRAWR -3,3:DRAWR 20,20:DRAWR 7,-7:D
RAWR -20,-20:DRAWR -4,4
2110 MOVE 371,240:DRAWR 10,10:DRAWR 0,16:DRAWR -4,0:DRAWR 0,28:DR
AWR 9,0:DRAWR 0,-29:DRAWR -6,0:MOVE 381,294:DRAWR 0,16
2120 MOVE 381,250:DRAWR 10,-10:DRAWR -3,-3:DRAWR 20,-20:DRAWR 7,7
:DRAWR -20,20:DRAWR -4,-4:MOVE 412,220:DRAWR 10,-10
2130 LOCATE 45,7:PRINT "R":LOCATE 46,8:PRINT "C":LOCATE 42,10:PRI
NT "R":LOCATE 43,11:PRINT "A":LOCATE 52,10:PRINT "R":LOCATE 53,11
:PRINT "B"
2140 LOCATE 42,13:PRINT "A":LOCATE 54,13:PRINT "B":LOCATE 48,5:PR
INT "C"
2150 MOVE 250,260:DRAWR 30,0:MOVE 250,276:DRAWR 30,0:MOVE 292,268
:DRAWR -15,15:MOVE 292,268:DRAWR -15,-15
2160 LOCATE 2,15:PRINT " PODAJ REZYSTANCJE "
2170 LOCATE 24,15:PRINT " REZYSTANCJE UKLADU W GWIAZDE :"
2180 LOCATE 28,17:PRINT "R  =":LOCATE 29,18:PRINT "A"
2190 LOCATE 28,19:PRINT "R  =":LOCATE 29,20:PRINT "B"
2200 LOCATE 28,21:PRINT "R  =":LOCATE 29,22:PRINT "C"
2210 LOCATE 3,17:PRINT "R  =":LOCATE 4,18:PRINT "1"
2220 LOCATE 8,17:INPUT r1
2230 IF r1<=0 OR r1>10000 THEN GOTO 2350
2240 LOCATE 3,19:PRINT "R  =":LOCATE 4,20:PRINT "2"
2250 LOCATE 8,19:INPUT r2
2260 IF r2<=0 OR r2>10000 THEN GOTO 2370
2270 LOCATE 3,21:PRINT "R  =":LOCATE 4,22:PRINT "3"
2280 LOCATE 8,21:INPUT r3
2290 IF r3<=0 OR r3>10000 THEN GOTO 2390
2300 ra=r1*r2/(r1+r2+r3)
2310 rb=r2*r3/(r1+r2+r3)
2320 rc=r1*r3/(r1+r2+r3)
2330 LOCATE 33,17:PRINT ra:LOCATE 33,19:PRINT rb:LOCATE 33,21:PRI
NT rc
2340 GOTO 570
2350 GOSUB 1690
2360 LOCATE 6,17:PRINT SPC(12):GOTO 2220
2370 GOSUB 1690
2380 LOCATE 6,19:PRINT SPC(12):GOTO 2250
2390 GOSUB 1690
2400 LOCATE 6,21:PRINT SPC(12):GOTO 2280
```

„Polin" znany jest na terenie całego województwa szczecińskiego. Tylko w 1986 roku zorganizował 37 różnorodnych pokazów i akcji. Z jego inicjatywy powstały kluby mikrokomputerowe w Stargardzie Szczecińskim, Golczewie, Gryficach, a nawet w Politechnice Szczecińskiej. Klub uczestniczył również w Targach Młodych Mistrzów Jutra w Rostocku (NRD), prezentując różnorodne wykorzystanie sprzętu komputerowego. W codziennej działalności dominuje praca szkoleniowa. Systematycznie organizowane są kursy programowania oraz pokazy i wykłady na temat zastosowania mikrokomputerów.

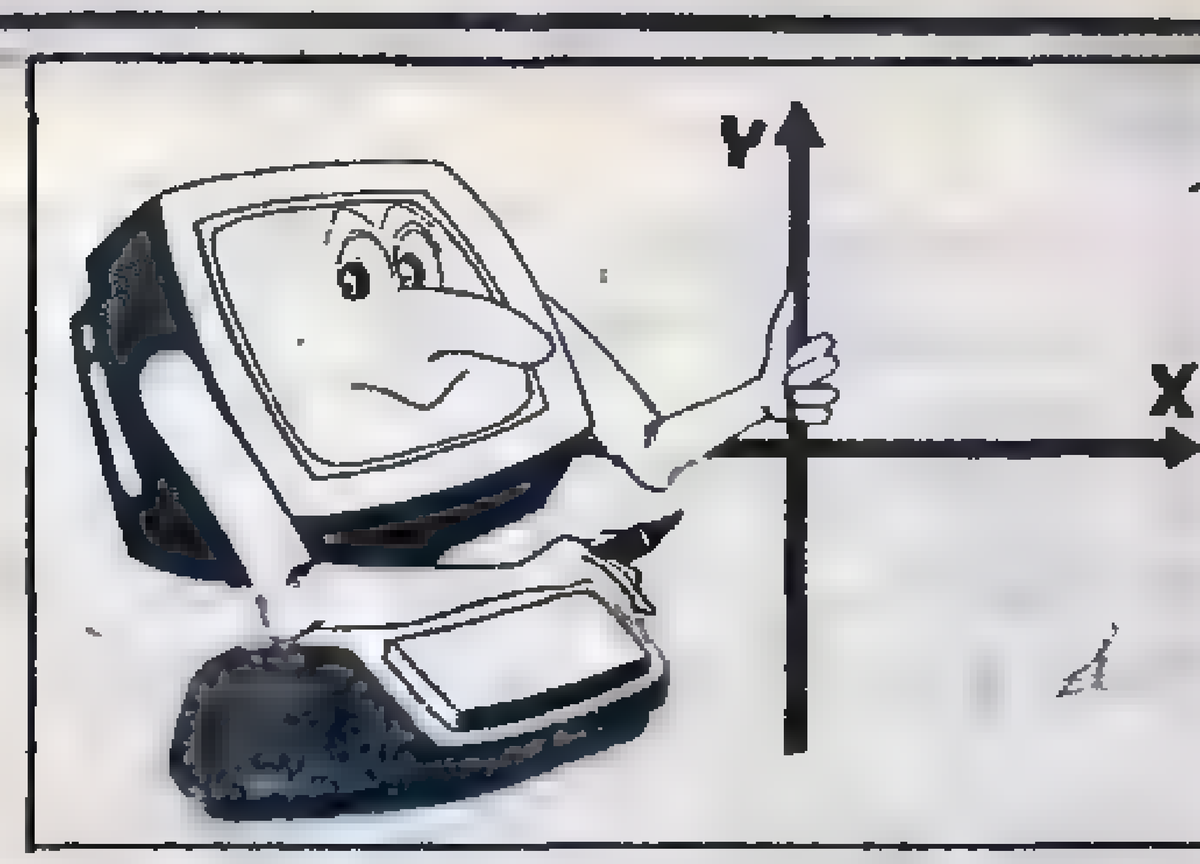
32



W roli nauczyciela uczeń szkoły podstawowej — słuchacze to: Bożena Janowicz (z zawodu informatyk) oraz jej klubowi koledzy: Andrzej Kaczmarek i Zbigniew Ignaszak.

AMSTRAD

PROGRAM 13—14



HIPERBOLE RÓWNOOSIOWE

Powyzsze programy przedstawiają jedną z krzywych stożkowych — hiperbolę w postaci hiperboli sprzężonych oraz hiperboli równoosiowych. Programy te stanowią kontynuację prezentowanych w pierwszym Dodatku Specjalnym „IKS-a" programów na krzywe stożkowe.

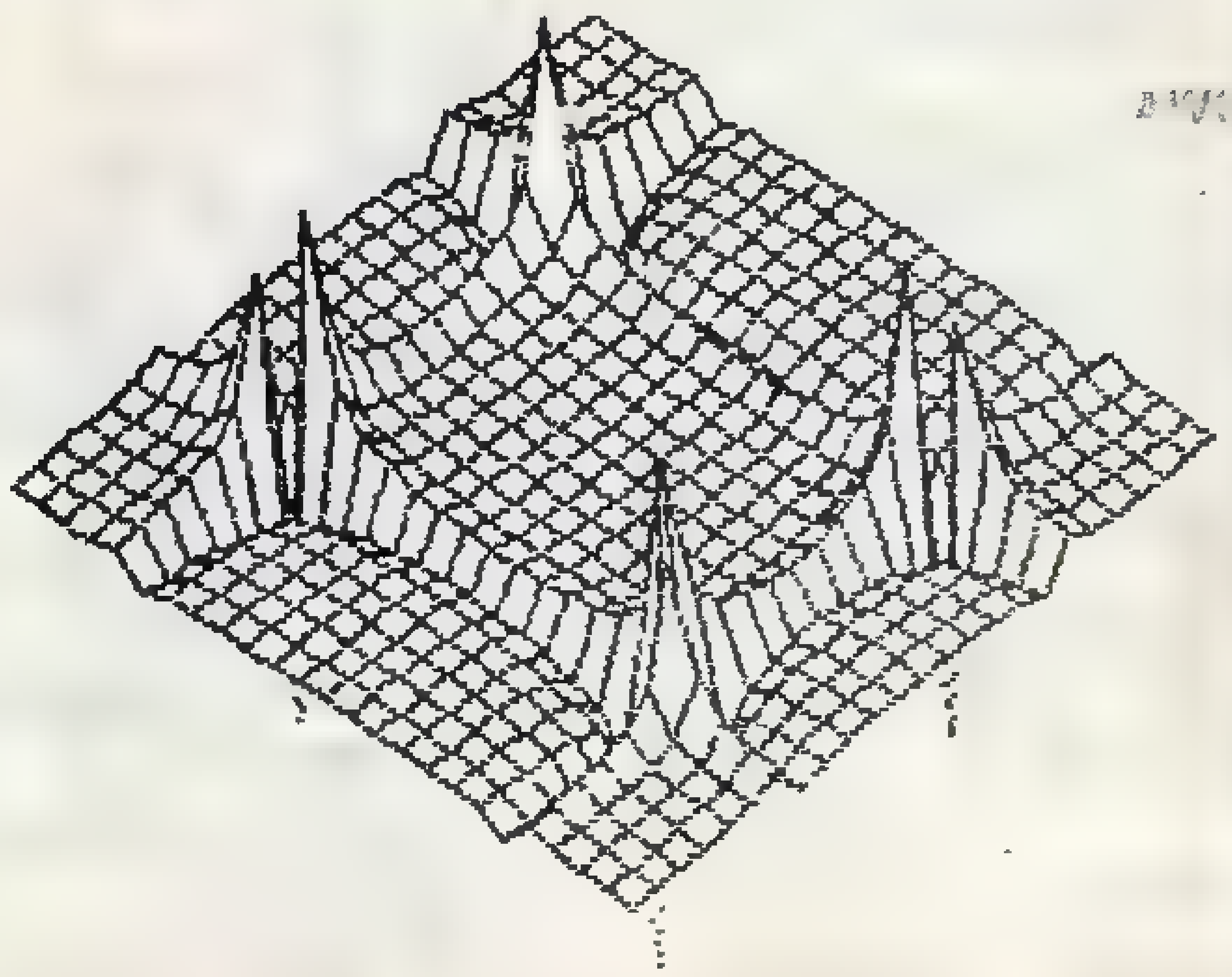
```
10 REM H I P E R B O L A
20 CLS
30 GOSUB 640
40 CLS:BORDER 3
50 WINDOW #2,9,33,23,25
60 PAPER #2,13:PEN #2,0:CLS #2
70 WINDOW #3,35,40,1,1
80 PAPER #3,13:PEN #3,0:CLS #3
90 PRINT #3," 0"
100 WINDOW #1,1,9,1,4
110 BORDER 9
120 PAPER #1,13:PEN #1,0:CLS #1
130 GOSUB 510
140 PLOT 100,220
150 FOR s=100 TO 600
160 PLOT s,220:NEXT
170 PLOT 598,222:PLOT 598,219
180 PLOT 596,224:PLOT 596,217
190 PLOT 340,400
200 FOR s=50 TO 390
210 PLOT 340,s:NEXT
220 PLOT 338,388:PLOT 336,386
230 PLOT 342,388:PLOT 344,386
240 LOCATE 38,13:PRINT "x"
250 LOCATE 20,1:PRINT "y"
260 l=0
270 INPUT #1,"a=";a
280 IF a=0 THEN 820
290 IF a<-9 OR a>9 THEN 860
300 IF a=-1 OR a=1 THEN 860
310 a=a*10:CLS #2
320 PRINT #2,"        1  2"
    "          1  2"
330 PRINT #2," xy = -";a/10;
    "  xy = - -";a/10;
340 PRINT #2,"          2";
    "           2  "
350 n=220
360 FOR x=-n TO 0 STEP 1
370 y=(a*a)/(2*x)
380 IF y<-170 THEN 410
390 PLOT 340+x,220+y
400 NEXT
410 FOR x=n TO 0 STEP -1
420 y=(a*a)/(2*x)
430 IF y>170 THEN 460
440 PLOT 340+x,220+y
450 NEXT
460 CLS #1:CLS #3:l=l+1
470 PRINT #3," ";l
480 IF l=2 THEN 550
490 IF l>=3 THEN 580
500 GOTO 270
510 PRINT #2,"        1  2";
    "          1  2 "
520 PRINT #2," xy = - a";
    "    xy = - - a  "
530 PRINT #2,"          2";
    "           2   "
540 RETURN
550 ENV 3,=9,9000
560 SOUND 1,300,0,13,3
570 GOTO 270
580 LOCATE 9,22
590 PRINT "koniec -";
    " [DOWOLNY KLAWISZ]"
600 ENV 3,=9,9000
610 SOUND 1,100,0,13,3
620 IF INKEY$="" THEN 620
630 ORIGIN 0,0:GOTO 40
640 t1$="H I P E R B O L A"
650 T2$=" "+CHR$(164)+
    "  "+" CJS "
660 MODE 0
670 ENV 1,=9,2000
680 ENT -1,6,3,1
690 ENV 2,127,0,0,127,
    0,0,127,0,0,127,0,0
700 ENV 3,=9,9000
710 FOR c=1 TO LEN(t1$)
720 LOCATE 2+c,10
730 FOR n2=LEN(t1$)
    TO c STEP -1
740 PRINT MID$(t1$,n2,1)
750 LOCATE 2+c,10
760 SOUND 135,20*n2,5,12,2,1
770 NEXT n2,c
780 SOUND 135,100,0,13,3,1,20
790 LOCATE 5,15:PRINT t2$
800 FOR c=1 TO 2000:NEXT
810 CLS:MODE 1:RETURN
820 LOCATE 9,23
830 PRINT "  WPISZ a";
    " ROZNE OD ZERA    "
840 FOR w=1 TO 2000:NEXT
850 GOSUB 510:GOTO 270
860 LOCATE 9,23
870 PRINT "PODAJ a";
    CHR$(180);"<-9,-2> ";
    CHR$(118);" a";
    CHR$(180);"<2,9>"
880 FOR w=1 TO 4000:NEXT
890 GOSUB 510:GOTO 270
```

HIPERBOLE SPRZĘŻONE

```
10 CLS
20 GOSUB 860
30 CLS:BORDER 3
40 WINDOW #2,6,37,21,25
50 PAPER #2,13:PEN #2,0:CLS #2
60 WINDOW #3,35,40,1,1
70 PAPER #3,13:PEN #3,0:CLS #3
80 PRINT #3," 0"
90 WINDOW #1,1,9,1,4
100 BORDER 9
110 PAPER #1,13:PEN #1,0:CLS #1
120 GOSUB 720
130 PLOT 100,240
140 FOR s=100 TO 600
150 PLOT s,240:NEXT:PLOT 598,242:PLOT 598,239:PL
OT 596,244:PLOT 596,237
160 PLOT 350,400
170 FOR s=80 TO 390
180 PLOT 350,s:NEXT:PLOT 348,388:PLOT 346,386:PL
OT 352,388:PLOT 354,386
190 LOCATE 38,12:PRINT "x"
200 LOCATE 21,1:PRINT "y"
210 l=0
220 INPUT #1,"a=";a
230 IF a=0 THEN 1020
240 IF a<-9 OR a>9 THEN 1080
250 a=a*10
260 INPUT #1,"b=";b
```

```
270 IF b=0 THEN 1050
280 IF b<-9 OR b>9 THEN 1090
290 b=b*10
300 CLS #2
310 PRINT #2, "   2    2       2    2    "
320 PRINT #2,"   x      y          y      x"

330 PRINT #2," --- - --- = 1  --- - --- = 1"
340 PRINT #2,"    2    2         2     2"
350 PRINT #2," ";a/10;"  ";b/10;"      ";b/10;
"  ";a/10;"
360 n=200
370 FOR x=0 TO -n STEP -1
380 k=(x-a)*(x+a):IF k<0 THEN 430
390 y=(b/a)*SQR(k):IF y>158 THEN 410
400 PLOT 350+x,240+y
410 y1=-(b/a)*SQR(k):IF y1<-158 THEN 440
420 PLOT 350+x,240+y1
430 NEXT
440 FOR z=0 TO n
450 k1=(z-a)*(z+a):IF k1<0 THEN 500
460 y2=(b/a)*SQR(k1):IF y2>158 THEN 480
470 PLOT 350+z,240+y2
480 y3=-(b/a)*SQR(k1):IF y3<-158 THEN 510
490 PLOT 350+z,240+y3
500 NEXT
510 FOR x=0 TO -n STEP -1
520 k=(x^2)+(a^2)
530 y=(b/a)*SQR(k):IF y>158 THEN 550
540 PLOT 350+x,240+y
550 y1=-(b/a)*SQR(k):IF y1<-158 THEN 580
560 PLOT 350+x,240+y1
570 NEXT
580 FOR z=0 TO n
590 k1=(z^2)+(a^2)
600 y2=(b/a)*SQR(k1):IF y2>158 THEN 620
610 PLOT 350+z,240+y2
620 y3=-(b/a)*SQR(k1):IF y3<-158 THEN 650
630 PLOT 350+z,240+y3
640 NEXT
650 CLS #1
660 CLS #3
670 l=l+1
680 PRINT #3," ";l
690 IF l=1 THEN 780
700 IF l>=2 THEN 800
710 GOTO 220
720 PRINT #2,"    2     2         2     2"
730 PRINT #2,"   x      y         y     x"
740 PRINT #2," --- - --- = 1  --- - --- = 1"
750 PRINT #2,"    2     2         2     2"
760 PRINT #2,"   a      b         b     a"
770 RETURN
780 ENV 3,=9,9000
790 SOUND 1,300,0,13,3:GOTO 220
800 LOCATE 9,21
810 PRINT " koniec - [DOWOLNY KLAWISZ]"
820 ENV 3,=9,9000
830 SOUND 1,100,0,13,3
840 IF INKEY$="" THEN 840
850 ORIGIN 0,0:GOTO 30
860 t1$="H I P E R B O L E"
870 T2$=CHR$(32)+CHR$(164)+CHR$(32)+CHR$(32)+" C
JS "
880 MODE 0
890 ENV 1,=9,2000:ENT -1,6,3,1
900 ENV 2,127,0,0,127,0,0,127,0,0,127,0,0:ENV 3,
=9,9000
910 FOR c=1 TO LEN(t1$)
920 LOCATE 2+c,10
930 FOR n2=LEN(t1$) TO c STEP -1
940 PRINT MID$(t1$,n2,1)
950 LOCATE 2+c,10
960 SOUND 135,20*n2,5,12,2,1
970 NEXT n2,c
980 SOUND 135,100,0,13,3,1,20
990 LOCATE 5,15:PRINT t2$
1000 FOR c=1 TO 2000:NEXT
1010 CLS:MODE 1:RETURN
1020 LOCATE 11,21:PRINT "WPISZ a ROZNE OD ZERA"
1030 FOR w=1 TO 2000:NEXT
1040 GOSUB 720:GOTO 220
1050 LOCATE 11,21:PRINT "WPISZ b ROZNE OD ZERA"
1060 FOR w=1 TO 2000:NEXT
1070 GOSUB 720:GOTO 260
1080 LOCATE 11,21:PRINT "   PODAJ a";CHR$(180);"
<-9,9>      ":FOR w=1 TO 3000:NEXT:GOSUB 720:GOTO
 220
1090 LOCATE 11,21:PRINT "   PODAJ b";CHR$(180);"
<-9,9>      ":FOR w=1 TO 3000:NEXT:GOSUB 720:GOTO
 260
```



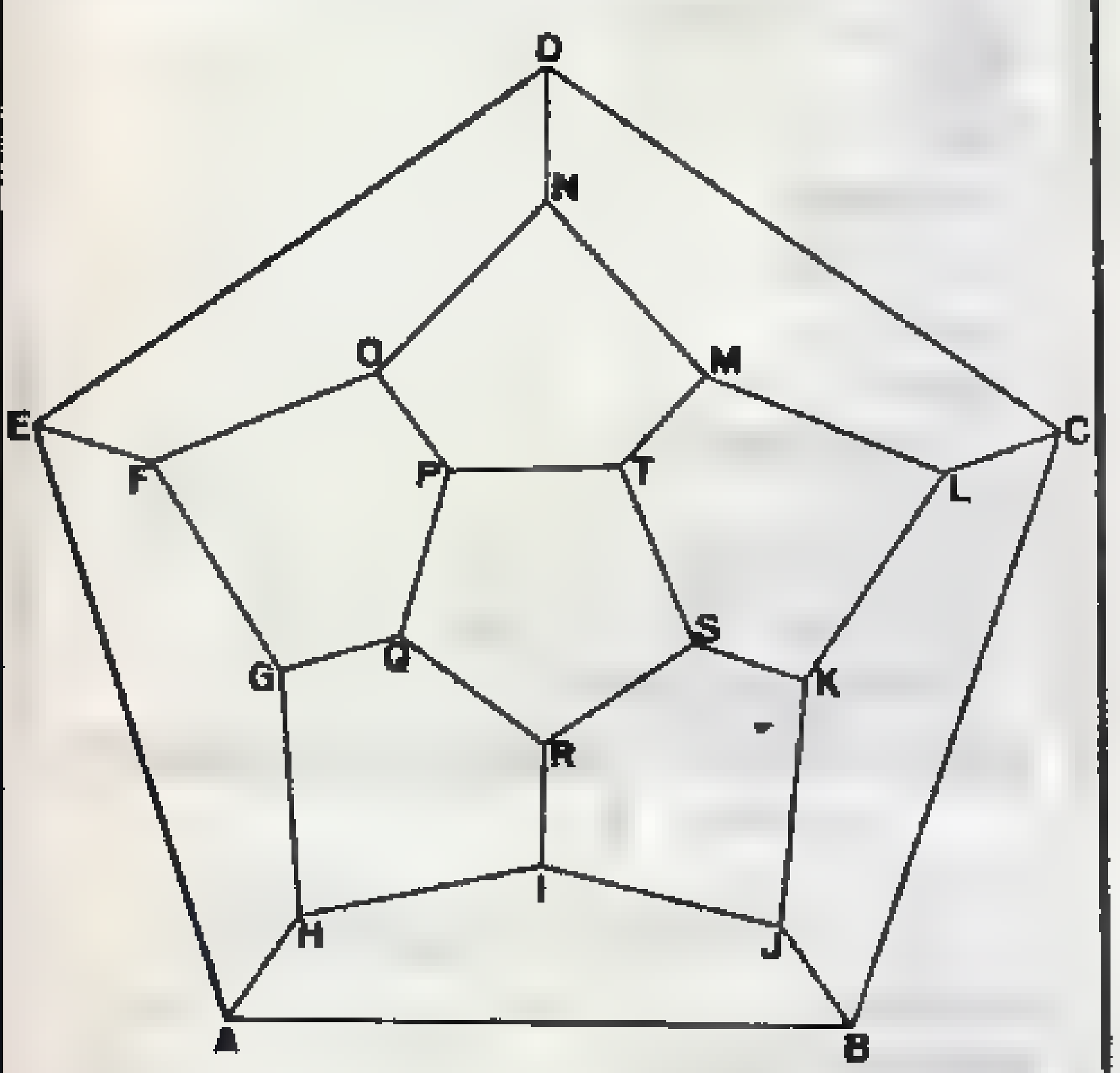
AMSTRAD

PROGRAM 15

Gra Hamiltona

William Hamilton, matematyk i astronom angielski z pierwszej połowy XIX wieku, obmyślił ciekawą rozrywkę zwaną „podróżą po dodekaedrze" (dwunastościanie). W „podróży" tej chodzi o jednorazowe odwiedziny wszystkich naroży dwunastościanu z powróceniem do naroża, z którego się wyszło.

Program umożliwia „podróż" lub demonstruje jej przebieg, przy czym dwunastościan został zastąpiony następującą figurą płaską:



Klawisze funkcyjne:
- 0 —wyjście z programu,
- 1 — powrót do „menu",
- 2 — demonstracja gry,
- 3 — gra,
- A ÷ T — wybór odpowiedniego punktu figury,
- TAB — cofnięcie ruchu.

UWAGA: program kończy się, gdy następuje powrót do punktu początkowego; wygrywa ten, kto przejdzie przez wszystkie punkty zgodnie z podanymi zasadami.

```
PROGRAM grahamiltona;
 CONST
  tagflag:boolean = false;
  tagaddr:integer = 0;
 TYPE
  nazwypunktow=(a,b,c,d,e,f,g,h,i,j,k,
               l,m,n,o,p,q,r,s,t);
  opispunktu=RECORD
             x,y:integer;
             wskp:boolean;
             sasiedzi:ARRAY[1..3] OF
               RECORD
                poprz,
                lewy,
                prawy:nazwypunktow
               END
            END;
  dodekaedr=ARRAY[nazwypunktow] OF
                          opispunktu;
  lzn=string[40];
 VAR
  fig:dodekaedr;
  opocz:nazwypunktow;
  ch:char;
  pozlit:ARRAY['A'..'T'] OF
          RECORD
           x,
           y:integer
          END;
  wykgr:boolean;


{*******************************************}
{*   prcedury pomocnicze do manipulacji   *}
{*   zobrazowaniem graficznym             *}
{*                                         *}

 PROCEDURE InitGrafik;
  BEGIN
   WRITE(#$1b,'0',#$1b,'y');
  END;
 PROCEDURE leaveGrafik;
  BEGIN
   WRITE(#$1b,'1',#$1b,'x');
  END;
 PROCEDURE draw(x,y:integer);
  BEGIN
   INLINE($2A/x/$EB/$2A/y/$cd/$5a/
          $fc/$F6/$BB);
  END;
 PROCEDURE plot(x,y:integer);
  BEGIN
   INLINE($2A/x/$EB/$2A/y/$cd/
          $5a/$fc/$EA/$BB);
  END;
 PROCEDURE grafpen(fs:byte);
  BEGIN
   INLINE($3A/fs/$cd/$5a/$fc/$DE/$BB);
  END;
 PROCEDURE grafpaper(fs:byte);
  BEGIN
   INLINE($3A/fs/$cd/$5a/$fc/$E4/$BB);
  END;
 PROCEDURE clg(fs:byte);
  BEGIN
   INLINE($cd/$5a/$fc/$DB/$BB);
   grafpaper(fs);
  END;
 PROCEDURE ink(fs,f1,f2:byte);
  BEGIN
   INLINE($3a/f1/$47/$3a/f2/$4f/
   $3a/fs/$cd/$5a/$fc/$32/$bc);
  END;
 PROCEDURE paper(fs:byte);
  BEGIN
   INLINE($3A/fs/$cd/$5a/$fc/$96/$BB);
  END;
 PROCEDURE mode(md:byte);
  BEGIN
   INLINE($3a/md/$cd/$5a/$fc/$0e/$bc);
  END;
 PROCEDURE border(v1,v2:byte);
  BEGIN
   INLINE($3a/v1/$47/$3a/v2/$4f/
   $cd/$5a/$fc/$38/$bc);
  END;
 PROCEDURE grafout(ch:char);
  BEGIN


   INLINE($3a/ch/$cd/$5a/$fc/$FC/$BB);
  END;
 PROCEDURE tag;
  BEGIN
   if not tagflag then
    BEGIN
     tagaddr := conoutptr;
     tagflag := true;
     conoutptr := addr(grafout);
    END;
  END;
 PROCEDURE tagoff;
  BEGIN
   if tagflag then
    BEGIN
     conoutptr := tagaddr;
     tagflag := false;
    END;
  END;
 PROCEDURE punkt(nazp:nazwypunktow);
  BEGIN
   plot(fig[nazp].x,fig[nazp].y)
  END;
 PROCEDURE linia(nazp:nazwypunktow);
  BEGIN
   draw(fig[nazp].x,fig[nazp].y)
  END;
 PROCEDURE litera(lit:char);
  BEGIN
   tag;
   plot(pozlit[lit].x,pozlit[lit].y);
   grafout(lit);
   tagoff
  END;
 PROCEDURE napis(l:lzn;nrlin,
                 nrkol:integer);
  VAR
   i,j,k:integer;
  BEGIN
   i:=nrkol*16-16;
```



```
       j:=400-nrlin*16;
       tag;
       FOR k:=1 TO length(l) DO
        BEGIN
         plot(i,j);
         grafout(l[k]);
         i:=i+16
        END;
       tagoff
      END;
{*****************************************}


{*****************************************}
(*       ustawienie wspolrzednych        *)
(*                                       *)
 PROCEDURE ustawwsp;
  PROCEDURE wspp(nazp:nazwypunktow;
                 xp,yp:integer);
   BEGIN
    WITH fig[nazp] DO
     BEGIN x:=xp; y:=yp; wskp:=false END
   END;
  BEGIN
   wspp(a,200,50);  wspp(b,400,50);
   wspp(c,464,240); wspp(d,300,356);
   wspp(e,136,240); wspp(f,176,226);
   wspp(g,216,162); wspp(h,222,84);
   wspp(i,300,100); wspp(j,378,84);
   wspp(k,384,162); wspp(l,424,226);
   wspp(m,354,258); wspp(n,300,314);
   wspp(o,246,258); wspp(p,270,228);
   wspp(q,254,174); wspp(r,300,140);
   wspp(s,346,174); wspp(t,330,228)
  END;
{*****************************************}


{*****************************************}
(*     opisy wierzcholkow sasiednich     *)
(*                                       *)
PROCEDURE ustawsas;
 PROCEDURE sd(nazp,p1,p2,
              p3:nazwypunktow);
  BEGIN
   WITH fig[nazp] DO
    BEGIN
     WITH sasiedzi[1] DO
      BEGIN
       poprz:=p1; lewy:=p2; prawy:=p3
      END;
     WITH sasiedzi[2] DO
      BEGIN
       poprz:=p2; lewy:=p3; prawy:=p1
      END;
     WITH sasiedzi[3] DO
      BEGIN
       poprz:=p3; lewy:=p1; prawy:=p2
      END
    END
  END;
 BEGIN
  sd(a,e,h,b); sd(b,a,j,c); sd(c,b,l,d);
  sd(d,c,n,e); sd(e,d,f,a); sd(f,e,o,g);
  sd(g,f,q,h); sd(h,g,i,a); sd(i,h,r,j);
  sd(j,i,k,b); sd(k,j,s,l); sd(l,k,m,c);
  sd(m,l,t,n); sd(n,m,o,d); sd(o,n,p,f);
  sd(p,o,t,q); sd(q,p,r,g); sd(r,q,s,i);
  sd(s,r,t,k); sd(t,s,p,m)
 END;
{*****************************************}


{*****************************************}
(*     wypelnienie tablicy opisujacej    *)
(*     polozenie nazw wierzcholkow       *)
(*                                       *)
```

```
PROCEDURE nazwyw;
 PROCEDURE wp(l:char;x,y:integer);
  BEGIN
   pozlit[l].x:=x;
   pozlit[l].y:=y
  END;
 BEGIN
  wp('A',195,40);  wp('B',395,40);
  wp('C',475,245); wp('D',295,376);
  wp('E',115,245); wp('F',174,246);
  wp('G',195,160); wp('H',225,78);
  wp('I',295,90);  wp('J',363,78);
  wp('K',392,160); wp('L',420,246);
  wp('M',361,276); wp('N',278,329);
  wp('O',231,276); wp('P',250,230);
  wp('T',317,247); wp('Q',244,164);
  wp('S',350,188); wp('R',307,139)
 END;
{*****************************************}


{*****************************************}
(*        wyswietlenie figury            *)
(*                                       *)

 PROCEDURE wyswfig;
  VAR
   i,j:nazwypunktow;
   l:char;
  BEGIN
   border(0,0); grafpaper(3);
   clg(3); grafpen(0);
   punkt(a);
   FOR i:=a TO d DO
    linia(succ(i));
   linia(a);
   punkt(f);
   FOR i:=g TO o DO
    linia(i);
   linia(f);
   linia(e);
   j:=h;
   FOR i:=a TO d DO
    BEGIN
     punkt(i);
     linia(j);
     j:=succ(succ(j))
    END;
   punkt(p);
   FOR i:=q TO t DO
    linia(i);
   linia(p);
   linia(o);
   j:=g;
   FOR i:=q TO t DO
    BEGIN
     punkt(i);
     linia(j);
     j:=succ(succ(j))
    END;
   nazwyw;
   grafpen(2);
   FOR l:='A' TO 'T' DO
    litera(l)
  END;
{*****************************************}


{*****************************************}
(*  pokaz wyznaczania drogi hamiltona    *)
(*                                       *)
PROCEDURE pokaz;
 VAR
  akt,pop,pom:nazwypunktow;
  il,i,l1,l2:integer;
 FUNCTION nre:integer;

  VAR zn:boolean;
      i:integer;
  BEGIN
   zn:=false;
```

```
   i:=1;
   REPEAT
    WITH fig[akt] DO
     IF sasiedzi[i].poprz=pop THEN
      BEGIN nre:=i; zn:=true END;
    i:=i+1
   UNTIL zn
  END;
PROCEDURE wlewo;
 BEGIN
  pom:=akt;
  akt:=fig[akt].sasiedzi[nre].lewy;
  pop:=pom;
  linia(akt)
 END;
PROCEDURE wprawo;
 BEGIN
  pom:=akt;
  akt:=fig[akt].sasiedzi[nre].prawy;
  pop:=pom;
  linia(akt)
 END;
PROCEDURE opoznij;
 VAR i:integer;
 BEGIN
  i:=1;
  REPEAT
   i:=i+1
  UNTIL i>20000
 END;
BEGIN
 ustawwsp;
 ustawsas;
 wyswfig;
 opoznij;
 il:=random(19);
 akt:=a;
 IF il>0 THEN
  FOR i:=0 TO il-1 DO
   akt:=succ(akt);
 pop:=fig[akt].sasiedzi[random(2)+1]
      .poprz;
 grafpen(1);
 litera(chr(ord('A')+ord(akt)));
 opoznij;
 punkt(akt);
 FOR l1:=1 TO 2 DO
  BEGIN
   FOR l2:=1 TO 3 DO
    BEGIN wlewo; opoznij END;
   FOR l2:=1 TO 3 DO
    BEGIN wprawo; opoznij END;
   wlewo; opoznij;
   wprawo; opoznij;
   wlewo; opoznij;
   wprawo; opoznij;
  END;
 napis('KONIEC POKAZU',24,1);
 wykgr:=false
END;

{*****************************************}


{*****************************************}
(*                 menu                  *)
(*                                       *)
 PROCEDURE menu;
  BEGIN
   border(0,0); grafpaper(3);
   clg(3); grafpen(2);
   napis('GRA HAMILTONA',2,5);
   napis('MENU:',4,5);
   napis('0 - WYJSCIE Z PROGRAMU',6,10);
   napis('1 - POWROT DO MENU',8,10);
   napis('2 - POKAZ',10,10);
   napis('3 - GRA',12,10);
   wykgr:=false
  END;
```

```
{*****************************************}

{*****************************************}
{*                  gra                  *}
{*                                       *}
 PROCEDURE gra;
  VAR
   koniec:boolean;
   ruchy:ARRAY[1..20] OF nazwypunktow;
   ppocz,pnast:nazwypunktow;
   nrakt:integer;
  FUNCTION nazp(ch:char):nazwypunktow;
   VAR
    i,j:integer;
    p:nazwypunktow;
  BEGIN
   IF (ch>='A') AND (ch<='T') THEN
     i:=ord(ch)-ord('A')
   ELSE
     i:=ord(ch)-ord('a');
   p:=a;
   FOR j:=1 TO i DO
    p:=succ(p);
   nazp:=p
  END;
 PROCEDURE wygaskom;
  VAR
   lpom:lzn;
   i:integer;
  BEGIN
   grafpen(3);
   FOR i:=1 TO 40 DO
    lpom[i]:=' ';
   napis(lpom,24,1);
   grafpen(1)
  END;
 FUNCTION rpopr(pnast:nazwypunktow):
            boolean;
  BEGIN
   WITH fig[pnast] DO
    IF (NOT wskp) AND
     ((sasiedzi[1].poprz=ruchy[nrakt]) OR
      (sasiedzi[2].poprz=ruchy[nrakt]) OR
      (sasiedzi[3].poprz=ruchy[nrakt]))
     THEN rpopr:=true
     ELSE rpopr:=false
  END;
 BEGIN
  ustawwsp;
  ustawsas;
  wyswfig;
  wykgr:=true;
  koniec:=false;
  grafpen(1);
  napis('PODAJ PUNKT POCZATKOWY',24,1);
  REPEAT
   REPEAT UNTIL keypressed;
   READ(kbd,ch);
  UNTIL ((ch>='A') AND (ch<='T')) OR
        ((ch>='a') AND (ch<='t'));
  ppocz:=nazp(ch);
  IF (ch>='a') AND (ch<='t') THEN
                ch:=chr(ord(ch)-32);
  litera(ch);
  nrakt:=1;
  ruchy[1]:=ppocz;
  wygaskom;
  napis('PODAJ NASTEPNY PUNKT',24,1);
  punkt(ppocz);
  WHILE NOT ((ch=#48) OR (ch=#49) OR
    (ch=#50) OR (ch=#51) OR koniec) DO
   BEGIN
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
    IF ch=#9 THEN
     BEGIN
  IF nrakt>1 THEN
  BEGIN
   grafpen(0);
   linia(ruchy[nrakt-1]);
   fig[ruchy[nrakt]].wskp:=false;
   nrakt:=nrakt-1;
   wygaskom;
   napis('PODAJ NASTEPNY PUNKT',24,1);
   punkt(ruchy[nrakt]);
  END
 END
ELSE
 IF ((ch>='A') AND (ch<='T')) OR
    ((ch>='a') AND (ch<='t')) THEN
  BEGIN
   pnast:=nazp(ch);
   IF rpopr(pnast) THEN
    BEGIN
     linia(pnast);
     IF pnast=ppocz THEN koniec:=true
     ELSE
      BEGIN
       nrakt:=nrakt+1;
       ruchy[nrakt]:=pnast;
       fig[pnast].wskp:=true;
       wygaskom;
       napis('PODAJ NASTEPNY PUNKT',
              24,1);
       punkt(ruchy[nrakt]);
      END
    END
   ELSE
    BEGIN
     wygaskom;
     napis(
      'ZLY RUCH-PODAJ NASTEPNY PUNKT'
      24,1);
     punkt(ruchy[nrakt])
    END
  END;
END;
   IF koniec THEN
    BEGIN
     wygaskom;
     IF nrakt=20 THEN
      napis(
       'KONIEC GRY. WYGRALES-GRATULACJE',
       24,1)
     ELSE
      napis(
       'KONIEC GRY. NIESTETY PRZEGRALES',
       24,1);
     wykgr:=false
    END
  END;
{*****************************************}
BEGIN
 crtinit;
 initgrafik;
 mode(1);
 ink(0,26,26);
 ink(1,15,15);
 ink(2,9,9);
 ink(3,0,0);
 menu;
 REPEAT
  IF NOT wykgr THEN
   BEGIN
    REPEAT UNTIL keypressed;
    READ(kbd,ch)
   END;
  CASE ch OF
   #49: menu;
   #50: pokaz;
   #51: gra
  END;
 UNTIL ch=#48;
 mode(2);
 leavegrafik;
END.
```

Stefan Rozmus



— Widzę, że nie lubi pani komputera.





KLUBY KOMPUTEROWE
MŁODYCH MISTRZOW TECHNIKI
ZSMP

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu), Ryszard Rogoń. Stali współpracownicy: Włodzimierz Gogołek, Krzysztof Mamcarz, Ireneusz Miernik, Janusz Miller, Jacek Szanlawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 8684. Nr. ind. 361682. K-82

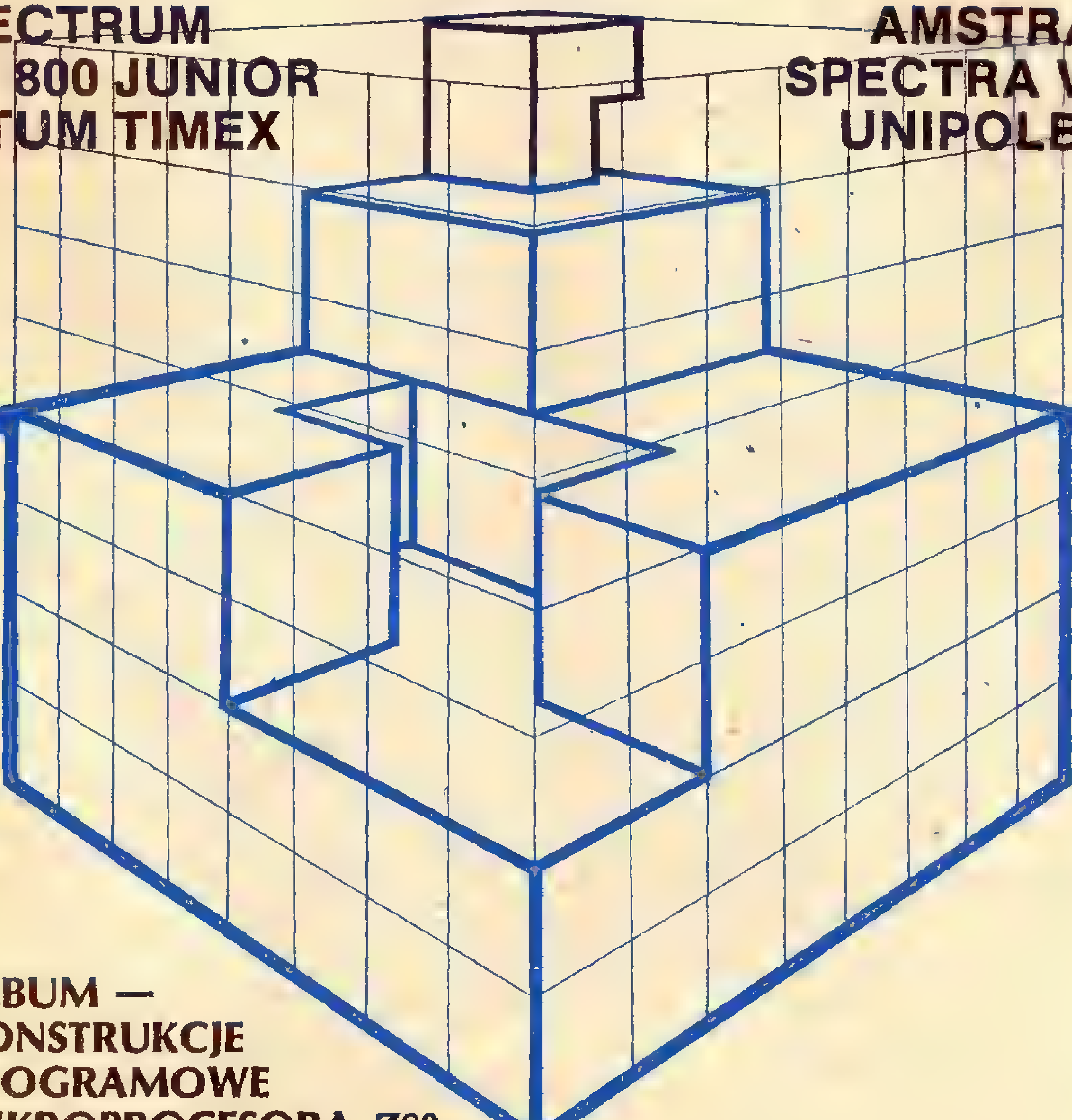
INFORMATYKA
KOMPUTERY
SYSTEMY

zeszyty programów komputerowych nr 3

Dodatek „Żołnierza Wolności" Cena 150 zł

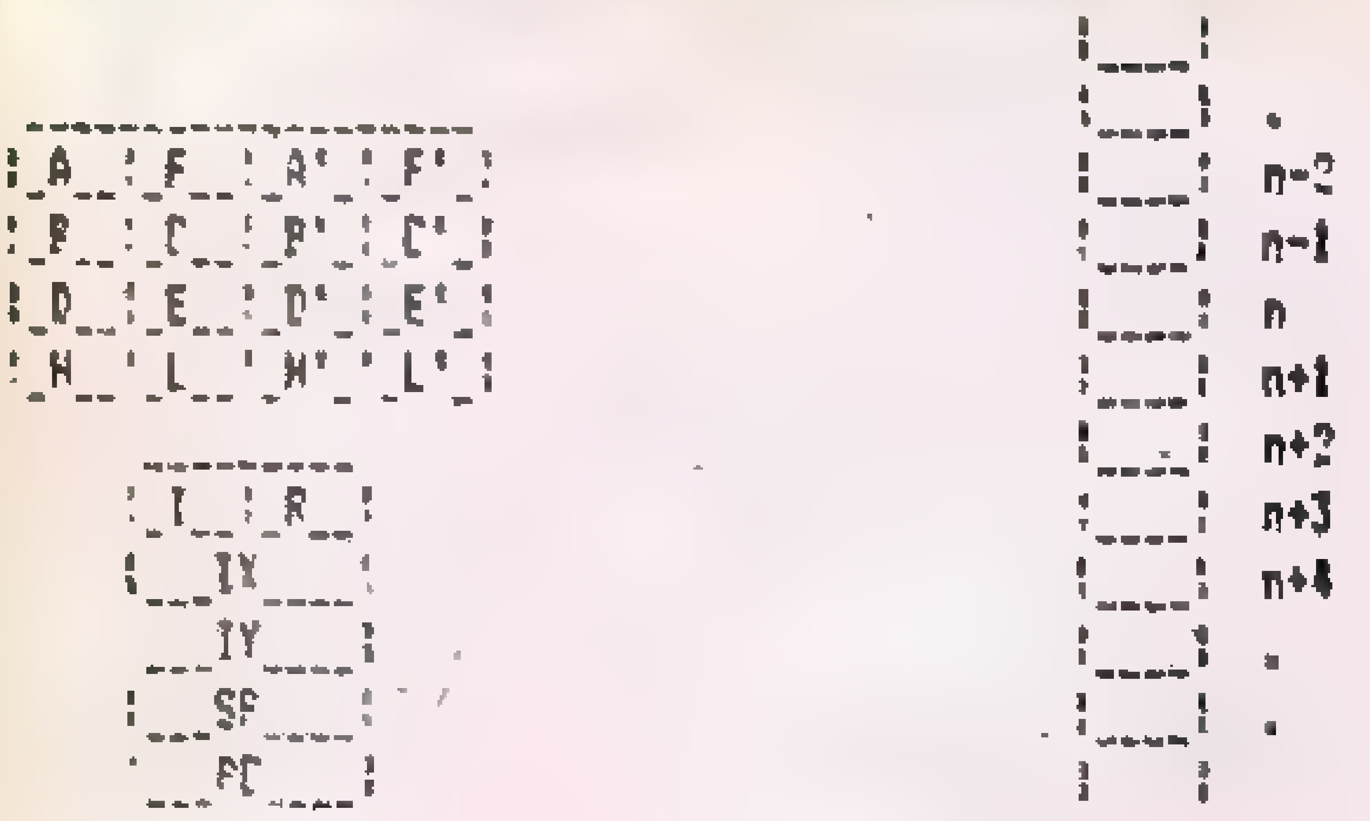
SPECTRUM
ELWRO 800 JUNIOR
MERITUM TIMEX

AMSTRAD
SPECTRA VIDEO
UNIPOLBRIT



ALBUM —
KONSTRUKCJE
PROGRAMOWE
MIKROPROCESORA Z80

PODSTAWOWE KONSTRUKCJE PROGRAMOWE MIKROPROCESORA Z80



SPIS TREŚCI

	strona
WPROWADZENIE	2
1. PRZESŁANIA POMIĘDZY REJESTRAMI I KOMÓRKAMI PAMIĘCI OPERACYJNEJ	4
1.1 Przesłania do rejestrów	4
1.1.1. Bezpośrednie przesłania zawartości komórek pamięci operacyjnej do rejestrów	4
1.1.2. Pośrednie przesłania zawartości komórek pamięci operacyjnej	4
1.1.3 Przesłania z wierzchołka stosu	5
1.1.4. Przesłania stałych	5
1.1.5. Przesłania pomiędzy rejestrami	6
1.2. Przesłania do komórek pamięci operacyjnej	7
1.2.1. Bezpośrednie przesłania zawartości rejestrów	7
1.2.2. Pośrednie przesłania zawartości rejestrów	8
1.2.3. Przesłania na wierzchołek stosu	8
1.2.4. Przesłania stałych	8
1.2.5. Przesłania pomiędzy komórkami pamięci operacyjnej	9
2. PĘTLE ITERACYJNE	10
2.1. Pętle typu "repeat"	10
2.1.1. Pętle wykonywane do 256 razy	10
2.1.2. Pętla wykonywana do 65536 razy	10
2.2. Pętla typu "while"	10
3. WARUNKI	11
3.1. Porównywanie 1-bajtowych tekstów znakowych lub liczb bez znaków	11
3.2. Porównywanie 1-bajtowych liczb całkowitych ze znakiem	12
3.3. Porównywanie 2-bajtowych liczb całkowitych bez znaku ze stałą nn	12
3.4. Porównywanie n-bajtowych tekstów znakowych	13
4. PODPROGRAMY, OPERACJE NA STOSIE	14
4.1. Wywoływanie podprogramów	14
4.2. Przekazywanie parametrów do podprogramów	15
5. KONWERSJE PÓL DANYCH	15
5.1. Liczby binarne i znakowe w kodzie BCD	15
5.2. Liczby binarne i znakowe w kodzie ASCII	16
6. OPERACJE NA POLACH BINARNYCH	16
6.1. Wprowadzanie określonych wartości na pozycje binarne komórki	16
6.2. Przesłania określonych pozycji binarnych	17
6.3. Badania wartości określonych pozycji binarnych	17
6.4. Przesuwanie pozycji binarnych	17
6.4.1. Przesuwanie cykliczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej	18
6.4.2. Przesuwanie cykliczne w lewo i w prawo z przeniesieniem (o jedną pozycję) wielkości 8-bitowej	18
6.4.3. Przesuwanie logiczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej	18
6.4.4. Przesuwanie arytmetyczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej	18
6.4.5. Przesuwanie logiczne w lewo i w prawo (o jedną pozycję) wielkości 16-bitowej	18
7. OPERACJE ARYTMETYCZNE	19
7.1. Dodawanie	19
7.1.1. Dodawanie liczb binarnych	19
7.1.2. Dodawanie liczb w kodzie BCD	19
7.2. Odejmowanie	20
7.2.1. Odejmowanie liczb binarnych	20
7.2.2. Odejmowanie liczb w kodzie BCD	20
7.3. Mnożenie liczb binarnych	20
7.4. Dzielenie liczb binarnych	20
8. OPERACJE NA TABLICACH	21
8.1. Sekwencyjne przeglądanie tablicy	21
8.1.1. Przeglądanie tablicy zbudowanej z elementów 1-bajtowych	21
8.1.2. Przeglądanie tablicy zbudowanej z elementów 2-bajtowych	21
8.2. Binarne (logarytmiczne) przeglądanie tablicy zbudowanej z elementów 1-bajtowych	22
8.3. Wyszukiwanie w tablicy elementu o wartości minimalnej i maksymalnej	22
8.4. Sortowanie elementów tablicy	23
DODATEK I. ZESTAWIENIE ROZKAZÓW MIKROPROCESORA Z80	24
DODATEK II. TABELE SZESNASTKOWYCH KODÓW ROZKAZÓW MIKROPROCESORA Z80	25
DODATEK III. TABELE KONWERSJI LICZB W RÓŻNYCH SYSTEMACH LICZENIA	29
LITERATURA	31

WPROWADZENIE

Przedstawiamy Czytelnikom album podstawowych konstrukcji programowych, występujących przy kodowaniu programów w języku asemblera dla mikroprocesora Z80. Konstrukcje te można znaleźć niemal w każdym programie, choć nie zawsze są w wyraźny sposób wyodrębnione. Wytrawni programiści z dużą praktyką zawodową znają je "na pamięć". Tajemnica ich dużej wydajności pracy (wyrażanej liczbą uruchomionych i przetestowanych instrukcji w jednostce czasu) wynika z operowania nie tylko pojedyńczymi instrukcjami, ale całymi ciągami instrukcji, realizującymi określone operacje - właśnie konstrukcjami programowymi. Niestety, okres dochodzenia do wprawy trwa stosunkowo długo. Praktycznie każdy programujący jest pozostawiony własnej inwencji. W dostępnej literaturze [2, 3, 4] można wyczytać jedynie opisy realizacji pojedyńczych rozkazów mikroprocesora Z80 i kilku najprostszych konstrukcji programowych. Bardzo trudno jest także "wyciągnąć" tajniki programowania od zawodowych programistów, dla których są to sprawy oczywiste, trywialne i nie warte wspomnienia.

Zadaniem tego materiału jest w jakimś stopniu pomóc początkującym programistom (zarówno zawodowym, jak i amatorom) przez dostarczenie wzorców gotowych konstrukcji programowych. Są one przedstawione w postaci szablonów, które należy przepisać do tekstu programu z ewentualnymi zmianami nazw obszarów danych i etykiet. Oczywiście, nie chodzi tu o zwykłe przepisywanie, konieczne jest zrozumienie zastosowanych rozwiązań. Szczególną uwagę należy zwrócić na warianty tych samych konstrukcji, różniące się głównie czasami realizacji, zajętością pamięci lub rodzajem wykorzystywanych rejestrów.

Przedstawione konstrukcje programowe zostały pogrupowane w następujących 8 rozdziałach:

1. Przesłania pomiędzy rejestrami i komórkami pamięci operacyjnej
2. Pętle iteracyjne
3. Warunki
4. Podprogramy, operacje na stosie
5. Konwersje pól danych
6. Operacje na polach binarnych
7. Operacje arytmetyczne
8. Operacje na tablicach

Dodatki na końcu tekstu zawierają informacje o rozkazach mikroprocesora Z80 i danych liczbowych w różnych systemach liczenia. Jest to materiał, do którego często będziemy zaglądać, szczególnie w początkowym okresie nauki programowania.

Zaproponowany układ treści ma ułatwić odszukanie wymaganych operacji. Przy czym od razu należy się zastrzec, że Czytelnik nie znajdzie wszystkich potrzebnych mu operacji. Liczba możliwych przypadków konstrukcji i ich wariantów jest zbyt duża, aby je zmieścić w ograniczonej objętości pisma.

Poszczególne konstrukcje programowe są scharakteryzowane funkcjami chronometrażowymi (t) i pamięciochłonności (p). Funkcja - t, określa czas przebiegu, wyrażony liczbą taktów zegara mikrokomputera, a funkcja p - zajętość pamięci, wyrażoną w bajtach. Wyliczanie czasu przebiegu w mikrosekundach (t') wymaga przekształcenia:

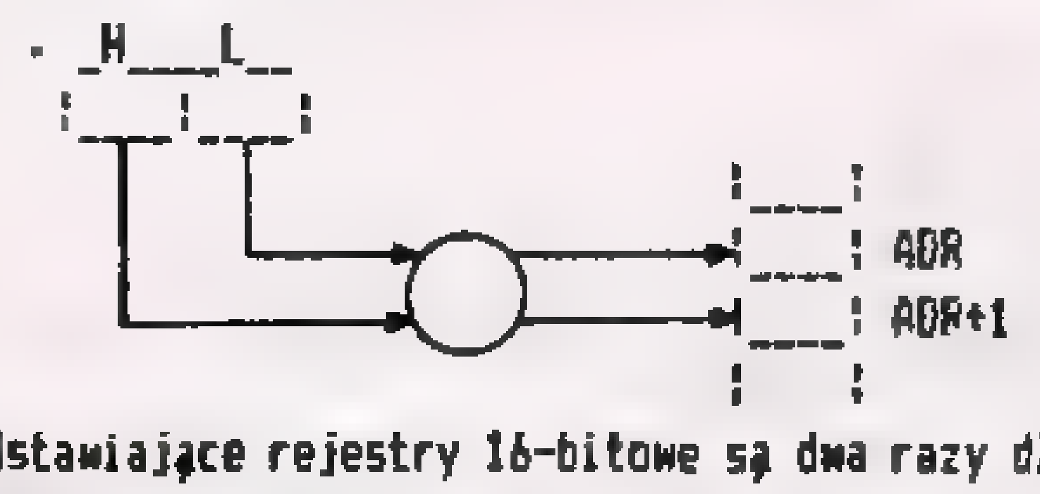
$$t' = t / f \qquad [\mu s]$$

gdzie: f - częstotliwość zegara [MHz]

Czasy przebiegu konstrukcji zostały wyznaczone według danych chronometrażowych pojedyńczych rozkazów zestawionych w dodatku I. (kolumna t). W dodatku tym zamieszczono także wyliczone czasy realizacji rozkazów w mikrosekundach dla trzech najczęściej spotykanych częstotliwości zegarów - odpowiednio: 2 MHz, 3.5 MHz i 4 MHz.

Dodatkowo dla celów ilustracyjnych poszczególne konstrukcje programowe zostały opatrzone tzw. "strukturami operacyjnymi", w których występują prostokąty i okręgi połączone liniami skierowanymi. Prostokąty symbolizują obszary danych w pamięci operacyjnej i rejestry, a okręgi - rozkazy. Przy obszarach danych i rejestrach występują ich nazwy. Strzałki przy liniach łączących prostokąty z okręgami wskazują na kierunek przesyłania danych. Odczyt danych jest oznaczony przez zwrot strzałki w kierunku do okręgu, zapis w kierunku do prostokąta symbolizującego obszar danych. Linie ze strzałkami skierowanymi w obydwie strony oznaczają odczyt i zapis.

Przykładowo poniższy rysunek ilustruje operację przesyłania zawartości rejestru L do komórki o symbolicznym adresie ADR i rejestru H do komórki o adresie ADR+1. Przyjęto konwencję rysowania komórek pamięci o wzrastających adresach kolejno jedna pod drugą.



Prostokąty przedstawiające rejestry 16-bitowe są dwa razy dłuższe od prostokątów przedstawiających rejestry 8-bitowe. W wypadku pary rejestrów (AF, BC, DE, HL), rejestr zawierający bardziej znaczące pozycje binarne jest narysowany na lewo od rejestru zawierającego mniej znaczące pozycje binarne.

Oprócz linii łączących okręgi z prostokątami (symbolizujące kierunek przepływu danych) na niektórych rysunkach występują także linie skierowane od rejestrów do komórek pamięci operacyjnej. Oznaczają one, że w rejestrach zapisane są adresy komórek (wskaźniki do komórek). Zmiany adresów powstałe w wyniku wykonywania operacji zaznaczone są linią przerywaną.

Wielobajtowe liczby binarne i znakowe w kodzie BCD zapisywane są w kolejnych komórkach pamięci operacyjnej, poczynając od najmniej znaczących pozycji. W ten sposób 2-bajtowa liczba całkowita ze znakiem jest zapamiętana jak niżej:



Wartość dziesiętna liczby zapisanej w komórkach o adresie n i n+1 wynosi 268 (256 + 8 + 4).

Wobec powszechnego stosowania wysokopoziomowych języków algebraicznych (takich jak BASIC, Pascal, C) asembler stosuje się w ściśle określonych przypadkach. Głównie wtedy, gdy czasy realizacji programów kodowanych w języku wysokopoziomowym są zbyt długie. Niewątpliwie najdłuższe czasy występują przy stosowaniu interpretera języka BASIC. W wielu zastosowaniach jest on nie do przyjęcia.

Jak duże różnice uzyskuje się w czasach przebiegu, stosując asembler zamiast BASIC-a, uzmysławią nam dane chronometrażowe 3 instrukcji podstawienia (na 2-bajtowych zmiennych typu INTEGER) zestawione w poniższej tabeli.

Instrukcja	MERITUM I (1.6 MHz) BASIC [µs]	MERITUM I (1.6 MHz) asembler [µs]	AMSTRAD 6128 (3.3 MHz) BASIC [µs]	AMSTRAD 6128 (3.3 MHz) asembler [µs]
I% = 1	1100	16.3	400	7.9
I% = J%	3100	20	900	10
I% = J%+K%	4500	39.4	1700	19
I% = J%*K%	4800	755	1800	366

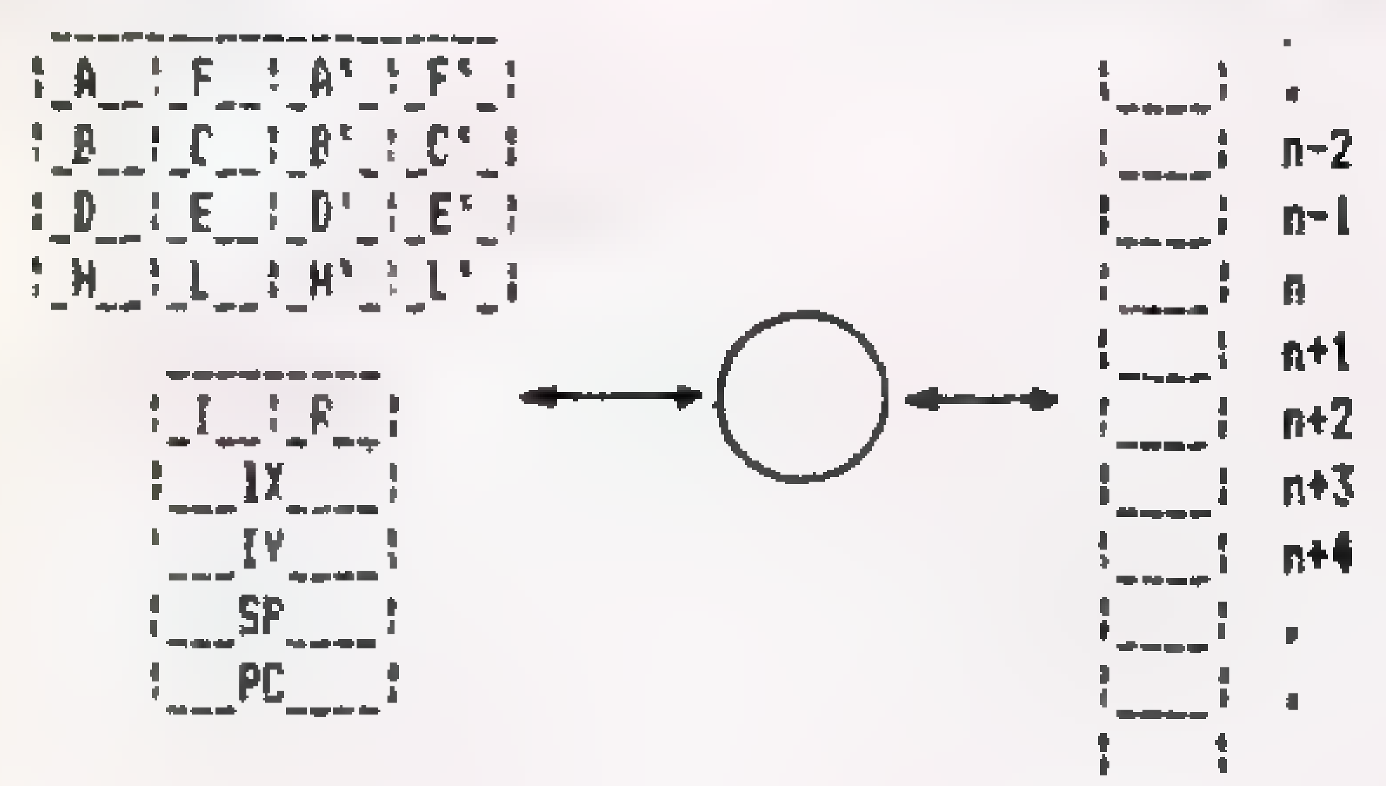
Wprawdzie porównanie jest wycinkowe (głębsza analiza wymaga oddzielnego opracowania), ale wyraźnie widać, że stosując oba rodzaje języków, ten sam mikrokomputer staje się narzędziem zupełnie innej klasy. Z mikrokomputerka przekształca się w "poważny" mikrokomputer. Jest to podstawowy powód, dla którego warto opanować niewątpliwie trudną sztukę programowania w języku asemblera.



Portret nowoczesnej szkolnej sekretarki...

Przesłania pomiędzy rejestrami i komórkami pamięci operacyjnej należą do najprostszych, ale najczęściej stosowanych operacji. Zwykle są zapisywane przy użyciu 1, 2 lub 3 rozkazów mikroprocesora. Początkujący programiści miewają sporo kłopotów z ich użyciem ze względu na dużą liczbę rejestrów i specyficzne, asymetryczne własności rozkazów. Termin asymetryczność rozkazów oznacza, że operują tylko na wybranych rejestrach lub parach rejestrów i w różny sposób mogą zmieniać wartość rejestru stanu wskaźników F.

Rejestry — Pamięć operacyjna

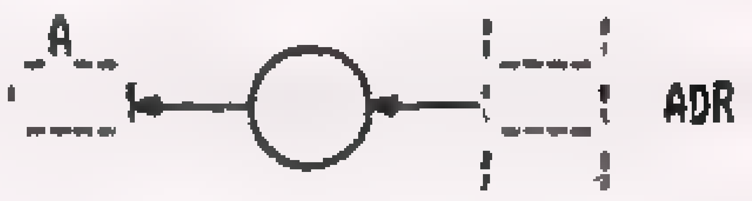


1.1. Przesłania do rejestrów

1.1.1. Bezpośrednie przesłania zawartości komórek pamięci operacyjnej do rejestrów

przesłanie zawartości komórki do rejestru A

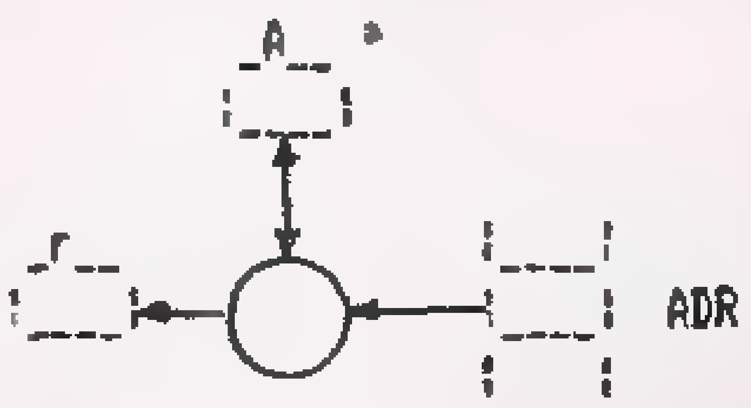
```
LD   A,(ADR)
```



t = 13, p = 3

przesłanie zawartości komórki do rejestru roboczego B, C, D, E, H lub L (r) za pośrednictwem rejestru A

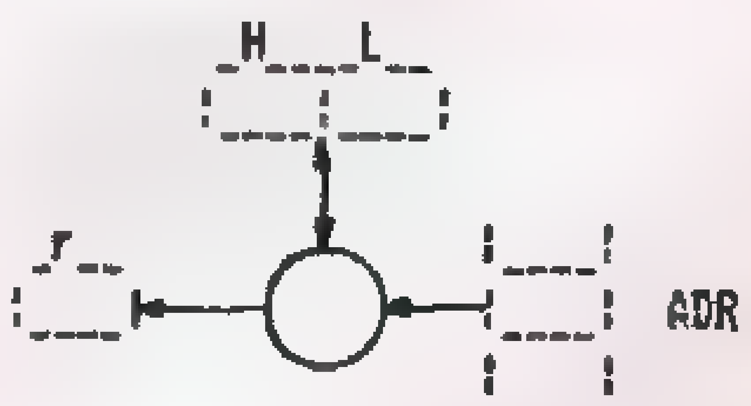
```
LD   A,(ADR)
LD   r,A
```



t = 17, p = 4

przesłanie zawartości komórki do rejestru roboczego A, B, C, D lub E (r) za pośrednictwem pary rejestrów HL

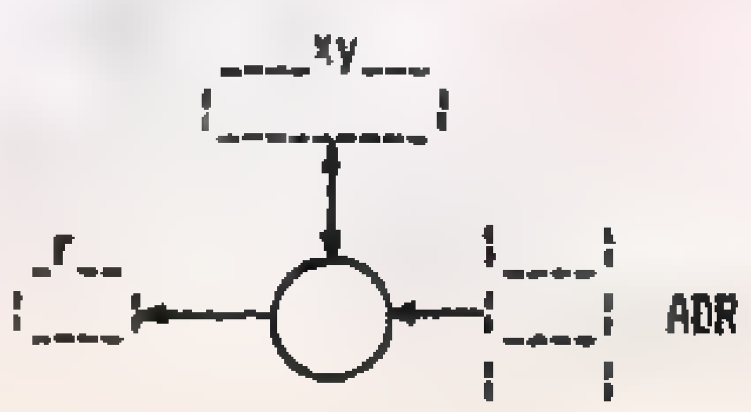
```
LD   HL,ADR
LD   r,(HL)
```



t = 17, p = 4

przesłanie zawartości komórki do rejestru roboczego A, B, C, D, E, H lub L (r) za pośrednictwem rejestru IX lub IY (xy)

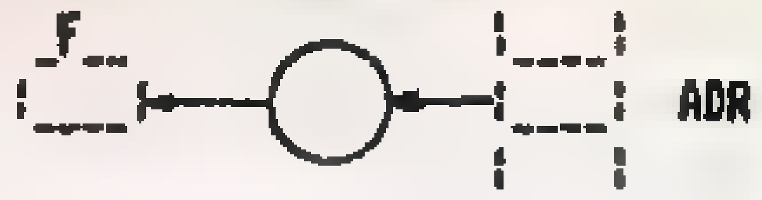
```
LD   xy,ADR
LD   r,(xy)
```



t = 33, p = 7

przesłanie zawartości komórki do rejestru znaczników F (zawartość rejestru A zostaje zniszczona)

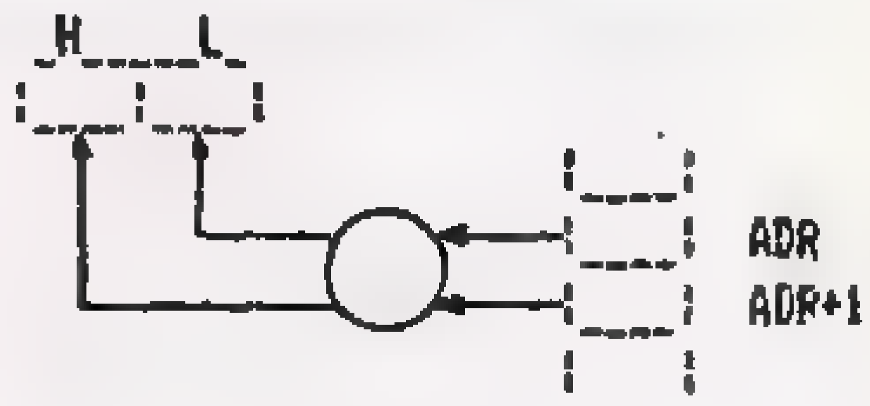
```
LD   HL,(ADR)
PUSH HL
POP  AF
```



t = 37, p = 5

przesłanie zawartości 2 komórek do pary rejestrów HL

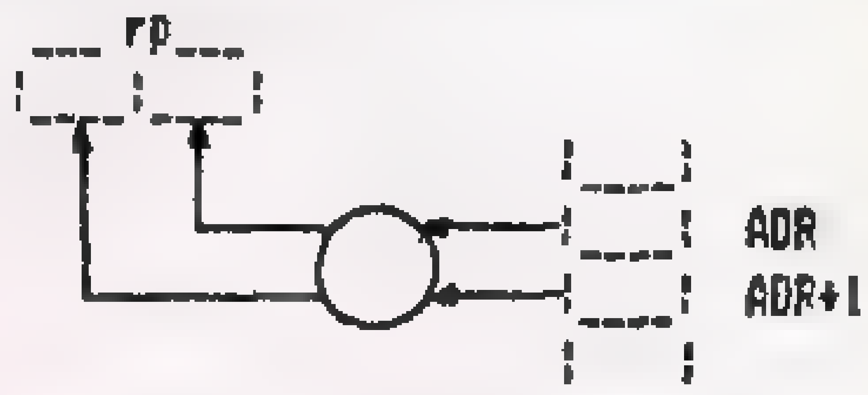
```
LD   HL,(ADR)
```



t = 16, p = 3

przesłanie zawartości 2 komórek do rejestrów BC, DE, SP, IX lub IY (rp)

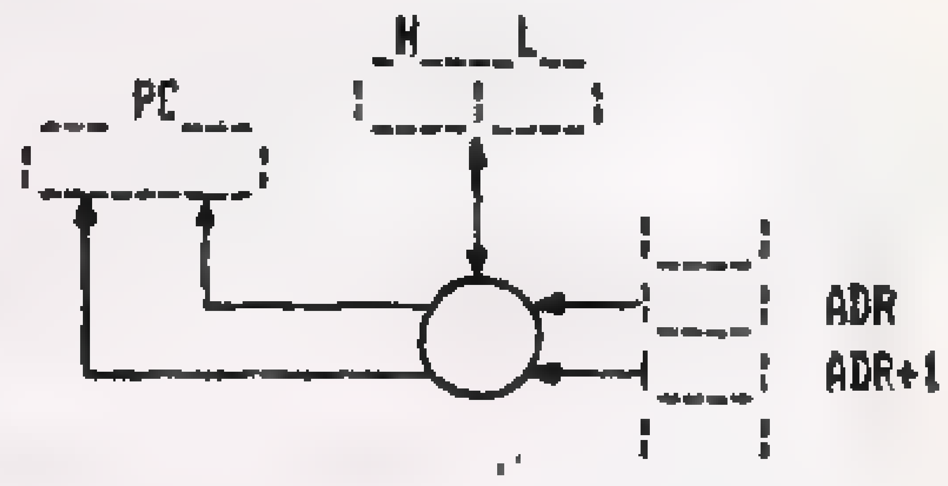
```
LD   rp,(ADR)
```



t = 20, p = 4

przesłanie zawartości 2 komórek do licznika rozkazów PC (skok do adresu zapisanego w tych komórkach)

```
LD   HL,(ADR)
JP   (HL)
```

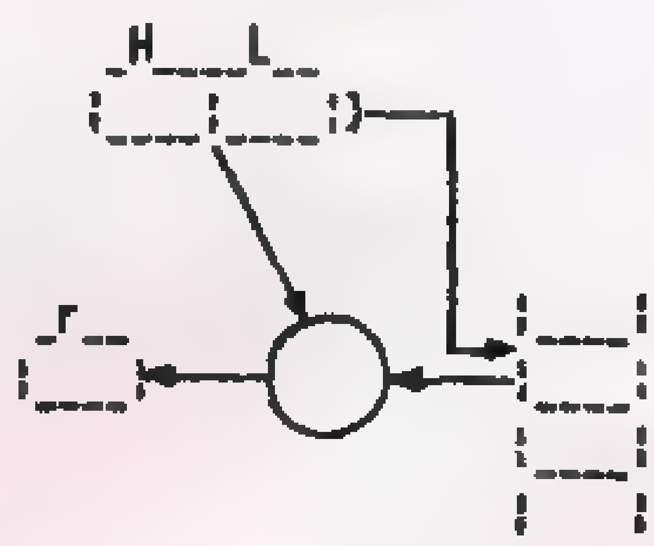


t = 20, p = 4

1.1.2. Pośrednie przesłania zawartości komórek pamięci operacyjnej

przesłanie zawartości komórki o adresie w HL do rejestru roboczego A, B, C, D lub E (r)

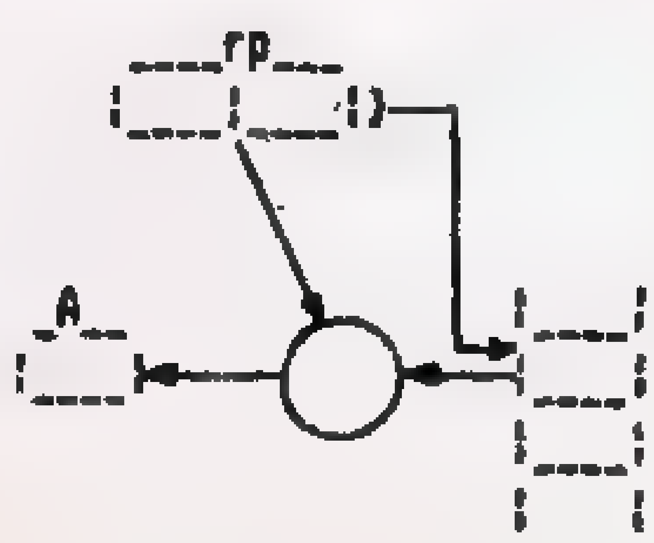
```
LD   r,(HL)
```



t = 7, p = 1

przesłanie zawartości komórki o adresie w HL, BC lub DE (rp) do rejestru A

```
LD   A,(rp)
```

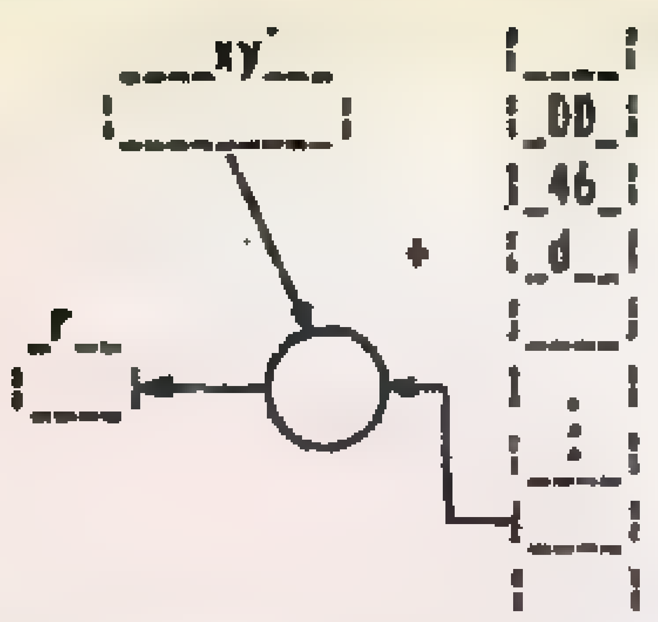


t = 7, p = 1

przesłanie zawartości komórki o adresie IX+d lub IY+d (xy+d, d=(-127,...,128)) do rejestru roboczego A, B, C, D, E, H lub L (r)

LD r,(xy+d)

t = 19, p = 3

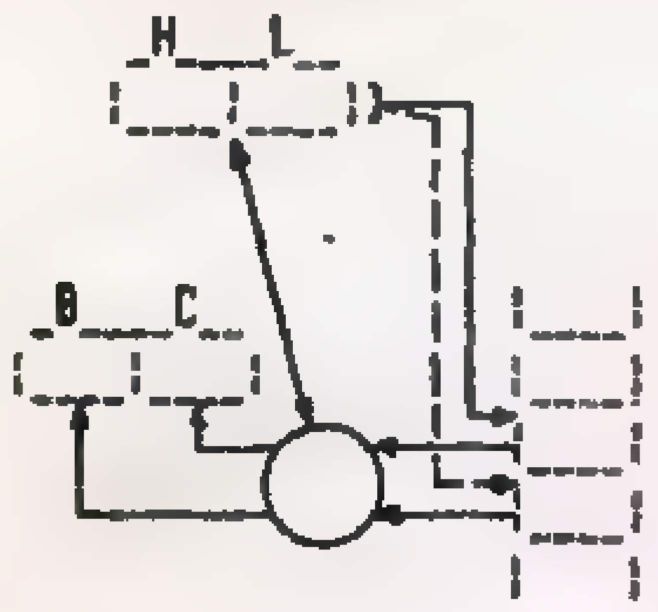


przesłanie zawartości 2 komórek o adresie w HL do pary rejestrów BC lub DE

1.1.3. Przesłania z wierzchołka stosu

LD C,(HL)
INC HL
LD B,(HL)

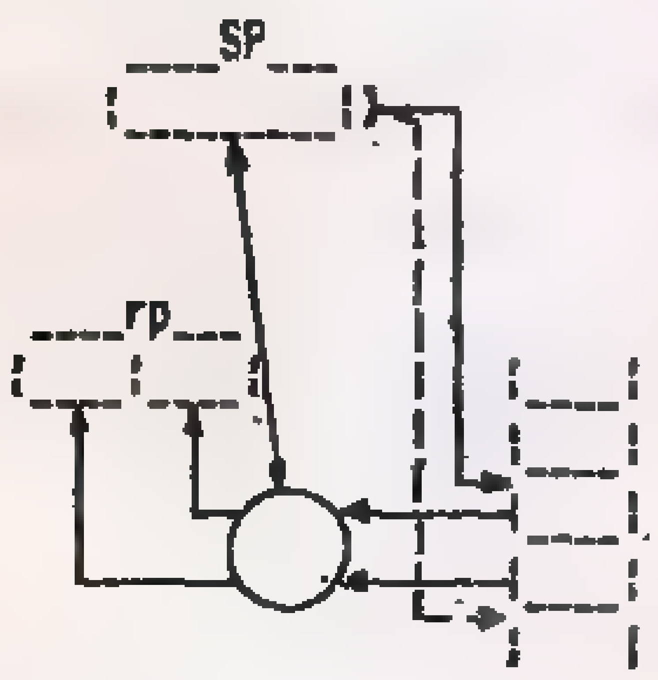
t = 20, p = 3



przesłanie zawartości 2 komórek z wierzchołka stosu do pary rejestrów AF, BC, DE lub HL (rp)

POP rp

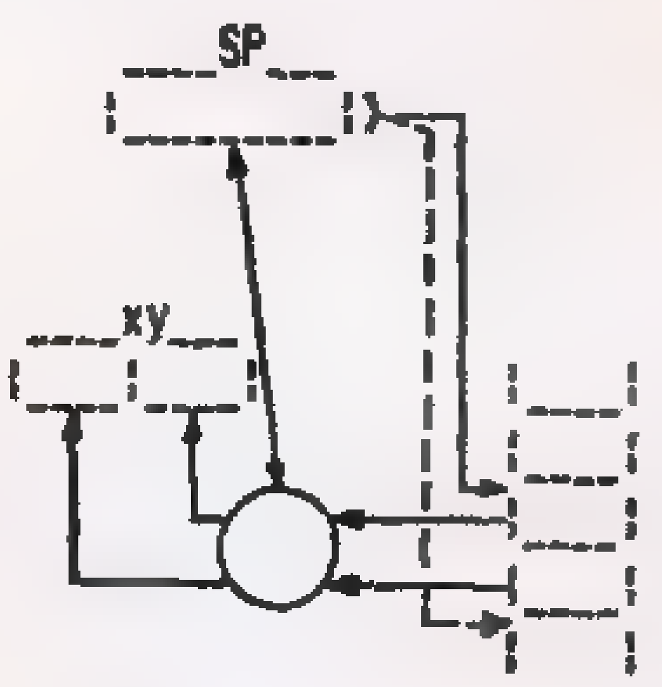
t = 10, p = 1



przesłanie zawartości 2 komórek z wierzchołka stosu do rejestru indeksowego IX lub IY (xy)

POP xy

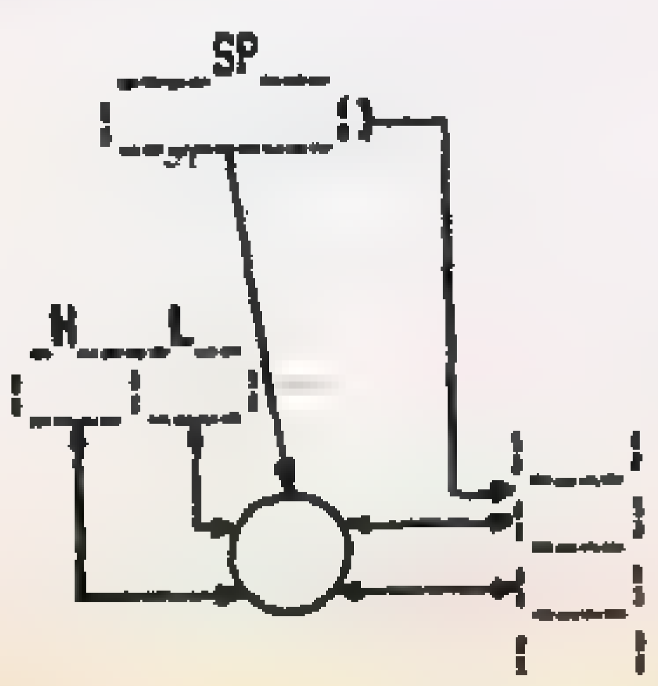
t = 14, p = 2



zamiana miejscami zawartości pary rejestrów HL z zawartością 2 komórek z wierzchołka stosu

EX (SP),HL

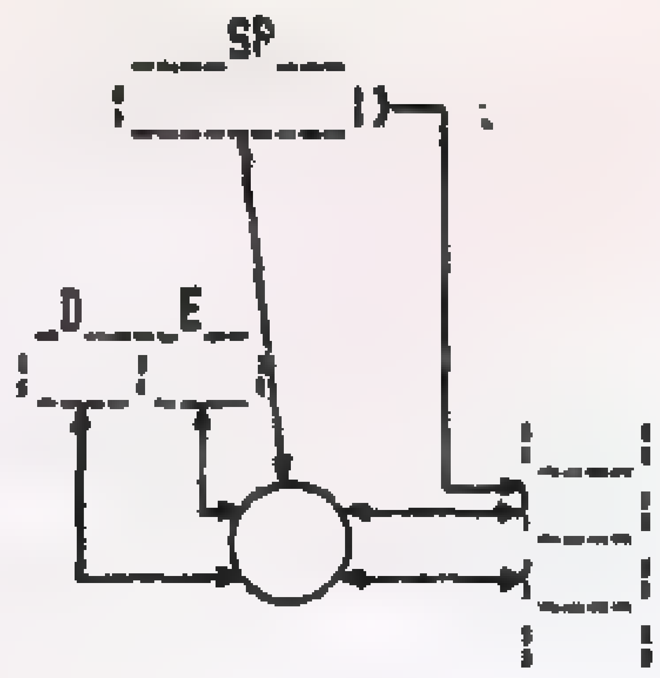
t = 19, p = 1



zamiana miejscami zawartości pary rejestrów DE z zawartością 2 komórek z wierzchołka stosu (bez naruszenia zawartości rejestrów HL)

EX DE,HL
EX (SP),HL
EX DE,HL

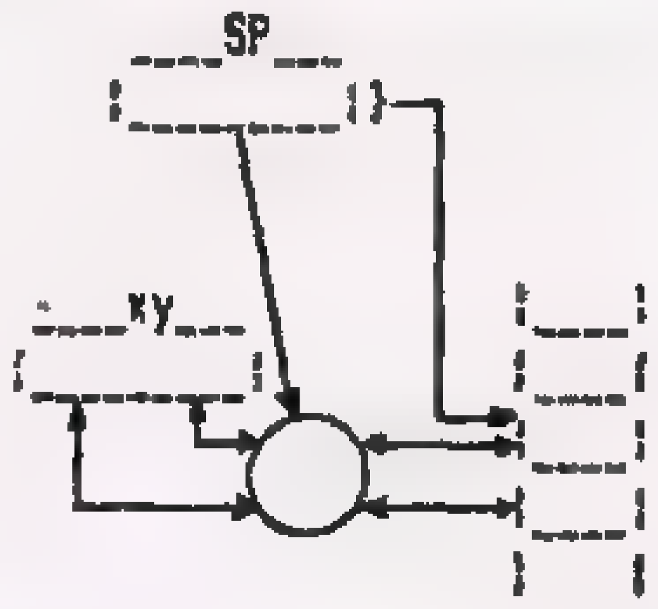
t = 27, p = 3



zamiana miejscami zawartości rejestru indeksowego IX lub IY (xy) z zawartością 2 komórek z wierzchołka stosu

EX (SP),xy

t = 23, p = 2

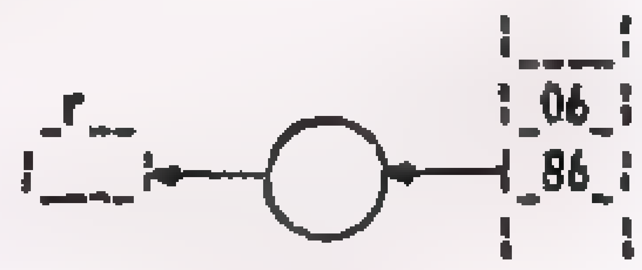


1.1.4. Przesłania stałych

przesłanie stałej (1 bajt) do rejestru roboczego A, B, C, D, E, H lub L (r)

LD r,86H

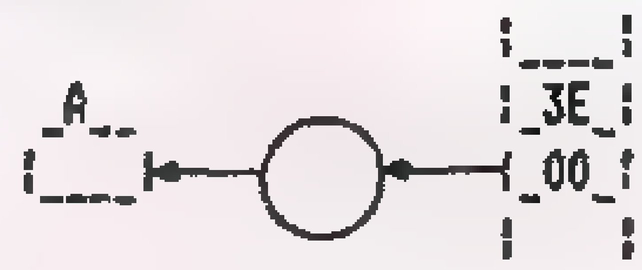
t = 7, p = 2



przesłanie 0 do rejestru A

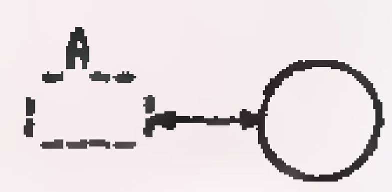
LD A,0

t = 7, p = 2



SUB A

t = 4, p = 1



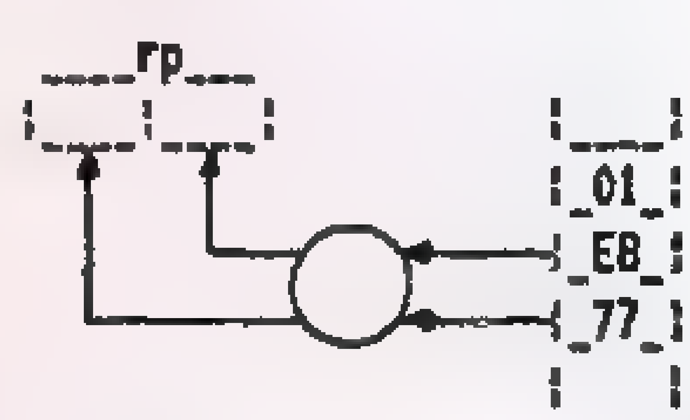
XOR A

t = 4, p = 1

przesłanie stałej (2 bajty) do pary rejestrów BC, DE, HL lub SP (rp)

LD rp,77E8H

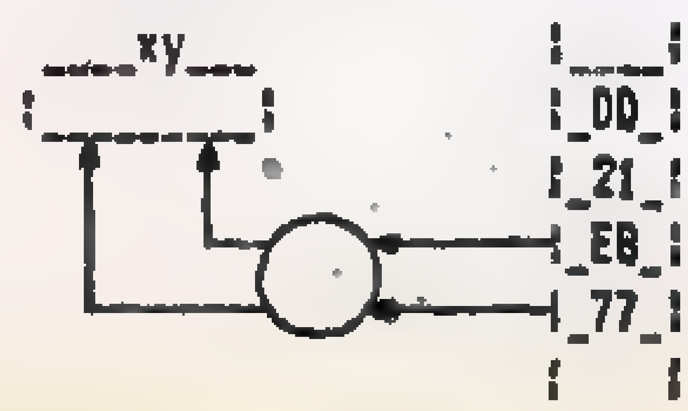
t = 10, p = 3



przesłanie stałej (2 bajty) do rejestru indeksowego IX lub IY (xy)

LD xy,77E8H

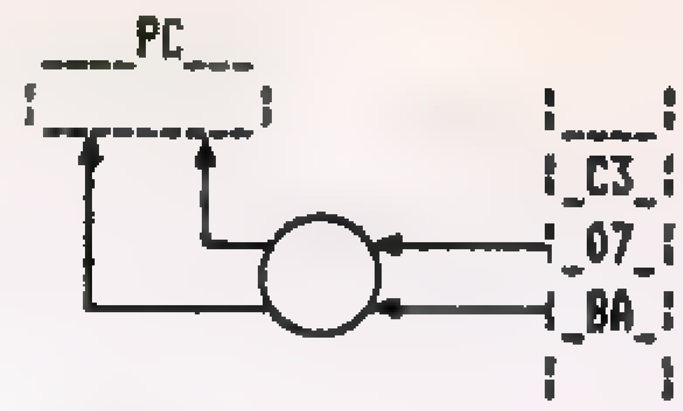
t = 14, p = 4



przesłanie stałej (2 bajty) do licznika rozkazów PC (skok wg podanego adresu)

```
JP   BA07H

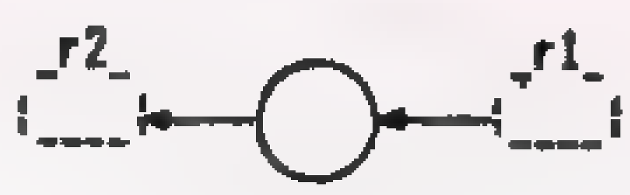
t = 10, p = 3
```



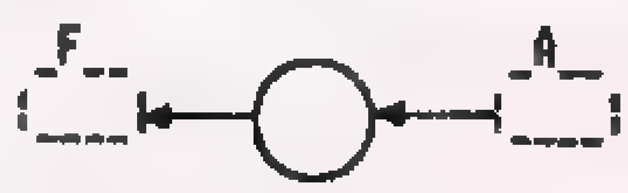
1.1.5. Przesłania pomiędzy rejestrami

przesłania pomiędzy rejestrami roboczymi A, B, C, D, E, H lub L (z r1 do r2)

```
LD   r2,r1
```



przesłanie zawartości rejestru A do rejestru znaczników F



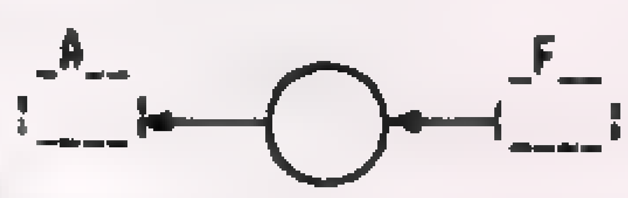

```
LD   H,A
LD   L,A
PUSH HL
POP  AF      ;zawartość rejestru A nie zostaje zniszczona

t = 29, p = 4

PUSH AF
INC  SP
POP  AF
DEC  SP      ;zawartość rejestru A zostaje zniszczona

t = 33, p = 4
```

przesłanie zawartości rejestru znaczników F do rejestru A



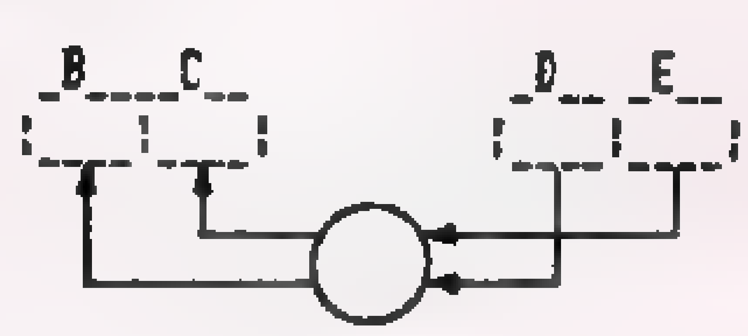

```
PUSH AF
POP  BC
LD   A,C     ;zawartość rejestru F nie zostaje zniszczona

t = 25, p = 3

PUSH AF
DEC  SP
POP  AF
INC  SP      ;zawartość rejestru F zostaje zniszczona

t = 33, p = 4
```

przesłania pomiędzy parami rejestrów BC, DE, HL (przykładowo pomiędzy rejestrami BC i DE)



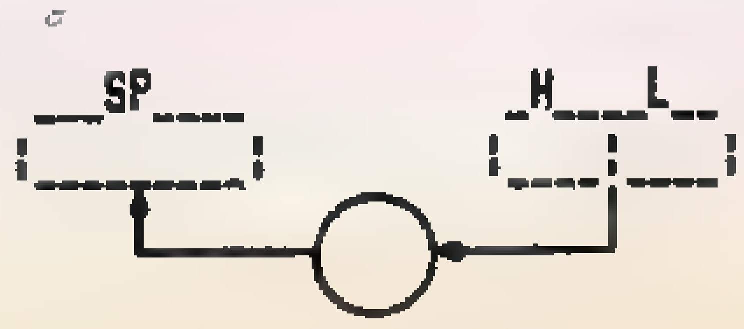

```
LD   C,E
LD   B,D

t = 8, p = 2

PUSH DE
POP  BC

t = 28, p = 2
```

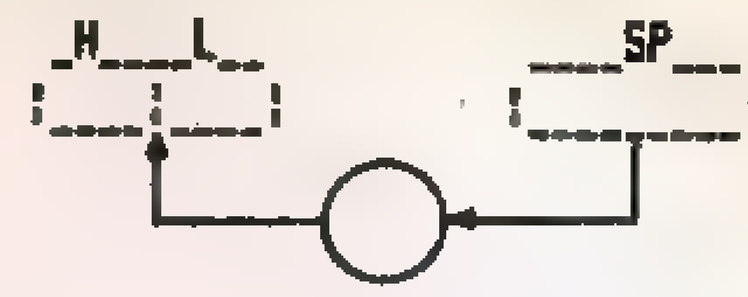
przesłanie zawartości pary rejestrów HL do wskaznika stosu SP




```
LD   SP,HL

t = 6, p = 1
```

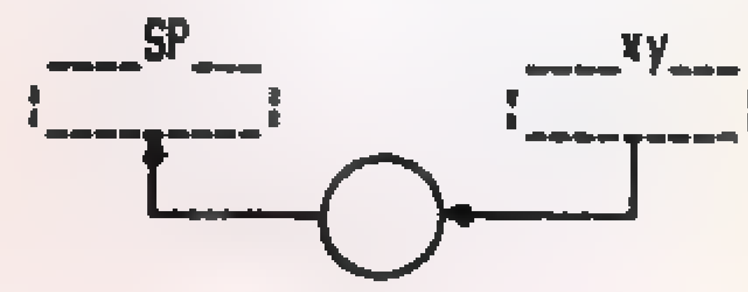
przesłanie zawartości wskaźnika stosu SP do pary rejestrów HL




```
LD   HL,0
ADD  HL,SP

t = 21, p = 4
```

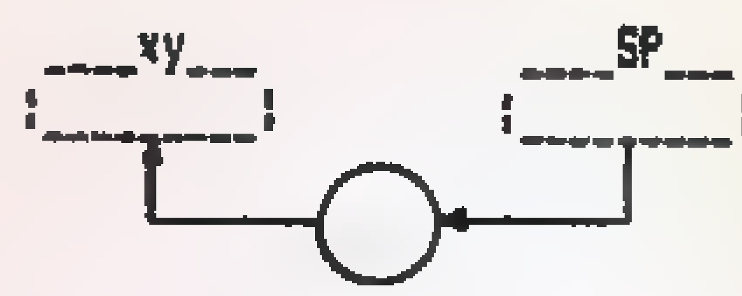
przesłanie zawartości rejestru indeksowego IX lub IY (xy) do wskaźnika stosu SP




```
LD   SP,xy

t = 10, p = 2
```

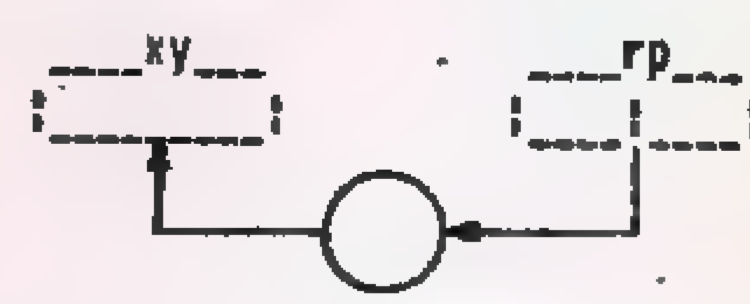
przesłanie zawartości wskaźnika stosu SP do rejestru indeksowego IX lub IY (xy)




```
LD   xy,0
ADD  xy,SP

t = 29, p = 6
```

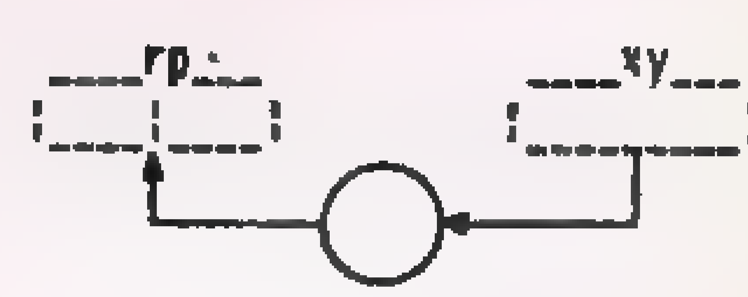
przesłanie zawartości pary rejestrów BC, DE lub HL (rp) do rejestru indeksowego IX lub IY (xy)




```
PUSH rp
POP  xy

t = 25, p = 3
```

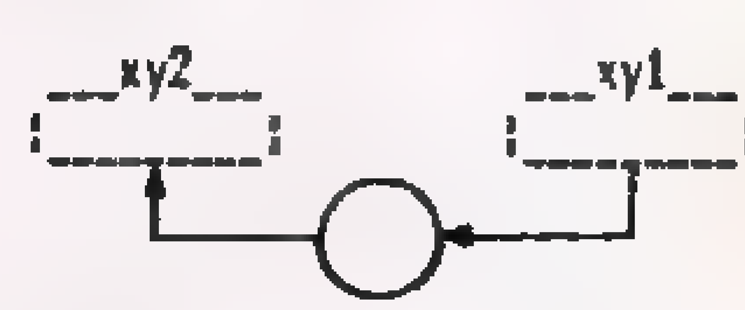
przesłanie zawartości rejestru indeksowego IX lub IY (xy) do pary rejestrów BC, DE lub HL (rp)




```
PUSH xy
POP  rp

t = 25, p = 3
```

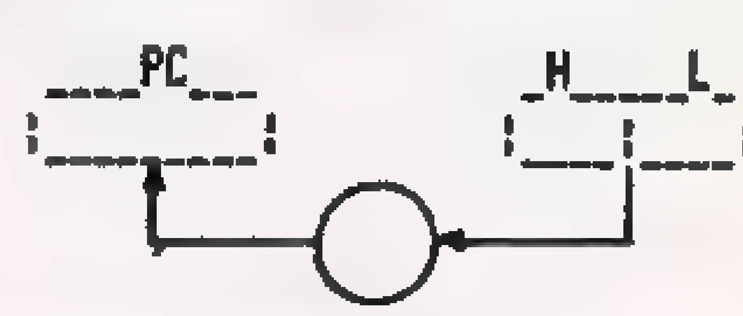
przesłania pomiędzy rejestrami indeksowymi IX, IY (z xy1 do xy2)




```
PUSH xy1
POP  xy2

t = 29, p = 4
```

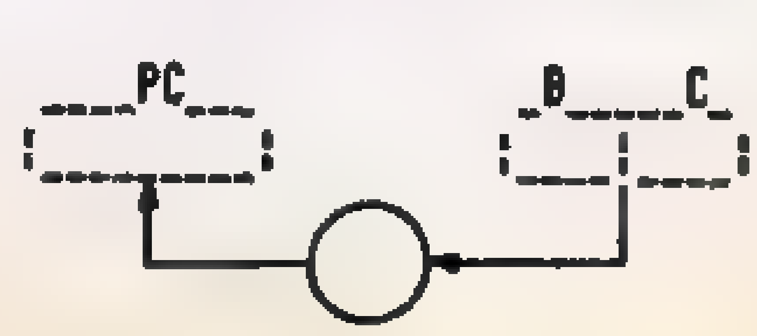
przesłanie zawartości pary rejestrów HL do licznika rozkazów PC (skok do adresu zapisanego w HL)




```
JP   (HL)

t = 4, p = 1
```

przesłanie zawartości pary rejestrów BC do licznika rozkazów PC (skok do adresu zapisanego w BC)




```
PUSH BC
RET

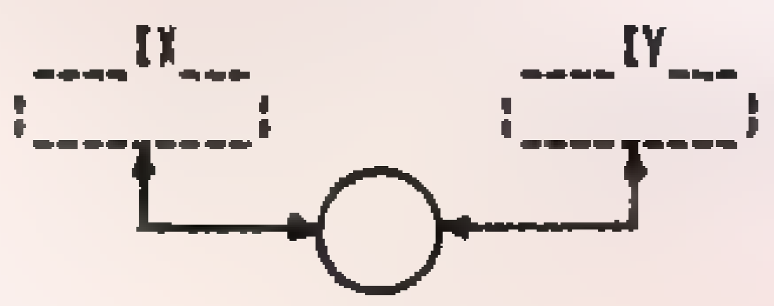
t = 21, p = 2
```

```
PUSH rp
EX   (SP),xy
POP  rp
```

t = 44, p = 4

-zamiana miejscami zawartości rejestru indeksowego IX z IY

```
PUSH IX
PUSH IY
POP  IX
POP  IY
```



t = 58, p = 8

```
PUSH IX
EX   (SP),IY
POP  IX
```

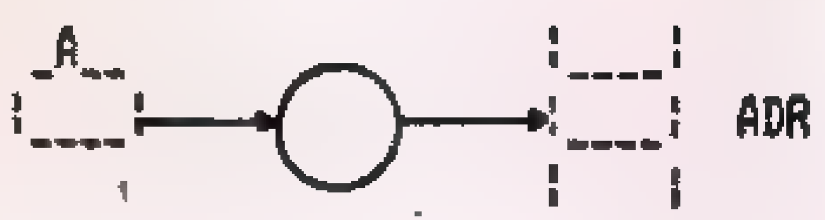
t = 52, p = 6

1.2. Przesłania do komórek pamięci operacyjnej

1.2.1. Bezpośrednie przesłania zawartości rejestrów

przesłanie zawartości akumulatora

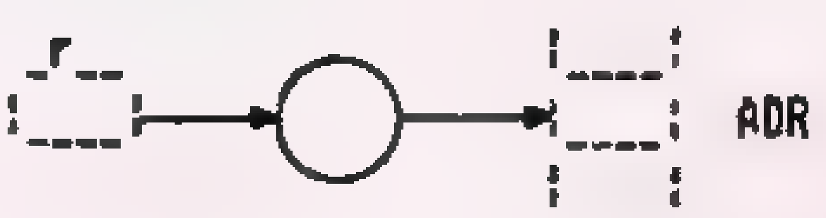
```
LD   (ADR),A
```



t = 13, p = 3

przesłanie zawartości rejestru roboczego B, C, D, E, H lub L (r) za pośrednictwem akumulatora

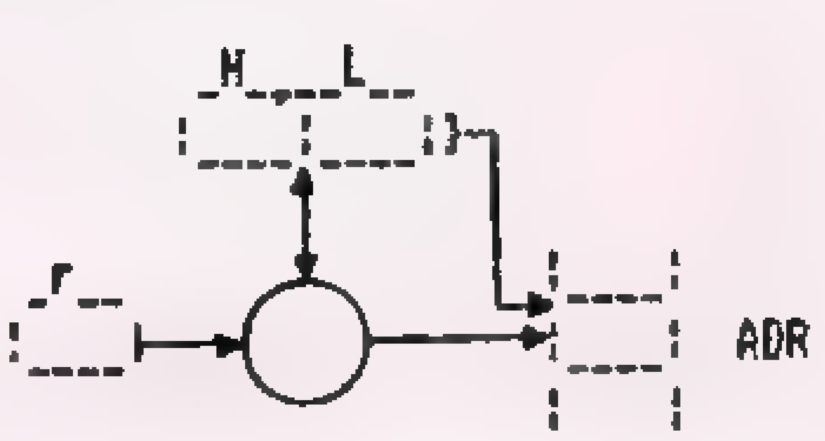
```
LD   A,r
LD   (ADR),A
```



t = 17, p = 4

przesłanie zawartości rejestru roboczego A, B, C, D lub E (r) za pośrednictwem pary rejestrów HL

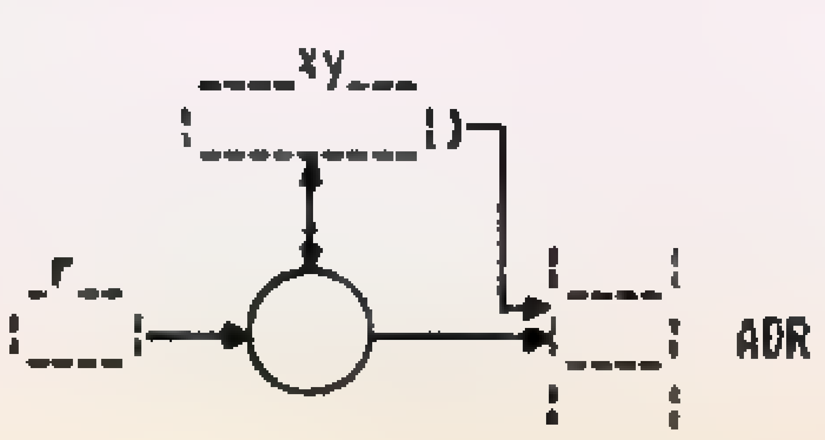
```
LD   HL,ADR
LD   (HL),r
```



t = 17, p = 4

przesłanie zawartości rejestru roboczego A, B, C, D, E, H lub L (r) za pośrednictwem rejestru IX lub IY (xy)

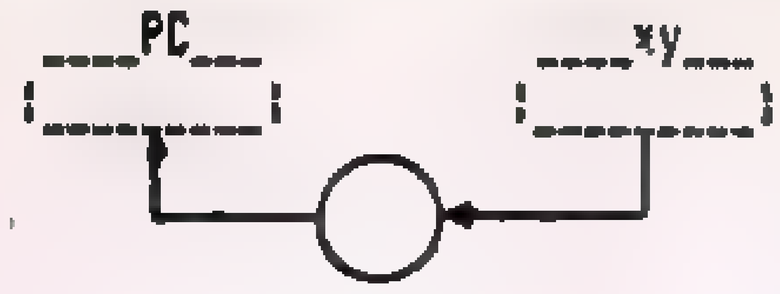
```
LD   xy,ADR
LD   (xy),r
```



t = 33, p = 7

przesłanie zawartości rejestru indeksowego IX lub IY (xy) do licznika rozkazów PC (skok do adresu zapisanego w xy)

```
JP   (xy)
```



t = 8, p = 2

zamiana miejscami zawartości rejestrów roboczych B, C, D, E, H, L (r1 z r2) za pośrednictwem rejestru A

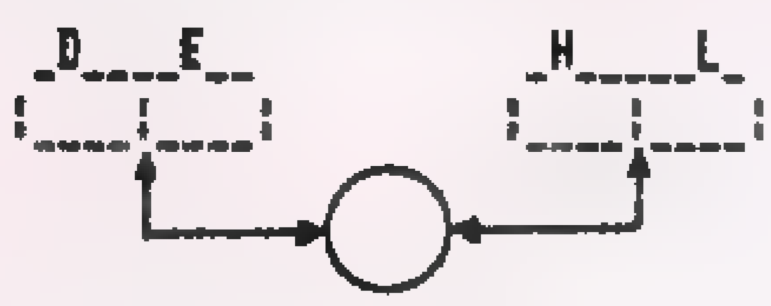
```
LD   A,r1
LD   r1,r2
LD   r2,A
```



t = 17, p = 4

zamiana miejscami zawartości pary rejestrów DE z HL

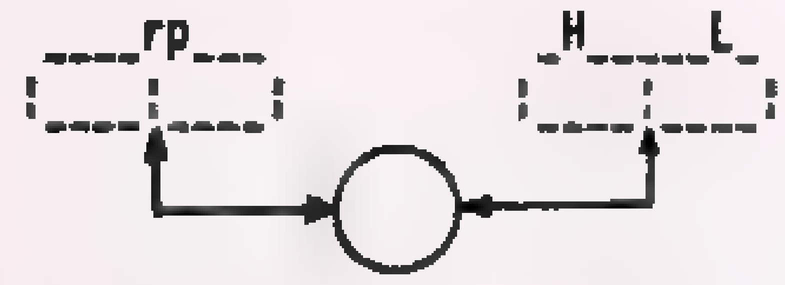
```
EX   DE,HL
```



t = 4, p = 1

zamiana miejscami zawartości pary rejestrów AF, BC lub DE (rp) z parą rejestrów HL

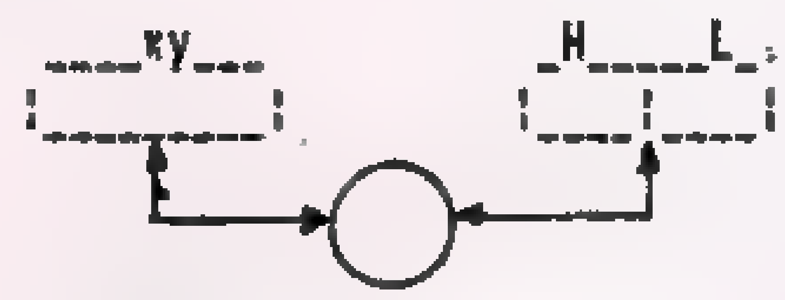
```
PUSH rp
EX   HL,(SP)
POP  rp
```



t = 40, p = 3

zamiana miejscami zawartości rejestrów indeksowych IX lub IY (xy) z parą rejestrów HL

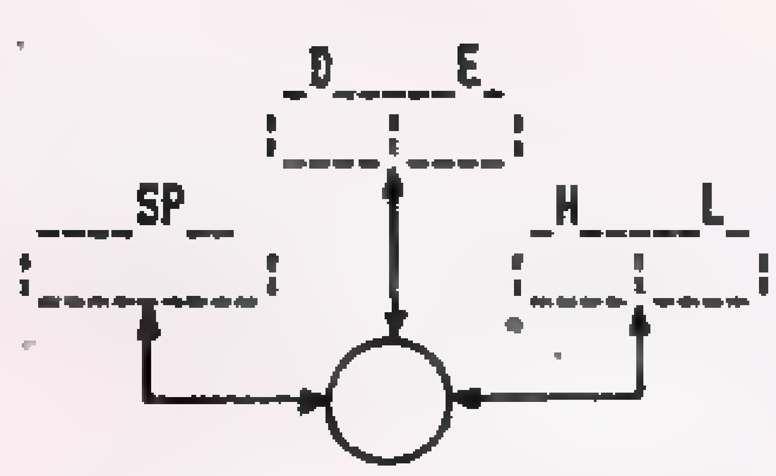
```
PUSH xy
EX   (SP),HL
POP  xy
```



t = 52, p = 6

zamiana miejscami zawartości wskaźnika stosu SP z parą rejestrów HL

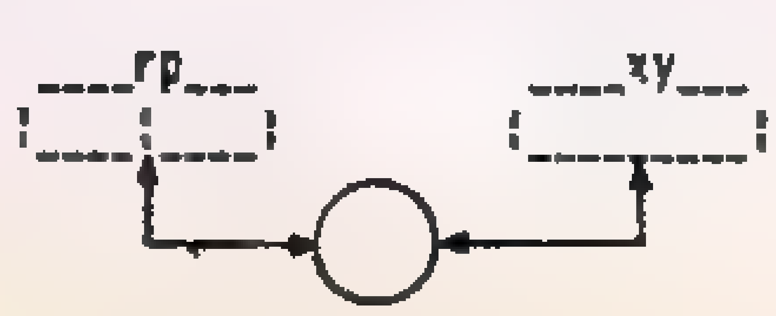
```
EX   DE,HL
LD   HL,0
ADD  HL,SP
EX   DE,HL
LD   SP,HL
EX   DE,HL
```



t = 39, p = 8

zamiana miejscami zawartości pary rejestrów BC, DE lub HL (rp) z rejestrami indeksowymi IX lub IY (xy)

```
PUSH rp
PUSH xy
POP  rp
POP  xy
```

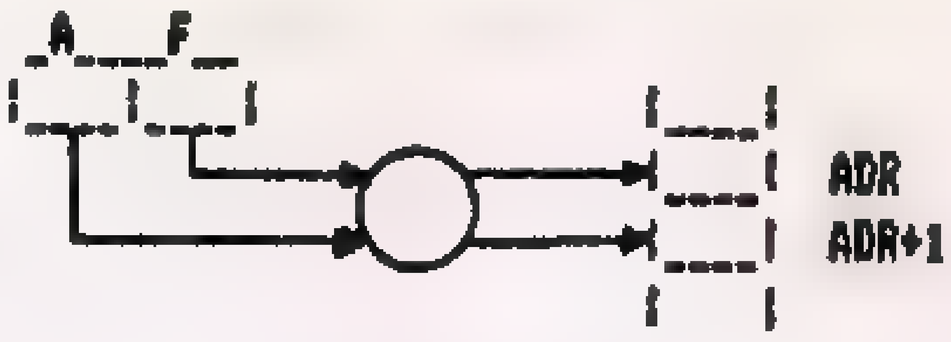


t = 50, p = 6

przesłanie zawartości rejestru znaczników F i akumulatora

PUSH AF
POP HL
LD   (ADR),HL

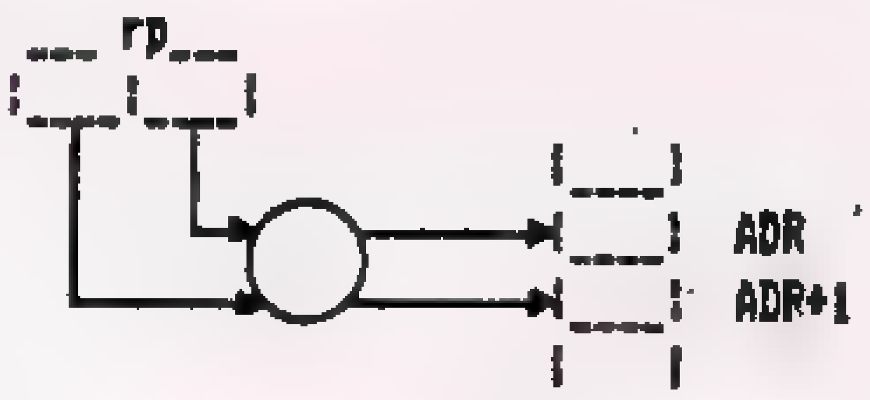
t = 37, p = 5



przesłanie zawartości rejestrów BC, DE, IX, IY lub SP (rp)

LD   (ADR),rp

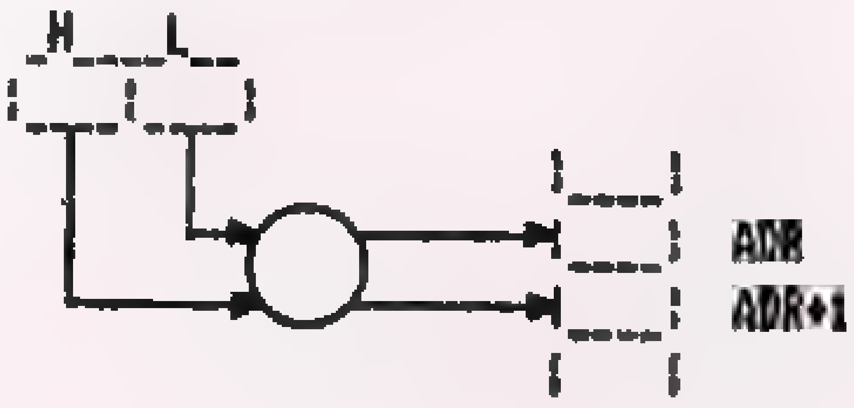
t = 20, p = 4



przesłanie zawartości pary rejestrów HL

LD   (ADR),HL

t = 16, p = 3

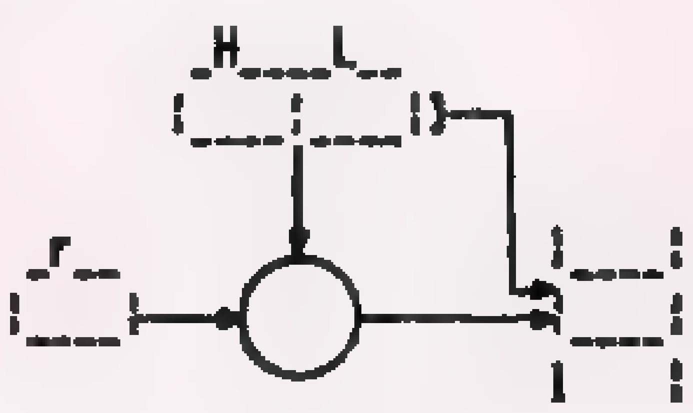


1.2.2. Pośrednie przesłania zawartości rejestrów

przesłanie zawartości rejestru roboczego A, B, C, D, E, H lub L (r) do pamięci o adresie w HL

LD   (HL),r

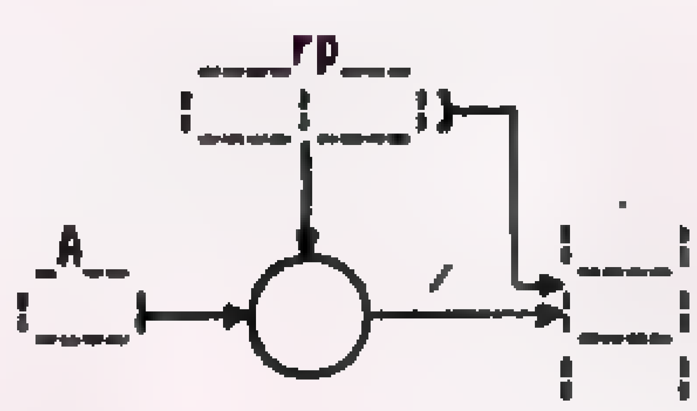
t = 7, p = 1



przesłanie zawartości rejestru A do pamięci o adresie w BC, DE lub HL (rp)

LD   (rp),A

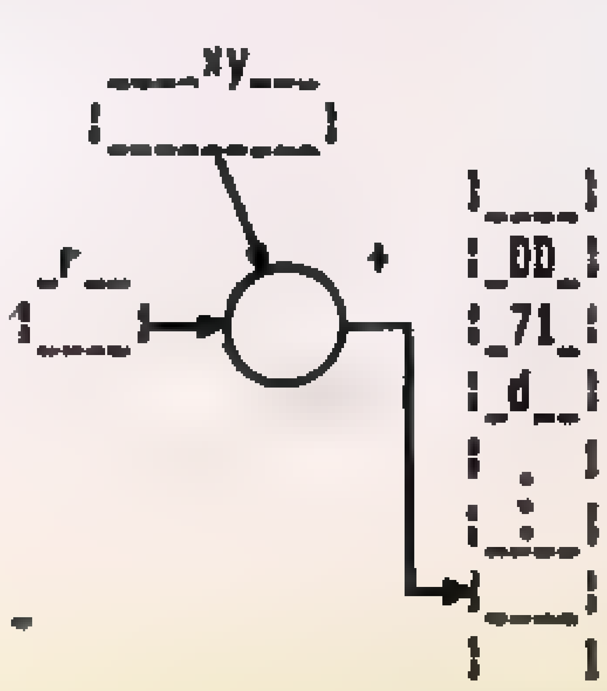
t = 7, p = 1



przesłanie zawartości rejestru A, B, C, D, E, H lub L (r) do pamięci o adresie IX+d lub IY+d (xy+d, d = (-127,...,128))

LD   (xy+d),r

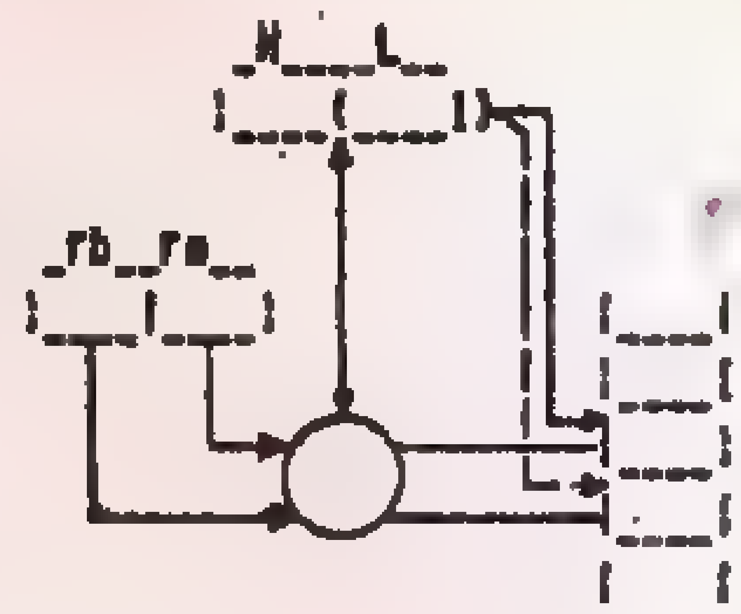
t = 19, p = 3



przesłanie zawartości pary rejestrów BC lub DE (rb = (B, D), ra = (C, E)) do pamięci o adresie w HL

LD   (HL),ra
INC HL
LD   (HL),rb

t = 20, p = 3

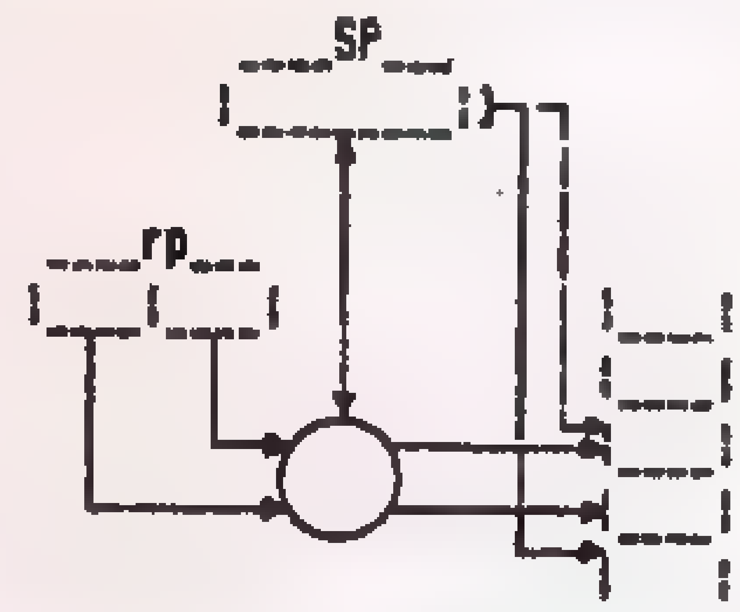


1.2.3. Przesłania na wierzchołek stosu

przesłanie na wierzchołek stosu zawartości pary rejestrów AF, BC, DE lub HL (rp)

PUSH rp

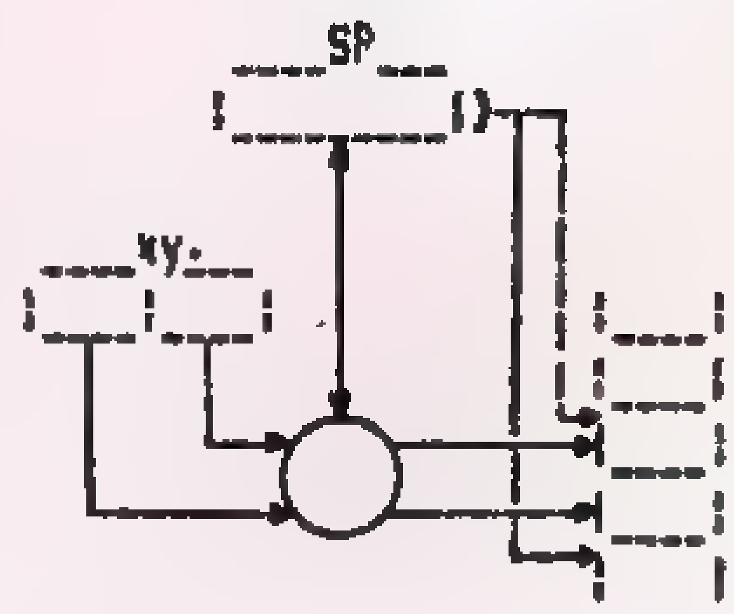
t = 11, p = 1



przesłanie na wierzchołek stosu zawartości rejestru indeksowego IX lub IY (xy)

PUSH xy

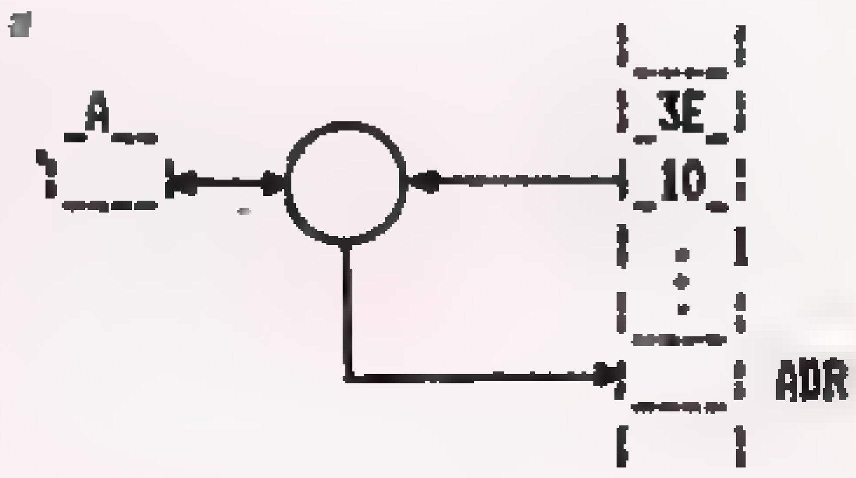
t = 15, p = 2



1.2.4. Przesłania stałych

przesłanie stałej (1 bajt) za pośrednictwem rejestru roboczego A

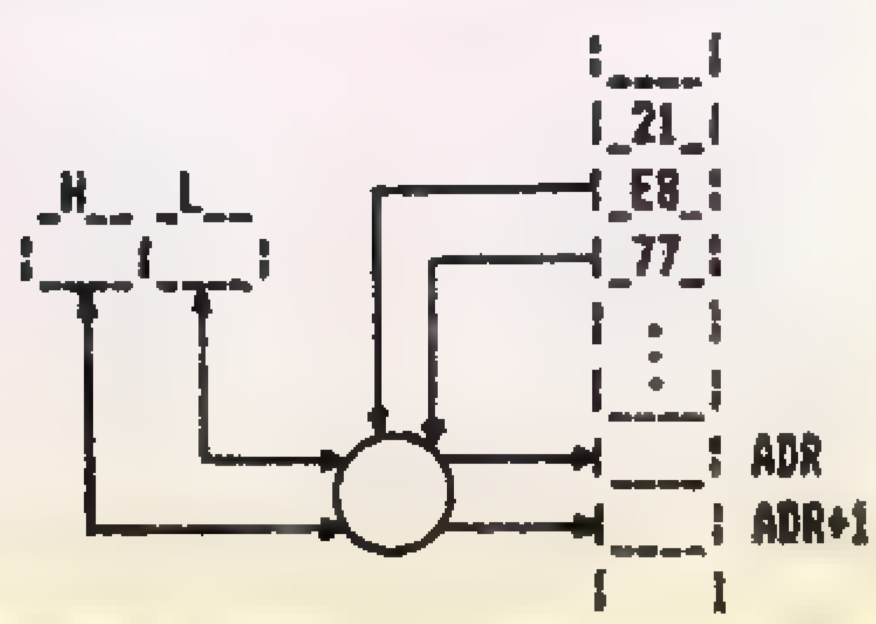
LD   A,10H
LD   (ADR),A
t = 20, p = 5



przesłanie stałej (2 bajty) za pośrednictwem pary rejestrów HL

LD   HL,77E8H
LD   (ADR),HL

t = 26, p = 6



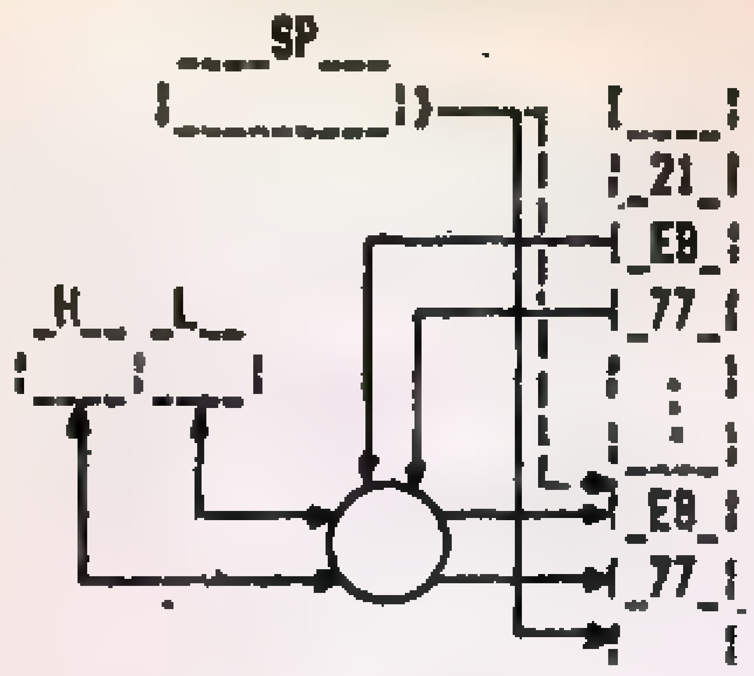
przesłanie stałej (2 bajty) na wierzchołek stosu

```
LD   HL,77EBH
PUSH HL
```

t = 21, p = 4

```
PUSH HL
LD   HL,77EBH
EX   (SP),HL
```

t = 40, p = 5

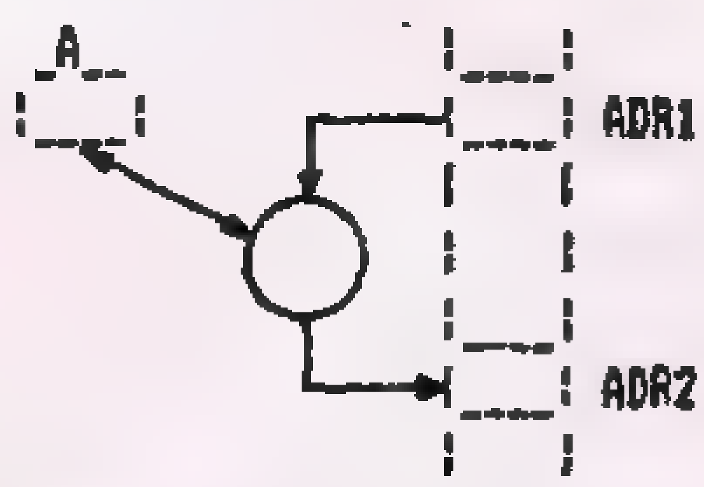


1.2.5. Przesłania pomiędzy komórkami pamięci operacyjnej

przesłanie bajtu za pośrednictwem rejestru A

```
LD   A,(ADR1)
LD   (ADR2),A
```

t = 26, p = 6



przesłanie bajtu przy użyciu rozkazu LDI lub LDD

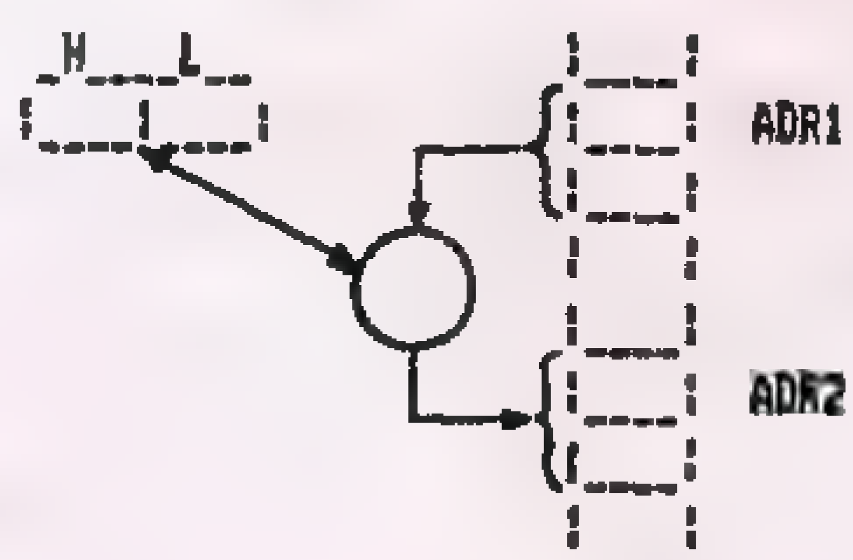
```
LD   BC,1          LD   BC,1
LD   HL,ADR1       LD   HL,ADR1
LD   DE,ADR2       LD   DE,ADR2
LDI                LDD
```

t = 46, p = 11

przesłanie 2 bajtów za pośrednictwem pary rejestrów HL oraz BC, DE, SP, IX lub IY (rp)

```
LD   HL,(ADR1)
LD   (ADR2),HL
```

t = 32, p = 6



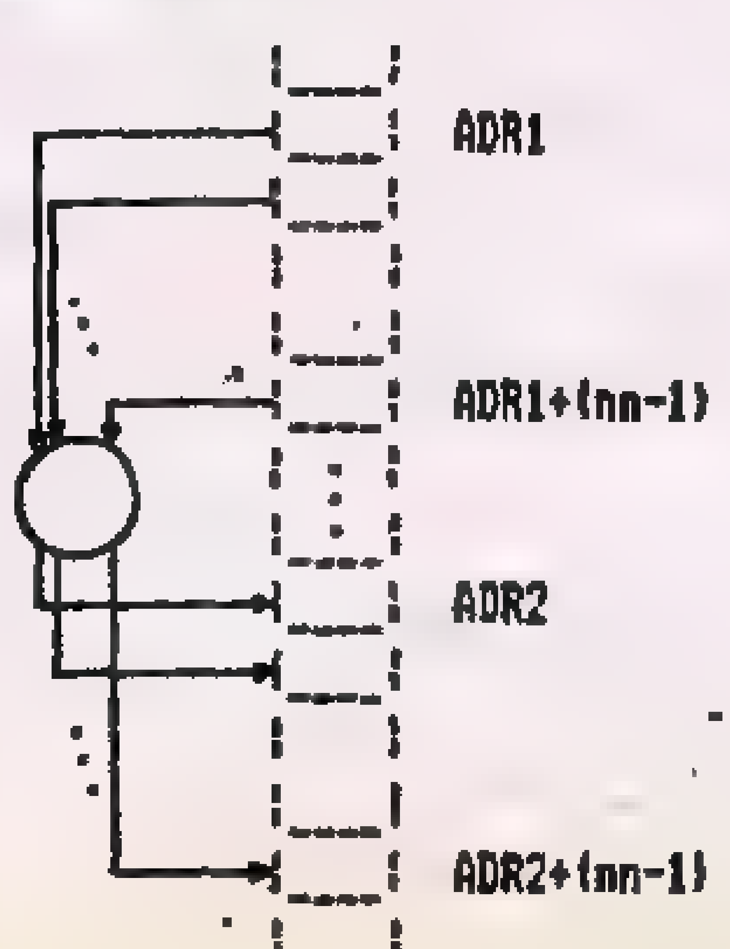

```
LD   rp,(ADR1)
LD   (ADR2),rp
```

t = 40, p = 8

przesłanie bloku nn bajtów przy użyciu rozkazu LDIR (adresy przesyłanych bajtów wzrastają od ADR1 do ADR1+(nn-1))

```
LD   BC,nn
LD   HL,ADR1
LD   DE,ADR2
LDIR
```

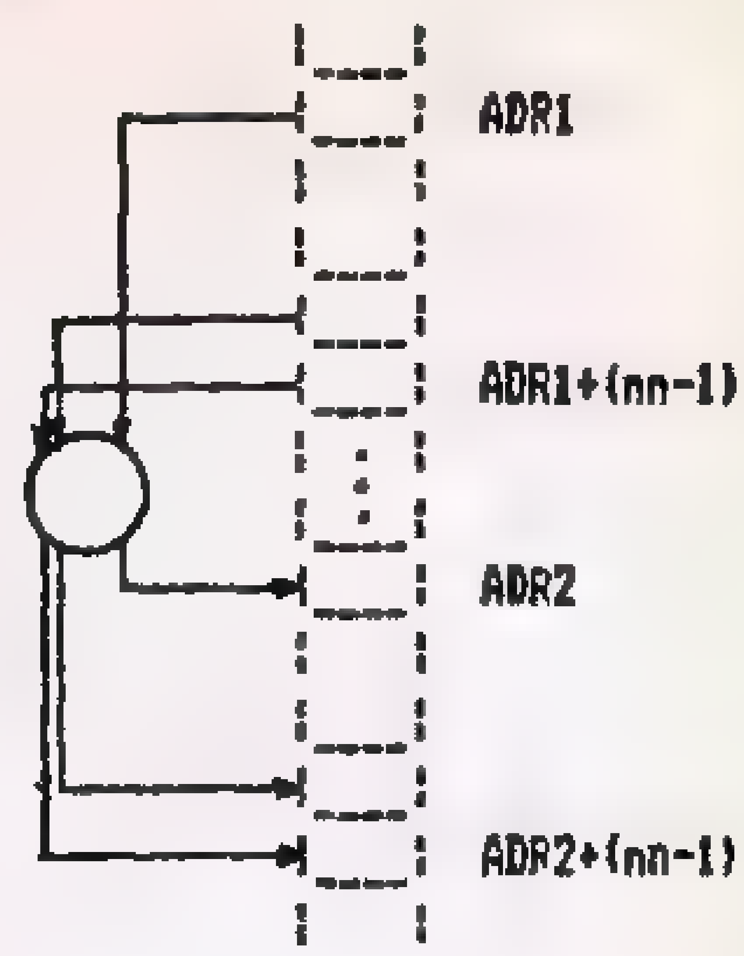
t = 21*nn + 25, p = 11
(0 <= nn <= 65355)



przesłanie bloku nn bajtów przy użyciu rozkazu LDDR (adresy przesłanych bajtów maleją od ADR1+(nn-1) do ADR1)

```
LD   BC,nn
LD   HL,ADR1+(nn-1)
LD   DE,ADR2+(nn-1)
LDDR
```

t = 21*nn + 25, p = 11
(0 <= nn <= 65355)



przesłania bloku n i nn bajtów za pośrednictwem rejestru A

```
    LD   HL,ADR1               LD   HL,ADR1
    LD   DE,ADR2               LD   DE,ADR2
    LD   B,n                   LD   BC,nn
P:  LD   A,(HL)            P:  LD   A,(HL)
    LD   (DE),A                LD   (DE),A
    INC  DE                    INC  DE
    INC  HL                    INC  HL
    DJNZ P                     DEC  BC
                               LD   A,B
                               OR   C
                               JR   NZ,P
```

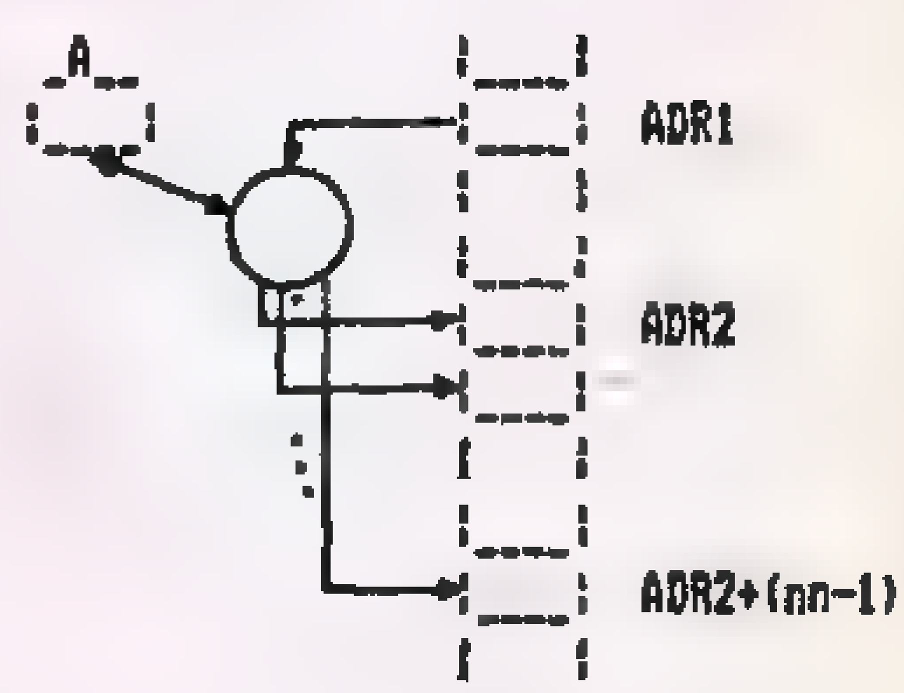
t = 39*n + 22, p = 14
(0 <= n <= 255)
dla n=0 przesłaniu podlega 256 bajtów

t = nn*52 + 25, p = 18
(0 <= nn <= 65535)
dla nn=0 przesłaniu podlega 65536 bajtów

powielenie zawartości komórki

```
LD   A,(ADR1)
LD   BC,nn-1
LD   HL,ADR2
LD   (HL),A
LD   DE,ADR2+1
LDIR
```

t = 21*nn + 24, p = 15
(0 <= nn <= 65355)



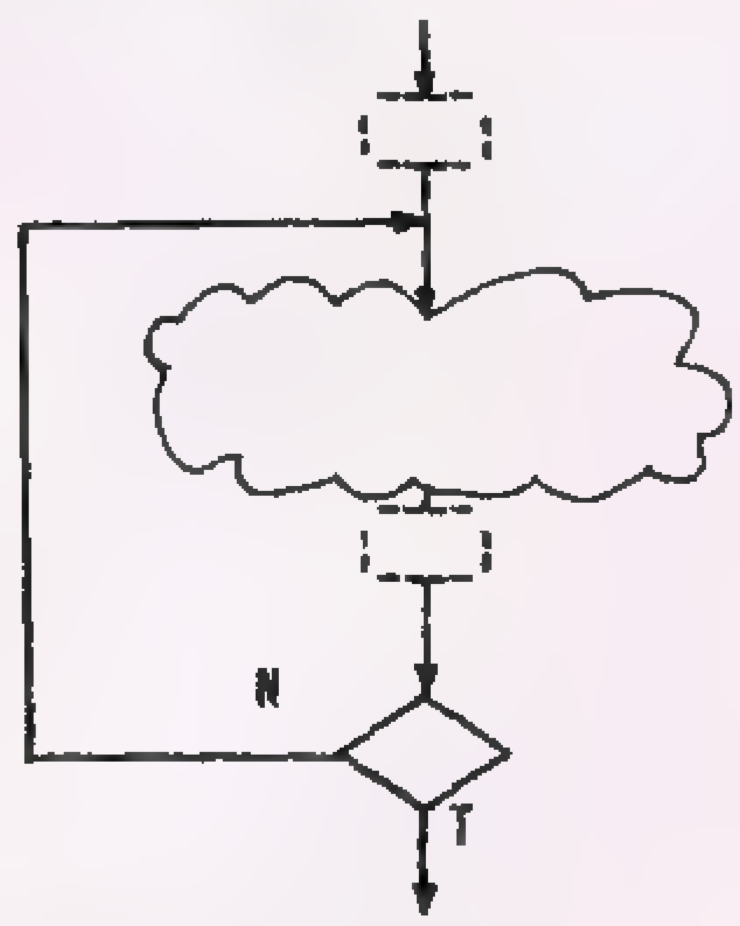

```
    LD   A,(ADR1)
    LD   B,n
    LD   HL,ADR2
P:  LD   (HL),A
    INC  HL
    DJNZ P
```

t = 26*n + 25, p = 12
(1 <= n <= 255)
dla n=0 powielanie realizowane jest 256 razy

2. PĘTLE ITERACYJNE

Najkrótsze czasy przebiegu pętli uzyskuje się dzięki użyciu rozkazu DJNZ i rejestru B, ale tylko dla pętli wykonywanej do 256 razy. Rozkazy mikroprocesora są lepiej przystosowane do kodowania pętli typu "repeat" niż pętli typu "while".

2.1. Pętle typu "repeat"



2.1.1. Pętle wykonywane do 256 razy

Dla n=0 rozkazy objęte pętlą będą wykonywane 256 razy

pętla kodowana przy użyciu rozkazu DJNZ i rejestru B

```
     LD   B,n
P:   .
     .
     .
     DJNZ P
```

$t = 13*n + 2$, $p = 4$
$(0 <= n <= 255)$

pętla kodowana przy użyciu rozkazu DEC i rejestru A, B, C, D, E, H lub L (r)

```
     LD   r,n
P:   .
     .
     .
     DEC  r
     JR   NZ,P
```

$t = 16*n + 2$, $p = 5$
$(0 <= n <= 255)$

pętla kodowana przy użyciu licznika zapamiętanego w komórce pamięci operacyjnej (LICZNIK)

```
     LD   A,n
     LD   (LICZNIK),A
P:   .
     .
     .
     LD   A,(LICZNIK)
     DEC  A
     LD   (LICZNIK),A
     JR   NZ,P
```

$t = 42*n + 15$, $p = 14$
$(0 <= n <= 255)$

pętla gniazdowana

```
     LD   C,n1          ; licznik pętli zewnętrznej
P1:  LD   B,n2          ; licznik pętli wewnętrznej
P2:  .
     .
     .
     DJNZ P2
     DEC  C
     JR   NZ,P1
```

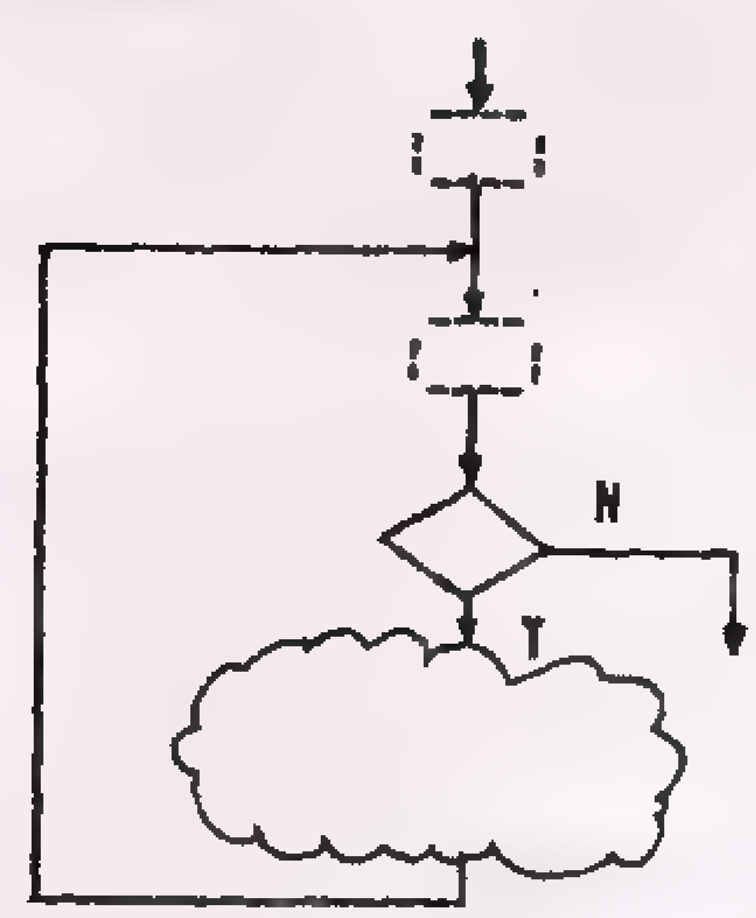
$t = 13*n1*n2 + 18*n1 + 2$, $p = 9$
$(0 <= n1 <= 255,\ 0 <= n2 <= 255)$
dla n1=0 i n2=0 instrukcje objęte pętlami będą wykonywane 65536 (256*256) razy

2.1.2. Pętla wykonywana do 65536 razy

```
     LD   BC,nn
P:   .
     .
     .
     DEC  BC
     LD   A,B
     OR   C
     JR   NZ,P
```

$t = 26*nn + 5$, $p = 8$
$(0 <= nn <= 65535)$
dla nn=0 instrukcje objęte pętlą będą wykonywane 65536 razy

2.2. Pętla typy "while"



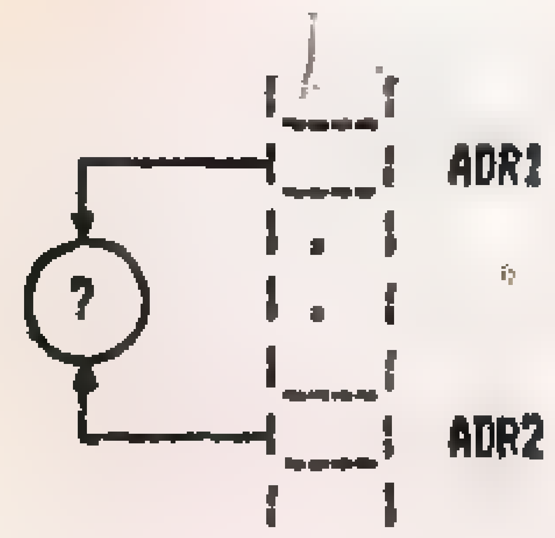
pętla kodowana przy użyciu rozkazu DEC i rejestru A, B, C, D, E, H lub L (r)

```
     LD   r,n+1
P1:  DEC  r
     JR   Z,P2
     .
     .
     .
     JR   P1
P2:  ...
```

$t = 23*n + 16$, $p = 7$
$(1 <= n <= 254)$

3. WARUNKI

3.1. Porównywanie 1-bajtowych tekstów znakowych lub liczb bez znaków



porównywanie zawartości 2 komórek pamięci operacyjnej o adresach ADR1 i ADR2

Warunek	Program	t	p
(ADR1) > (ADR2)	LD A,(ADR2) LD HL,ADR1 CP (HL) JR NC,NIE TAK: ...	37 – tak; 42 – nie	9
(ADR1) < (ADR2)	LD A,(ADR1) LD HL,ADR2 CP (HL) JR C,TAK NIE: ...	42 – tak; 37 – nie	9
(ADR1) >= (ADR2)	LD A,(ADR1) LD HL,ADR2 CP (HL) JR NC,TAK NIE: ...	42 – tak; 37 – nie	9
(ADR1) <= (ADR2)	LD A,(ADR2) LD HL,ADR1 CP (HL) JR NC,TAK NIE: ...	42 – tak; 37 – nie	9
(ADR1) = (ADR2)	LD A,(ADR1) LD HL,ADR2 CP (HL) JR Z,TAK NIE: ...	42 – tak; 37 – nie	9
(ADR1) # (ADR2)	LD A.(ADR1) LD HL,ADR2 CP (HL) JR NZ,TAK NIE: ...	42 – tak; 37 – nie	9

porównywanie zawartości komórki pamięci operacyjnej o adresie ADR ze stałą n

Warunek	Program	t	p
(ADR) > n	LD A,(ADR) CP n JR C,NIE JR Z,NIE TAK: ...	28 – tak; 26 lub 33 – nie	6
(ADR) < n	LD A,(ADR) CP n JR C,TAK NIE: ...	26 – tak; 21°– nie	4
(ADR) >= n	LD A,(ADR) CP n JR NC,TAK NIE: ...	26 – tak; 21 – nie	4
(ADR) <= n	LD A,(ADR) CP n JR C,TAK JR Z,TAK NIE: ...	26 lub 33 – tak; 28 – nie	6
(ADR)= n	LD A,(ADR) CP n JR Z,TAK NIE: ...	26 – tak; 21 – nie	4
(ADR) # n	LD A,(ADR) CP n JR NZ,TAK NIE: ...	26 – tak; 21 – nie	4

n – wielkość 8-bitowa

porównywanie zawartości rejestru A, B, C, D, E, H lub L (r) ze stałą n

Warunek	Program	t	p
r > n	LD A,n CP r JR NC,NIE TAK: ...	18 – tak; 23 – nie	5
r < n	LD A,r CP n JR C,TAK NIE: ...	23 – tak; 18 – nie	5
r >= n	LD A,r CP n JR NC,TAK NIE: ...	23 – tak; 18 – nie	4
r <= n	LD A,n CP r JR NC,TAK NIE: ...	23 – tak; 18 – nie	6
r = n	LD A,r CP n JR Z,TAK NIE: ...	23 – tak; 18 – nie	5
r # n	LD A,r CP n JR NZ,TAK NIE: ...	23 – tak; 18 – nie	5

7.2. Porównywanie 1-bajtowych liczb całkowitych ze znakiem

porównywanie zawartości 2 komórek pamięci operacyjnej o adresach ADR1 i ADR2

```
(ADR1) > (ADR2)  LD  A,(ADR1)       (ADR1) < (ADR2)  LD  A,(ADR1)
                 LD  HL,ADR2                         LD  HL,ADR2
                 CP  (HL)                            CP  (HL)
                 JR  Z,NIE                           JP  M,P1
                 JP  M,P1                            JP  PE,TAK
                 JP  PO,TAK                          JR  NIE
                 JR  NIE                         P1: JP  PO,TAK
             P1: JP  PE,TAK                     NIE: ...
            NIE: ...

t = {57 - tak              , p = 20  t = {50 - tak              , p = 18
    {42,57 lub 69 - nie                  {50 lub 62 - nie
```

```
(ADR1) >= (ADR2) LD  A,(ADR1)       (ADR1) <= (ADR2) LD  A,(ADR1)
                 LD  HL,ADR2                         LD  HL,ADR2
                 CP  (HL)                            CP  (HL)
                 JP  M,P1                            JR  Z,TAK
                 JP  PO,TAK                          JP  M,P1
                 JR  NIE                             JP  PE,TAK
             P1: JP  PE,TAK                          JR  NIE
            NIE: ...                             P1: JP  PO,TAK
                                                NIE: ...

t = {50 - tak              , p = 18  t = {42 lub 57 - tak       , p = 20
    {50 lub 62 - nie                     {57 lub 69 - nie
```

```
(ADR1) = (ADR2)  LD  A,(ADR1)       (ADR1) # (ADR2)  LD  A,(ADR1)
                 LD  HL,ADR2                         LD  HL,ADR2
                 CP  (HL)                            CP  (HL)
                 JR  Z,TAK                           JR  NZ,TAK
            NIE: ...                            NIE: ...

t = {42 - tak              , p = 9   t = {42 - tak              , p = 9
    {37 - nie                            {37 - nie
```

7.3. Porównywanie 2-bajtowych liczb bez znaku ze stałą nn

porównywanie stałej nn z liczbą zapisaną w komórkach pamięci operacyjnej o adresach ADR i ADR+1

```
(ADR) > nn       LD  HL,(ADR)       (ADR) < nn       LD  HL,(ADR)
                 LD  BC,nn                           LD  BC,nn
                 OR  A                               OR  A
                 SBC HL,BC                           SBC HL,BC
                 JR  C,NIE                           JR  C,TAK
                 JR  Z,NIE                      NIE: ...
            TAK: ...

t = {63 - tak              , p = 14  t = {61 - tak              , p = 12
    {61 lub 68 - nie                     {56 - nie
```

```
(ADR) >= nn      LD  HL,(ADR)       (ADR) <= nn      LD  HL,(ADR)
                 LD  BC,nn                           LD  BC,nn
                 OR  A                               OR  A
                 SBC HL,BC                           SBC HL,BC
                 JR  NC,TAK                          JR  C,TAK
            NIE: ...                                 JR  Z,TAK
                                                NIE:

t = {61 - tak              , p = 12  t = {61 lub 68 - tak       , p = 14
    {56 - nie                            {63 - nie
```

```
(ADR) = nn       LD  HL,(ADR)       (ADR) # nn       LD  HL,(ADR)
                 LD  BC,nn                           LD  BC,nn
                 OR  A                               OR  A
                 SBC HL,BC                           SBC HL,BC
                 JR  Z,TAK                           JR  NZ,TAK
            NIE: ...                            NIE: ...

t = {61 - tak              , p = 12  t = {61 - tak              , p = 12
    {56 - nie                            {56 - nie
```

Rozkazy OR A, użyte w powyższej tablicy i trzech następnych powodują wyzerowanie wskaźnika przeniesienia CY, ten sam efekt można uzyskać przez zastosowanie sekwencji rozkazów SCF i CCF.
nn – 16-bitowa liczba bez znaku

porównywanie stałej nn z zawartością pary rejestrów DE

```
DE > nn          LD  HL,nn          DE < nn          LD  L,E
                 OR  A                               LD  H,D
                 SBC HL,DE                           LD  BC,nn
                 JR  NC,NIE                          OR  A
            TAK: ...                                 SBC HL,BC
                                                     JR  C,TAK
                                                NIE: ...

t = {36 - tak              , p = 8   t = {49 - tak              , p = 10
    {41 - nie                            {44 - nie
```

```
DE >= nn         LD  L,E            DE <= nn         LD  HL,nn
                 LD  H,D                             OR  A
                 LD  BC,nn                           SBC HL,DE
                 OR  A                               JR  NC,TAK
                 SBC HL,BC                      NIE: ...
                 JR  NC,TAK
            NIE: ...

t = {49 - tak              , p = 10  t = {41 - tak              , p = 8
    {44 - nie                            {36 - nie
```

```
DE = nn          LD  HL,nn          DE # nn          LD  HL,nn
                 OR  A                               OR  A
                 SBC HL,DE                           SBC HL,DE
                 JR  Z,TAK                           JR  NZ,TAK
            NIE: ...                            NIE: ...

t = {41 - tak              , p = 8   t = {41 - tak              , p = 8
    {36 - nie                            {36 - nie
```

porównywanie stałej nn z zawartością rejestru SP

```
SP > nn        LD  HL,nn
               OR  A
               SBC HL,SP
               JR  NC,NIE
          TAK: ...

       ┌36 - tak
   t = ┤         , p = 8
       └41 - nie
```

```
SP < nn        LD  HL,0
               ADD HL,SP
               LD  BC,nn
               OR  A
               SBC HL,BC
               JR  C,TAK
          NIE: ...

       ┌62 - tak
   t = ┤         , p = 12
       └57 - nie
```

```
SP >= nn       LD  HL,0
               ADD HL,SP
               LD  BC,nn
               OR  A
               SBC HL,BC
               JR  NC,TAK
          NIE: ...

       ┌62 - tak
   t = ┤         , p = 12
       └57 - nie
```

```
SP <= nn       LD  HL,nn
               OR  A
               SBC HL,SP
               JR  NC,TAK
          NIE: ...

       ┌41 - tak
   t = ┤         , p = 8
       └36 - nie
```

```
SP = nn        LD  HL,nn
               OR  A
               SBC HL,SP
               JR  Z,TAK
          NIE: ....

       ┌41 - tak
   t = ┤         , p = 8
       └36 - nie
```

```
SP # nn        LD  HL,nn
               OR  A
               SBC HL,SP
               JR  NZ,TAK
          NIE: ...

       ┌41 - tak
   t = ┤         , p = 8
       └36 - nie
```

nn - 16-bitowa liczba bez znaku

porównywanie stałej nn z zawartością pary rejestrów HL

```
HL > nn        LD  DE,nn
               EX  DE,HL
               OR  A
               SBC HL,DE
               EX  DE,HL
               JR  NC,NIE
          TAK: ...

       ┌44 - tak
   t = ┤         , p = 10
       └49 - nie
```

```
HL < nn        PUSH HL
               LD  DE,nn
               OR  A
               SBC HL,DE
               POP HL
               JR  C,TAK
          NIE: ...

       ┌62 - tak
   t = ┤         , p = 10
       └57 - nie
```

```
HL >= nn       PUSH HL
               LD  DE,nn
               OR  A
               SBC HL,DE
               POP HL
               JR  NC,TAK
          NIE: ...

       ┌62 - tak
   t = ┤         , p = 10
       └57 - nie
```

```
HL <= nn       LD  DE,nn
               EX  DE,HL
               OR  A
               SBC HL,DE
               EX  DE,HL
               JR  NC,TAK
          NIE:

       ┌49 - tak
   t = ┤         , p = 10
       └44 - nie
```

```
HL = nn        LD  DE,nn
               EX  DE,HL
               OR  A
               SBC HL,DE
               EX  DE,HL
               JR  Z,TAK
          NIE: ...

       ┌49 - tak
   t = ┤         , p = 10
       └44 - nie
```

```
HL # nn        LD  DE,nn
               EX  DE,HL
               OR  A
               SBC HL,DE
               EX  DE,HL
               JR  NZ,TAK
          NIE: ...

       ┌49 - tak
   t = ┤         , p = 10
       └44 - nie
```

nn - 16-bitowa liczba bez znaku

3.4. Porównywanie -bajtowych tekstów znakowych

porównywanie tekstów znakowych, zapisanych w komórkach począwszy od adresów ADR1 i ADR2

```
ADR1 > ADR2        LD  DE,ADR1
                   LD  HL,ADR2
                   LD  B,n
              P1:  LD  A,(DE)
                   CP  (HL)
                   JR  NZ,P2
                   INC DE
                   INC HL
                   DJNZ P1
                   JR  NIE
              P2:  JR  C,NIE
              TAK: ...

  p = 19
```

```
ADR1 < ADR2        LD  DE,ADR1
                   LD  HL,ADR2
                   LD  B,n
              P1:  LD  A,(DE)
                   CP  (HL)
                   JR  NZ,P2
                   INC DE
                   INC HL
                   DJNZ P1
                   JR  NIE
              P2:  JR  C,TAK
              NIE: ...

  p = 19
```

```
ADR1 >= ADR2       LD  DE,ADR1
                   LD  HL,ADR2
                   LD  B,n
              P1:  LD  A,(DE)
                   CP  (HL)
                   JR  NZ,P2
                   INC DE
                   INC HL
                   DJNZ P1
              TAK: ...
              P2:  JR  NC,TAK
              NIE: ...

  p = 17
```

```
ADR1 <= ADR2       LD  DE,ADR1
                   LD  HL,ADR2
                   LD  B,n
              P1:  LD  A,(DE)
                   CP  (HL)
                   JR  NZ,P2
                   INC DE
                   INC HL
                   DJNZ P1
              TAK: ...
              P2:  JR  C,TAK
              NIE: ...

  p = 17
```

```
ADR1 = ADR2        LD  DE,ADR1
                   LD  HL,ADR2
                   LD  B,n
              P1:  LD  A,(DE)
                   CP  (HL)
                   JR  NZ,NIE
                   INC DE
                   INC HL
                   DJNZ P1
              TAK: ...

  p = 15
```

```
ADR1 # ADR2        LD  DE,ADR1
                   LD  HL,ADR2
                   LD  B,n
              P1:  LD  A,(DE)
                   CP  (HL)
                   JR  NZ,TAK
                   INC DE
                   INC HL
                   DJNZ P1
              NIE: ...

  p = 15
```

(0 <= n <= 255)
dla n=0 porównywane są teksty o długości 256 znaków

czasy przebiegu operacji porównywania n-bajtowych tekstów znakowych dla przypadków spełnienia i nie spełnienia warunków

warunek	tak	nie
ADR1 > ADR2	t = 46*k + 14	t = 46*k + 19 gdy ADR1 < ADR2 t = 46*n + 34 gdy ADR1 = ADR2
ADR1 < ADR2	t = 46*k + 19	t = 46*k + 14 gdy ADR1 > ADR2 t = 46*n + 34 gdy ADR1 = ADR2
ADR1 >= ADR2	t = 46*k + 19 gdy ADR1 > ADR2 t = 46*n + 22 gdy ADR1 = ADR2	t = 46*k + 19
ADR1 <= ADR2	t = 46*k + 19 gdy ADR1 < ADR2 t = 46*n + 22 gdy ADR1 = ADR2	t = 46*k + 19
ADR1 = ADR2	t = 46*n + 22	t = 46*k + 7
ADR1 # ADR2	t = 46*k + 7	t = 46*n + 22

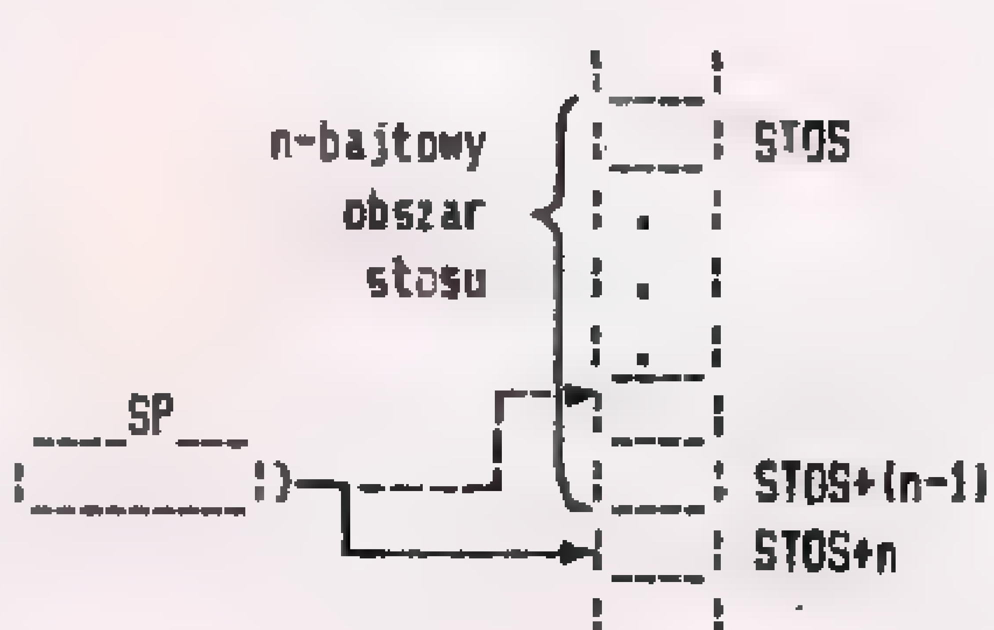
k - liczba porównań znaków (1 <= k <= n)

4/ PODPROGRAMY, OPERACJE NA STOSIE

Operacje wykonywane na elementach stosu muszą być poprzedzone ustaleniem wartości początkowej rejestru wskaźnika stosu - SP. Dla n-bajtwego obszaru stosu, rozpoczynającego się od symbolicznego adresu STOS, realizowane jest to rozkazem:

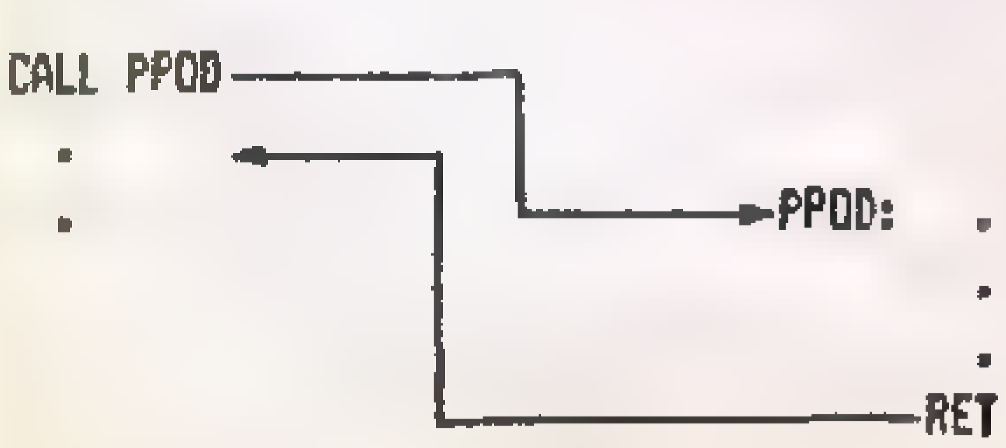
LD SP,STOS+n

t = 10, p = 3



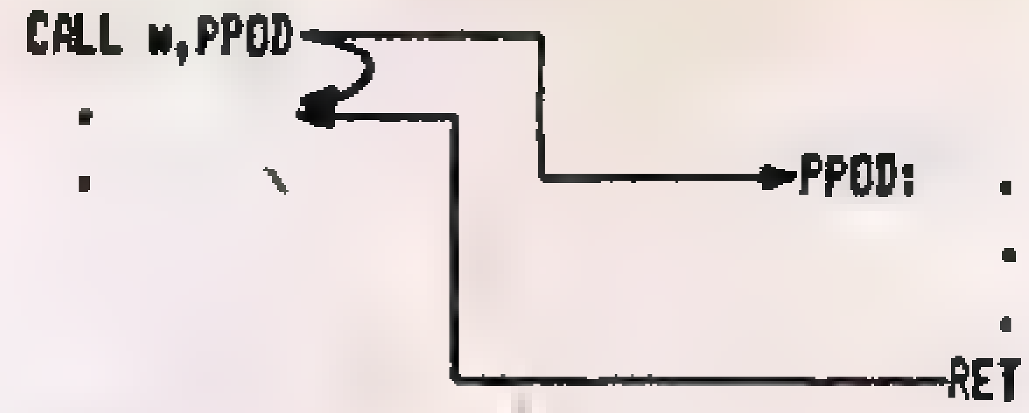
4.1. Wywołanie podprogramów

wywołanie i powrót z podprogramu



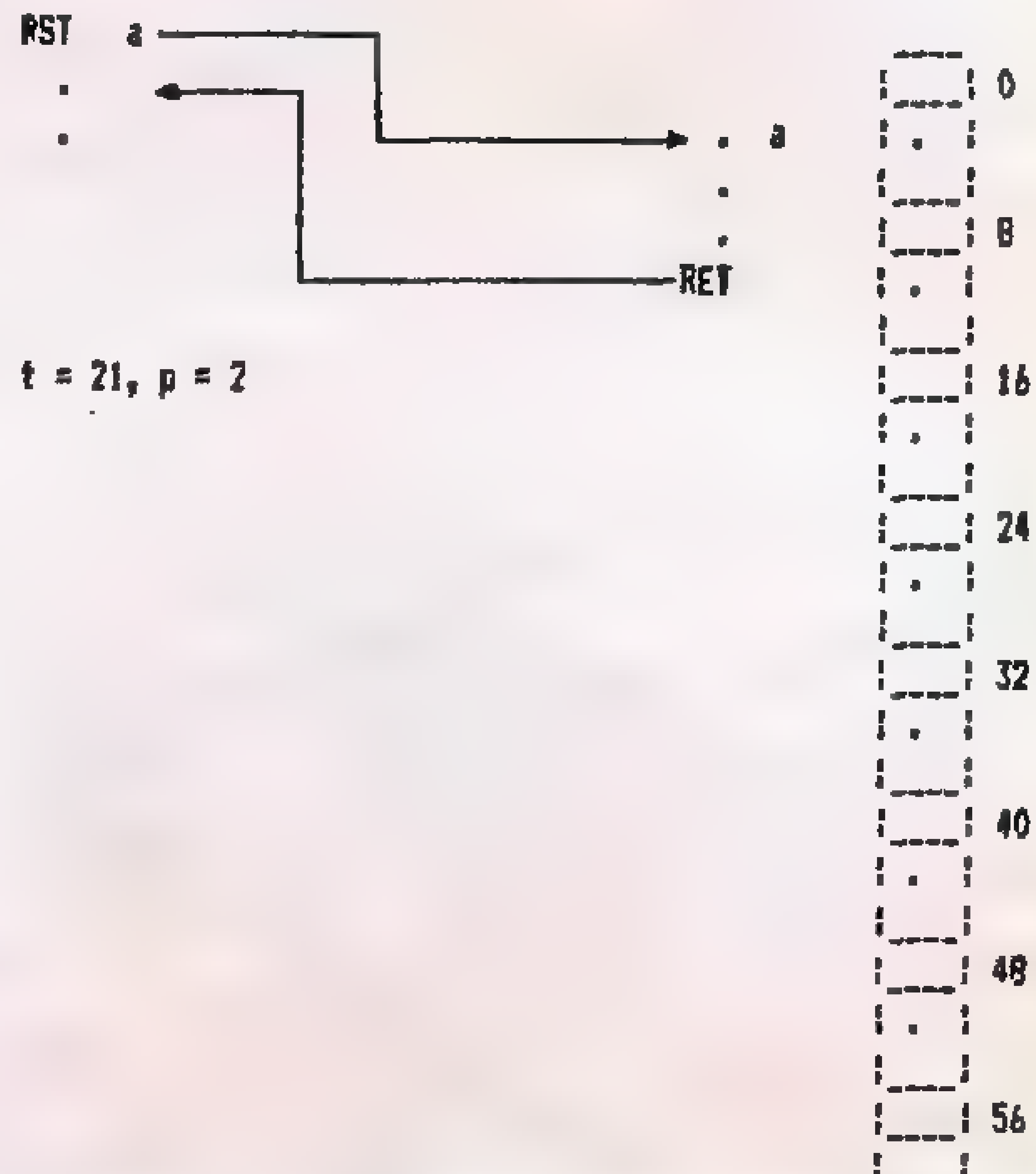
t = 27, p = 4

wywołanie warunkowe i powrót z podprogramu (w = (C, M, NC, NZ, P, PE, PO, Z))



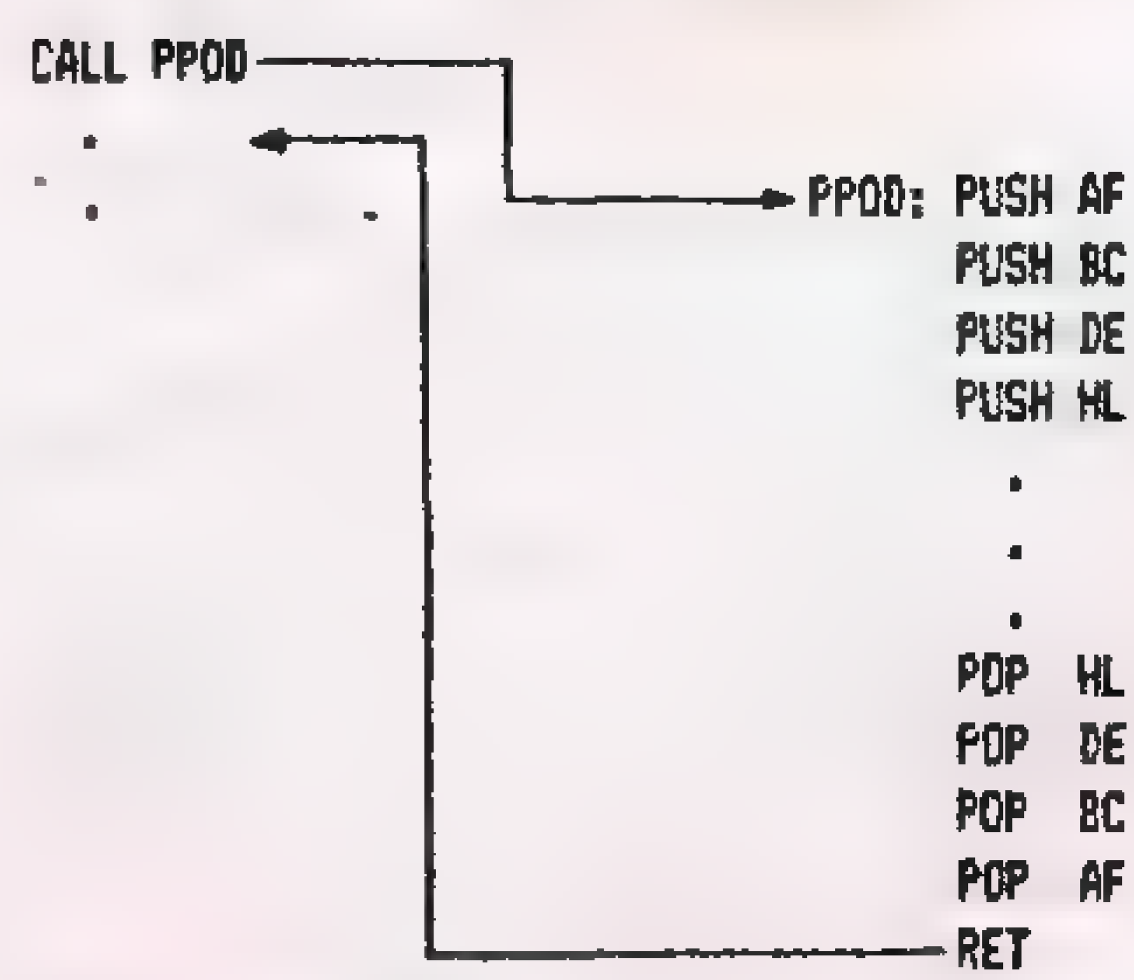
t = 27, p = 4

wywołanie podprogramu przy użyciu rozkazu <RST a>, umieszczonego w pamięci o adresie - a (a = (0, 8, 16, 24, 32, 40, 48, 56))



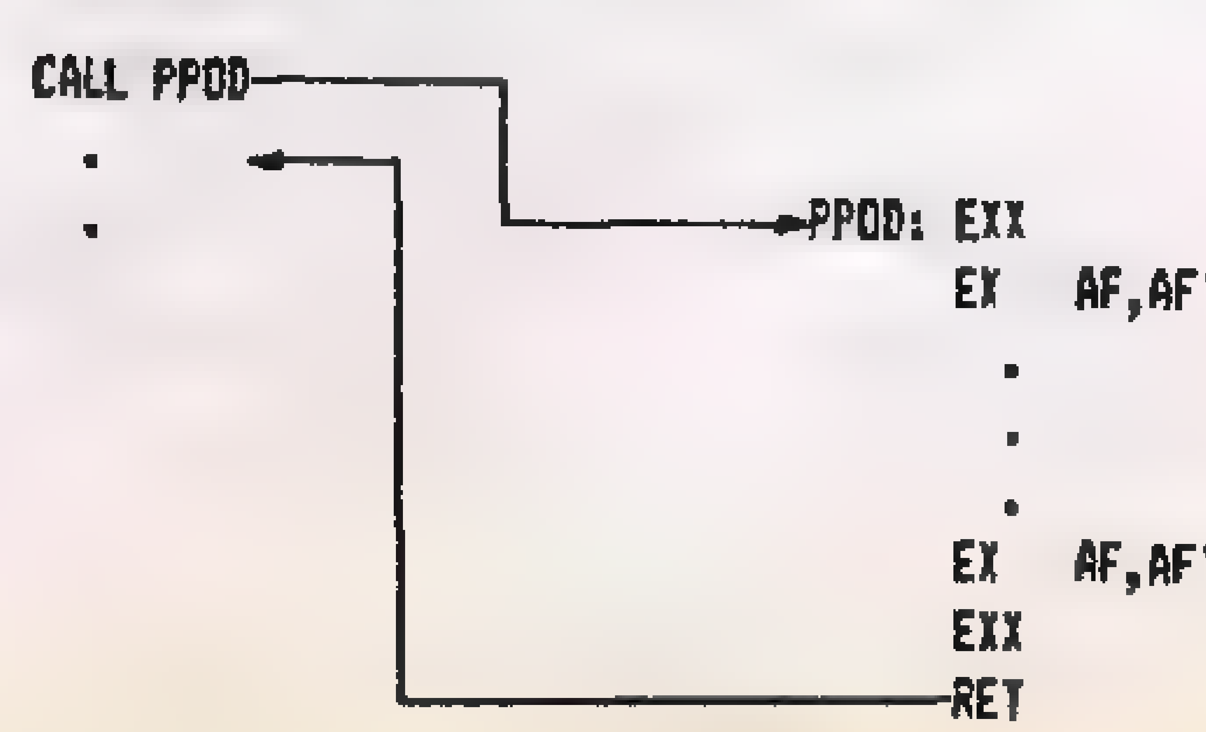
t = 21, p = 2

wywołanie i powrót z podprogramu z zapamiętaniem zawartości rejestrów roboczych na stosie



t = 111, p = 12

wywołanie i powrót z podprogramu z zapamiętaniem zawartości rejestrów roboczych w rejestrach pomocniczych



t = 43, p = 8

4.2. Przekazywanie parametrów do podprogramów

Przedstawione poniżej metody zilustrowano dla przypadku przekazywania parametru jednobajtowego (PAR8) i dwubajtowego (PAR16). Pierwszy przesyłany jest do rejestru A, drugi do pary rejestrów DE.

```
!___!
!___! PAR8
!___! PAR16
!___!
!   !
```

przekazywanie parametrów poprzez wspólny obszar pamięci

```
CALL PPOD
  .
  .            PPOD:  .
                      .
                 LD   A,(PAR8)
                 LD   DE,(PAR16)
                      .
                      .
                 RET
t = 60, p = 11
```

przekazywanie parametrów poprzez podanie adresu ich początku w rejestrach HL

```
LD   HL,PAR8
CALL PPOD
  .
  .            PPOD:  .
                      .
                 LD   A,(HL)
                 INC  HL
                 LD   E,(HL)
                 INC  HL
                 LD   D,(HL)
                      .
                      .
t = 70, p = 12   RET
```

przekazywanie parametrów poprzez rejestry

```
LD   A,(PAR8)
LD   DE,(PAR16)
CALL PPOD
  .
  .            PPOD:  .
                      .
                      .
                 RET
t = 60, p = 11
```

przekazywanie parametrów za rozkazem CALL przy użyciu pary rejestrów HL

```
CALL PPOD
DEFB 10        PPOD:  .
DEFW 2345             .
  .              POP  HL
  .              LD   A,(HL)
                 INC  HL
                 LD   E,(HL)
                 INC  HL
                 LD   D,(HL)
                 INC  HL
                 PUSH HL
                      .
                      .
t = 104, p = 12  RET
```

przekazywanie parametrów za rozkazem CALL przy użyciu rejestru indeksowego IX lub IY (xy)

```
CALL PPOD
DEFB 10
DEFW 2345   PPOD:  .            PPOD:  .
  .                .                   .
  .         POP  xy             EX   (SP),xy
            LD   A,(xy)         LD   A,(xy)
            LD   E,(xy+1)       INC  xy
            LD   D,(xy+2)       LD   E,(xy)
            LD   BC,3           INC  xy
            ADD  xy,BC          LD   D,(xy)
            PUSH xy             INC  xy
                 .              EX   (SP),xy
                 .                   .
                 .                   .
            RET                 RET
        t = 138, p = 22     t = 160, p = 23
```

przekazywanie parametrów poprzez stos

```
LD   BC,(PAR8-1)
PUSH BC
LD   BC,(PAR16)
PUSH BC
CALL PPOD
  .
  .            PPOD:  .
                      .
                      .
                 POP  BC
                 POP  DE
                 POP  AF
                 PUSH BC
                      .
                      .
t = 120, p = 17  RET
```

5. KONWERSJE PÓL DANYCH

5.1. Liczby binarne i znakowe w kodzie BCD

konwersja 1-bajtowej liczby binarnej bez znaku (ADR1) na 2-bajtową liczbę w kodzie BCD (ADR2)

```
!___!
!___! ADR1
! . !      dziesiątki
! . !      jedności
! . !
!___! ADR2
!___!
!   !      setki
```

```
      LD   A,(ADR1)
      LD   C,-1
P:    INC  C            ;obliczanie
      SUB  100          ;setek
      JR   NC,P
      ADD  A,100
;
      LD   H,A
      LD   A,C
      LD   (ADR2+1),A
      LD   A,H
;
      LD   C,-1
P1:   INC  C            ;obliczanie
      SUB  10           ;dziesiątek
      JR   NC,P1
      ADD  A,10
```

```
        LD   H,A
        LD   A,C
        RLCA             ;przesuwanie dziesiątek
        RLCA             ;na 4 bardziej znaczące
        RLCA             ;pozycje
        RLCA
        OR   H           ;dołączanie jedności
        LD   (ADR2),A
```

t = 23*(s + d) + 143, p = 37
s - liczba setek (0 <= s <= 2)
d - liczba dziesiątek (0 <= d <= 9)

konwersja 1-bajtowej liczby bez znaku w kodzie BCD (ADR1) na 1-bajtową liczbę binarną (ADR2)

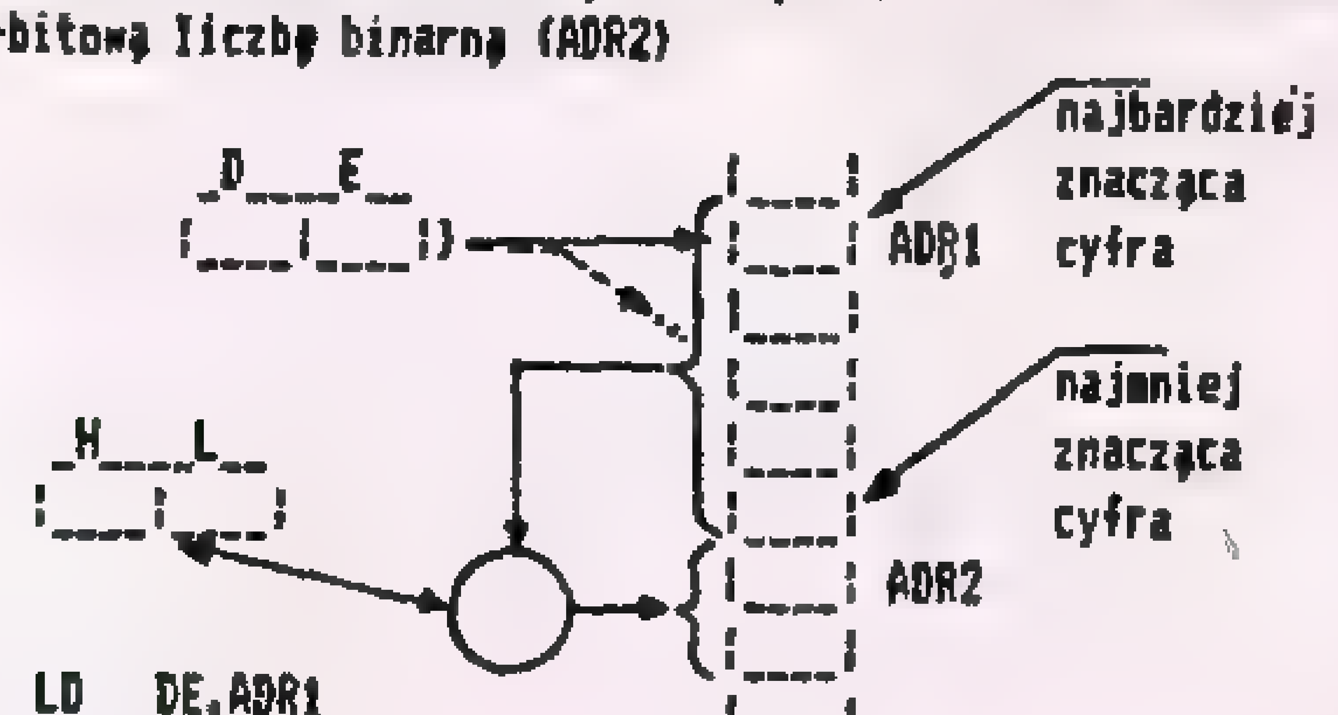



```
        LD   A,(ADR1)
        LD   B,A
        AND  11110000B   ;wycięcie dziesiątek
        RRCA             ;dziesiątki*8
        LD   C,A
        RRCA
        RRCA             ;dziesiątki*2
        ADD  A,C         ;dziesiątki*(8+2)
        LD   C,A
        LD   A,B
        AND  00001111B   ;wycięcie jedności
        ADD  A,C
        LD   (ADR2),A
```

t = 76, p = 19

5.2. Liczby binarne i znakowe w kodzie ASCII

konwersja 5-cio znakowej liczby dziesiętnej w kodzie ASCII (ADR1) na 16-bitową liczbę binarną (ADR2)




```
        LD   DE,ADR1
        LD   HL,0
        LD   B,5
P:      LD   A,(DE)      ;dodanie kolejnej
        SUB  '0'         ;cyfry
        PUSH BC
        LD   C,A
        LD   B,0
        ADD  HL,BC
        INC  DE
        POP  BC
        DEC  B
        JR   Z,P1
        PUSH DE
;                        mnożenie przez 10
        ADD  HL,HL       ; HL*2
        LD   E,L
        LD   D,H
        ADD  HL,HL       ; HL*4
        ADD  HL,HL       ; HL*8
        ADD  HL,DE       ; HL*(8+2)
        POP  DE
        JR   P
P1:     LD   (ADR2),HL
```

t = 755, p = 34

6. OPERACJE NA POLACH BINARNYCH

Operacje na polach binarnych wykonywane są przy użyciu specjalnie do tego celu przeznaczonych rozkazów (SET, RES, BIT), rozkazów logicznych (OR, AND, XOR, CPL), oraz licznej grupy rozkazów przesuwania.

```
 _______________
|_|_|_|_|_|_|_|_|
 7 6 5 4 3 2 1 0
```

Oznaczenie pozycji binarnych bajtu

6.1. Wprowadzanie określonych wartości na pozycje binarne komórki

wprowadzanie "1" i "0" na b-tą pozycję binarną (b=(0,1,...,7))

wprowadzenie "1"	wprowadzenie "0"
LD HL,ADR	LD HL,ADR
SET b,(HL)	RES b,(HL)
t = 25, p = 5	t = 25, p = 5

wprowadzanie "1" na określone pozycje binarne z pozostawieniem bez zmiany pozostałych pozycji (przykładowo wprowadzono "1" na pozycje nr 2, 1 i 0)

```
                         zawartość komórki
        LD   A,(ADR)       przed wykonaniem operacji:    01111001
        OR   00000111B
        LD   (ADR),A       po wykonaniu operacji:        01111111
```

t = 33, p = 8

```
        LD   HL,ADR
        SET  2,(HL)
        SET  1,(HL)
        SET  0,(HL)
```

t = 55, p = 9

wprowadzanie "0" na określone pozycje binarne z pozostawieniem bez zmiany pozostałych pozycji (przykładowo wprowadzono "0" na pozycje nr 2, 1 i 0)

```
                         zawartość komórki
        LD   A,(ADR)       przed wykonaniem operacji:    11101011
        AND  11111000B
        LD   (ADR)         po wykonaniu operacji:        11101000
```

t = 33, p = 8

```
        LD   HL,ADR
        RES  2,(HL)
        RES  1,(HL)
        RES  0,(HL)
```

t = 55, p = 9

zamiana na wartość przeciwną (z "1" na "0" a z "0" na "1") określonych pozycji binarnych z pozostawieniem bez zmiany pozostałych pozycji (przykładowo zamieniono wartości pozycji nr 2, 1 i 0)

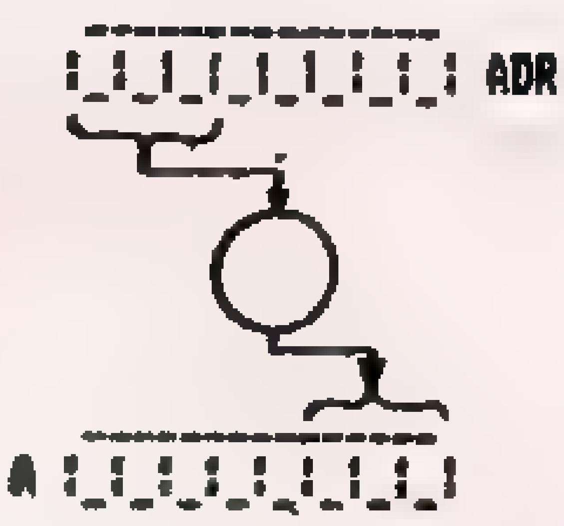
```
                        zawartość komórki
LD   A,(ADR)            przed wykonaniem operacji:     11101011
XOR  00000111B
LD   (ADR),A            po wykonaniu  operacji:        11101100

t = 33, p = 8
```

6.2. Przesłania określonych pozycji binarnych

przesłanie określonych pozycji komórki do rejestru A z prawostronnym przesunięciem (przykładowo przesłano pozycje nr 7, 6 i 5)

```
LD   A,(ADR)
AND  11100000B
SRL  A
SRL  A
SRL  A
SRL  A
SRL  A
```



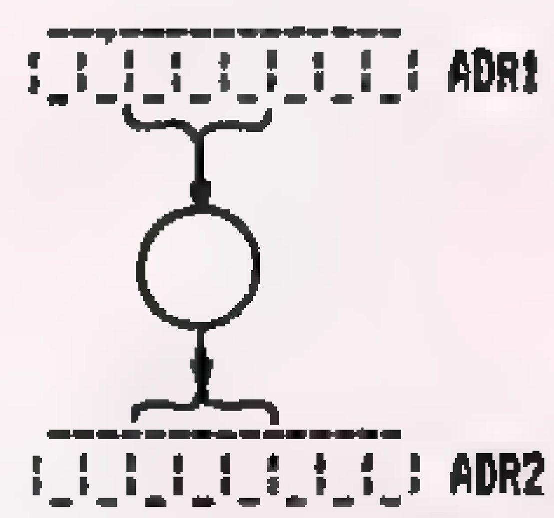

```
t = 60, p =13

LD   A,(ADR)
AND  11100000B
RLCA
RLCA
RLCA

t = 32, p = 8
```

przesłanie określonych pozycji binarnych jednej komórki (ADR1) na te same pozycje drugiej komórki (ADR2) z pozostawieniem bez zmiany pozostałych pozycji (przykładowo przesłano pozycje nr 5, 4 i 3)

```
LD   A,(ADR1)
AND  00111000B
LD   B,A
LD   A,(ADR2)
OR   B
LD   (ADR2),A
```




```
t = 54, p = 13
```

6.3. Badanie wartości określonych pozycji binarnych

badanie wystąpienia "1" i "0" na określonej pozycji binarnej komórki

```
------------------------------------------------
!     (b) = "1"?      !      (b) = "0"?        !
!---------------------!------------------------!
!    LD   A,(ADR)     !    LD   A,(ADR)        !
!    BIT  b,A         !    BIT  b,A            !
!    JR   NZ,TAK      !    JR   Z,TAK          !
! NIE: ...            ! NIE: ...               !
!---------------------!------------------------!
!              (33 - tak                       !
!          t = {         , p = 7               !
!              (28 - nie                       !
!----------------------------------------------!
!    LD   HL,ADR      !    LD   HL,ADR         !
!    BIT  b,(HL)      !    BIT  b,(HL)         !
!    JR   NZ,TAK      !    JR   Z,TAK          !
! NIE: ...            ! NIE: ...               !
!---------------------!------------------------!
!              (34 - tak                       !
!          t = {         , p = 7               !
!              (29 - nie                       !
```

```
!----------------------------------------------!
!   przykładowo badana jest pozycja nr 0       !
!                                              !
!    LD   A,(ADR)     !    LD   A,(ADR)        !
!    AND  00000001B   !    AND  00000001B      !
!    JR   NZ,TAK      !    JR   Z,TAK          !
! NIE: ...            ! NIE: ...               !
!---------------------!------------------------!
!              (32 - tak                       !
!          t = {         , p = 7               !
!              (27 - nie                       !
!----------------------------------------------!
b={0,1,...,7}
```

badanie wystąpienia "1" na określonych pozycjach binarnych komórki (przykładowo badane są pozycje nr 2, 1 i 0)

```
     LD   A,(ADR)
     AND  00000111B
     XOR  00000111B
     JR   Z,TAK
NIE:  ...

    (39 - tak
t = {         , p = 9
    (34 - nie

     LD   A,(ADR)
     BIT  2,A
     JR   Z,NIE
     BIT  1,A
     JR   Z,NIE
     BIT  0,A
     JR   NZ,TAK
NIE:  ...

    (63 - tak
t = {                 , p = 15
    (33, 48, 58 - nie
```

badanie wystąpienia "0" na określonych pozycjach binarnych komórki (przykładowo badane są pozycje nr 2, 1 i 0)

```
     LD   A,(ADR)
     AND  00000111B
     JR   Z,TAK
NIE:  ...

    (32 - tak
t = {         , p = 7
    (27 - nie

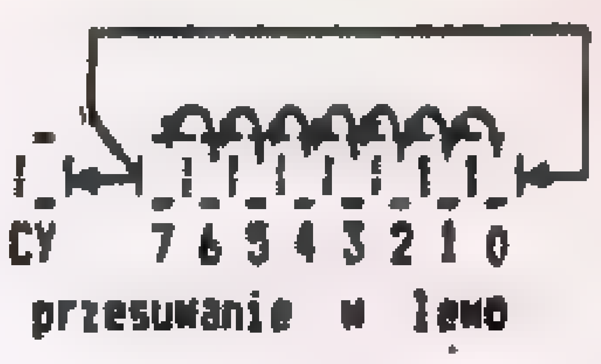
     LD   A,(ADR)
     BIT  2,A
     JR   NZ,NIE
     BIT  1,A
     JR   NZ,NIE
     BIT  0,A
     JR   Z,TAK
NIE:  ...

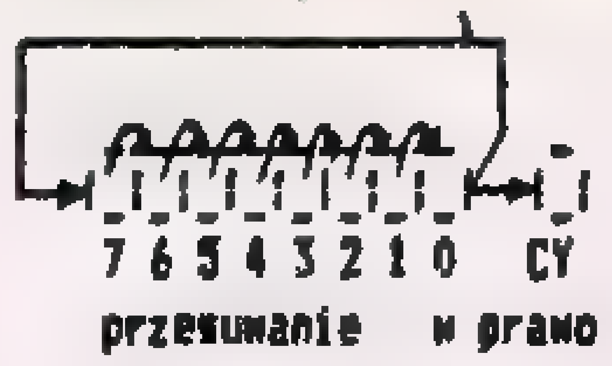
    (63 - tak
t = {                 , p = 15
    (33, 48, 58 - nie
```

6.4. Przesuwanie pozycji binarnych

Oznaczenia:
CY - wskaźnik przeniesienia
bzb - bardziej znaczący bajt
mzb - mniej znaczący bajt

6.4.1. Przesuwanie cykliczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej





przesuwanie zawartości akumulatora

```
RLCA                          RRCA
            t = 4, p = 1
```

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
RLC  r                        RRC  r
            t = 8, p = 2
```

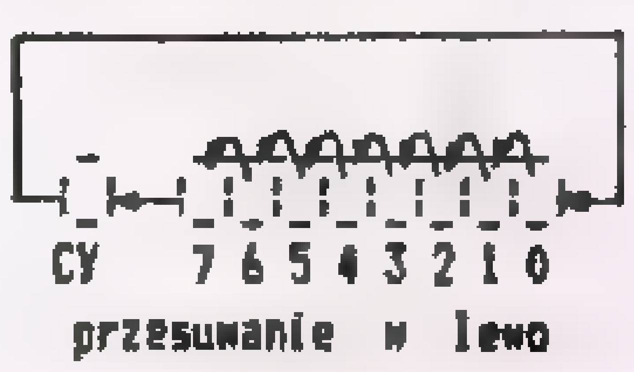
przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

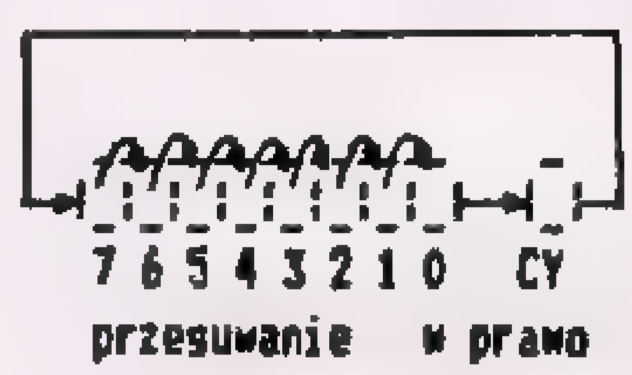
```
LD   HL,ADR                   LD   HL,ADR
RLC  (HL)                     RRC  (HL)
            t = 25, p = 5
```

przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                   LD   xy,ADR
RLC  (xy)                     RRC  (xy)
            t = 37, p = 8
```

6.4.2. Przesuwanie cykliczne w lewo i w prawo z przeniesieniem (o jedną pozycję) wielkości 8-bitowej





przesuwanie zawartości akumulatora

```
RLA                           RRA
            t = 4, p = 1
```

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
RL   r                        RR   r
            t = 8, p = 2
```

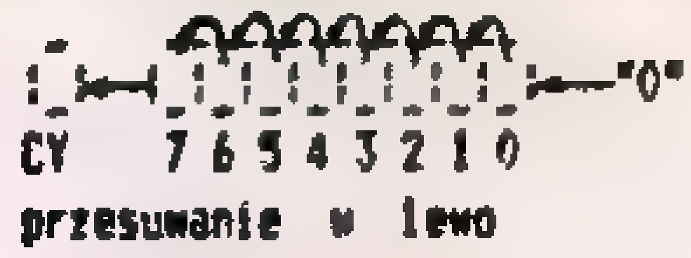
przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

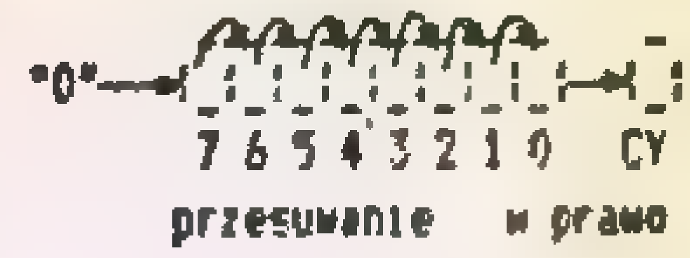
```
LD   HL,ADR                   LD   HL,ADR
RL   (HL)                     RR   (HL)
            t = 25, p = 5
```

przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                   LD   xy,ADR
RL   (xy)                     RR   (xy)
            t = 37, p = 8
```

6.4.3. Przesuwanie logiczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej





przesuwanie zawartości akumulatora

```
ADD  A
t = 4, p = 1
```

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
SLA  r                        SRL  r
            t = 8, p = 2
```

przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

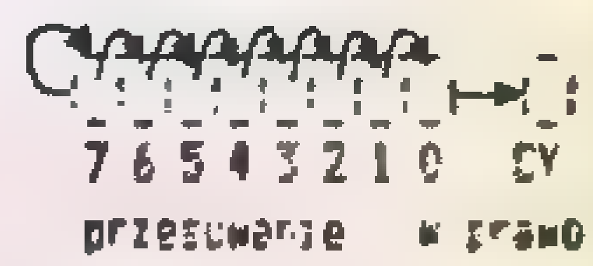
```
LD   HL,ADR                   LD   HL,ADR
SLA  (HL)                     SRL  (HL)
            t = 25, p = 5
```

przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                   LD   xy,ADR
SLA  (xy)                     SRL  (xy)
            t = 37, p = 9
```

6.4.4. Przesuwanie arytmetyczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej





przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
SLA  r                        SRA  r
            t = 8, p = 2
```

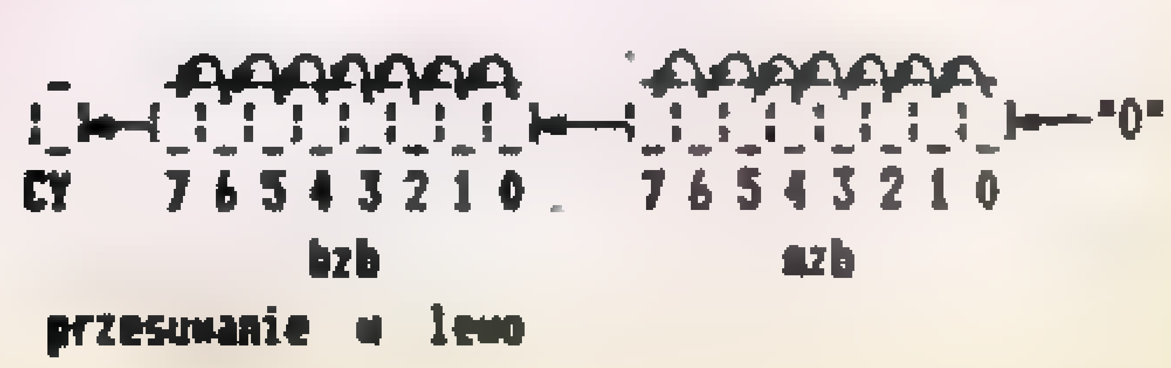
przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

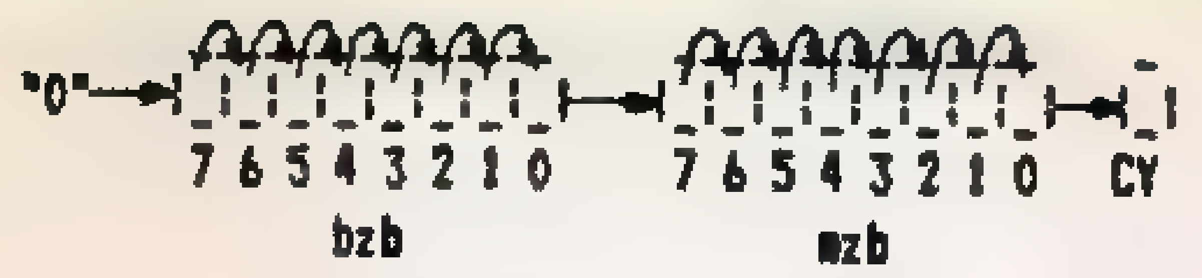
```
LD   HL,ADR                   LD   HL,ADR
SLA  (HL)                     SRA  (HL)
            t = 25, p = 5
```

przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                   LD   xy,ADR
SLA  (xy)                     SRA  (xy)
            t = 37, p = 8
```

6.4.5. Przesuwanie logiczne w lewo i w prawo (o jedną pozycję) wielkości 16-bitowej





przesuwanie w prawo

przesuwanie zawartości rejestrów HL

```
ADD  HL,HL                          SRL  H
                                    RR   L
t = 11, p = 1
                                    t = 16, 4
```

przesuwanie zawartości rejestru IX lub IY (xy)

```
ADD xy,xy

t = 15 p = 2
```

przesuwanie zawartości pary rejestrów BC, DE, HL (rb = (B, D, H), rm = (C, E, L))

```
SLA  rm                             SRL  rb
RL   rb                             RR   rm
            t = 16, p = 4
```

przesuwanie zawartości 2 komórek (ADR i ADR+1) przy użyciu pary rejestrów HL i rejestru IX lub IY (xy)

```
LD   HL,ADR                         LD   ADR+1
SLA  (HL)                           SRL  (HL)
INC  HL                             DEC  HL
RL   (HL)                           RR   (HL)
            t = 46, p = 8

LD   xy,ADR                         LD   xy,ADR
SLA  (xy)                           SRL  (xy+1)
RL   (xy+1)                         RR   (xy)
            t = 60, p = 12
```

7. OPERACJE ARYTMETYCZNE

Ze względu na brak w mikroprocesorze Z80 rozkazów mnożenia i dzielenia, operacje te wykonywane są przy użyciu rozkazów dodawania, odejmowania i przesuwania.

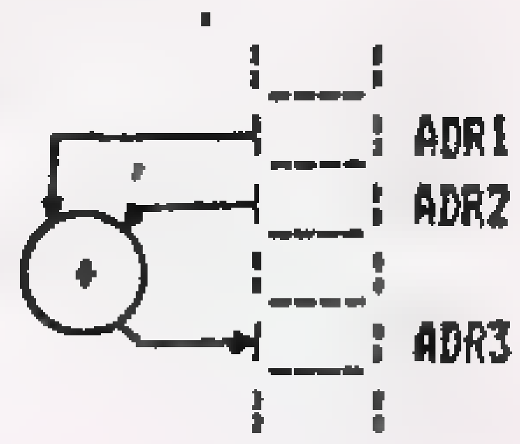
7.1. Dodawanie

7.1.1. Dodawanie liczb binarnych

dodawanie liczb 1-bajtowych

```
LD   A,(ADR1)   ;ADR3:=ADR1+ADR2
LD   HL,ADR2
ADD  A,(HL)
LD   (ADR3),A

t = 43, p = 10
```



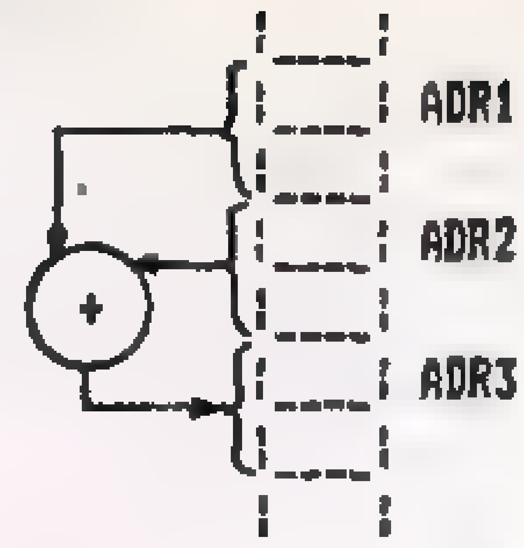

```
LD   A,(ADR1)   ;wersja bez użycia
LD   B,A        ;rejestrów HL
LD   A,(ADR2)
ADD  A,B
LD   (ADR3),A

t = 47, P = 11
```

dodawanie liczb 2-bajtowych

```
LD   A,(ADR1)   ;ADR3:=ADR1+ADR2
LD   HL,ADR2
ADD  A,(HL)
LD   (ADR3),A
LD   A,(ADR1+1)
DEC  HL
ADC  A,(HL)
LD   (ADR3+1),A

t = 82, p = 18
```



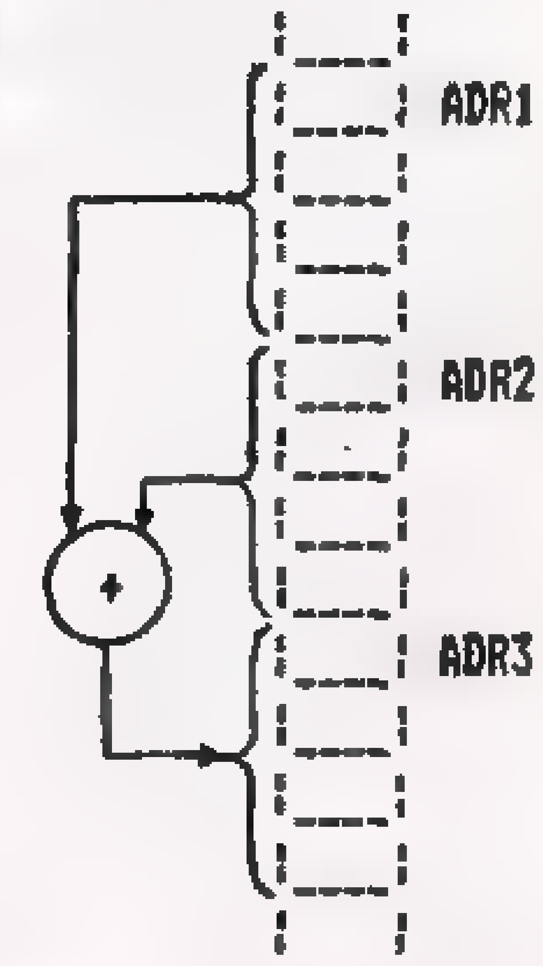

```
LD   HL,(ADR1)  ;ADR3:=ADR1+ADR2
LD   BC,(ADR2)
ADD  HL,BC
LD   (ADR3),HL

t = 63, p = 11
```

dodawanie liczb 4-bajtowych

```
LD   HL,(ADR1)  ;ADR3:=ADR1+ADR2
LD   BC,(ADR2)
ADD  HL,BC
LD   (ADR3),HL
LD   HL,(ADR1+2)
LD   BC,(ADR2+2)
ADC  HL,BC
LD   (ADR3+2),HL

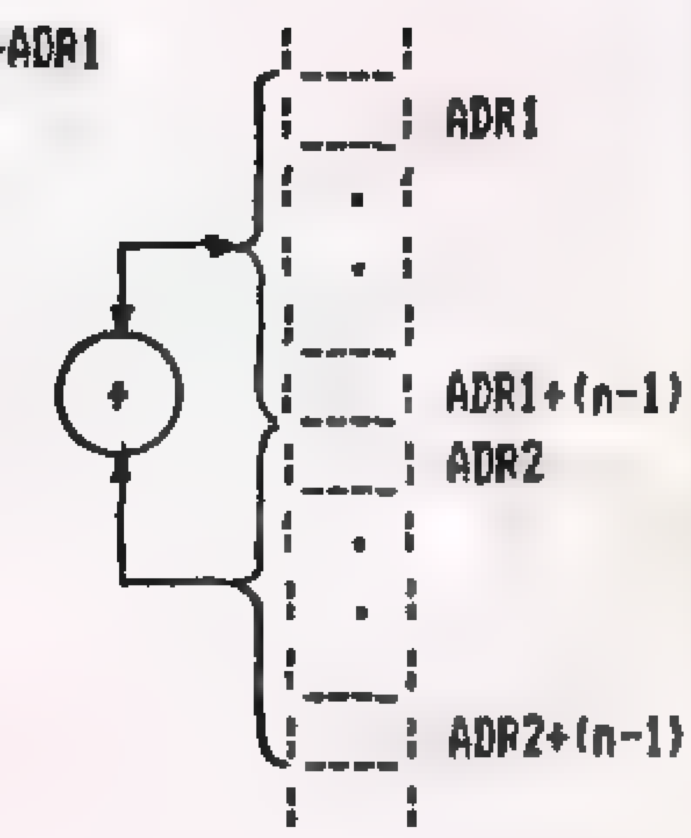
t = 130, p = 23
```



dodawanie liczb n-bajtowych

```
     LD   HL,ADR1   ;ADR1:=ADR2+ADR1
     LD   DE,ADR2
     LD   B,n
     AND  A
P:   LD   A,(DE)
     ADC  A,(HL)
     LD   (HL),A
     INC  HL
     INC  DE
     DJNZ P

t = 46*n + 29, p = 17
(1 <= n <= 255)
```

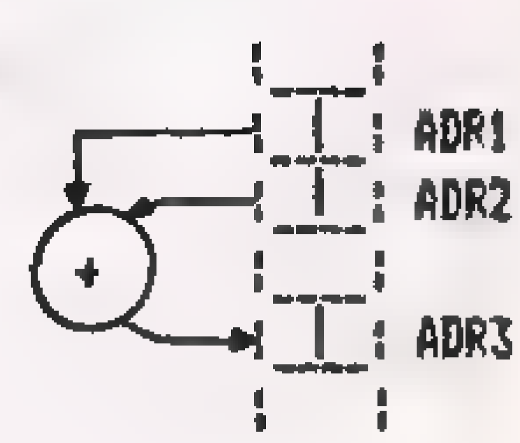


7.1.2. Dodawanie liczb w kodzie BCD

dodawanie liczb 1-bajtowych bez znaku

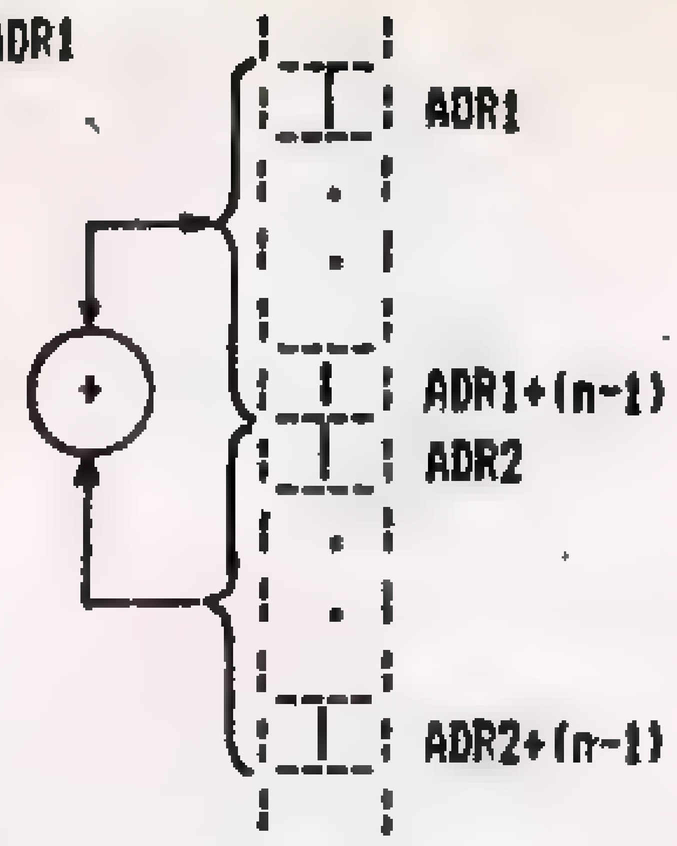
```
LD   A,(ADR1)   ;ADR3:=ADR1+ADR2
LD   HL,ADR2
ADD  A,(HL)
DAA
LD   (ADR3),A

t = 47,  p = 11
```



dodawanie liczb n-bajtowych bez znaku

```
     LD   HL,ADR1   ;ADR1:=ADR2+ADR1
     LD   DE,ADR2
     LD   B,n
     AND  A
P:   LD   A,(DE)
     ADC  A,(HL)
     DAA
     LD   (HL),A
     INC  HL
     INC  DE
     DJNZ P
```



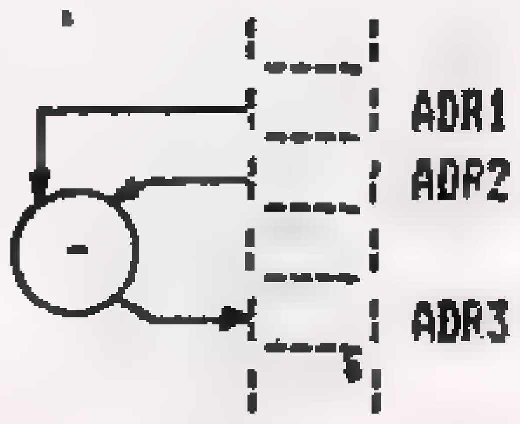
t = 56*n + 29, p = 18
(1 <= n <= 255)

7.2. Odejmowanie

7.2.1. Odejmowanie liczb binarnych

odejmowanie liczb 1-bajtowych

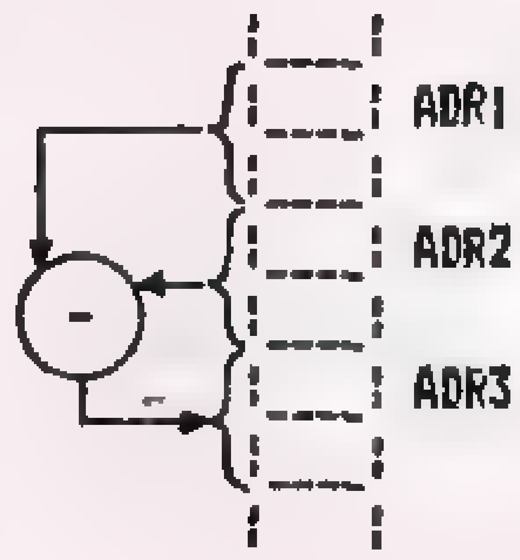
```
LD   A,(ADR1)  ;ADR3:=ADR1-ADR2
LD   HL,ADR2
SUB  A,(HL)
LD   (ADR3),A
```



t = 43, p = 10

odejmowanie liczb 2-bajtowych

```
LD   HL,(ADR1) ;ADR3:=ADR1-ADR2
LD   DE,ADR2
AND  A
SBC  HL,DE
LD   (ADR3),HL
```

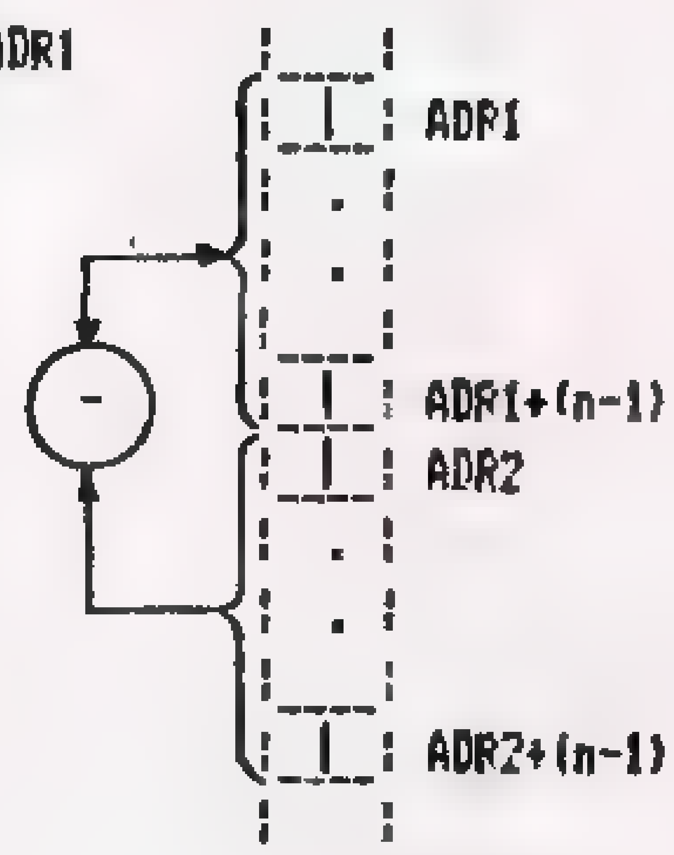


t = 82, p = 18

7.2.2. Odejmowanie liczb w kodzie BCD

odejmowanie liczb n-bajtowych

```
     LD   HL,ADR1   ;ADR1:=ADR2-ADR1
     LD   DE,ADR2
     LD   B,n
     AND  A
P:   LD   A,(DE)
     SBC  A,(HL)
     DAA
     LD   (HL),A
     INC  HL
     INC  DE
     DJNZ P
```

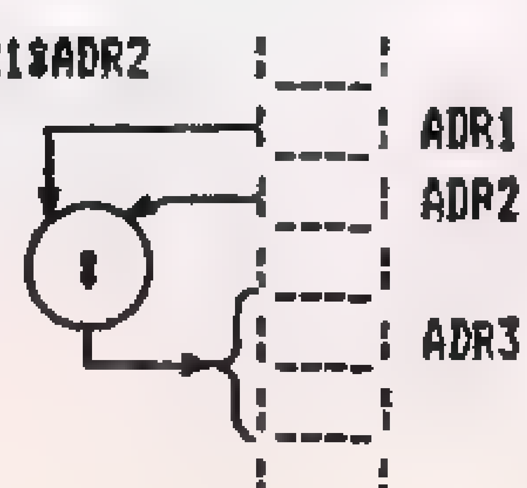


t = 56*n + 29, p = 18
(1 <= n <= 255)

7.3 Mnożenie liczb binarnych

mnożenie liczb 1-bajtowych bez znaku (wynik 2-bajtowy)

```
     LD   HL,(ADR1-1) ;ADR3:=ADR1*ADR2
     LD   L,0
     LD   D,0
     LD   A,(ADR2)
     LD   E,A
     LD   B,8
P:   ADD  HL,HL
     JR   NC,P1
     ADD  HL,DE
P1:  DJNZ P
     LD   (ADR3),HL
```



t ≅ 377, p = 22

mnożenie zawartości rejestru A przez 2

```
ADD  A,A     ;A:=A*2
```

t = 4, p = 1

```
SLA  A       ;A:=A*2
```

t = 8 ,p = 2

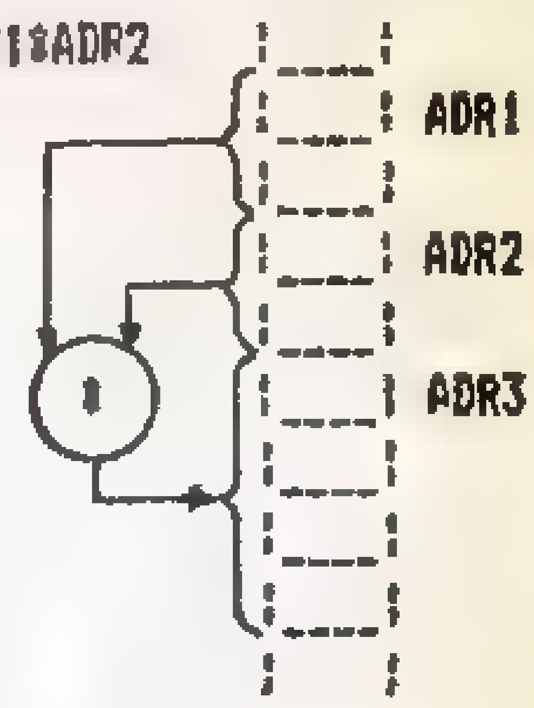
mnożenie zawartości rejestru A przez 5

```
LD   B,A     ;A*(2+2+1)
ADD  A,A
ADD  A,A
ADD  A,B
```

t = 16, p = 4

mnożenie liczb 2-bajtowych bez znaku (wynik 4-bajtowy)

```
     LD   BC,(ADR1)   ;ADR3:=ADR1*ADR2
     LD   DE,(ADR2)
     LD   HL,0
     LD   A,16
P:   ADD  HL,HL
     EX   DE,HL
     ADC  HL,HL
     JR   NC,P1
     ADD  HL,BC
     JR   NC,P1
     INC  DE
P1:  DEC  A
     JR   NZ,P
     LD   (ADR3),HL
     LD   (ADR3+2),DE
```



t ≅ 1208, p = 33

mnożenie liczby 2-bajtowej przez 2

```
LD   HL,ADR   ;ADR:=ADR*2
SLA  (HL)
INC  HL
RL   (HL)
```

t = 46, p = 8

```
LD   xy,ADR   ;ADR:=ADR*2
SLA  (xy)
RL   (xy+1)
```

t = 60, p = 12

mnożenie liczby 2-bajtowej przez 10

```
LD   HL,(ADR) ;ADR:=ADR*10
ADD  HL,HL    ;HL*2
LD   E,L
LD   D,H
ADD  HL,HL    ;HL*4
ADD  HL,HL    ;HL*8
ADD  HL,DE    ;HL*(8+2)
LD   (ADR),HL
```

t = 84, p = 12

7.4 Dzielenie liczb binarnych

dzielenie liczby bez znaku w rejestrze A przez 2

```
SRL  A        ;A:=A/2
```

t = 8, p = 2

dzielenie liczby ze znakiem w rejestrze A przez 2

```
SRA  A        ;A:=A/2
```

t = 8, p = 2

dzielenie liczby 2-bajtowej bez znaku przez 2

```
LD   xy,ADR   ;ADR:=ADR/2
SRL  (xy+1)
RR   (xy)
```

t = 60, p = 12

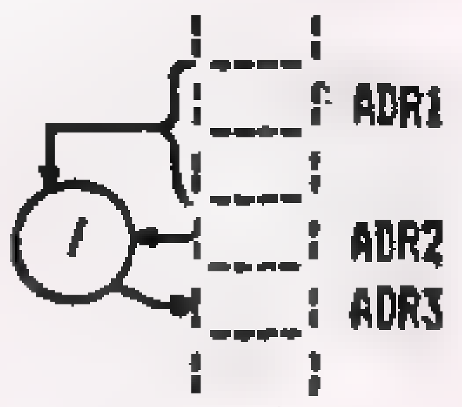
dzielenie liczby 2-bajtowej ze znakiem przez 2

```
LD   xy,ADR   ;ADR:=ADR/2
SRA  (xy+1)
RR   (xy)
```

t = 60, p = 12

dzielenie liczby 2-bajtowej przez liczbę 1-bajtową

```
     LD   HL,(ADR1)   ;ADR3:=ADR1/ADR2
     LD   A,(ADR2)
     LD   C,A
     LD   B,8
P:   ADD  HL,HL
     LD   A,H
     SUB  C
     JR   C,P1
     LD   H,A
     INC  L
P1:  DJNZ P
     LD   A,L
     LD   (ADR3),A
  (reszta z dzielenia
   w rejestrze H)
```



t ≅ 452, p = 16

8. OPERACJE NA TABLICACH

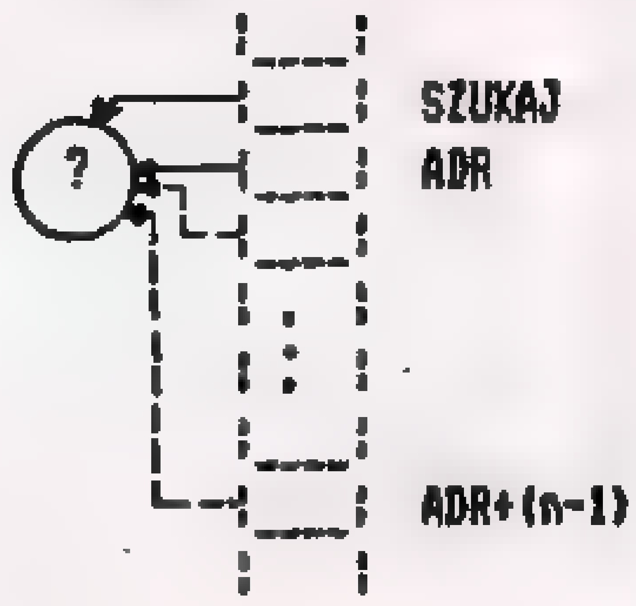
Oznaczenia:
n - liczba elementów tablicy
k - liczba porównań w tablicy

8.1. Sekwencyjne przeglądanie tablicy

8.1.1. Przeglądanie tablicy zbudowanej z elementów 1-bajtowych

Przeglądanie przy użyciu rozkazu CPIR

```
       LD   A,(SZUKAJ)
       LD   HL,ADR
       LD   BC,n
       CPIR
       JR   Z,JEST
   (brak szukanego
      elementu)
JEST:
   (w HL adres
   szukanego elementu)
```



t = 21*k + 40 - znaleziono szukany element / 21*n + 49 - brak szukanego elementu, p = 13
(1 <= k <= n)
(1 <= n <= 65535)

Dla n-elementowej tablicy liczba porównań k zależy od aktualnego położenia szukanego elementu. W najbardziej korzystnym przypadku k = 1, a najbardziej niekorzystnym k = n. Zakładając, że położenie poszukiwanego elementu w tablicy podlega równomiernemu rozkładowi prawdopodobieństwa, średnio należy przeprowadzić (n + 1)/2 porównań.

Przeglądanie przy użyciu rozkazu CPI

```
      LD   A,(SZUKAJ)
      LD   HL,ADR
      LD   BC,n
P:    CPI
      JR   Z,JEST
      JP   PE,P
BRAK:
   (brak szukanego
      elementu)
JEST:
   (w HL adres
   szukanego elementu)
```

t = 33*k + 28 - znaleziono szukany element / 33*n + 28 - brak szukanego elementu, p = 16
(1 <= k <= n)
(1 <= n <= 65535)

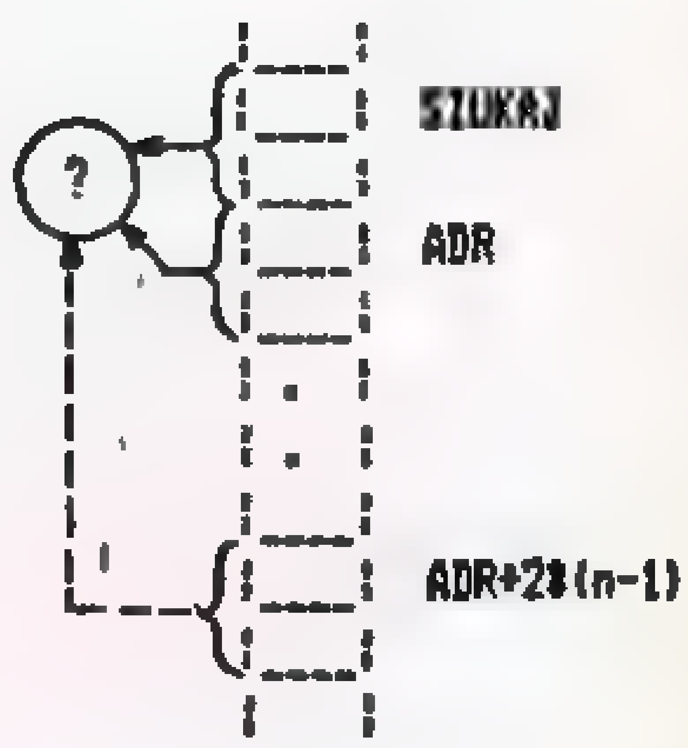
Przeglądanie przy użyciu rozkazu CP (HL)

```
      LD   A,(SZUKAJ)
      LD   HL,ADR
      LD   B,n
P:    CP   (HL)
      JR   Z,JEST
      INC  HL
      DJNZ P
   (brak szukanego
      elementu)
JEST:
   (w HL adres
   szukanego elementu)
```

t = 33*k + 16 - znaleziono szukany element / 33*n + 25 - brak szukanego elementu, p = 13
(1 <= k <= n)
(1 <= n <= 255)

8.1.2. Przeglądanie tablicy zbudowanej z elementów 2-bajtowych

```
      LD   A,(SZUKAJ)
      LD   HL,ADR
      LD   B,n
P1:   CP   (HL)
      JR   Z,P2
      INC  HL
      INC  HL
      DJNZ P1
      JR   BRAK
P2:   LD   A,(SZUKAJ+1)
      INC  HL
      LD   D,(HL)
      CP   D
      JR   Z,JEST
      INC  HL
      LD   A,(SZUKAJ)
      DJNZ P1
```



BRAK:
(brak szukanego elementu)

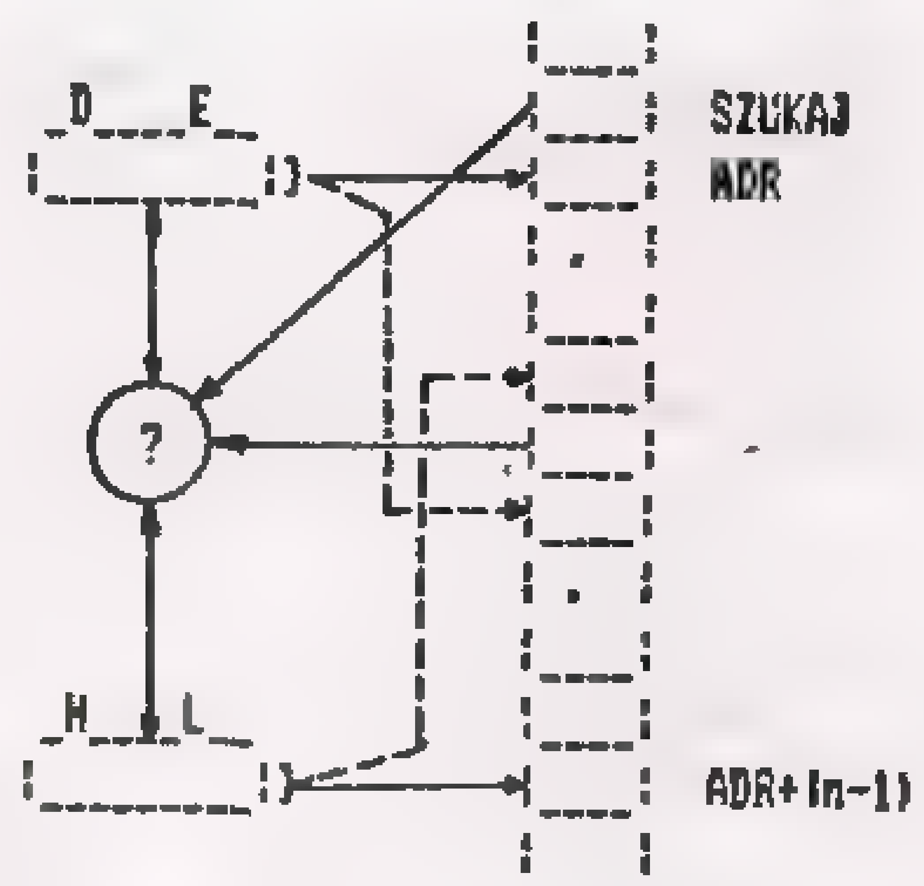
JEST:
(w HL adres do drugiego bajtu szukanego elementu)

t = 398k + 49a + 15, p = 31
(1 <= n <= 255)

a - liczba elementów tablicy, których pierwszy bajt jest identyczny z pierwszym bajtem poszukiwanego elementu, a drugie są różne (0 <= a <= n)

8.2. Binarne (logarytmiczne) przeglądanie tablicy zbudowanej z elementów 1-bajtowych

Warunkiem stosowania przeglądania binarnego jest uporządkowanie elementów tablicy wg wartości wzrastających. Metoda ta polega na porównywaniu poszukiwanej wartości ze środkowym elementem tablicy i jeżeli nie są to te same wartości, podobszar dalszego przeszukiwania jest ograniczony do górnej lub dolnej połowy obszaru tablicy, wyznaczonej przez element środkowy. Z kolei w połowie tej badany jest jej element środkowy i znowu, jeżeli warunek porównywania nie jest spełniony, to podobszar przeszukiwania zmniejsza się o połowę. Operacja ta jest powtarzana, aż do znalezienia poszukiwanego elementu lub stwierdzenia jego braku w tablicy.




```
      LD   HL,ADR
      LD   E,L        ;DE:=ADR
      LD   D,H
      LD   BC,n-1
      ADD  HL,BC      ;HL:=ADR+(n-1)
      LD   A,(SZUKAJ)
      LD   C,A
P:    PUSH HL
      ADD  HL,DE      ;HL:=(HL+DE)/2
      RR   H          ;dzielenie 17-bitowej
      RR   L          ;liczby przez 2
      LD   A,(HL)     ;sprawdź element środkowy
      CP   C
      JR   NC,P1
      EX   DE,HL      ;dolna podtablica
      INC  DE
      POP  HL
      JR   P2
P1:   INC  SP
      INC  SP
      JR   Z,JEST
      DEC  HL         ;górna podtablica
P2:   LD   A,L        ;DE > HL gdy brak elementu
      CP   E
      LD   A,H
      SBC  A,D
      JR   NC,P
```

(brak szukanego elementu)

JEST:
(w HL adres szukanego elementu)

t = 113k + 23, p = 41

Maksymalna liczba porównań w tablicy wynosi:

$$k = \lfloor \log_2 n \rfloor + 1$$

Gdy brak poszukiwanego elementu:

$$k = \lfloor \log_2 n \rfloor + 2$$

Zakładając, że położenie poszukiwanego elementu w tablicy podlega równomiernemu rozkładowi prawdopodobieństwa, średnia liczba porównań wynosi [6]:

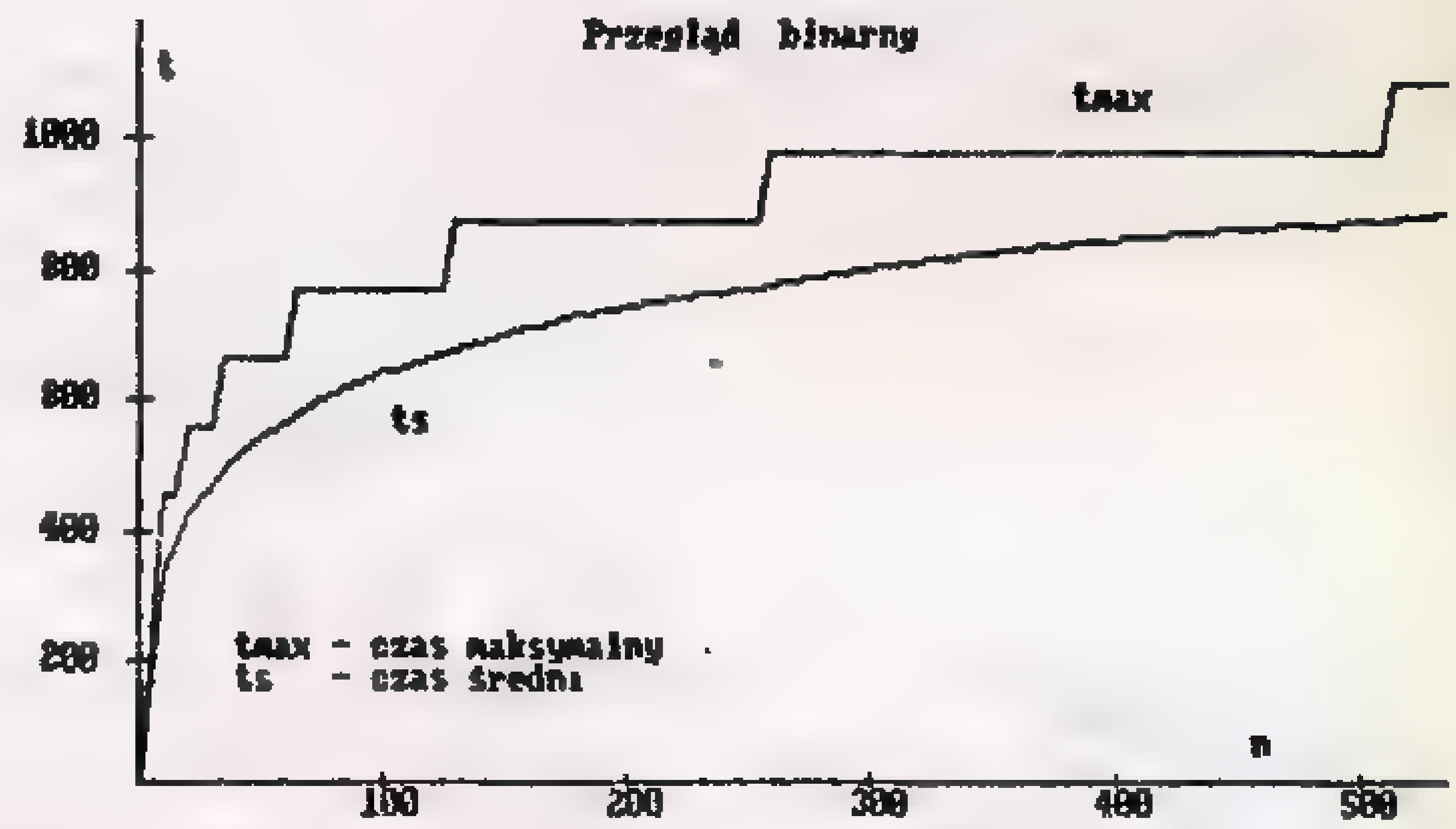
$$k = q + 1 - (2^{q+1} - q - 2)/n$$

gdzie:

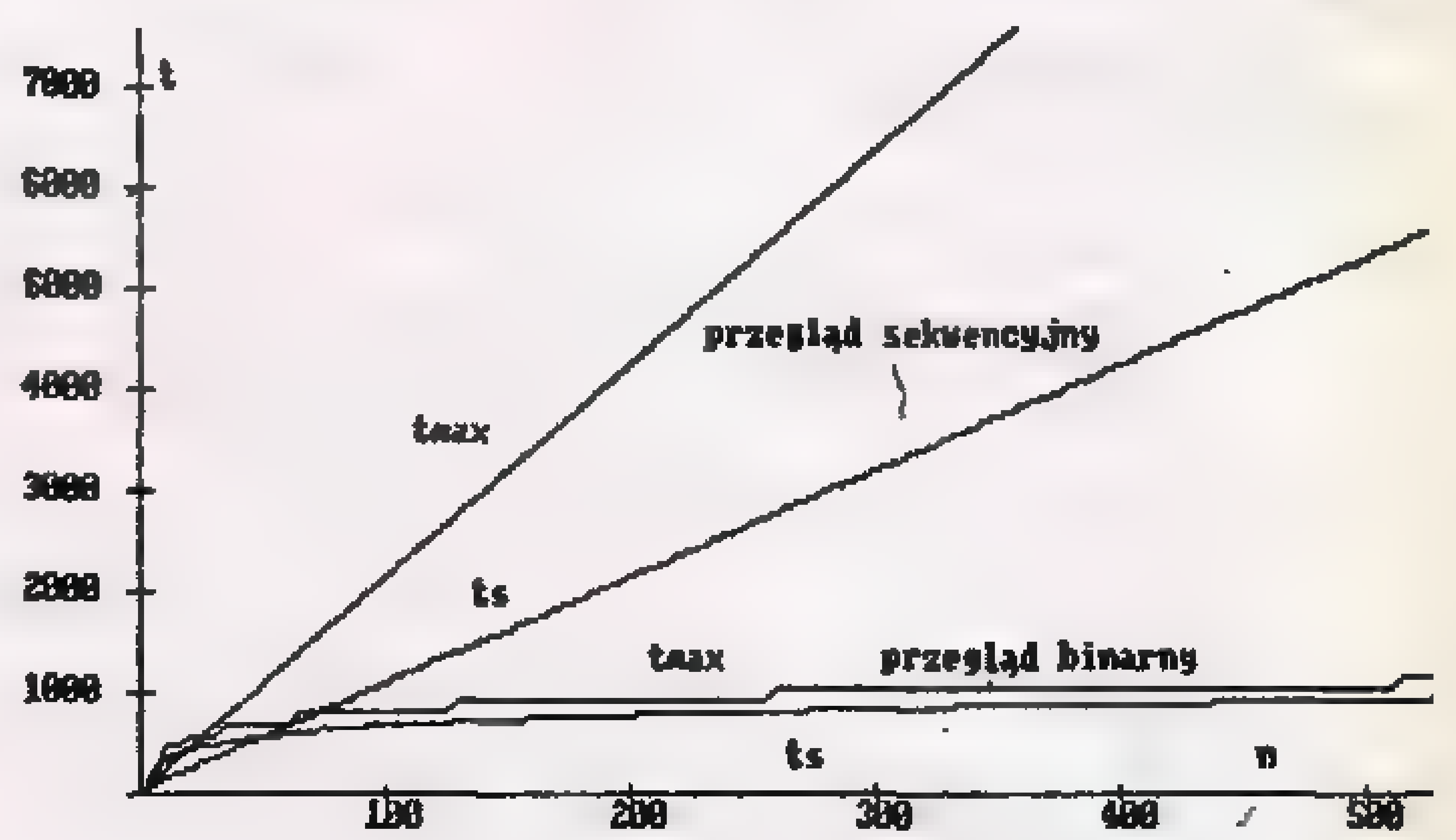
$$q = \lfloor \log_2 n \rfloor$$

$\lfloor r \rfloor$ - oznaczenie części całkowitej liczby r

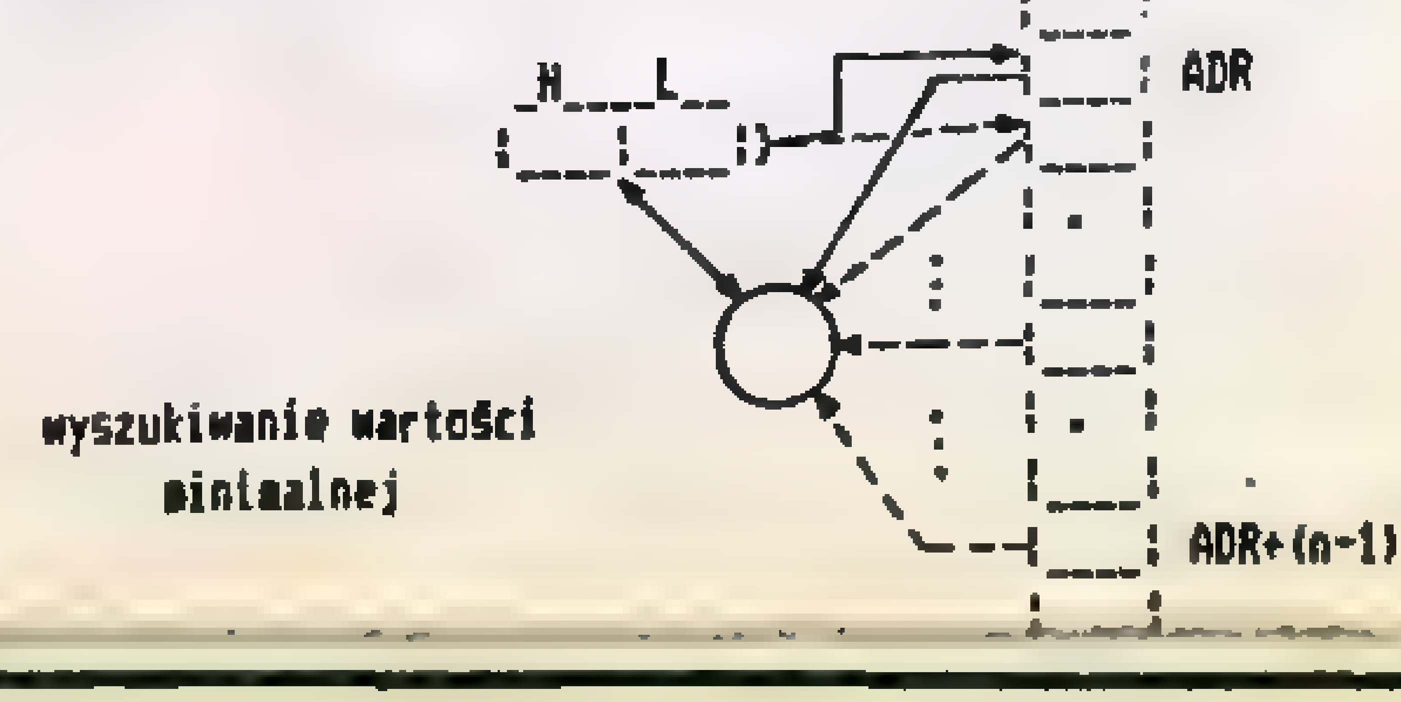
Poniżej zestawiono wykresy ilustrujące zależność średniego i maksymalnego czasu przeglądu binarnego od rozmiarów tablicy. Cechą charakterystyczną jest stosunkowo mała różnica pomiędzy czasem maksymalnym i średnim.



Kolejne wykresy przedstawiają zestawienie charakterystyk chronometrażowych przeglądu sekwencyjnego (przy użyciu rozkazu CPIR) i binarnego. Wykazują one, że dla tablic o niewielkich rozmiarach (do 50 elementów) lepiej jest stosować przegląd sekwencyjny, oraz wzrost przewagi przeglądu binarnego ze wzrostem rozmiarów tablicy.



8.3. Wyszukiwanie z tablicy elementu o wartości minimalnej i maksymalnej



```
        LD   HL,ADR
        LD   B,n
P:      LD   A,(HL)
        LD   E,L
        LD   D,H
P1:     DEC  B
        JR   Z,P2
        INC  HL
        CP   (HL)
        JR   C,P1   ;A < (HL)
        JR   Z,P1   ;A = (HL)
        JR   P      ;A > (HL)
P2:     EX   DE,HL
        (w HL adres elementu
         o wartości minimalnej)
```

t = 39*n + 26*52 + 52, p = 20

a - liczba przypadków, gdy kolejny element tablicy ma wartość mniejszą od wartości najmniejszej z dotychczas znalezionych (1 <= a <= n), (2 <= n <= 255)

wyszukiwanie wartości maksymalnej

```
        LD   HL,ADR
        LD   B,n
P:      LD   A,(HL)
        LD   E,L
        LD   D,H
P1:     DEC  B
        JR   Z,P2
        INC  HL
        CP   (HL)
        JR   NC,P1  ;A >= (HL)
        JR   P      ;A <  (HL)
P2:     EX   DE,HL
        (w HL adres elementu
         o wartości maksymalnej)
```

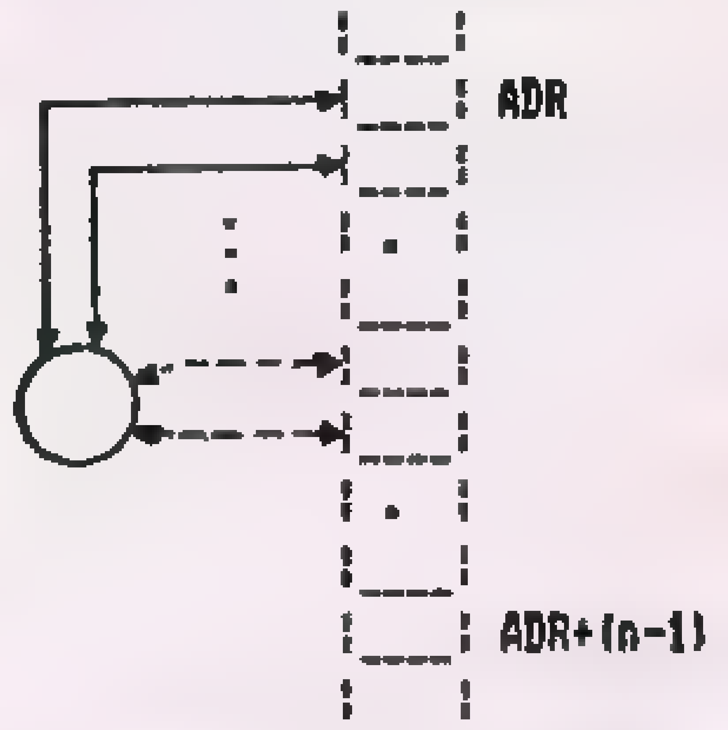
t = 36*n + 22*a + 40, p = 18

a - liczba przypadków, gdy kolejny element tablicy ma wartość większą od wartości największej z dotychczas znalezionych (1 <= a <= n), (2 <= n <= 255)

8.4. Sortowanie elementów tablicy

Sortowanie tablicy (zbudowanej z elementów 1-bajtowych) metodą zamiany sąsiednich elementów.

Metoda ta polega na zasadzie porównywania i zamiany par sąsiadujących ze sobą elementów tablicy tak długo, aż wszystkie zostaną posortowane. Na początku pierwszy element tablicy jest porównywany z drugim i zamieniany z nim w razie potrzeby. Potem drugi z trzecim, trzeci z czwartym i tak aż do końca tablicy. W razie wystąpienia co najmniej jednej zamiany, cała ta operacja jest powtarzana od początku tablicy. Brak zamiany oznacza koniec sortowania.




```
P:      LD   HL,ADR
        LD   DE,ADR+1
        LD   B,n-1
        LD   C,0
P1:     LD   A,(DE)
        CP   (HL)                ;porównanie 2 kolejnych
        JR   NC,P2               ;elementów
        LD   C,(HL)
        LD   (HL),A
        LD   A,C                 ;zamiana sąsiednich
                                 ;elementów
        LD   (DE),A
        LD   C,1                 ;była zamiana
P2:     INC  HL
        INC  DE
        DJNZ P1
        DEC  C                   ;czy była zamiana?
        JR   Z,P
```

p = 27, (2 <= n <= 255)

Czas wykonania operacji sortowania w znacznym stopniu zależy od początkowego ułożenia elementów tablicy. W najbardziej korzystnym wypadku, gdy przy pierwszym przeglądnięciu tablicy nie wystąpiła żadna zamiana (czyli elementy tablicy były już posortowane) czas ten wynosi:

t = n*46 - 6

W najbardziej niekorzystnym przypadku, gdy elementy tablicy są uporządkowane odwrotnie, tzn. od wartości największej do najmniejszej czas sortowania wynosi:

$t = n^{2}*52 + n*26$

DODATEK I.
ZESTAWIENIE ROZKAZOW MIKROPROCESORA Z80

	t	p	2 MHz	3.5 MHz	4 MHz	
ADC A,n	7	2	3.5	2	1.8	n - wielkość 8-bitowa
ADC A,r	4	1	2	1.1	1	r=(A,B,C,D,E,H,L)
ADC A,(HL)	7	1	3.5	2	1.8	
ADC A,(xy+d)	19	3	9.5	5.4	4.8	xy=(IX,IY)
						d - liczba 8-bitowa ze znakiem
ADC HL,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,HL,SP)
ADD A,n	7	2	3.5	2	1.8	
ADD A,r	4	1	2	1.1	1	
ADD A,(HL)	7	1	3.5	2	1.8	
ADD A,(xy+d)	19	3	9.5	5.4	4.8	
ADD HL,rp	11	1	5.5	3.1	2.8	rp=(BC,DE,HL,SP)
ADD IX,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,IX,SP)
ADD IY,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,IY,SP)
AND n	7	2	3.5	2	1.8	
AND r	4	1	2	1.1	1	
AND (HL)	7	1	3.5	2	1.8	
AND (xy+d)	19	3	9.5	5.4	4.8	
BIT b,r	8	2	4	2.3	2	b=(0,1,...,7)
BIT b,(HL)	12	2	6	3.4	3	
BIT b,(xy+d)	20	4	10	5.7	5	
CALL nn	17	3	8.5	4.9	4.3	nn -wielkość 16-bitowa
CALL C,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL M,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL NC,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	nie-(10/17)-tak
CALL NZ,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL P,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL PE,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL PO,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL Z,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CCF	4	1	2	1.1	1	
CP n	7	2	3.5	2	1.8	
CP r	4	1	2	1.1	1	
CP (HL)	7	1	3.5	2	1.8	
CP (xy+d)	19	3	9.5	5.4	4.8	
CPD	16	2	8	4.6	4	BC=0 OR A=(HL)
CPDR	21/16	2	10.5/8	6/4.6	5.3/4	
CPI	16	2	8	4.6	4	(21/16)
CPIR	21/16	2	10.5/8	6/4.6	5.3/4	
						BC#0 AND A#(HL)
CPL	4	1	2	1.1	1	

	t	p	2 MHz	3.5 MHz	4 MHz	
DAA	4	1	2	1.1	1	
DEC r	4	1	2	1.1	1	
DEC (HL)	11	1	5.5	3.1	2.8	
DEC (xy+d)	23	3	11.5	6.6	5.8	
DEC rp	6	1	3	1.7	1.5	rp=(BC,DE,HL,SP)
DEC xy	10	2	5	2.9	2.5	
DI	4	1	2	1.1	1	
DJNZ d	13/8	2	6.5/4	3.7/2.3	3.3/2	B#0-(13/8)-B=0
EI	4	1	2	1.1	1	
EX AF,AF'	4	1	2	1.1	1	
EX DE,HL	4	1	2	1.1	1	
EX (SP),HL	19	1	9.5	5.4	4.8	
EX (SP),xy	23	2	11.5	6.6	5.8	
EXX	4	1	2	1.1	1	
HALT	4	1	2	1.1	1	
IM 0	8	2	4	2.3	2	
IM 1	8	2	4	2.3	2	
IM 2	8	2	4	2.3	2	
IN A,(n)	11	1	5.5	3.1	2.8	
IN r,(C)	12	2	6	3.4	3	
INC r	4	1	2	1.1	1	
INC (HL)	11	1	5.5	3.1	2.8	
INC (xy+d)	23	3	11.5	6.6	5.8	
INC rp	6	1	3	1.7	1.5	rp=(BC,DE,HL,SP)
INC xy	10	2	5	2.9	2.5	
IND	15	2	7.5	4.3	3.8	
INDR	20/15	2	10/7.5	5.7/4.3	5/3.8	B#0-(20/15)-B=0
INI	15	2	7.5	4.3	3.8	
INIR	20/15	2	10/7.5	5.7/4.3	5/3.8	
JP nn	10	3	5	2.9	2.5	
JP (HL)	4	1	2	1.1	1	
JP (xy)	8	2	4	2.3	2	
JP C,nn	10	3	5	2.9	2.5	
JP M,nn	10	3	5	2.9	2.5	
JP NC,nn	10	3	5	2.9	2.5	
JP NZ,nn	10	3	5	2.9	2.5	
JP P,nn	10	3	5	2.9	2.5	
JP PE,nn	10	3	5	2.9	2.5	
JP PO,nn	10	3	5	2.9	2.5	
JP Z,nn	10	3	5	2.9	2.5	

	t	p	2 MHz	3.5 MHz	4 MHz	
JR d	12	2	6	3.4	3	
JR C,d	7/12	2	3.5/6	2/3.4	1.8/3	nie-(7/12)-tak
JR NC,d	7/12	2	3.5/6	2/3.4	1.8/3	
JR NZ,d	7/12	2	3.5/6	2/3.4	1.8/3	
JR Z,d	7/12	2	3.5/6	2/3.4	1.8/3	
LD r1,r2	4	1	2	1.1	1	r1=r2=(A,B,C,D,E,H,L)
LD r,n	7	2	3.5	2	1.8	
LD r,(HL)	7	1	3.5	2	1.8	
LD r,(xy+d)	19	3	9.5	5.4	4.8	
LD (HL),n	10	2	5	2.9	2.5	
LD (HL),r	7	1	3.5	2	1.8	
LD (xy+d),r	19	3	9.5	5.4	4.8	
LD (xy+d),n	19	4	9.5	5.4	4.8	
LD A,(rp)	7	1	3.5	2	1.8	rp=(BC,DE)
LD A,(nn)	13	3	6.5	3.7	3.3	
LD (rp),A	7	1	3.5	2	1.8	rp=(BC,DE)
LD (nn),A	13	3	6.5	3.7	3.3	
LD A,I	9	2	4.5	2.6	2.3	
LD A,R	9	2	4.5	2.6	2.3	
LD I,A	9	2	4.5	2.6	2.3	
LD R,A	9	2	4.5	2.6	2.3	
LD rp,nn	10	3	5	2.9	2.5	rp=(BC,DE,HL,SP)
LD xy,nn	14	4	7	4	3.5	
LD HL,(nn)	16	3	8	4.6	4	
LD rp,(nn)	20	4	10	5.7	5	rp=(BC,DE,SP,IX,IY)
LD (nn),HL	16	3	8	4.6	4	
LD (nn),rp	20	4	10	5.7	5	rp=(BC,DE,SP,IX,IY)
LD SP,HL	6	1	3	1.7	1.5	
LD SP,xy	10	2	5	2.9	2.5	
LDD	16	2	8	4.6	4	
LDDR	21/16	2	10.5/8	6/4.6	5.3/4	BC#0-(21/16)-BC=0
LDI	16	2	8	4.6	4	
LDIR	21/16	2	10.5/8	6/4.6	5.3/4	
NEG	8	2	4	2.3	2	
NOP	4	1	2	1.1	1	
OR n	7	2	3.5	2	1.8	
OR r	4	1	2	1.1	1	
OR (HL)	7	1	3.5	2	1.8	
OR (xy+d)	19	3	9.5	5.4	4.8	
OUT (C),r	12	2	6	3.4	3	
OUT (n),A	11	2	5.5	3.1	2.8	
OUTI	16	2	8	4.6	4	
OTIR	21/16	2	10.5/8	6/4.6	5.3/4	B#0-(21/16)-B=0
OUTD	16	2	8	4.6	4	
OTDR	21/16	2	10.5/8	6/4.6	5.3/4	

	t	p	2 MHz	3.5 MHz	4 MHz	
POP rp	10	1	5	2.9	2.5	rp=(BC.DE.HL.AF)
POP xy	14	2	7	4	3.5	
PUSH rp	11	1	5.5	3.1	2.8	rp=(BC,DE.HL.AF)
PUSH xy	15	2	7.5	4.3	3.8	
RES b,r	8	2	4	2.3	2	
RES b,(HL)	15	2	7.5	4.3	3.8	
RES b,(xy+d)	23	4	11.5	6.6	5.8	
RET	10	1	5	2.9	2.5	
RET C	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	nie-(5/11)-tak
RET M	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET NC	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET NZ	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET P	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET PE	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET PO	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET Z	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RETI	14	2	7	4	3.5	
RETN	14	2	7	4	3.5	
RLCA	4	1	2	1.1	1	
RLA	4	1	2	1.1	1	
RRCA	4	1	2	1.1	1	
RRA	4	1	2	1.1	1	
RLC r	8	2	4	2.3	2	
RLC (HL)	15	2	7.5	4.3	3.8	
RLC (xy+d)	23	4	11.5	6.6	5.8	
RL r	8	2	4	2.3	2	
RL (HL)	15	2	7.5	4.3	3.8	
RL (xy+d)	23	4	11.5	6.6	5.8	
RRC r	8	2	4	2.3	2	
RRC (HL)	15	2	7.5	4.3	3.8	
RRC (xy+d)	23	4	11.5	6.6	5.8	
RR r	8	2	4	2.3	2	
RR (HL)	15	2	7.5	4.3	3.8	
RR (xy+d)	23	4	11.5	6.6	5.8	
RLD	18	2	9	5.1	4.5	
RRD	18	2	9	5.1	4.5	
RST a	11	1	5.5	3.1	2.8	a=(0,8,16,24,32,40,48,56)
SBC A,n	7	2	3.5	2	1.8	
SBC A,r	4	1	2	1.1	1	
SBC A,(HL)	7	1	3.5	2	1.8	
SBC A,(xy+d)	19	3	9.5	5.4	4.8	
SBC HL,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,HL,SP)

	t	p	2 MHz	3.5 MHz	4 MHz
SCF	4	1	2	1.1	1
SET b,r	8	2	4	2.3	2
SET b,(HL)	15	2	7.5	4.3	3.8
SET b,(xy+d)	23	4	11.5	6.6	5.8
SLA r	8	2	4	2.3	2
SLA (HL)	15	2	7.5	4.3	3.8
SLA (xy+d)	23	4	11.5	6.6	5.8
SRA r	8	2	4	2.3	2
SRA (HL)	15	2	7.5	4.3	3.8
SRA (xy+d)	23	4	11.5	6.6	5.8
SRL r	8	2	4	2.3	2
SRL (HL)	15	2	7.5	4.3	3.8
SRL (xy+d)	23	4	11.5	6.6	5.8
SUB n	7	2	3.5	2	1.8
SUB r	4	1	2	1.1	1
SUB (HL)	7	1	3.5	2	1.8
SUB (xy+d)	19	3	9.5	5.4	4.0
XOR n	7	2	3.5	2	1.8
XOR r	4	1	2	1.1	1
XOR (HL)	7	1	3.5	2	1.8
XOR (xy+d)	19	3	9.5	5.4	4.0

DODATEK II.

TABELE SZESNASTKOWYCH KODOW ROZKAZOW MIKROPPOCESORA Z80

Szęsnastkowy kod określonego rozkazu jest wyznaczany przez sumowanie 2-cyfrowej liczby szesnastkowej (od 00 do F0) określającej wiersz w którym rozkaz jest zapisany z 1-cyfrową liczbą szesnastkową (od 0 do F) określającą kolumnę.
Przykładowo, rozkaz CALL nn, zapisany w wierszu C0 i kolumnie D tablicy D.II.1 identyfikowany jest szesnastkowym kodem C0 + D = CD.

Oznaczenia przyjęte w tablicach są analogiczne jak w dodatku I, oprócz różnic, które każdorazowo zaznaczono oraz innego zapisu argumentów rozkazów RST. Argumentami są liczby z zakresu od 0 do 7, wyznaczające odpowiednio adresy 0, 8, 16,... 56.

W tablicach zamieszczono także kody rozkazów nie opublikowanych w oficjalnych materiałach firmy Zilog oraz przypadki interpretacji przez mikroprocesor różnych kodów jako ten sam rozkaz (przykładowo liczba szesnastkowa 22 i dwie liczby szesnastkowe ED, 63 są interpretowane jako kod rozkazu LD (nn),HL). Rozkazom tym nadano symboliczne kody, ale nie są one akceptowane przez standardowe asemblery i disasemblery. W programach można je umieścić w postaci kodów liczbowych (w tablicach D.II.2. do D.II.5. zostały zaznaczone symbolem *).
Pola z wpisanym oznaczeniem tablicy oznaczają kody wielobajtowe (głównie 2-bajtowe), opisane w odpowiedniej tablicy.

Tablica D.II.1. Rozkazy o kodach 1-bajtowych

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	NOP	LD BC,nn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA
10	DJNZ n	LD DE,nn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR n	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RRA
20	JR NZ,n	LD HL,nn	LD (nn),HL	INC HL	INC H	DEC H	LD H,n	DAA	JR Z,n	ADD HL,HL	LD HL,(nn)	DEC HL	INC L	DEC L	LD L,n	CPL
30	JR NC,n	LD SP,nn	LD (nn),A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR C,n	ADD HL,SP	LD A,(nn)	DEC SP	INC A	DEC A	LD A,n	CCF
40	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
50	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
60	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
70	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
80	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
90	SUB A,B	SUB A,C	SUB A,D	SUB A,E	SUB A,H	SUB A,L	SUB A,(HL)	SUB A,A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
A0	AND A,B	AND A,C	AND A,D	AND A,E	AND A,H	AND A,L	AND A,(HL)	AND A,A	XOR A,B	XOR A,C	XOR A,D	XOR A,E	XOR A,H	XOR A,L	XOR A,(HL)	XOR A,A
B0	OR A,B	OR A,C	OR A,D	OR A,E	OR A,H	OR A,L	OR A,(HL)	OR A,A	CP A,B	CP A,C	CP A,D	CP A,E	CP A,H	CP A,L	CP A,(HL)	CP A,A
C0	RET NZ	POP BC	JP NZ,nn	JP nn	CALL NZ,nn	PUSH BC	ADD A,n	RST 0	RET Z	RET	JP Z,nn	D.II.3.	CALL Z,nn	CALL nn	ADC A,n	RST 1
D0	RET NC	POP DE	JP NC,nn	OUT (n),A	CALL NC,nn	PUSH DE	SUB A,n	RST 2	RET C	EXX	JP C,nn	IN (n),A	CALL C,nn	D.II.4.	SBC A,n	RST 3
E0	RET PO	POP HL	JP PO,nn	EX (SP),HL	CALL PO,nn	PUSH HL	AND A,n	RST 4	RET PE	JP (HL)	JP PE,nn	EX DE,HL	CALL PE,nn	D.II.2.	XOR A,n	RST 5
F0	RET P	POP AF	JP P,nn	DI	CALL P,nn	PUSH AF	OR A,n	RST 6	RET M	LD SP,HL	JP M,nn	EI	CALL M,nn	D.II.5.	CP A,n	RST 7

Tablica D.II.2. Kody rozkazów poprzedzone bajtem ED

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00																
10																
20																
30																
40	IN B,(C)	OUT (C),B	SBC HL,BC	LD (nn),BC	NEG	RETN	IM 0	LD I,A	IN C,(C)	OUT (C),C	ADC HL,BC	LD BC,(nn)	* NEG	RETI		LD R,A
50	IN D,(C)	OUT (C),D	SBC HL,DE	LD (nn),DE	* NEG	* RETN	IM 1	LD A,I	IN E,(C)	OUT (C),E	ADC HL,DE	LD DE,(nn)	* NEG	* RETI	IM 2	LD A,R
60	IN H,(C)	OUT (C),H	SBC HL,HL	* LD (nn),HL	* NEG	* RETN		RRD	IN L,(C)	OUT (C),L	ADC HL,HL	* LD HL,(nn)	* NEG	* RETI		RLD
70	IN F,(C)		SBC HL,SP	LD (nn),SP	* NEG	* RETN			IN A,(C)	OUT (C),A	ADC HL,SP	LD SP,(nn)	* NEG	* RETI		
80																
90																
A0	LDI	CPI	INI	OUTI					LDD	CPD	IND	OUTD				
B0	LDIR	CPIR	INIR	OTIR					LDDR	CPDR	INDR	OTDR				
C0																
D0																
E0																
F0																

Kody odpowiadające pozycjom nie wypełnionym powyższej tablicy są interpretowane przez mikroprocesor jednakowo jako rozkaz NOP. Kody ED 63 i ED 6B nie są ujęte w firmowej liście rozkazów, asemblery generują odpowiednio 1-bajtowe kody 22 i 2A (wg tablicy D.II.1.).
Dla rozkazów NEG, RETN i RETI generowane są kody: ED 44, ED 45 i ED 4D. Pozostałe kody dla tych rozkazów należą do grupy nie opublikowanych.

Tablica D.II.3. Kody rozkazów poprzedzone bajtem CB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC (HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
10	RL B	RL C	RL D	RL E	RL H	RL L	RL (HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
20	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA (HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
30	* SLA1 B	* SLA1 C	* SLA1 D	* SLA1 E	* SLA1 H	* SLA1 L	* SLA1 (HL)	* SLA1 A	SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
40	BIT 0,B	BIT 0,C	BIT 0,D	BIT 0,E	BIT 0,H	BIT 0,L	BIT 0,(HL)	BIT 0,A	BIT 1,B	BIT 1,C	BIT 1,D	BIT 1,E	BIT 1,H	BIT 1,L	BIT 1,(HL)	BIT 1,A
50	BIT 2,B	BIT 2,C	BIT 2,D	BIT 2,E	BIT 2,H	BIT 2,L	BIT 2,(HL)	BIT 2,A	BIT 3,B	BIT 3,C	BIT 3,D	BIT 3,E	BIT 3,H	BIT 3,L	BIT 3,(HL)	BIT 3,A
60	BIT 4,B	BIT 4,C	BIT 4,D	BIT 4,E	BIT 4,H	BIT 4,L	BIT 4,(HL)	BIT 4,A	BIT 5,B	BIT 5,C	BIT 5,D	BIT 5,E	BIT 5,H	BIT 5,L	BIT 5,(HL)	BIT 5,A
70	BIT 6,B	BIT 6,C	BIT 6,D	BIT 6,E	BIT 6,H	BIT 6,L	BIT 6,(HL)	BIT 6,A	BIT 7,B	BIT 7,C	BIT 7,D	BIT 7,E	BIT 7,H	BIT 7,L	BIT 7,(HL)	BIT 7,A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80	RES 0,B	RES 0,C	RES 0,D	RES 0,E	RES 0,H	RES 0,L	RES 0,(HL)	RES 0,A	RES 1,B	RES 1,C	RES 1,D	RES 1,E	RES 1,H	RES 1,L	RES 1,(HL)	RES 1,A
90	RES 2,B	RES 2,C	RES 2,D	RES 2,E	RES 2,H	RES 2,L	RES 2,(HL)	RES 2,A	RES 3,B	RES 3,C	RES 3,D	RES 3,E	RES 3,H	RES 3,L	RES 3,(HL)	RES 3,A
A0	RES 4,B	RES 4,C	RES 4,D	RES 4,E	RES 4,H	RES 4,L	RES 4,(HL)	RES 4,A	RES 5,B	RES 5,C	RES 5,D	RES 5,E	RES 5,H	RES 5,L	RES 5,(HL)	RES 5,A
B0	RES 6,B	RES 6,C	RES 6,D	RES 6,E	RES 6,H	RES 6,L	RES 6,(HL)	RES 6,A	RES 7,B	RES 7,C	RES 7,D	RES 7,E	RES 7,H	RES 7,L	RES 7,(HL)	RES 7,A
C0	SET 0,B	SET 0,C	SET 0,D	SET 0,E	SET 0,H	SET 0,L	SET 0,(HL)	SET 0,A	SET 1,B	SET 1,C	SET 1,D	SET 1,E	SET 1,H	SET 1,L	SET 1,(HL)	SET 1,A
D0	SET 2,B	SET 2,C	SET 2,D	SET 2,E	SET 2,H	SET 2,L	SET 2,(HL)	SET 2,A	SET 3,B	SET 3,C	SET 3,D	SET 3,E	SET 3,H	SET 3,L	SET 3,(HL)	SET 3,A
E0	SET 4,B	SET 4,C	SET 4,D	SET 4,E	SET 4,H	SET 4,L	SET 4,(HL)	SET 4,A	SET 5,B	SET 5,C	SET 5,D	SET 5,E	SET 5,H	SET 5,L	SET 5,(HL)	SET 5,A
F0	SET 6,B	SET 6,C	SET 6,D	SET 6,E	SET 6,H	SET 6,L	SET 6,(HL)	SET 6,A	SET 7,B	SET 7,C	SET 7,D	SET 7,E	SET 7,H	SET 7,L	SET 7,(HL)	SET 7,A

SLA1 - nie opublikowany rozkaz podobny w działaniu do SLA, różniący się od niego wprowadzeniem "1" zamiast "0" na zerową pozycję binarną argumentu.

Tablica D.II.4. kody rozkazów poprzedzone bajtem DD

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	* NOP	* LD BC,nn	* LD (BC),A	* INC BC	* INC B	* DEC B	* LD B,n	* RLCA	* EX AF,AF'	ADD X,BC	* LD A,(BC)	* DEC BC	* INC C	* DEC C	* LD C,n	* RRCA
10	* DJNZ n	* LD DE,nn	* LD (DE),A	* INC DE	* INC D	* DEC D	* LD D,n	* RLA	* JR n	ADD X,DE	* LD A,(DE)	* DEC DE	* INC E	* DEC E	* LD E,n	* RRA
20	* JR NZ,n	LD X,nn	LD (nn),X	INC X	* INC HX	* DEC HX	* LD HX,n	* DAA	* JR Z,n	ADD X,X	LD X,(nn)	DEC X	* INC LX	* DEC LX	* LD LX,n	* CPL
30	* JR NC,n	* LD SP,nn	* LD (nn),A	* INC SP	INC (X+d)	DEC (X+d)	LD (X+d),n	* SCF	* JR C,n	ADD X,SP	* LD A,(nn)	* DEC SP	* INC A	* DEC A	* LD A,n	* CCF
40	* LD B,B	* LD B,C	* LD B,D	* LD B,E	* LD B,HX	* LD B,LX	LD B,(X+d)	* LD B,A	* LD C,B	* LD C,C	* LD C,D	* LD C,E	* LD C,HX	* LD C,LX	LD C,(X+d)	* LD C,A
50	* LD D,B	* LD D,C	* LD D,D	* LD D,E	* LD D,HX	* LD D,LX	LD D,(X+d)	* LD D,A	* LD E,B	* LD E,C	* LD E,D	* LD E,E	* LD E,HX	* LD E,LX	LD E,(X+d)	* LD E,A
60	* LD HX,B	* LD HX,C	* LD HX,D	* LD HX,E	* LD HX,HX	* LD HX,LX	LD H,(X+d)	* LD HX,A	* LD LX,B	* LD LX,C	* LD LX,D	* LD LX,E	* LD LX,HX	* LD LX,LX	LD L,(X+d)	* LD LX,A
70	LD (X+d),B	LD (X+d),C	LD (X+d),D	LD (X+d),E	LD (X+d),H	LD (X+d),L	* HALT	LD (X+d),A	* LD A,B	* LD A,C	* LD A,D	* LD A,E	* LD A,HX	* LD A,LX	LD A,(X+d)	* LD A,A
80	* ADD A,B	* ADD A,C	* ADD A,D	* ADD A,E	* ADD A,HX	* ADD A,LX	ADD A,(X+d)	* ADD A,A	* ADC A,B	* ADC A,C	* ADC A,D	* ADC A,E	* ADC A,HX	* ADC A,LX	ADC A,(X+d)	* ADC A,A
90	* SUB A,B	* SUB A,C	* SUB A,D	* SUB A,E	* SUB A,HX	* SUB A,LX	SUB A,(X+d)	* SUB A,A	* SBC A,B	* SBC A,C	* SBC A,D	* SBC A,E	* SBC A,HX	* SBC A,LX	SBC A,(X+d)	* SBC A,A
A0	* AND A,B	* AND A,C	* AND A,D	* AND A,E	* AND A,HX	* AND A,LX	AND A,(X+d)	* AND A,A	* XOR A,B	* XOR A,C	* XOR A,D	* XOR A,E	* XOR A,HX	* XOR A,LX	XOR A,(X+d)	* XOR A,A
B0	* OR A,B	* OR A,C	* OR A,D	* OR A,E	* OR A,HX	* OR A,LX	OR A,(X+d)	* OR A,A	* CP A,B	* CP A,C	* CP A,D	* CP A,E	* CP A,HX	* CP A,LX	CP A,(X+d)	* CP A,A
C0	* RET NZ	* POP BC	* JP NZ,nn	* JP nn	* CALL NZ,nn	* PUSH BC	* ADD A,n	* RST 0	* RET Z	* RET	* JP Z,nn	D.II.6.	* CALL Z,nn	* CALL nn	* ADC A,n	* RST 1
D0	* RET NC	* POP DE	* JP NC,nn	* OUT (n),A	* CALL NC,nn	* PUSH DE	* SUB A,n	* RST 2	* RET C	* EXX	* JP C,nn	* IN (n),A	* CALL C,nn	D.II.4.	* SBC A,n	* RST 3
E0	* RET PO	POP X	* JP PO,nn	EX (SP),X	* CALL PO,nn	PUSH X	* AND A,n	* RST 4	* RET PE	JP (X)	* JP PE,nn	* EX DE,HL	* CALL PE,nn	D.II.2.	* XOR A,n	* RST 5
F0	* RET P	* POP AF	* JP P,nn	* DI	* CALL P,nn	* PUSH AF	* OR A,n	* RST 6	* RET M	LD SP,X	* JP M,nn	* EI	* CALL M,nn	D.II.5.	* CP A,n	* RST 7

Wszystkie rozkazy operujące na zawartości połowy rejestru IX oraz będące powtórzeniami z tablicy D.II.1 są nie opublikowane. W ciągach kodów DD DD, DD ED i DD FD pierwszy bajt jest pomijany (traktowany jako rozkaz NOP), drugi wchodzi w skład rozkazu odpowiednio wg tablicy D.II.4. (DD), D.II.2 (ED) i D.II.5. (FD).

X - 16-bitowy rejestr IX

LX - 8 mniej znaczących pozycji binarnych rejestru IX

HX - 8 bardziej znaczących pozycji binarnych rejestru IX

Tablica D.II.5. Kody rozkazów poprzedzone bajtem FD

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	* NOP	* LD BC,nn	* LD (BC),A	* INC BC	* INC B	* DEC B	* LD B,n	* RLCA	* EX AF,AF'	ADD Y,BC	* LD A,(BC)	* DEC BC	* INC C	* DEC C	* LD C,n	* RRCA
10	* DJNZ n	* LD DE,nn	* LD (DE),A	* INC DE	* INC D	* DEC D	* LD D,n	* RLA	* JR n	ADD Y,DE	* LD A,(DE)	* DEC DE	* INC E	* DEC E	* LD E,n	* RRA
20	* JR NZ,n	LD Y,nn	LD (nn),Y	INC Y	* INC HY	* DEC HY	* LD HY,n	* DAA	* JR Z,n	ADD Y,Y	LD Y,(nn)	DEC Y	* INC LY	* DEC LY	* LD LY,n	* CPL
30	* JR NC,n	* LD SP,nn	* LD (nn),A	* INC SP	INC (Y+d)	DEC (Y+d)	LD (Y+d),n	* SCF	* JR C,n	ADD Y,SP	* LD A,(nn)	* DEC SP	* INC A	* DEC A	* LD A,n	* CCF
40	* LD B,B	* LD B,C	* LD B,D	* LD B,E	* LD B,HY	* LD B,LY	LD B,(Y+d)	* LD B,A	* LD C,B	* LD C,C	* LD C,D	* LD C,E	* LD C,HY	* LD C,LY	LD C,(Y+d)	* LD C,A
50	* LD D,B	* LD D,C	* LD D,D	* LD D,E	* LD D,HY	* LD D,LY	LD D,(Y+d)	* LD D,A	* LD E,B	* LD E,C	* LD E,D	* LD E,E	* LD E,HY	* LD E,LY	LD E,(Y+d)	* LD E,A
60	* LD HY,B	* LD HY,C	* LD HY,D	* LD HY,E	* LD HY,HY	* LD HY,LY	LD H,(Y+d)	* LD HY,A	* LD LY,B	* LD LY,C	LD LY,D	LD LY,E	LD LY,HY	LD LY,LY	LD L,(Y+d)	LD LY,A
70	LD (Y+d),B	LD (Y+d),C	LD (Y+d),D	LD (Y+d),E	LD (Y+d),H	LD (Y+d),L	* HALT	LD (Y+d),A	* LD A,B	* LD A,C	* LD A,D	* LD A,E	LD A,HY	* LD A,LY	LD A,(Y+d)	* LD A,A
80	* ADD A,B	* ADD A,C	* ADD A,D	* ADD A,E	* ADD A,HY	* ADD A,LY	ADD A,(Y+d)	* ADD A,A	* ADC A,B	* ADC A,C	* ADC A,D	* ADC A,E	* ADC A,HY	* ADC A,LY	ADC A,(Y+d)	* ADC A,A
90	* SUB A,B	* SUB A,C	* SUB A,D	* SUB A,E	* SUB A,HY	* SUB A,LY	SUB A,(Y+d)	* SUB A,A	* SBC A,B	* SBC A,C	* SBC A,D	* SBC A,E	* SBC A,HY	* SBC A,LY	SBC A,(Y+d)	* SBC A,A
A0	* AND A,B	* AND A,C	* AND A,D	* AND A,E	* AND A,HY	* AND A,LY	AND A,(Y+d)	* AND A,A	* XOR A,B	* XOR A,C	* XOR A,D	* XOR A,E	* XOR A,HY	* XOR A,LY	XOR A,(Y+d)	* XOR A,A
B0	* OR A,B	* OR A,C	* OR A,D	* OR A,E	* OR A,HY	* OR A,LY	OR A,(Y+d)	* OR A,A	* CP A,B	* CP A,C	* CP A,D	* CP A,E	* CP A,HY	* CP A,LY	CP A,(Y+d)	* CP A,A
C0	* RET NZ	* POP BC	* JP NZ,nn	* JP nn	* CALL NZ,nn	* PUSH BC	* ADD A,n	* RST 0	* RET Z	* RET	* JP Z,nn	D.II.7.	* CALL Z,nn	* CALL nn	* ADC A,n	* RST 1
D0	* RET NC	* POP DE	* JP NC,nn	* OUT (n),A	* CALL NC,nn	* PUSH DE	* SUB A,n	* RST 2	* RET C	* EXX	* JP C,nn	* IN (n),A	* CALL C,nn	D.II.4.	* SBC A,n	* RST 3
E0	* RET PO	POP Y	* JP PO,nn	EX (SP),Y	* CALL PO,nn	PUSH Y	* AND A,n	* RST 4	* RET PE	JP (Y)	* JP PE,nn	* EX DE,HL	* CALL PE,nn	D.II.2.	* XOR A,n	* RST 5
F0	* RET P	* POP AF	* JP P,nn	* DI	* CALL P,nn	* PUSH AF	* OR A,n	* RST 6	* RET M	* LD SP,Y	* JP M,nn	* EI	* CALL M,nn	D.II.5.	* CP A,n	* RST 7

Wszystkie rozkazy operujące na zawartości połowy rejestru IY oraz będące powtórzeniami z tablicy D.II.1 są nie opublikowane. W ciągach kodów FD DD, FD ED i FD FD pierwszy bajt jest pomijany (traktowany jako rozkaz NOP), drugi wchodzi w skład rozkazu odpowiednio wg tablicy D.II.4. (DD), D.II.2 (ED) i D.II.5. (FD).

X - 16-bitowy rejestr IY

LY - 8 mniej znaczących pozycji binarnych rejestru IY

HY - 8 bardziej znaczących pozycji binarnych rejestru IY

Tablica D.II.6. Kody rozkazów poprzedzone bajtami DD i CB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	RLC:B (X+d)	RLC:C (X+d)	RLC:D (X+d)	RLC:E (X+d)	RLC:H (X+d)	RLC:L (X+d)	RLC (X+d)	RLC:A (X+d)	RRC:B (X+d)	RRC:C (X+d)	RRC:D (X+d)	RRC:E (X+d)	RRC:H (X+d)	RRC:L (X+d)	RRC (X+d)	RRC:A (X+d)
10	RL:B (X+d)	RL:C (X+d)	RL:D (X+d)	RL:E (X+d)	RL:H (X+d)	RL:L (X+d)	RL (X+d)	RL:A (X+d)	RR:B (X+d)	RR:C (X+d)	RR:D (X+d)	RR:E (X+d)	RR:H (X+d)	RR:L (X+d)	RR (X+d)	RR:A (X+d)
20	SLA:B (X+d)	SLA:C (X+d)	SLA:D (X+d)	SLA:E (X+d)	SLA:H (X+d)	SLA:L (X+d)	SLA (X+d)	SLA:A (X+d)	SRA:B (X+d)	SRA:C (X+d)	SRA:D (X+d)	SRA:E (X+d)	SRA:H (X+d)	SRA:L (X+d)	SRA (X+d)	SRA:A (X+d)
30	SLL:B (X+d)	SLL:C (X+d)	SLL:D (X+d)	SLL:E (X+d)	SLL:H (X+d)	SLL:L (X+d)	SLL (X+d)	SLL:A (X+d)	SRL:B (X+d)	SRL:C (X+d)	SRL:D (X+d)	SRL:E (X+d)	SRL:H (X+d)	SRL:L (X+d)	SRL (X+d)	SRL:A (X+d)
40	BIT:B 0,(X+d)	BIT:C 0,(X+d)	BIT:D 0,(X+d)	BIT:E 0,(X+d)	BIT:H 0,(X+d)	BIT:L 0,(X+d)	BIT 0,(X+d)	BIT:A 0,(X+d)	BIT:B 1,(X+d)	BIT:C 1,(X+d)	BIT:D 1,(X+d)	BIT:E 1,(X+d)	BIT:H 1,(X+d)	BIT:L 1,(X+d)	BIT 1,(X+d)	BIT:A 1,(X+d)
50	BIT:B 2,(X+d)	BIT:C 2,(X+d)	BIT:D 2,(X+d)	BIT:E 2,(X+d)	BIT:H 2,(X+d)	BIT:L 2,(X+d)	BIT 2,(X+d)	BIT:A 2,(X+d)	BIT:B 3,(X+d)	BIT:C 3,(X+d)	BIT:D 3,(X+d)	BIT:E 3,(X+d)	BIT:H 3,(X+d)	BIT:L 3,(X+d)	BIT 3,(X+d)	BIT:A 3,(X+d)
60	BIT:B 4,(X+d)	BIT:C 4,(X+d)	BIT:D 4,(X+d)	BIT:E 4,(X+d)	BIT:H 4,(X+d)	BIT:L 4,(X+d)	BIT 4,(X+d)	BIT:A 4,(X+d)	BIT:B 5,(X+d)	BIT:C 5,(X+d)	BIT:D 5,(X+d)	BIT:E 5,(X+d)	BIT:H 5,(X+d)	BIT:L 5,(X+d)	BIT 5,(X+d)	BIT:A 5,(X+d)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	BIT:B 6,(X+d)	BIT:C 6,(X+d)	BIT:D 6,(X+d)	BIT:E 6,(X+d)	BIT:H 6,(X+d)	BIT:L 6,(X+d)	BIT 6,(X+d)	BIT:A 6,(X+d)	BIT:B 7,(X+d)	BIT:C 7,(X+d)	BIT:D 7,(X+d)	BIT:E 7,(X+d)	BIT:H 7,(X+d)	BIT:L 7,(X+d)	BIT 7,(X+d)	BIT:A 7,(X+d)
80	RES:B 0,(X+d)	RES:C 0,(X+d)	RES:D 0,(X+d)	RES:E 0,(X+d)	RES:H 0,(X+d)	RES:L 0,(X+d)	RES 0,(X+d)	RES:A 0,(X+d)	RES:B 1,(X+d)	RES:C 1,(X+d)	RES:D 1,(X+d)	RES:E 1,(X+d)	RES:H 1,(X+d)	RES:L 1,(X+d)	RES 1,(X+d)	RES:A 1,(X+d)
90	RES:B 2,(X+d)	RES:C 2,(X+d)	RES:D 2,(X+d)	RES:E 2,(X+d)	RES:H 2,(X+d)	RES:L 2,(X+d)	RES 2,(X+d)	RES:A 2,(X+d)	RES:B 3,(X+d)	RES:C 3,(X+d)	RES:D 3,(X+d)	RES:E 3,(X+d)	RES:H 3,(X+d)	RES:L 3,(X+d)	RES 3,(X+d)	RES:A 3,(X+d)
A0	RES:B 4,(X+d)	RES:C 4,(X+d)	RES:D 4,(X+d)	RES:E 4,(X+d)	RES:H 4,(X+d)	RES:L 4,(X+d)	RES 4,(X+d)	RES:A 4,(X+d)	RES:B 5,(X+d)	RES:C 5,(X+d)	RES:D 5,(X+d)	RES:E 5,(X+d)	RES:H 5,(X+d)	RES:L 5,(X+d)	RES 5,(X+d)	RES:A 5,(X+d)
B0	RES:B 6,(X+d)	RES:C 6,(X+d)	RES:D 6,(X+d)	RES:E 6,(X+d)	RES:H 6,(X+d)	RES:L 6,(X+d)	RES 6,(X+d)	RES:A 6,(X+d)	RES:B 7,(X+d)	RES:C 7,(X+d)	RES:D 7,(X+d)	RES:E 7,(X+d)	RES:H 7,(X+d)	RES:L 7,(X+d)	RES 7,(X+d)	RES:A 7,(X+d)
C0	SET:B 0,(X+d)	SET:C 0,(X+d)	SET:D 0,(X+d)	SET:E 0,(X+d)	SET:H 0,(X+d)	SET:L 0,(X+d)	SET 0,(X+d)	SET:A 0,(X+d)	SET:B 1,(X+d)	SET:C 1,(X+d)	SET:D 1,(X+d)	SET:E 1,(X+d)	SET:H 1,(X+d)	SET:L 1,(X+d)	SET 1,(X+d)	SET:A 1,(X+d)
D0	SET:B 2,(X+d)	SET:C 2,(X+d)	SET:D 2,(X+d)	SET:E 2,(X+d)	SET:H 2,(X+d)	SET:L 2,(X+d)	SET 2,(X+d)	SET:A 2,(X+d)	SET:B 3,(X+d)	SET:C 3,(X+d)	SET:D 3,(X+d)	SET:E 3,(X+d)	SET:H 3,(X+d)	SET:L 3,(X+d)	SET 3,(X+d)	SET:A 3,(X+d)
E0	SET:B 4,(X+d)	SET:C 4,(X+d)	SET:D 4,(X+d)	SET:E 4,(X+d)	SET:H 4,(X+d)	SET:L 4,(X+d)	SET 4,(X+d)	SET:A 4,(X+d)	SET:B 5,(X+d)	SET:C 5,(X+d)	SET:D 5,(X+d)	SET:E 5,(X+d)	SET:H 5,(X+d)	SET:L 5,(X+d)	SET 5,(X+d)	SET:A 5,(X+d)
F0	SET:B 6,(X+d)	SET:C 6,(X+d)	SET:D 6,(X+d)	SET:E 6,(X+d)	SET:H 6,(X+d)	SET:L 6,(X+d)	SET 6,(X+d)	SET:A 6,(X+d)	SET:B 7,(X+d)	SET:C 7,(X+d)	SET:D 7,(X+d)	SET:E 7,(X+d)	SET:H 7,(X+d)	SET:L 7,(X+d)	SET 7,(X+d)	SET:A 7,(X+d)

Wszystkie rozkazy zaznaczone rejestrem poprzedzonym znakiem : są nie opublikowane. Wprowadzona notacja oznacza, że rozkaz wykonuje się na argumencie określonym w dolnej pozycji pola tablicy, a dodatkowo uzyskany wynik jest przesyłany do tak zaznaczonego rejestru. Jedynie rozkazy z kolumny 6 i E należą do grupy opublikowanych. X – rejestr IX

Tablica D.I1.7. Kody rozkazów poprzedzone bajtami FD i CB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	RLC:B (Y+d)	RLC:C (Y+d)	RLC:D (Y+d)	RLC:E (Y+d)	RLC:H (Y+d)	RLC:L (Y+d)	RLC (Y+d)	RLC:A (Y+d)	RRC:B (Y+d)	RRC:C (Y+d)	RRC:D (Y+d)	RRC:E (Y+d)	RRC:H (Y+d)	RRC:L (Y+d)	RRC (Y+d)	RRC:A (Y+d)
10	RL:B (Y+d)	RL:C (Y+d)	RL:D (Y+d)	RL:E (Y+d)	RL:H (Y+d)	RL:L (Y+d)	RL (Y+d)	RL:A (Y+d)	RR:B (Y+d)	RR:C (Y+d)	RR:D (Y+d)	RR:E (Y+d)	RR:H (Y+d)	RR:L (Y+d)	RR (Y+d)	RR:A (Y+d)
20	SLA:B (Y+d)	SLA:C (Y+d)	SLA:D (Y+d)	SLA:E (Y+d)	SLA:H (Y+d)	SLA:L (Y+d)	SLA (Y+d)	SLA:A (Y+d)	SRA:B (Y+d)	SRA:C (Y+d)	SRA:D (Y+d)	SRA:E (Y+d)	SRA:H (Y+d)	SRA:L (Y+d)	SRA (Y+d)	SRA:A (Y+d)
30	SLL:B (Y+d)	SLL:C (Y+d)	SLL:D (Y+d)	SLL:E (Y+d)	SLL:H (Y+d)	SLL:L (Y+d)	SLL (Y+d)	SLL:A (Y+d)	SRL:B (Y+d)	SRL:C (Y+d)	SRL:D (Y+d)	SRL:E (Y+d)	SRL:H (Y+d)	SRL:L (Y+d)	SRL (Y+d)	SRL:A (Y+d)
40	BIT:B 0,(Y+d)	BIT:C 0,(Y+d)	BIT:D 0,(Y+d)	BIT:E 0,(Y+d)	BIT:H 0,(Y+d)	BIT:L 0,(Y+d)	BIT 0,(Y+d)	BIT:A 0,(Y+d)	BIT:B 1,(Y+d)	BIT:C 1,(Y+d)	BIT:D 1,(Y+d)	BIT:E 1,(Y+d)	BIT:H 1,(Y+d)	BIT:L 1,(Y+d)	BIT 1,(Y+d)	BIT:A 1,(Y+d)
50	BIT:B 2,(Y+d)	BIT:C 2,(Y+d)	BIT:D 2,(Y+d)	BIT:E 2,(Y+d)	BIT:H 2,(Y+d)	BIT:L 2,(Y+d)	BIT 2,(Y+d)	BIT:A 2,(Y+d)	BIT:B 3,(Y+d)	BIT:C 3,(Y+d)	BIT:D 3,(Y+d)	BIT:E 3,(Y+d)	BIT:H 3,(Y+d)	BIT:L 3,(Y+d)	BIT 3,(Y+d)	BIT:A 3,(Y+d)
60	BIT:B 4,(Y+d)	BIT:C 4,(Y+d)	BIT:D 4,(Y+d)	BIT:E 4,(Y+d)	BIT:H 4,(Y+d)	BIT:L 4,(Y+d)	BIT 4,(Y+d)	BIT:A 4,(Y+d)	BIT:B 5,(Y+d)	BIT:C 5,(Y+d)	BIT:D 5,(Y+d)	BIT:E 5,(Y+d)	BIT:H 5,(Y+d)	BIT:L 5,(Y+d)	BIT 5,(Y+d)	BIT:A 5,(Y+d)
70	BIT:B 6,(Y+d)	BIT:C 6,(Y+d)	BIT:D 6,(Y+d)	BIT:E 6,(Y+d)	BIT:H 6,(Y+d)	BIT:L 6,(Y+d)	BIT 6,(Y+d)	BIT:A 6,(Y+d)	BIT:B 7,(Y+d)	BIT:C 7,(Y+d)	BIT:D 7,(Y+d)	BIT:E 7,(Y+d)	BIT:H 7,(Y+d)	BIT:L 7,(Y+d)	BIT 7,(Y+d)	BIT:A 7,(Y+d)
80	RES:B 0,(Y+d)	RES:C 0,(Y+d)	RES:D 0,(Y+d)	RES:E 0,(Y+d)	RES:H 0,(Y+d)	RES:L 0,(Y+d)	RES 0,(Y+d)	RES:A 0,(Y+d)	RES:B 1,(Y+d)	RES:C 1,(Y+d)	RES:D 1,(Y+d)	RES:E 1,(Y+d)	RES:H 1,(Y+d)	RES:L 1,(Y+d)	RES 1,(Y+d)	RES:A 1,(Y+d)
90	RES:B 2,(Y+d)	RES:C 2,(Y+d)	RES:D 2,(Y+d)	RES:E 2,(Y+d)	RES:H 2,(Y+d)	RES:L 2,(Y+d)	RES 2,(Y+d)	RES:A 2,(Y+d)	RES:B 3,(Y+d)	RES:C 3,(Y+d)	RES:D 3,(Y+d)	RES:E 3,(Y+d)	RES:H 3,(Y+d)	RES:L 3,(Y+d)	RES 3,(Y+d)	RES:A 3,(Y+d)
A0	RES:B 4,(Y+d)	RES:C 4,(Y+d)	RES:D 4,(Y+d)	RES:E 4,(Y+d)	RES:H 4,(Y+d)	RES:L 4,(Y+d)	RES 4,(Y+d)	RES:A 4,(Y+d)	RES:B 5,(Y+d)	RES:C 5,(Y+d)	RES:D 5,(Y+d)	RES:E 5,(Y+d)	RES:H 5,(Y+d)	RES:L 5,(Y+d)	RES 5,(Y+d)	RES:A 5,(Y+d)
B0	RES:B 6,(Y+d)	RES:C 6,(Y+d)	RES:D 6,(Y+d)	RES:E 6,(Y+d)	RES:H 6,(Y+d)	RES:L 6,(Y+d)	RES 6,(Y+d)	RES:A 6,(Y+d)	RES:B 7,(Y+d)	RES:C 7,(Y+d)	RES:D 7,(Y+d)	RES:E 7,(Y+d)	RES:H 7,(Y+d)	RES:L 7,(Y+d)	RES 7,(Y+d)	RES:A 7,(Y+d)
C0	SET:B 0,(Y+d)	SET:C 0,(Y+d)	SET:D 0,(Y+d)	SET:E 0,(Y+d)	SET:H 0,(Y+d)	SET:L 0,(Y+d)	SET 0,(Y+d)	SET:A 0,(Y+d)	SET:B 1,(Y+d)	SET:C 1,(Y+d)	SET:D 1,(Y+d)	SET:E 1,(Y+d)	SET:H 1,(Y+d)	SET:L 1,(Y+d)	SET 1,(Y+d)	SET:A 1,(Y+d)
D0	SET:B 2,(Y+d)	SET:C 2,(Y+d)	SET:D 2,(Y+d)	SET:E 2,(Y+d)	SET:H 2,(Y+d)	SET:L 2,(Y+d)	SET 2,(Y+d)	SET:A 2,(Y+d)	SET:B 3,(Y+d)	SET:C 3,(Y+d)	SET:D 3,(Y+d)	SET:E 3,(Y+d)	SET:H 3,(Y+d)	SET:L 3,(Y+d)	SET 3,(Y+d)	SET:A 3,(Y+d)
E0	SET:B 4,(Y+d)	SET:C 4,(Y+d)	SET:D 4,(Y+d)	SET:E 4,(Y+d)	SET:H 4,(Y+d)	SET:L 4,(Y+d)	SET 4,(Y+d)	SET:A 4,(Y+d)	SET:B 5,(Y+d)	SET:C 5,(Y+d)	SET:D 5,(Y+d)	SET:E 5,(Y+d)	SET:H 5,(Y+d)	SET:L 5,(Y+d)	SET 5,(Y+d)	SET:A 5,(Y+d)
F0	SET:B 6,(Y+d)	SET:C 6,(Y+d)	SET:D 6,(Y+d)	SET:E 6,(Y+d)	SET:H 6,(Y+d)	SET:L 6,(Y+d)	SET 6,(Y+d)	SET:A 6,(Y+d)	SET:B 7,(Y+d)	SET:C 7,(Y+d)	SET:D 7,(Y+d)	SET:E 7,(Y+d)	SET:H 7,(Y+d)	SET:L 7,(Y+d)	SET 7,(Y+d)	SET:A 7,(Y+d)

Wszystkie rozkazy zaznaczone rejestrem poprzedzonym znakiem : są nie opublikowane. Wprowadzona notacja oznacza, że rozkaz wykonuje się na argumencie określonym w dolnej pozycji pola tablicy, a dodatkowo uzyskany wynik jest przesyłany do tak zaznaczonego rejestru. Jedynie rozkazy z kolumny 6 i E należą do grupy opublikowanych. Y – rejestr IY

DODATEK III.
TABELE KONWERSJI LICZB W ROZNYCH SYSTEMACH LICZENIA

Tablica D.III.1. Interpretacja zawartości 8-bitowej komórki w różnych systemach liczenia i znakowo

dziesiętnie bez znaku	szesnastkowo	dziesiętnie ze znakiem	binarnie	znakowo w kodzie ASCII
0	00	0	00000000	NUL
1	01	1	00000001	SOH
2	02	2	00000010	STX
3	03	3	00000011	ETX
4	04	4	00000100	EOT
5	05	5	00000101	ENQ
6	06	6	00000110	ACK
7	07	7	00000111	BEL
8	08	8	00001000	BS
9	09	9	00001001	HT
10	0A	10	00001010	LF
11	0B	11	00001011	VT
12	0C	12	00001100	FF
13	0D	13	00001101	CR
14	0E	14	00001110	SO
15	0F	15	00001111	SI
16	10	16	00010000	DLE
17	11	17	00010001	DC1
18	12	18	00010010	DC2
19	13	19	00010011	DC3
20	14	20	00010100	DC4
21	15	21	00010101	NAK
22	16	22	00010110	SYN
23	17	23	00010111	ETB
24	18	24	00011000	CAN
25	19	25	00011001	EM
26	1A	26	00011010	SUB
27	1B	27	00011011	ESC
28	1C	28	00011100	FS
29	1D	29	00011101	GS
30	1E	30	00011110	RS
31	1F	31	00011111	US
32	20	32	00100000	spacja
33	21	33	00100001	!
34	22	34	00100010	"
35	23	35	00100011	#
36	24	36	00100100	$
37	25	37	00100101	%
38	26	38	00100110	&
39	27	39	00100111	'
40	28	40	00101000	(
41	29	41	00101001	)
42	2A	42	00101010	*
43	2B	43	00101011	+
44	2C	44	00101100	,
45	2D	45	00101101	-
46	2E	46	00101110	.
47	2F	47	00101111	/
48	30	48	00110000	0
49	31	49	00110001	1
50	32	50	00110010	2
51	33	51	00110011	3
52	34	52	00110100	4
53	35	53	00110101	5
54	36	54	00110110	6
55	37	55	00110111	7
56	38	56	00111000	8
57	39	57	00111001	9
58	3A	58	00111010	:
59	3B	59	00111011	;
60	3C	60	00111100	<
61	3D	61	00111101	=
62	3E	62	00111110	>
63	3F	63	00111111	?
64	40	64	01000000	@
65	41	65	01000001	A
66	42	66	01000010	B
67	43	67	01000011	C
68	44	68	01000100	D
69	45	69	01000101	E
70	46	70	01000110	F
71	47	71	01000111	G
72	48	72	01001000	H
73	49	73	01001001	I
74	4A	74	01001010	J
75	4B	75	01001011	K
76	4C	76	01001100	L
77	4D	77	01001101	M
78	4E	78	01001110	N
79	4F	79	01001111	O
80	50	80	01010000	P
81	51	81	01010001	Q
82	52	82	01010010	R
83	53	83	01010011	S
84	54	84	01010100	T
85	55	85	01010101	U
86	56	86	01010110	V
87	57	87	01010111	W
88	58	88	01011000	X
89	59	89	01011001	Y
90	5A	90	01011010	Z
91	5B	91	01011011	[
92	5C	92	01011100	\
93	5D	93	01011101	]
94	5E	94	01011110	^
95	5F	95	01011111	_
96	60	96	01100000	`
97	61	97	01100001	a
98	62	98	01100010	b
99	63	99	01100011	c
100	64	100	01100100	d
101	65	101	01100101	e
102	66	102	01100110	f
103	67	103	01100111	g
104	68	104	01101000	h
105	69	105	01101001	i
106	6A	106	01101010	j
107	6B	107	01101011	k
108	6C	108	01101100	l
109	6D	109	01101101	m
110	6E	110	01101110	n
111	6F	111	01101111	o
112	70	112	01110000	p
113	71	113	01110001	q
114	72	114	01110010	r
115	73	115	01110011	s
116	74	116	01110100	t
117	75	117	01110101	u
118	76	118	01110110	v
119	77	119	01110111	w
120	78	120	01111000	x
121	79	121	01111001	y
122	7A	122	01111010	z
123	7B	123	01111011	{
124	7C	124	01111100	|
125	7D	125	01111101	}
126	7E	126	01111110	~
127	7F	127	01111111	DEL
128	80	-128	10000000	
129	81	-127	10000001	
130	82	-126	10000010	
131	83	-125	10000011	
132	84	-124	10000100	
133	85	-123	10000101	
134	86	-122	10000110	
135	87	-121	10000111	
136	88	-120	10001000	

dziesiętnie bez znaku	szesnastkowo	dziesiętnie ze znakiem	binarnie	znakowo w kodzie ASCII
137	89	-119	10001001	
138	8A	-118	10001010	
139	8B	-117	10001011	
140	8C	-116	10001100	
141	8D	-115	10001101	
142	8E	-114	10001110	
143	8F	-113	10001111	
144	90	-112	10010000	
145	91	-111	10010001	
146	92	-110	10010010	
147	93	-109	10010011	
148	94	-108	10010100	
149	95	-107	10010101	
150	96	-106	10010110	
151	97	-105	10010111	
152	98	-104	10011000	
153	99	-103	10011001	
154	9A	-102	10011010	
155	9B	-101	10011011	
156	9C	-100	10011100	
157	9D	-99	10011101	
158	9E	-98	10011110	
159	9F	-97	10011111	
160	A0	-96	10100000	
161	A1	-95	10100001	
162	A2	-94	10100010	
163	A3	-93	10100011	
164	A4	-92	10100100	
165	A5	-91	10100101	
166	A6	-90	10100110	
167	A7	-89	10100111	
168	A8	-88	10101000	
169	A9	-87	10101001	
170	AA	-86	10101010	
171	AB	-85	10101011	
172	AC	-84	10101100	
173	AD	-83	10101101	
174	AE	-82	10101110	
175	AF	-81	10101111	
176	B0	-80	10110000	
177	B1	-79	10110001	
178	B2	-78	10110010	
179	B3	-77	10110011	
180	B4	-76	10110100	
181	B5	-75	10110101	
182	B6	-74	10110110	
183	B7	-73	10110111	
184	B8	-72	10111000	
185	B9	-71	10111001	
186	BA	-70	10111010	
187	BB	-69	10111011	
188	BC	-68	10111100	
189	BD	-67	10111101	
190	BE	-66	10111110	
191	BF	-65	10111111	
192	C0	-64	11000000	
193	C1	-63	11000001	
194	C2	-62	11000010	
195	C3	-61	11000011	
196	C4	-60	11000100	
197	C5	-59	11000101	
198	C6	-58	11000110	
199	C7	-57	11000111	
200	C8	-56	11001000	
201	C9	-55	11001001	
202	CA	-54	11001010	
203	CB	-53	11001011	
204	CC	-52	11001100	
205	CD	-51	11001101	
206	CE	-50	11001110	
207	CF	-49	11001111	
208	D0	-48	11010000	
209	D1	-47	11010001	
210	D2	-46	11010010	
211	D3	-45	11010011	
212	D4	-44	11010100	
213	D5	-43	11010101	
214	D6	-42	11010110	
215	D7	-41	11010111	
216	D8	-40	11011000	
217	D9	-39	11011001	
218	DA	-38	11011010	
219	DB	-37	11011011	
220	DC	-36	11011100	
221	DD	-35	11011101	
222	DE	-34	11011110	
223	DF	-33	11011111	
224	E0	-32	11100000	
225	E1	-31	11100001	
226	E2	-30	11100010	
227	E3	-29	11100011	
228	E4	-28	11100100	
229	E5	-27	11100101	
230	E6	-26	11100110	
231	E7	-25	11100111	
232	E8	-24	11101000	
233	E9	-23	11101001	
234	EA	-22	11101010	
235	EB	-21	11101011	
236	EC	-20	11101100	
237	ED	-19	11101101	
238	EE	-18	11101110	
239	EF	-17	11101111	
240	F0	-16	11110000	
241	F1	-15	11110001	
242	F2	-14	11110010	
243	F3	-13	11110011	
244	F4	-12	11110100	
245	F5	-11	11110101	
246	F6	-10	11110110	
247	F7	-9	11110111	
248	F8	-8	11111000	
249	F9	-7	11111001	
250	FA	-6	11111010	
251	FB	-5	11111011	
252	FC	-4	11111100	
253	FD	-3	11111101	
254	FE	-2	11111110	
255	FF	-1	11111111	

Tablica D.III.2. Konwersja liczb 1-bajtowych w systemie dziesiętnym (bez znaku) i szesnastkowym

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	80	81	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
60	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
80	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
[illegible]	[illegible]	[illegible]	210	211	212	213	214	215	216	217	218	219	220	221	222	223

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Liczby dziesiętne od 0 do 255 zamieszczone w tablicy są identyfikowane szesnastkowymi numerami wierszy (2-cyfrowe liczby od 00 do F0) i kolumn (1-cyfrowe liczby od 0 do F).

Wyznaczenie wartości szesnastkowej wybranej liczby dziesiętnej polega na szesnastkowym dodaniu wartości określonego numeru wiersza do kolumny. Przykładowo liczba dziesiętna 171, zapisana w wierszu A0 i kolumnie B odpowiada liczbie AB (A0 + B) w systemie szesnastkowym.

W wypadku liczby szesnastkowej, jej postać dziesiętną znajduje się w kolumnie wyznaczonej przez mniej znaczącą cyfrę tej liczby i wierszu wyznaczonym przez odjęcie od tej liczby mniej znaczącej cyfry. Dla przykładu szesnastkowej liczbie 9A odpowiada liczba dziesiętna z kolumny A i wiersza 9A − A = 90, czyli 154.

LITERATURA

[1] Grabowski J., Koślacz S.: Podstawy i praktyka programowania mikroprocesorów. Warszawa, WNT 1980.
[2] Lipowski J., Małysiak H. i in.: Modułowe systemy mikrokomputerowe. Warszawa, WNT 1984.
[3] Misiurewicz P.: Układy mikroprocesorowe. Struktury i programowanie. Warszawa, WNT 1983.
[4] Sacha K., Rydzewski A.: Mikroprocesor w pytaniach i odpowiedziach. Warszawa, WNT 1985.
[5] Knuth D.: The art of computer programming. Fundamental algorithms. Vol. 1, Addison-Wesley, 1968.
[6] Knuth D.: The art of computer programming. Searching and sorting. Vol. 3, Addison-Wesley, 1973.
[7] Leventhal L., Saville W., : Z80 assembly language subroutines. Osborne/McGraw-Hill, 1983.

Tablica D.III.3. Konwersja liczb 2-bajtowych w systemie dziesiętnym (bez znaku) i szesnastkowym

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0 0	1 256	2 512	3 768	4 1024	5 1280	6 1536	7 1792	8 2048	9 2304	10 2560	11 2816	12 3072	13 3328	14 3584	15 3840
10	16 4096	17 4352	18 4608	19 4864	20 5120	21 5376	22 5632	23 5888	24 6144	25 6400	26 6656	27 6912	28 7168	29 7424	30 7680	31 7936
20	32 8192	33 8448	34 8704	35 8960	36 9216	37 9472	38 9728	39 9984	40 10240	41 10496	42 10752	43 11008	44 11264	45 11520	46 11776	47 12032
30	48 12288	49 12544	50 12800	51 13056	52 13312	53 13568	54 13824	55 14080	56 14336	57 14592	58 14848	59 15104	60 15360	61 15616	62 15872	63 16128
40	64 16384	65 16640	66 16896	67 17152	68 17408	69 17664	70 17920	71 18176	72 18432	73 18688	74 18944	75 19200	76 19456	77 19712	78 19968	79 20224
50	80 20480	81 20736	82 20992	83 21248	84 21504	85 21760	86 22016	87 22272	88 22528	89 22784	90 23040	91 23296	92 23552	93 23808	94 24064	95 24320
60	96 24576	97 24832	98 25088	99 25344	100 25600	101 25856	102 26112	103 26368	104 26624	105 26880	106 27136	107 27392	108 27648	109 27904	110 28160	111 28416
70	112 28672	113 28928	114 29184	115 29440	116 29696	117 29952	118 30208	119 30464	120 30720	121 30976	122 31232	123 31488	124 31744	125 32000	126 32256	127 32512
80	128 32768	129 33024	130 33280	131 33536	132 33792	133 34048	134 34304	135 34560	136 34816	137 35072	138 35328	139 35584	140 35840	141 36096	142 36352	143 36608
90	144 36864	145 37120	146 37376	147 37632	148 37888	149 38144	150 38400	151 38656	152 38912	153 39168	154 39424	155 39680	156 39936	157 40192	158 40448	159 40704
A0	160 40960	161 41216	162 41472	163 41728	164 41984	165 42240	166 42496	167 42752	168 43008	169 43264	170 43520	171 43776	172 44032	173 44288	174 44544	175 44800
B0	176 45056	177 45312	178 45568	179 45824	180 46080	181 46336	182 46592	183 46848	184 47104	185 47360	186 47616	187 47872	188 48128	189 48384	190 48640	191 48896
C0	192 49152	193 49408	194 49664	195 49920	196 50176	197 50432	198 50688	199 50944	200 51200	201 51456	202 51712	203 51968	204 52224	205 52480	206 52736	207 52992
D0	208 53248	209 53504	210 53760	211 54016	212 54272	213 54528	214 54784	215 55040	216 55296	217 55552	218 55808	219 56064	220 56320	221 56576	222 56832	223 57088
E0	224 57344	225 57600	226 57856	227 58112	228 58368	229 58624	230 58880	231 59136	232 59392	233 59648	234 59904	235 60160	236 60416	237 60672	238 60928	239 61184
F0	240 61440	241 61696	242 61952	243 62208	244 62464	245 62720	246 62976	247 63232	248 63488	249 63744	250 64000	251 64256	252 64512	253 64768	254 65024	255 65280

Oznaczenie wierszy i kolumn oraz górne liczby dziesiętne w poszczególnych polach są identyczne jak w tablicy D.III.2. Dolne liczby dziesiętne powstały w wyniku mnożenia liczb górnych przez 256 (256*0, 256*1, 256*2,...256*255)

Konwersja 2-bajtowej (4-cyfrowej) liczby szesnastkowej na postać dziesiętną polega na dodaniu wartości dziesiętnych dwóch bardziej znaczących cyfr (dolna liczba odpowiedniego pola) i dwóch mniej znaczących cyfr (górna liczba odpowiedniego pola). Przykładowo, wartość liczby szesnastkowej ABCD w systemie dziesiętnym wynosi 43776 + 205 = 43891.

Konwersja liczby dziesiętnej na postać szesnastkową wymaga operacji odejmowania. W pierwszym kroku należy znaleźć wsród dolnych pozycji poszczególnych pól tabeli maksymalną liczbę, nieprzekraczającą wartości danej liczby. Kolumna i wiersz wyznaczają postać szesnastkową dwóch bardziej znaczących cyfr. Znalezioną liczbę dziesiętną odejmujemy od danej liczby, a wynik posłuży do odczytu dwóch mniej znaczących cyfr szesnastkowych.

Dla przykładu wyznaczymy wartość szesnastkową liczby dziesiętnej 45778. Maksymalna liczba, nieprzekraczająca wartości 45778 zapisana w wierszu B0 i kolumnie 2 (B0 + 2 = B2) wynosi 45568. Obliczamy różnicę: 45778 − 45568 = 210. Liczba 210 zapisana jest w wierszu B0 i kolumnie 2 (D0 + 2 = D2). Ostatecznie liczbie dziesiętnej 45778 odpowiada liczba B2D2 w systemie szesnastkowym.

W.Z.Graf. W-wa Zam. 9343 K-97

INFORMATYKA
KOMPUTERY
SYSTEMY

Dodatek „Żołnierza Wolności" Cena 150 zł

zeszyty programów komputerowych nr 4

ATARI



SPIS TREŚCI

Atari Basic w przykładach 3
Wyprowadzanie informacji na ekran 4
Sterowanie kursorem 5
Programowanie wejścia 6
Maskowanie wejścia 9
Sprawdzanie odpowiedzi 9
Wprowadzanie odpowiedzi 9
Instrukcja TRAP 11
Procedura obsługi błędu 11
Użyteczne podprogramy wprowadzania 12
Wprowadzanie danych grupowych 12
Przeglądanie i zmienianie wprowadzonych danych 13
Używanie minipulatorów do ograniczenia odpowiedzi 13
Sterowanie obrazem za pomocą manipulatora 13
Wprowadzanie liczb manipulatorem 15
Wykorzystanie menu do ograniczenia wyborów 16
Używanie manipulatora do wyboru z menu 16
Wprowadzanie znaków manipulatorem 17
Wprowadzanie daty 17
Wprowadzanie ciągu znaków 17
Działania na zmiennych tekstowych 17
Unieszkodliwienie klawisza BREAK 19
Asembler 19
Konwerter języka maszynowego 20
Disasembler 22
Budowa zamku 22
Atari 24
Dynamiczne bajty 25
Prywatny język programowania 27
Katalog gier 29

„IKS" — dodatek „Żołnierza Wolności". Redaguje: Wiesław Cetera (kierownik zespołu); stali współpracownicy: Włodzimierz Gogołek, Janusz Janiec, Krzysztof Mamcarz, Ireneusz Miernik, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-60 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem Wydawnictwa „Czasopisma Wojskowe". Nr zam. 9385. Nr ind. 390313 K-69

Ten zeszyt dedykujemy wszystkim tym, którzy postanowili programować samodzielnie. Wszystkie przykłady oparte zostały o dialekt Basica zastosowany w mikrokomputerach Atari. Jest ich w Polsce najwięcej. Łatwo też można zdobyć ich oprogramowanie. W koleżeński obieg ruszyły nie tylko gry, ale też i programy użytkowe. Czy zatem nasz zeszyt jest potrzebny?

Nie mamy wątpliwości — dobrej literatury informatycznej nadal brakuje. Oczywiście nie każdy musi umieć programować, nie każdy przecież zostanie informatykiem, ale wierzymy, że każdy kto chce bliżej poznać swój komputer sięgnie po niniejszy zeszyt.

Wprowadzanie długich programów zniechęca. Wiemy o tym, toteż tasiemcowych wydruków w zeszycie nie ma. Poza tym sumy kontrolne skutecznie zabezpieczają przed popełnieniem błędów podczas wprowadzania programu do pamięci komputera.

Od chwili informatycznej „eksplozji" w Polsce wysłuchaliśmy już wielu sporów na temat ułomności niektórych typów mikrokomputerów, krytykowano Juniora, Mazovię, Kraka 86; wiele czasu stracono na udowadnianie wyższości polskiego Logo nad **BASIC**-iem. Tymczasem czas upływał. Jak to często bywa dobre pomysły utopiono w potoku akademickich dyskusji.

Każdy język programowania ma swoje zalety, wyrabia dobre, bądź złe nawyki u programisty. BASIC obok swoich niewątpliwych wad ma tę wielką zaletę, że jest. W jego interpretery wyposażone są wszystkie występujące w kraju mikrokomputery. I czy ktoś chce czy nie to właśnie BASIC jest dla posiadaczy mikrokomputerów pierwszym językiem, w którym porozumiewają się ze swoim urządzeniem. Dlatego czwarty zeszyt programów to właśnie BASIC — teraz można krytykować.

Na końcu zeszytu zamieściliśmy również kilka programów dla posiadaczy starszego typu mikrokomputera — ZX Spectrum. Cierpliwość przy wprowadzaniu z pewnością będzie nagrodzona atrakcyjnością programu.

Życzymy powodzenia.

Aby wprowadzić programy do pamięci komputera, radzimy wykorzystać Edytor BASIC'u, zamieszczony w nr. 4 „IKS-a" z bieżącego roku. Pamiętajmy: dwa znaki z lewej strony numeru linii nie są częścią programu, lecz sumą kontrolną danej linii, dlatego nie wprowadzamy ich do komputera.

ATARI BASIC w przykładach

TOMASZ MROWIEC
LUDWIK PIELA

Po otrzymaniu propozycji przygotowania numeru specjalnego „IKS", w części poświęconego komputerom Atari, byliśmy w rozterce; co zaproponować czytelnikom, aby mogli jak najwięcej skorzystać? Wiedzieliśmy doskonale, czego nie chcemy umieścić, chociaż kosztowałoby to nas najmniej wysiłku. Spędzamy dużo czasu przy komputerach i wiemy, jak żmudną i męczącą czynnością jest „wklepywanie" programów, szczególnie wtedy, gdy nie bardzo wiemy, co mają robić. Dlatego odrzuciliśmy wszelkie propozycje umieszczania długich programów. Nie chcieliśmy również dostarczać gier polegających na zabiciu przeciwnika lub uniknięciu unicestwienia. Wystarczająca ich podaż jest we wszelkiego typu „wypożyczalniach" i na giełdach.

Jednocześnie zauważyliśmy, że wielu programom, realizującym nieraz bardzo interesujące pomysły, brakuje ostatecznego wykończenia, są mało przejrzyste i trudne dla ewentualnych modyfikacji. Dlatego zdecydowaliśmy się zaprezentować czytelnikom, szczególnie tym, którzy stawiają pierwsze kroki w programowaniu, a także bardziej zaawansowanym pewne ciekawe procedury, które mogą wykorzystać w swoich programach. Nie są to żadne „sztuczki" programowe, które są często efektowne, lecz mało czytelne. **Prezentowane moduły programowe, bo tak je można nazwać, są jasne i zrozumiałe dla każdego znającego nieco BASIC.** Jeśli chociaż niektórzy z was uznają je za przydatne i wykorzystają w tworzonych przez siebie programach, będzie to dla was wystarczającą satysfakcją.

Dodatkowo prezentujemy kilka programów narzędziowych, które przeznaczone są dla bardziej zaawansowanych programistów oraz przykłady gry edukacyjnej dla najmłodszych.

Najbardziej niedoświadczeni programiści szybko odkrywają, że sekcje wejścia i wyjścia programu są jego najtrudniejszymi częściami.

Prawie każdy program wymaga danych, które muszą być wprowadzane z klawiatury. Czy wystarczy kilka instrukcji INPUT? W większości wypadków nie. A co będzie, jeśli użytkownik przypadkiem naciśnie zły klawisz? Lub gorzej, co będzie, jeśli stwierdzi, po wprowadzeniu dwóch lub trzech pozycji danych, że są one niewłaściwe? Dobry programista musi przewidzieć, że użytkownik jego programu jest człowiekiem i popełnia nieprzewidziane, ludzkie błędy.

Podobna sytuacja jest z wynikami. Nie mogą one być po prostu wyświetlane lub drukowane przez grupę przypadkowych instrukcji PRINT. Będą przecież czytane przez człowieka. Dopóki ich postać nie jest starannie zaprojektowana, mogą być bardzo trudne do odczytania. W konsekwencji informacja może być źle odczytana lub całkowicie niewidoczna.

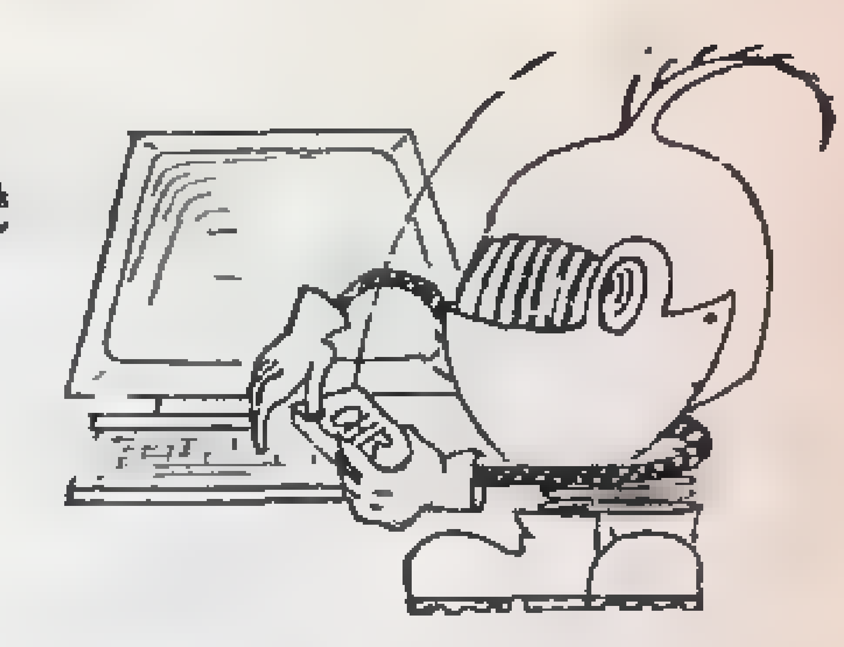
```
QO 1 REM ****************
ZR 2 REM * Program nr 1  *
QQ 3 REM ****************
TW 10 PRINT "Podaj cene za litr "
 ;
NK 20 INPUT C1L
HQ 30 PRINT "Ile litrow na 100 km
 . ";
CK 40 INPUT LKM
HQ 50 PRINT " Km."," LITRY","CENA
 "
SS 60 PRINT "-----","-------","--
 ---"
TL 70 FOR KM=100 TO 1700 STEP 100
OU 80 LITR=KM*LKM/100
TR 90 CENA=INT(C1L*LITR*100)/100
NN 100 PRINT KM,LITR,CENA
NC 110 NEXT KM
TG 120 PRINT
VV 130 PRINT "L/100km =";LKM,C1L;
 " zl. za litr"
NY 140 END
```

```
QO 1 REM ****************
AL 2 REM * Program nr 2  *
QQ 3 REM ****************
QL 5 DIM N$(10),T$(10),BL$(40)
KF 6 REM Wypelnienie BL$ spacjami
AH 7 BL$=" ":BL$(40)=BL$:BL$(2)=B
 L$
TW 10 PRINT "Podaj cene za litr "
 ;
NK 20 INPUT C1L
HQ 30 PRINT "Ile litrow na 100 km
 . ";
CK 40 INPUT LKM
HQ 50 PRINT " Km."," LITRY","CENA
 "
SS 60 PRINT "-----","-------","--
 ---"
TL 70 FOR KM=100 TO 1700 STEP 100
OU 80 LITR=KM*LKM/100
TR 90 CENA=INT(C1L*LITR*100)/100
SJ 100 NS=5:N=KM:GOSUB 11000:PRIN
 T N$(1,NS),
GR 102 NS=7:N=LITR:GOSUB 11000:PR
 INT N$(1,NS),
SI 104 NS=5:N=CENA:GOSUB 11000:PR
 INT N$(1,NS)
NC 110 NEXT KM
TG 120 PRINT
VV 130 PRINT "L/100km =";LKM,C1L;
 " zl. za litr"
NY 140 END
IK 10000 REM *****************
YT 10001 REM *   Podprogram   *
HY 10002 REM *wyrownania liczb*
KY 10003 REM *    w prawo     *
JE 10004 REM *****************
TC 11000 T$=STR$(N)
GS 11010 N$=BL$
TV 11020 N$(NS-LEN(T$)+1,NS)=T$
DL 11030 RETURN
```

```
QO 1 REM ****************
BF 2 REM * Program nr 3  *
QQ 3 REM ****************
QL 5 DIM N$(10),T$(10),BL$(40)
KF 6 REM Wypelnienie BL$ spacjami
AH 7 BL$=" ":BL$(40)=BL$:BL$(2)=B
   L$
TW 10 PRINT "Podaj cene za litr "
   ;
NK 20 INPUT C1L
HQ 30 PRINT "Ile litrow na 100 km
   .  ";
CK 40 INPUT LKM
VK 50 PRINT " Km.","  LITRY",". CE
   NA"
KT 60 PRINT "-----","-------","--
   -----"
TL 70 FOR KM=100 TO 1700 STEP 100
OU 80 LITR=KM*LKM/100
TR 90 CENA=INT(C1L*LITR*100)/100
TC 100 NS=6:DD=0:N=KM:GOSUB 11000
   :PRINT N$(1,NS),
NL 102 NS=7:DD=2:N=LITR:GOSUB 110
   00:PRINT N$(1,NS),
JV 104 NS=7:DD=1:N=CENA:GOSUB 110
   00:PRINT N$(1,NS)
NC 116 NEXT KM
TG 120 PRINT
VV 130 PRINT "L/100km =";LKM,C1L;
   " zl. za litr"
NY 140 END
IK 10000 REM *****************
YT 10001 REM *   Podprogram    *
HY 10002 REM *wyrownania liczb*
NY 10003 REM *   w/g kropki    *
GT 10004 REM *   dziesietnej   *
JJ 10005 REM *****************
TC 11000 T$=STR$(N)
GS 11010 N$=BL$
DK 11030 DP=LEN(T$)+1
YZ 11040 FOR J=1 TO LEN(T$)
QY 11050 IF T$(J,J)="." THEN DP=J
   :J=NS
GV 11060 NEXT J
SG 11070 NL=DP+DD
GH 11080 N$(NS-NL+1,NS)=T$
EJ 11090 RETURN
```

```
FK 1 REM ***************
ZH 2 REM * Program nr 4 *
FM 3 REM ***************
OY 10 DIM R$(1),CU$(24),CR$(40)
QL 20 CU$=CHR$(28):CU$(24)=CU$:CU
   $(2)=CU$
YB 30 CR$=CHR$(31):CR$(40)=CR$:CR
   $(2)=CR$
YW 40 PRINT CHR$(125);"Przyszla w
   artosc wkladu w P K O"
VG 50 PRINT :PRINT
ZE 60 PRINT "Wklad pierwotny"
OR 70 PRINT "Stopa oprocentowania
   "
VP 80 PRINT "Ile lat";CU$(1,3)
ZU 100 PRINT CR$(1,15);:INPUT AMT
SI 110 PRINT CR$(1,20);:INPUT IR
CK 120 PRINT CR$(1,7);:INPUT YR
AF 130 CMP=1
OP 140 IP=IR/CMP/100
EB 150 FV=AMT*(1+IP)^(CMP*YR)
QY 160 PRINT "Przyszla wartosc: "
   ;INT(FV*100+0.5)/100;"       "
TQ 170 PRINT
UH 180 PRINT "Czy zmieniasz wklad
   ";
HE 190 INPUT R$
KJ 200 IF R$="T" THEN PRINT CU$(1
   ,6);:GOTO 100
NT 210 END
```

Wyprowadzenie informacji na ekran



Przedstawienie informacji na ekranie powinno być tak zorganizowane, aby była ona łatwa do zrozumienia lub przyjemna dla oczu. Podstawowym narzędziem wyświetlania informacji jest instrukcja **PRINT**.

Rozważmy wprowadzanie na ekran informacji przedstawionych w postaci tabel. Zwykle dużo łatwiej jest przeglądać wykaz pozycji, gdy są one zestawione w kolumny. Dotyczy to zarówno liczb, jak i danych tekstowych. Atari **BASIC** dysponuje dwoma sposobami wyprowadzania informacji w postaci kolumn. Jednym jest użycie przecinków między wartościami w instrukcjach **PRINT**. Drugim jest użycie klawisza **TAB**.

Jeśli komputer natrafi na przecinek po wartości (zmiennej) instrukcji **PRINT**, przesuwa kursor w prawo. Wypełnia spacjami obszar między końcem wyświetlanej wartości i początkiem następnej kolumny. Druga kolumna zaczyna się o dziesięć odstępów od lewego marginesu, dodatkowe kolumny występują co każde dziesięć odstępów. **PROGRAM NR 1 używa dwóch z trzech dostępnych kolumn**.

Używając przecinków należy zachować ostrożność. Dwie pozycje przed końcem kolumny muszą być puste. W przeciwnym razie następuje przejście do początku następnej kolumny. Możliwa jest zmiana szerokości kolumn za pomocą **POKE 201,*n***, gdzie n jest nową szerokością kolumn.

Oba sposoby wyprowadzania informacji kolumnami mają jedną wadę. Wyrównują one wartości do lewego brzegu kolumny. Takie wyrównanie do lewego marginesu jest wygodne dla słów i innych wartości alfabetycznych. Liczby są łatwiejsze do odczytania, jeśli są wyrównane w prawo. **PROGRAM NR 2 zawiera podprogram wyrównania liczb do prawego marginesu**. Podprogram ten wymaga indywidualnego dostępu do każdej cyfry liczby, która ma być wyrównana. BASIC umożliwia taki dostęp tylko w zmiennych tekstowych, zatem podprogram tłumaczy liczbę na ciąg znaków (linia 11000). Następnie wypełnia ciąg wyjściowy spacjami (linia 11010). Gwarantuje to stałą wartość w częściach ciągu, które nie kończą się cyfrą. W końcu wyrównuje liczbę w prawo (linia 11020). Sprawdza długość liczby i określa jak blisko prawego brzegu kolumny musi się rozpocząć, aby mogła być umieszczona właściwie.

Program ten działa prawidłowo dla liczb całkowitych. Kolumny liczb z kropkami dziesiętnymi byłyby łatwiejsze do odczytania, gdyby liczby były wyrównane według kropki dziesiętnej. W tym celu musimy określić, gdzie

```
FK 1 REM ****************
AB 2 REM * Program nr 5 *
FM 3 REM ****************
RN 9 REM USTAWIENIE LEWEGO MARGIN
   ESU
FX 10 POKE 82,0
FB 19 REM USTAWIENIE PRAWEGO MARG
   INESU
GF 20 POKE 83,31
XQ 30 GRAPHICS 0:DS=PEEK(88)+PEEK
   (89)*256
DT 35 REM ZMIANA PROGRAMU WYSWIET
   LANIA
NQ 40 DL=1539
WR 50 FOR I=1536 TO 1538
NE 60 POKE I,112:NEXT I
JI 70 FOR I=1 TO 24
YC 80 POKE DL,66
FX 90 POKE DL+2,INT(DS/256):POKE
   DL+1,DS-PEEK(DL+2)*256:DS=DS+4
   0
SL 100 DL=DL+3
FT 110 NEXT I
FM 115 REM USTAWIENIE WASKIEGO TL
   A
QI 120 POKE 559,33
EA 130 POKE DL+2,6:POKE DL,65:POK
   E DL+1,0
ZK 140 POKE 560,0:POKE 561,6
```

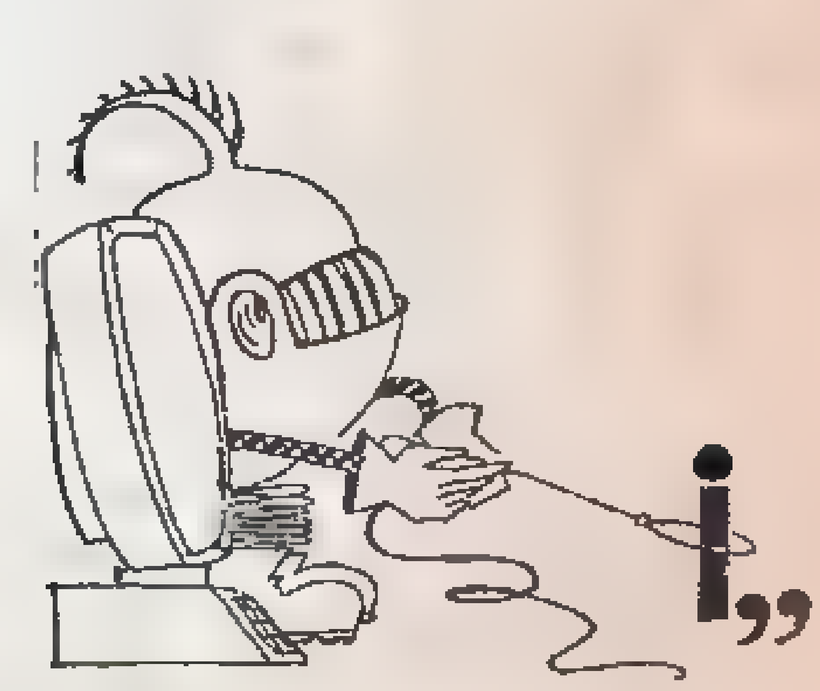
```
FK 1 REM ****************
AV 2 REM * Program nr 6 *
FM 3 REM ****************
HH 10 DIM BL$(40),CMD$(1),R$(1)
OS 50 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
OS 300 AMPSUB=2000
TW 310 PRINT CHR$(125):REM KASOWA
   NIE EKRANU
HT 320 POSITION 2,1
XG 330 PRINT "POLECENIA (ACDEP, H
   =Pomoc)";
UV 340 INPUT CMD$
HL 360 POSITION 2,0
XJ 370 PRINT "Mam wykonac polecen
   ie ";
RY 380 ON ASC(CMD$)-64 GOSUB 600,
   31767,800,1000,1200,31767,3176
   7,1400
XS 390 IF CMD$<>"P" THEN 310
WB 400 PRINT "DRUKUJ":POP :GOSUB
   31766:GOTO 310
FA 600 PRINT "DODAJ ":GOSUB 31766
   :RETURN
VS 800 PRINT "ZMIEN ":GOSUB 31766
   :RETURN
JJ 1000 PRINT "KASUJ ":GOSUB 3176
   6:RETURN
XY 1200 PRINT "KONIEC":POSITION 2
   ,10:END
DL 1400 PRINT "POMOC"
XG 1410 GOSUB 2000
ZT 1420 POP :GOTO 320
YJ 2000 GOSUB 2300
KQ 2010 PRINT "______Wykaz komen
   d______"
BX 2030 PRINT " A=dodaj  C=zmien
   D=kasuj"
MZ 2040 PRINT " E=koniec F=drukuj
   H=pomoc";
AP 2050 RETURN
IU 2300 POSITION 2,19
RG 2310 FOR J=20 TO 23
SK 2330 PRINT BL$(1,38);
FT 2340 NEXT J
DU 2350 POSITION 2,20
AY 2360 RETURN
CJ 31766 FOR I=0 TO 400:NEXT I:RE
   TURN
GD 31767 RETURN
```

umieścić kropkę dziesiętną w każdej kolumnie. Następnie trzeba sprawdzić, gdzie jest kropka w każdej liczbie. Nie jest to zadanie trywialne, ponieważ BASIC używa liczb zmiennoprzecinkowych. Kropka dziesiętna może być gdziekolwiek. Po znalezieniu musimy przesunąć ją w prawo lub w lewo. Może to oznaczać gubienie dodatkowych cyfr z prawej strony lub wypełnianie dodatkowymi spacjami.

Nowy podprogram (z PROGRAMU NR 3) **zawiera wszystkie wymagania poprzedniego plus dodatkowe.** Wstępnie zakłada, że nie ma kropki dziesiętnej i za pomocą pętli **FOR-NEXT** poszukuje kropki w ciągu numerycznym. W wyniku działania tego podprogramu otrzymujemy kolumny liczb łatwe do przeglądania. Zauważmy, że BASIC nie drukuje kropek dla liczb całkowitych. Nie drukuje też zer końcowych liczb, które nie zmieniają ich wartości.

Istnieje niebezpieczeństwo, że liczba będzie zbyt duża i nie zmieści się w wydzielonej kolumnie. Możliwe jest wstawienie do programu lub podprogramu instrukcji sprawdzających wielkość liczby i ustawiających odpowiednią szerokość kolumny w celu dostosowania jej do największych możliwych wartości.

Sterowanie kursorem



Atari BASIC umożliwia bezpośrednie sterowanie kursorem dwoma sposobami. Jednym jest programowanie ruchu kursora znakami, przy użyciu funkcji **CHR$** lub ciągów „escape". Drugim sposobem jest używanie instrukcji **POSITION**.

Czasami program wymaga oczyszczenia ekranu. Robi się to przez wyświetlenie znaku ATASCII o kodzie 125, który czyści ekran i ustawia kursor w lewym górnym rogu. W celu wygenerowania niezbędnego znaku można użyć: **CHR$** (125), **ESC/CTRL**— < lub **ESC/SHIFT**— <.

Przesuwanie kursora po ekranie umożliwiają znaki: , , → i . Nie kasują żadnego znaku, przez który przechodzi kursor. Są dokładnie takie same w trybie programowania, jak w trybie bezpośrednim.

PROGRAM NR 4, po obliczeniu wyniku, przesuwa kursor do góry, do miejsca wprowadzenia pierwszej liczby. Pozwala to wprowadzić nową liczbę lub nacisnąć RETURN, aby pozostawić poprzednią wartość.

Instrukcja POSITION działa nieco inaczej. Nie przesuwa kursorem, lecz aktualizuje zawartość pewnych komórek pamięci komputera nowym położeniem kur-

```
FK 1 REM ***************
BP 2 REM * Program nr 7 *
FM 3 REM ***************
HH 10 DIM BL$(40),CMD$(1),R$(1)
OS 50 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
OS 300 AMPSUB=2000
TW 310 PRINT CHR$(125):REM KASOWA
   NIE EKRANU
HT 320 POSITION 2,1
XG 330 PRINT "POLECENIA (ACDEP, H
   =Pomoc)";
UV 340 INPUT CMD$
HL 360 POSITION 2,0
XJ 370 PRINT "Mam wykonac polecen
   ie ";
RY 380 ON ASC(CMD$)-64 GOSUB 600,
   31767,800,1000,1200,31767,3176
   7,1400
XS 390 IF CMD$<>"P" THEN 310
WB 400 PRINT "DRUKUJ":POP :GOSUB
   31766:GOTO 310
FA 600 PRINT "DODAJ ":GOSUB 31766
   :RETURN
VS 800 PRINT "ZMIEN ":GOSUB 31766
   :RETURN
JJ 1000 PRINT "KASUJ ":GOSUB 3176
   6:RETURN
XY 1200 PRINT "KONIEC":POSITION 2
   ,10:END
DL 1400 PRINT "POMOC"
XG 1410 GOSUB 2000
ZT 1420 POP :GOTO 320
VP 1994 REM ****************
XV 1995 REM *  Podprogram   *
NP 1996 REM * wyswietlania  *
IV 1997 REM * rozszerzonego *
NU 1998 REM * wykazu komend *
WJ 1999 REM ****************
YJ 2000 GOSUB 2300
PO 2010 PRINT "__________Wykaz kom
   end______"
ZF 2020 PRINT "Wiecej danych (ACD
   P, N jesli nie)"
BX 2030 PRINT " A=dodaj C=zmien
   D=kasuj"
MZ 2040 PRINT " E=koniec P=drukuj
   H=pomoc";
RO 2050 POSITION 36,21
KZ 2060 INPUT R$
IY 2070 IF R$="N" THEN RETURN
HP 2080 ON ASC(R$)-64 GOTO 2150,3
   2767,2210,2210
BN 2090 IF R$<>"P" THEN GOTO 2050
YL 2100 GOSUB 2300
CD 2110 PRINT "_________komenda:
   DRUKUJ_____"
AU 2120 PRINT " Tekst okreslajacy
    dzialanie"
HK 2130 PRINT " komendy drukuj "
AO 2140 RETURN
ZA 2150 GOSUB 2300
VK 2160 PRINT "_________komenda:
   DODAJ _____"
BJ 2170 PRINT " Tekst okreslajacy
    dzialanie"
DW 2180 PRINT " komendy dodaj"
BD 2190 RETURN
YQ 2210 GOSUB 2300
ZJ 2220 PRINT "__komenda: ZMIEN l
   ub KASUJ___"
AZ 2230 PRINT " Tekst okreslajacy
    dzialanie"
VA 2240 PRINT " komend zmien i ka
   suj"
AT 2250 RETURN
IF 2300 POSITION 1,19
VH 2310 FOR J=19 TO 23
SK 2330 PRINT BL$(1,38);
FT 2340 NEXT J
DU 2350 POSITION 2,20
AY 2360 RETURN
CJ 31766 FOR I=0 TO 400:NEXT I:RE
   TURN
GD 31767 RETURN
```

sora. Nowy numer wiersza jest w komórce o adresie 84, nowy numer kolumny — w komórce 85. W dowolnej chwili można użyć funkcji **PEEK**, aby sprawdzić, gdzie następnie będzie kursor: **PEEK(84)** — dla numeru wiersza, **PEEK (85)** — dla numeru kolumny. W pewnych trybach graficznych komputer Atari używa do przechowania numeru kolumny dwóch komórek. W tym wypadku **PEEK(86)** × 256 + **PEEK(85)** daje numer kolumny.

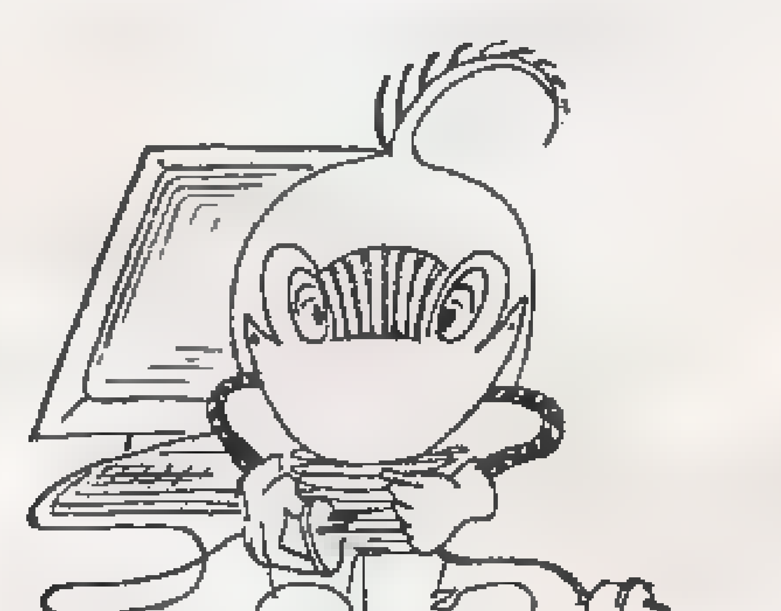
Po każdym wyświetleniu informacji użycia instrukcji **PRINT** następuje również aktualizacja dwóch innych komórek pamięci przechowujących ostatnią pozycję kursora. Komórka 90 przechowuje numer wiersza, a 91 — numer kolumny.

Dzięki instrukcji **POKE** możemy zmieniać szerokość obrazu przesuwając lewy i prawy margines. W komórce o adresie 82 przechowywana jest wartość lewego marginesu, a w komórce 83 — prawego. Standardowo lewy margines jest w kolumnie 2. Aby go zmienić na 0 używamy instrukcji **POKE 82,0**. Prawy margines standardowo jest w kolumnie 39. Aby go przesunąć używamy: **POKE 83,38.** **Zmianę szerokości obrazu demonstruje PROGRAM NR 5.**

Po ponownym ustawieniu marginesów musimy pamiętać, że instrukcja PRINT „obserwuje" ich wartości. Nie ma znaku, który umożliwi przesunięcie kursora poza marginesy. Po osiągnięciu prawego marginesu następuje przejście do nowej linii.

Jedynym sposobem ustawienia kursora poza marginesami jest użycie funkcji POSITION.

Należy również pamiętać, że zwężenie obrazu zmniejsza długości nie tylko linii fizycznej programu, lecz także linii logicznej, która nigdy nie przekracza trzech linii fizycznych, bez względu na ich długość.



Programowanie wejścia

Prawie każdy program wymaga wprowadzenia informacji. Celem sekcji wejścia dowolnego programu powinno być minimalizowanie błędów wprowadzania oraz wygodne wykrywanie i poprawianie popełnionych błędów. Istnieją różne sposoby wprowadzania, które dążą do minimalizacji błędów. Poniżej przedstawione będą następujące:

```
FK 1 REM ***************
FJ 2 REM *             *
CK 3 REM * Program nr 8 *
FL 4 REM *             *
FO 5 REM ***************
QN 10 DIM NMSK$(40),SMSK$(40),R$(
   40),CL$(40)
MB 20 NMSK$="#":NMSK$(40)=NMSK$(1
   ):NMSK$(2)=NMSK$(1)
HZ 30 SMSK$="_":SMSK$(40)=SMSK$(1
   ):SMSK$(2)=SMSK$(1)
WU 40 CL$=CHR$(30):CL$(40)=CL$:CL
   $(2)=CL$
JS 90 PRINT CHR$(125);"WPROWADZAN
   IE CENNIKA:":?
OB 100 PRINT "NAZWA TOWARU ";SMSK
   $(1,9);CL$(1,10);
GO 110 INPUT R$
EJ 120 PRINT "CENA JEDNOSTKOWA ";
   NMSK$(1,8);CL$(1,9);
GS 130 INPUT R$
NY 140 END
```

```
FK 1 REM ***************
FJ 2 REM *             *
DE 3 REM * Program nr 9 *
FL 4 REM *             *
FO 5 REM ***************
VX 10 DIM R$(40)
TD 200 PRINT
TB 210 PRINT "CZY WPROWADZASZ INN
   A POZYCJE ";
GW 220 INPUT R$:R$=R$(1,1)
KZ 230 IF R$="T" OR R$="t" THEN 9
   0
GN 240 IF R$="N" OR R$="n" THEN E
   ND
MM 250 GOTO 210
```

```
QO 1 REM ****************
IB 2 REM *              *
MQ 3 REM * Program nr 10 *
ID 4 REM *              *
QS 5 REM ****************
WM 10 DIM X$(10)
IV 50 TRAP 8000
UD 200 PRINT "WPROWADZ CIAG ZNAKO
   W ";
JD 210 INPUT X$
EO 220 IF X$="K" THEN 500:REM Kon
   iec programu ?
EG 230 PRINT "WPROWADZ WARTOSC LI
   CZBOWA";
ST 240 INPUT X
CC 249 REM Blad jesli wprowadzono
    0
NW 250 X=X*X/X
JL 499 REM Koniec programu
KR 500 PRINT "OSTATNIE DANE: ";X$
   ;" i ";X
TF 510 TRAP 40000:REM Wylaczenie
   TRAP
NY 520 END
WQ 7998 REM +++++Obsluga bledow++
   +++
HS 7999 REM Pobranie numeru bledu
HN 8000 E=PEEK(195)
CS 8009 REM Pobranie numeru linii
    w ktorej wystapil blad
CD 8010 EL=PEEK(186)+256*PEEK(187
   )
BC 8020 IF E=3 OR E=8 THEN 8100
DJ 8029 REM Blad w programie
NF 8030 PRINT "UWAGA! Blad nr ";E
DD 8040 PRINT "w linii ";EL
FS 8080 END
AB 8100 REM Blad danych wejsciowy
   ch
JH 8110 PRINT CHR$(253);
SV 8120 PRINT "BLAD... WPROWADZ J
   ESZCZE RAZ"
MQ 8130 TRAP 8000:REM Kasowanie T
   RAP
KX 8140 GOTO EL
```

— wyświetlanie pomocniczych komunikatów,
— oczekiwanie naturalnych, intuicyjnych odpowiedzi,
— sprawdzanie wprowadzonych informacji,
— używanie podprogramu obsługi błędów,
— logiczne grupowanie wprowadzanych danych,
— umożliwienie obejrzenia i zmiany pogrupowanych danych,
— ograniczanie odpowiedzi: używanie manipulatorów,
— ograniczanie wyborów: używanie menu.

Jednym ze sposobów ułatwiania współpracy użytkownika z programem, umożliwiającym wyeliminowanie błędów wprowadzania informacji, jest wyświetlanie krótkich komunikatów (podpowiedzi), które opisują oczekiwane dane. W miarę możliwości powinniśmy wstawiać podpowiedzi w jednej linii, a odpowiedzi w następnej. Ponieważ instrukcja **INPUT** zawsze wyświetla znak zapytania, najlepiej jest przedstawiać podpowiedzi w formie zapytań.

Czasami niemożliwe jest sformułowanie zadowalającej podpowiedzi; jest albo zbyt lakoniczna albo zajmuje zbyt dużo miejsca. W takim wypadku możemy wyświetlić rozwiniętą podpowiedź w innym miejscu ekranu. Normalnie program wyświetla krótki komunikat. Umożliwia też użytkownikowi uzyskanie rozwiniętej instrukcji po naciśnięciu klawisza „H" (od Help). Wyprowadza wtedy instrukcje w pewnym stałym miejscu ekranu, powiedzmy w czterech dolnych liniach. Po wyświetleniu instrukcji program musi wrócić do miejsca, gdzie nastąpiło wezwanie pomocy (oczekiwanie na wprowadzenie).

PROGRAM NR 6 **oczekuje na jednoliterowe polecenia w drugiej linii ekranu.** Rozszerza polecenia literowe do poleceń słownych i wyświetla słowo z prawej strony górnej linii. Instrukcja może być wywołana przez dowolny znak lub słowo. Literę H wybraliśmy samowolnie. Instrukcje możemy umieszczać w dowolnym miejscu ekranu, lcze zalecane jest wyświetlanie każdego zestawu instrukcji w tym samym obszarze. Oczywiście program musi czekać, aż użytkownik zakończy czytanie każdej części instrukcji, zanim przejdzie do następnej. Realizuje to pojedyncza instrukcja **INPUT**. Jednocześnie program może zezwolić użytkownikowi na przerwanie instrukcji i powrót do normalnego ciągu wprowadzania. W PROGRAMIE NR 7 **podmieniony został podprogram zaczynający się od linii 2000.**

Innym miejscem umieszczania instrukcji jest początek programu. Wyświetlane instrukcje nie zastąpią dobrze napisanej instrukcji drukowanej, lecz są często wystarczającym przypomnieniem dla kogoś, kto jest trochę roztargniony.

```
QO 1 REM ****************
IB 2 REM *              *
NL 3 REM * Prosram nr 11 *
ID 4 REM *              *
QS 5 REM ****************
EQ 10 DIM PRMT$(40),MSK$(40),BL$(
   40),ERM$(20),R$(20)
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$
   (2)=MSK$
GM 40 ERRHDL=8000
IC 90 GOTO 1000
ME 97 REM *****************
UE 98 REM * Kasowanie linii *
MI 99 REM *****************
EI 100 FOR J=L1 TO L2
VX 110 POSITION 0,J
YX 120 PRINT BL$(1,38);
GH 130 NEXT J
ZF 140 RETURN
IB 167 REM ******************
ZX 168 REM * Komunikat bledu *
IH 169 REM ******************
NE 170 FOR J=1 TO 3
MT 180 POSITION 20,0
UD 190 PRINT ERM$;CHR$(253);
LE 200 FOR J2=1 TO 100:NEXT J2:RE
   M Opoznienie
MG 210 POSITION 20,0
ND 220 PRINT BL$(1,19):REM Kasowa
   nie
GI 230 NEXT J
ZG 240 RETURN
NX 246 REM *************
EW 247 REM * Kasowanie  *
LM 248 REM * instrukcji *
OG 249 REM *************
EG 250 L1=20:L2=23:GOSUB 100
OR 260 POSITION 2,20
ZM 270 RETURN
OK 596 REM *************
SR 597 REM * Podprosram *
ZJ 598 REM *   wejscia  *
OT 599 REM *************
IB 600 TRAP ERRHDL
TM 610 GOSUB 250
QV 620 REM
AV 630 POSITION IC+1,IR
LG 640 PRINT MSK$(1,IL);BL$
FE 650 POSITION IC,IR
SC 659 REM
HD 660 INPUT R$
RP 664 REM
MB 665 FOR J1=1 TO LEN(R$):IF R$(
   J1,J1)<>MSK$(1,1) THEN NEXT J1
AG 666 IF J1>1 THEN R$=R$(1,J1-1)
   :GOTO 670
CG 667 R$=""
ZU 670 IF R$="?" THEN GOSUB AMPSU
   B:GOTO 630
VH 680 IF LO>HI THEN GOTO 750:REM
   cias
OD 690 REM Zerowe wejscie
GF 700 IF R$="" THEN R$="0"
YV 705 IF ASC(R$(1,1))<48 OR ASC(
   R$(1,1))>57 THEN 730
UW 710 R=VAL(R$)
PA 720 IF R>LO AND R<HI THEN RETU
   RN
AX 730 ERM$="ZLA WARTOSC !"
FU 740 GOSUB 170:GOTO 610
ZD 749 REM Wprowadzanie ciasu
PG 750 IF LEN(R$)<=IL THEN RETURN
OW 759 REM Zla dlusosc ciasu
YG 760 ERM$="ZLA DLUGOSC CIAGU !"
GA 770 GOSUB 170:GOTO 610
MK 796 REM ***************
VF 797 REM * Wprowadzanie *
MQ 798 REM ***************
SL 799 REM
CU 800 L1=1:L2=2:GOSUB 100
RU 809 REM
HW 810 POSITION 2,1
AA 820 PRINT PRMT$
EZ 829 REM Wartosc wejsciowa
XI 830 IC=2:IR=2:GOSUB 600
ZM 840 RETURN
GY 999 REM ---PROGRAM GLOWNY---
EF 1000 PRINT CHR$(125);:REM Czys
   zczenie ekranu
JW 1009 REM Wprowadzanie ciasu
BE 1010 PRMT$="Podaj imie sracza
   ?"
AN 1020 IL=18:LO=1:HI=0
RR 1030 AMPSUB=31767
YJ 1040 GOSUB 800
LJ 1049 REM Wprowadzanie wartosci
    liczbowej
GB 1050 PRMT$=R$
HX 1060 PRMT$(LEN(PRMT$)+1)="-Jak
   i wynik ?"
IR 1080 IL=7:LO=0:HI=300
CD 1090 AMPSUB=7000
XZ 1100 GOSUB 800
NG 1110 GOTO 1010
SO 6997 REM Instrukcja wprowadzan
   ia wartosci liczbowej
XW 7000 GOSUB 250
YW 7010 PRINT " Wprowadz liczbe d
   odatnia,"
PW 7020 PRINT "        mniejsza od
   300."
AO 7030 RETURN
NT 7996 REM *****************
CH 7997 REM * Obslusa bledow *
OB 7998 REM *****************
HS 7999 REM Pobranie numeru bledu
HN 8000 E=PEEK(195)
OI 8009 REM Pobranie numeru linii
BT 8010 EL=PEEK(187)*256+PEEK(186
   )
BC 8020 IF E=3 OR E=8 THEN 8100
NF 8030 PRINT "UWAGA! Blad nr ";E
KF 8040 PRINT "w linii nr ";EL
FS 8080 END
IF 8100 REM
NA 8110 ERM$="BLAD WE"
YT 8120 GOSUB 170
VQ 8130 TRAP ERRHDL
KX 8140 GOTO EL
GD 31767 RETURN
```



Maskowanie wejścia

Zawsze istnieją pewne ograniczenia długości odpowiedzi. Program może wyświetlać ciąg znaków, które ograniczają tę długość. Taki ciąg nazywamy maską wprowadzania.

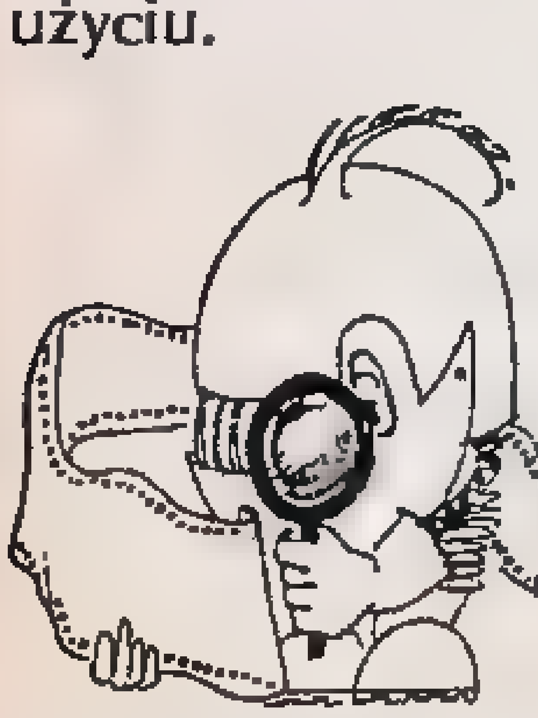
Do tego celu nadaje się dowolny znak. Powszechnie stosowane są znaki podkreślenia, gwiazdki, plus i minus. Program może używać jednego rodzaju znaków do maskowania tekstu i innego do maskowania liczb. Dostarcza to dodatkowej informacji o oczekiwanej odpowiedzi. **PROGRAM NR 8 używa znaku podkreślenia dla wprowadzania tekstów i znaku # dla wejścia numerycznego.**

Każda instrukcja **PRINT,** która drukuje maskę, drukuje także ciąg znaków, który przesuwa kursor na początek maski wprowadzania (linie 100 i 120).



Wybieranie odpowiedzi

Możemy zmniejszyć szansę powstania błędu przez ostrożne wybieranie odpowiedzi wejściowych. Nasz program powinien umożliwić użytkownikowi odpowiedzi w sposób naturalny, intuicyjny. Podczas pisania programu wygodnie jest, aby użytkownik kodował wszystkie informacje wejściowe. Zmusza go to jednak do wykonywania mechanicznych czynności podczas każdego używania programu. Ponieważ komputer jest lepszy w czynnościach rutynowych, pozwólmy mu wykonywać kodowanie? Jeśli naturalną odpowiedzią jest słowo lub litera, których program będzie ewentualnie potrzebował przetłumaczyć na liczbę, dajmy mu to zrobić. **Właśnie to wykonują linie 310 do 390 PROGRAMU NR 7.** Użytkownik wprowadza kody mnemoniczne poleceń: A,C,D,E,P lub H. Program bada, który podprogram wywołać (linia 380), w celu wykonania polecenia. Wyobraźmy sobie, o ile łatwiej byłoby napisać program, który wymaga od użytkownika wprowadzania polecenia w postaci numerycznej, lecz o ile trudniejszy byłby on w użyciu.



Sprawdzanie odpowiedzi

Nawet przy najostrożniejszym zaprojektowaniu wprowadzania nie możemy być pewni, w jaki sposób będą odpowiadać użytkownicy. Jeśli wprowadzenie błędnej informacji może spowodować jakąś trudność, program powinien sprawdzić, czy tekst nie jest zbyt długi? czy

```
QO 1 REM ****************
IB 2 REM *              *
OG 3 REM * Program nr 12 *
ID 4 REM *              *
QS 5 REM ****************
OS 9 REM Inicjacja zmiennych
GM 10 DIM PRMT$(40),MSK$(40),BL$(
   40),ERM$(23)
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$
   (2)=MSK$
GM 40 ERRHDL=8000
YU 50 DIM NA$(20),STT$(20),CI$(20
   ),ST$(2),ZI$(9),R$(25)
GB 89 REM Skok do poczatku progra
   mu
IC 90 GOTO 1000
ME 97 REM ******************
UE 98 REM * Kasowanie linii *
MI 99 REM ******************
EI 100 FOR J=L1 TO L2
VX 110 POSITION 0,J
YX 120 PRINT BL$(1,38);
GH 130 NEXT J
ZF 140 RETURN
LV 165 REM ***************
YT 166 REM * Wyswietlanie *
ZU 167 REM *    bledow    *
ME 168 REM ***************
NE 170 FOR J=1 TO 3
UT 180 POSITION 10,20
UD 190 PRINT ERM$;CHR$(253);
TX 200 FOR J2=1 TO 100:NEXT J2
UG 210 POSITION 10,20
BL 220 PRINT BL$(1,23)
GI 230 NEXT J
ZG 240 RETURN
NU 245 REM *************
ET 246 REM * Kasowanie  *
QK 247 REM *    pola    *
LM 248 REM * instrukcji *
OG 249 REM *************
EG 250 L1=20:L2=23:GOSUB 100
OA 260 POSITION 2,20
ZM 270 RETURN
MI 596 REM ***************
BK 597 REM *  Podprogram  *
FK 598 REM * wprowadzania *
MR 599 REM ***************
IB 600 TRAP ERRHDL
TO 620 GOSUB 250
AV 630 POSITION IC+1,IR
TR 640 PRINT MSK$(1,IL);BL$(1,23-
   IL)
FE 650 POSITION IC,IR
XO 659 REM Wprowadzenie i sprawdz
   enie odpowiedzi
HD 660 INPUT R$
MB 665 FOR J1=1 TO LEN(R$):IF R$(
   J1,J1)<>MSK$(1,1) THEN NEXT J1
AG 666 IF J1>1 THEN R$=R$(1,J1-1)
   :GOTO 670
CG 667 R$=""
ZU 670 IF R$="?" THEN GOSUB AMPSU
   B:GOTO 630
GE 680 IF LO>HI THEN GOTO 750
OD 690 REM Zerowe wejscie
```

```
GF 700 IF R$="" THEN R$="0"
YV 705 IF ASC(R$(1,1))<48 OR ASC(
   R$(1,1))>57 THEN 730
UW 710 R=VAL(R$)
B6 720 IF R>=LO AND R<=HI THEN RE
   TURN
AX 730 ERM$="ZLA WARTOSC !"
GP 740 GOSUB 170:GOTO 620
SB 749 REM
PG 750 IF LEN(R$)<=IL THEN RETURN
OW 759 REM Zla dlugosc ciagu
ZU 760 ERM$="CIAG ZBYT DLUGI "
GV 770 GOSUB 170:GOTO 620
MH 795 REM ***************
VC 796 REM * Wprowadzanie *
GG 797 REM *  wspomagane  *
MQ 798 REM ***************
CU 800 L1=1:L2=2:GOSUB 100
QG 809 REM Wyswietlenie komunikat
   u
HW 810 POSITION 2,1
AA 820 PRINT PRMT$
EZ 829 REM Wartosc wejsciowa
XI 830 IC=2:IR=2:GOSUB 600
ZM 840 RETURN
AC 999 REM Kasowanie ekranu i wys
   wietlanie
IZ 1000 PRINT CHR$(125);"PODAJ NA
   ZWISKO I ADRES"
AP 1010 PRINT :PRINT
IC 1020 PRINT "1)Nazwisko:"
RZ 1030 PRINT "2)   Ulica:"
JC 1040 PRINT "3)  Miasto:"
UD 1050 PRINT "4) Panstwo:"
UW 1070 PRINT "5)     Kod:"
WR 1099 REM Wprowadzenie 5 pol
WZ 1100 FOR F=1 TO 5
XA 1110 GOSUB 2000
DQ 1120 NEXT F
CQ 1129 REM Wprowadzanie zmian
IN 1130 PRMT$="Czy chcesz cos zmi
   enic?"
XB 1140 LO=1:HI=0:AMPSUB=31767
VD 1150 IL=1:GOSUB 800
MO 1159 REM Analiza odpowiedzi
EE 1160 IF R$="N" OR R$="n" THEN
   1000
PB 1170 IF R$="T" OR R$="t" THEN
   1200
JO 1180 ERM$="Prosze napisac T lu
   b N":GOSUB 170
PQ 1190 GOTO 1130

XE 1199 REM Pobranie numeru pola
HV 1200 PRMT$="Ktore pole"
AK 1210 LO=1:HI=5:AMPSUB=32767
UW 1220 IL=1:GOSUB 800
WF 1230 F=R:GOSUB 2000:GOTO 1130
FF 1900 END
XM 2000 LO=1:HI=0:AMPSUB=32767
AZ 2010 ON F GOTO 2100,2200,2300,
   2400,2500
KA 2097 REM
IP 2098 REM Wprowadzenie nazwiska
KI 2099 REM
PV 2100 IC=13:IR=3:IL=20:GOSUB 60
   0
KR 2110 NA$=R$:RETURN
```



```
KC 2197 REM
OE 2198 REM Wprowadzenie ulicy
KK 2199 REM
QM 2200 IC=13:IR=4:IL=20:GOSUB 60
   0
TV 2210 STT$=R$:RETURN
KE 2297 REM
HZ 2298 REM Wprowadzenie miasta
KM 2299 REM
RD 2300 IC=13:IR=5:IL=20:GOSUB 60
   0
KL 2310 CI$=R$:RETURN
KG 2397 REM
RS 2398 REM Wprowadzenie panstwa
KO 2399 REM
HU 2400 IC=13:IR=6:IL=3:GOSUB 600
RE 2410 ST$=R$:RETURN
KI 2497 REM
ZW 2498 REM Wprowadzenie kodu
KQ 2499 REM
KT 2500 IC=13:IR=7:IL=6:GOSUB 600
PX 2510 ZI$=R$:RETURN
NT 7996 REM *****************
CH 7997 REM * Obsluga bledow *
OB 7998 REM *****************
HS 7999 REM Pobranie numeru bledu
HN 8000 E=PEEK(195)
NF 8009 REM Pobranie numeru bledn
   ej linii
BT 8010 EL=PEEK(187)*256+PEEK(186
   )
BC 8020 IF E=3 OR E=8 THEN 8100
CP 8030 PRINT "ZNALEZIONO BLAD NR
   ";E
AJ 8040 PRINT "  W LINII ";EL
FS 8080 END
IF 8100 REM
HF 8110 ERM$="BLAD WPROWADZANIA"
YT 8120 GOSUB 170
VQ 8130 TRAP ERRHDL
KX 8140 GOTO EL
GD 31767 RETURN
```

dane numeryczne mieszczą się w wybranym zakresie? czy wprowadzona informacja nie spowoduje później błędu w programie?

Jeśli chcemy napisać dobry program, musimy włożyć trochę wysiłku w przeciwdziałanie błędom, które może popełnić ktoś używający naszego programu. Program powinien wykrywać błędy wprowadzania i wymuszać na użytkowniku ponowne wprowadzenie danych, które mogłyby spowodować nienormalne zatrzymanie programu.

Prawdą jest, że BASIC wychwytuje pewne błędy. Jednak jego możliwości są ograniczone. Możliwe jest wprowadzenie wartości poprawnego typu, której wielkość jest nie do przyjęcia. Oznacza to, że wprowadzona wartość może spowodować błąd w dalszej części programu. Poniższy przykład ilustruje ten problem:

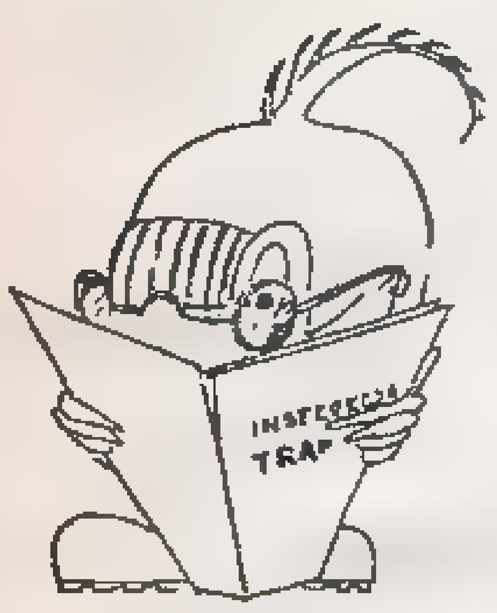
```
100 INPUT X
200 PRINT 100/X
300 END
```

Jeśli wprowadzimy 0 (w linii 100), program upadnie podczas próby dzielenia przez zero (linia 200) w instrukcji PRINT. Można tego uniknąć dosyć łatwo. Dodajmy do powyższego programu następujące linie w celu sprawdzenia, czy wprowadzona wartość nie jest zerem i zażądajmy ponownego wprowadzenia, jeśli jest.

110 IF X<>0 THEN 200
120 PRINT „Niedozwolone ... wprowadź ponownie"
130 GOTO 100

Przez rozwinięcie zilustrowanego sposobu możemy zobaczyć, jak łatwe jest sprawdzenie pod względem wartości. W zależności od okoliczności powinniśmy to robić raczej instrukcją ON—GOTO lub ON—GOSUB niż ciągiem instrukcji IF—THEN.

Czasami kontrola błędów jest kosztowna. Może zająć trochę czasu programowania, obszaru pamięci programu i czasu jego działania. Rozważmy, na przykład, typowe pytanie *tak* lub *nie*. Program powinien umożliwić dowolne poprawne odpowiedzi „naturalne", którymi są: *tak, nie, Tak, Nie, TAK, NIE, t, n, T* lub *N*. Istnieje 10 możliwych odpowiedzi, każdą z nich program musi sprawdzić. **Możemy łatwo zredukować liczbę sprawdzanych wartości, wystarczy sprawdzać pierwszy wprowadzany znak, jak w PROGRAMIE NR 9.** Jeśli odpowiedź nie jest dozwolona, program powtarza pytanie.



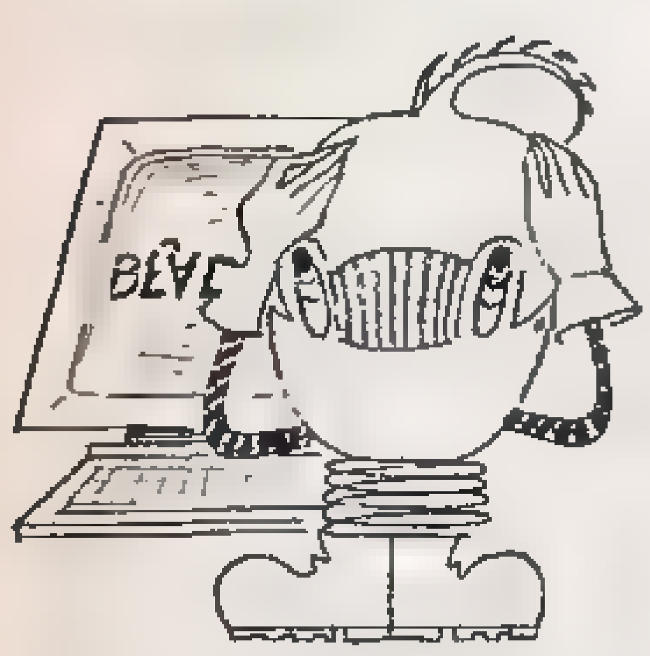
Instrukcja TRAP

Atari BASIC ma specjalną instrukcję, która umożliwia śledzenie błędów, które przechwytuje, zanim wyświetli komunikat i wstrzyma realizację programu. Po wykonaniu następującej instrukcji:

100 TRAP 20000

program przejdzie do podanego numeru linii w razie wykrycia błędu. Umieści także kod numeryczny błędu w komórce o adresie 195, który możemy zbadać funkcją PEEK. BASIC przechowuje także numer linii, w której wystąpił błąd. Możemy go pobrać dzięki wyrażeniu: **PEEK(187)+256*PEEK(186).**

Instrukcja **TRAP** jest deaktywowana po każdym wystąpieniu błędu. Program musi zatem wykonać inną instrukcję TRAP w celu jej reaktywowania. W celu deaktywacji tej instrukcji należy użyć **TRAP 40000.**



Procedura obsługi błędu

Zwykłą reakcją na błędy wykryte instrukcją TRAP jest napisanie procedury obsługi błędów, do której przechodzi program po wykryciu błędu. Na końcu tej procedury program może przejść z powrotem do początku linii, w której wystąpił błąd, lub do dowolnej innej linii programu. Procedura obsługi błędów może podjąć różne akcje w zależności od natury błędu i aktualnego stanu programu.

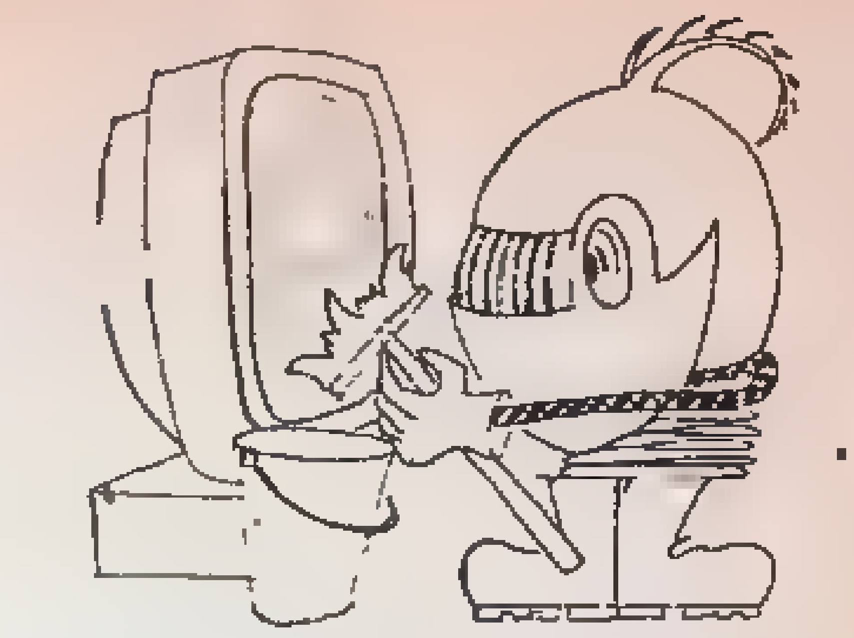
PROGRAM NR 10 demonstruje użycie instrukcji TRAP. Program ten traktuje błędy, które nie są związane z wprowadzaniem danych, jako fatalne. Informuje o numerze błędu i numerze linii oraz zatrzymuje program. Błędy wprowadzenia nie są fatalne. Program sygnalizuje je i żąda ponownego wprowadzenia.

```
QO 1 REM ****************
IB 2 REM *              *
PB 3 REM * Prosram nr 13 *
ID 4 REM *              *
QS 5 REM ****************
EG 10 DIM X(50,14)
RZ 20 DIM BL$(40),X$(4)
OQ 30 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
GN 40 POKE 752,2
SL 49 REM Inicjowanie tablicy
FE 50 PRINT CHR$(125);"Prosze chw
   ile poczekac"
IY 60 FOR K=0 TO 14
IN 70 FOR J=0 TO 50
WQ 80 X(J,K)=(J+1)*100+K+1
JG 90 NEXT J
GL 100 NEXT K
HX 199 REM Prosram slowny
DS 200 PRINT CHR$(125);"Uzyj mani
   pulatora do przesuniecia okna"
RI 205 R=25:C=7
QL 210 GOSUB 1000
RI 220 FOR K=1 TO 10
MX 230 FOR J=1 TO 3
IU 240 POSITION J*10-1,K+4
PI 250 X$=STR$(X(R+K-1,C+J-1))
ON 260 PRINT BL$(1,10-LEN(X$));X$
   ;
GQ 270 NEXT J
HC 280 NEXT K
IG 290 POSITION 2,1
ZH 299 REM Odczyt manipulatora
DL 300 IF STICK(0)=7 AND C<12 THE
   N C=C+1:GOTO 210
EO 310 IF STICK(0)=11 AND C>0 THE
   N C=C-1:GOTO 210
AT 320 IF STICK(0)=13 AND R<41 TH
   EN R=R+1:GOTO 210
HW 330 IF STICK(0)=14 AND R>0 THE
   N R=R-1:GOTO 210
PV 340 GOTO 290
YG 998 REM +++PODPROGRAM 1000+++
XD 1000 FOR J=1 TO 3
JZ 1010 POSITION 3+J*10,3
GH 1020 PRINT "KOLUMNA";
FJ 1030 NEXT J
WM 1040 FOR J=0 TO 2
EX 1050 POSITION 16+J*10,4
LP 1060 IF C+J+1<10 THEN PRINT "
   ";
JD 1070 PRINT C+J+1;
FY 1080 NEXT J
BQ 1090 FOR J=0 TO 9
AV 1100 POSITION 0,J+5
RG 1110 PRINT "WIERSZ ";
QK 1120 IF R+J+1<10 THEN PRINT "
   ";
PR 1130 PRINT R+J+1;
FO 1140 NEXT J
AQ 1150 RETURN
```

```
QO 1 REM ****************
IB 2 REM *              *
PW 3 REM * Program nr 14 *
ID 4 REM *              *
QS 5 REM ****************
FM 1000 R=1:LO=0:HI=100
PB 1010 POKE 752,2
JE 6509 REM Kasowanie starej wart
   osci
MP 6510 PRINT CHR$(125)
XU 6519 REM Wyswietlenie nowej wa
   rtosci
JU 6530 PRINT R;
PK 6539 REM Zakonczenie programu
RX 6550 IF STRIG(0)=0 THEN END
NW 6559 REM Opoznienie
XF 6560 FOR J=1 TO 30:NEXT J
IX 6579 REM Ruch do tylu
FZ 6580 IF STICK(0)=11 AND R<>LO
   THEN R=R-1:GOTO 6510
NE 6589 REM Ruch do przodu
QU 6590 IF STICK(0)=7 AND R<>HI T
   HEN R=R+1:GOTO 6510
BD 6599 REM Bez zmian
UD 6600 GOTO 6550
```

```
QO 1 REM ****************
IB 2 REM *              *
QR 3 REM * Program nr 15 *
ID 4 REM *              *
QS 5 REM ****************
ZQ 10 DIM BL$(40)
HC 19 POKE 752,2
OP 20 BL$=" ":BL$(40)=BL$:BL$(2)=
   BL$
LX 1000 PRINT CHR$(125)
ER 1200 LO=0:HI=200:R=100:INC=0.5
TW 1210 IC=10:IR=11:IL=7
BO 1220 GOSUB 6500
RM 1230 PRINT
WQ 1240 PRINT "Wybrana liczba : "
   ;R
FF 1900 END
BK 6496 REM ********************
VO 6497 REM * WE z manipulatora*
BS 6498 REM ********************
AG 6500 POSITION IC,IR
WI 6510 PRINT BL$(1,IL);
AM 6520 POSITION IC,IR
UU 6530 PRINT R;
IP 6539 REM Koniec programu
ZA 6550 IF STRIG(0)=0 THEN RETURN
NW 6559 REM Opoznienie
FZ 6560 FOR J=1 TO SD:NEXT J
SU 6569 REM Opoznienie minimalne
VG 6570 SD=1
YO 6579 REM Ruch w tyl
GW 6580 IF STICK(0)=11 AND R<>LO
   THEN R=R-INC:GOTO 6500
CT 6589 REM Z powrotem
LF 6590 IF STICK(0)=7 AND R<>HI T
   HEN R=R+INC:GOTO 6500
BD 6599 REM Bez zmian
EQ 6600 SD=30:GOTO 6550
```


Użyteczne podprogramy wprowadzania

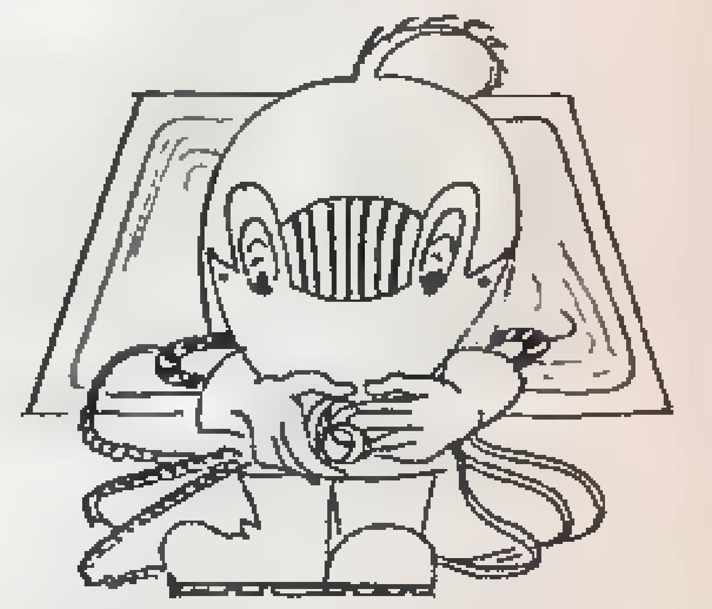


Obecnie spróbujemy zbudować ogólny podprogram wprowadzania. Będzie on używał wszystkich technik, które omówiliśmy dotychczas: podpowiedzi, rozszerzeń instrukcji, maskowania wejścia, sprawdzania odpowiedzi i procedury obsługi błędu.

Podprogram wprowadzania będzie używał kilku innych podprogramów. Jeden z nich czyści (kasuje) linie na ekranie, inny wyświetla komunikat błędu w prawym, górnym rogu ekranu oraz generuje sygnał dźwiękowy. Potrzebny jest także podprogram czyszczenia obszaru u dołu ekranu, w którym pojawiają się rozszerzone instrukcje. **PROGRAM NR 11 zawiera wszystkie niezbędne podprogramy oraz wzajemne powiązania między nimi.** Podprogramy te są wyróżnione odpowiednimi komentarzami. Używane są dwie zmienne do przechowywania numerów linii: **ERRHDL i AMPSUB.** Umożliwia to dostarczenie do programu wywołującego własnych procedur, a zatem różnych sposobów obsługi błędów i rozwinięć instrukcji. **AMPSUB** musi być rzeczywistym numerem linii, która może zawierać samą instrukcję **RETURN,** lecz musi istnieć.

Na początku PROGRAMU NR 11 deklarowane są i inicjowane zmienne używane przez poszczególne podprogramy. Następnie przechodzi on do realizacji głównego ciągu instrukcji (linia 90), gdzie ustawia 20-znakową zmienną tekstową (linie 1010 do 1040) oraz dane numeryczne (linie 1050 do 1070). Zauważmy, że wejście tekstowe nie ma rozwinięć instrukcji — **AMPSUB** ma wartość 31767, adres procedury, która nic nie robi. Istnieją jednak rozwinięcia instrukcji do wprowadzenia wartości numerycznych (linie 7000 do 7040).

Wprowadzanie danych grupowych



Bardzo często program potrzebuje kilku części informacji logicznie powiązanych ze sobą. Może wprowadzać pozycje danych na kilka sposobów. Jednym z nich jest wprowadzanie każdej pozycji w tym samym miejscu ekranu, używając różnych znaków zachęty dla każdej pozycji. **Ten sposób używany jest w procedurze wprowadzania nazw i wartości w PROGRAMIE NR 11.** Program przypomina, którą wartość należy wprowadzić, przez wypisanie odpowiedniego komunikatu.

Lepszym sposobem manipulowania wielokrotnymi pozycjami danych jest wyświetlenie odpowiedniego szablonu na ekranie i wypełnianie go w miarę wprowadzania danych. W tym celu program najpierw wyświetla szablon zawierający opis każdej pozycji i wystarczający obszar dla wprowadzania danych. Tak opisane pozycje nazywane są polami, każde z nich ma numer. Dane wprowadzamy sekwencyjnie, rozpoczynając od pierwszego pola i kończąc na ostatnim.

Do uzyskania takiego efektu potrzebny jest niewielki wysiłek. Załóżmy, że chcemy wprowadzić nazwisko i adres. Mamy zatem do wprowadzenia pięć pozycji: nazwi-

sko, ulica, miasto, państwo i kod pocztowy. Opisany program wprowadzania wykona większość pracy. Wystarczy wprowadzić go do pamięci. Po uruchomieniu program wyświetla maskę wejściową dla każdego z pięciu pól. Informuje także, które pole należy aktualnie wprowadzić.



Przeglądanie i zmienianie wprowadzonych danych

Po zakończeniu wprowadzania program może umożliwić dokonanie zmian poszczególnych pól. Wymaga to znajomości numeru zmienianego pola.

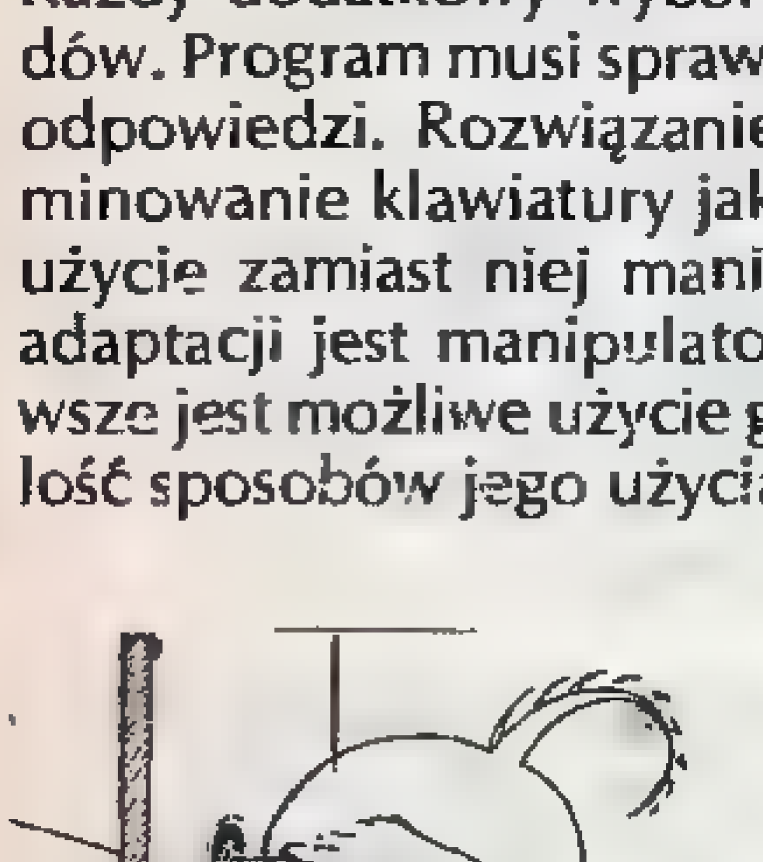
Możemy dodać do tego programu możliwość dokonywania zmian. Po zakończeniu wprowadzania jesteśmy pytani, czy chcemy dokonać zmian. Jeśli tak, to należy podać numer pola, które chcemy zmienić, i użyć odpowiedniej instrukcji do wywołania właściwego podprogramu wprowadzania.

PROGRAM NR 12 realizuje wszystkie opisane wyżej funkcje. Możliwe jest dodanie innych własności, według uznania.



Używanie manipulatorów do ograniczenia odpowiedzi

Jedną z trudności wszystkich sposobów wprowadzania danych jest mnogość wyborów, które ma użytkownik. Każdy dodatkowy wybór to możliwość powstania błędów. Program musi sprawdzać i eliminować niewłaściwe odpowiedzi. Rozwiązaniem tego problemu jest wyeliminowanie klawiatury jako urządzenia wprowadzania i użycie zamiast niej manipulatorów. Najłatwiejszym do adaptacji jest manipulator drążkowy (joystick). Nie zawsze jest możliwe użycie go zamiast klawiatury, lecz wielość sposobów jego użycia jest nieoczekiwana.



Sterowanie obrazem za pomocą manipulatora

Kiedy mamy do czynienia z dużymi ilościami danych, tylko część z nich może być wyświetlona jednocześnie na ekranie. Jednym ze sposobów jest użycie ekranu jako okna nad danymi. W dowolnej chwili pokazuje on tylko część z dostępnych danych. Przeglądanie danych w ten sposób jest łatwe, jeśli są one umieszczone w tablicy numerycznej lub nawet pseudotablicy tekstowej. Wyobraźmy sobie, że tablica danych zapisana jest na dużej

```
QO 1 REM ****************
IB 2 REM *              *
RM 3 REM * Program nr 16 *
ID 4 REM *              *
QS 5 REM ****************
FC 999 REM Kasowanie ekranu i wys
   wietlanie podzialki
LX 1000 PRINT CHR$(125)
BY 1010 POSITION 0,11
XL 1020 PRINT "01234567890123456
   7890123456789012345678 9"
FR 1030 FOR J=0 TO 23
UO 1040 POSITION 21,J
KE 1050 PRINT J-INT(J/10)*10;
FS 1060 NEXT J
ZN 1069 REM Wyswietlanie kursora
NK 1070 POSITION 21,11
JT 1080 PRINT CHR$(253);
KE 1089 REM
AY 1090 SC=21:SR=11
WZ 1099 REM Ograniczenia ruchu
CR 1100 LC=2:RC=39:TR=0:BR=23
VA 1109 REM Ustawienie szybkosci
   zmian
QD 1110 DLY1=20
CW 1119 REM Wywolanie podprogramu
    manipulatora
YZ 1120 GOSUB 6000
FF 1900 END
YF 5996 REM ********************
OF 5997 REM *Podprogram kursora *
YN 5998 REM ********************
TK 5999 REM Nacisnij przycisk w c
   elu zatrzymania kursora
YB 6000 IF STRIG(0)=0 THEN RETURN
NP 6009 REM Wolno w dol
KQ 6010 FOR J=1 TO DLY1:NEXT J
QD 6019 REM Sprawdzenie pozycji m
   anipulatora
UQ 6020 IF STICK(0)=7 AND SC<>RC
   THEN SC=SC+1:PRINT CHR$(31);
MX 6030 IF STICK(0)=11 AND SC<>LC
    THEN SC=SC-1:PRINT CHR$(30);
RO 6040 IF STICK(0)=13 AND SR<>BR
    THEN SR=SR+1:PRINT CHR$(29);
NN 6050 IF STICK(0)=14 AND SR<>TR
    THEN SR=SR-1:PRINT CHR$(28);
PO 6060 GOTO 6000
```

```
QO 1 REM ****************
IB 2 REM *              *
SH 3 REM * Program nr 17 *
ID 4 REM *              *
QS 5 REM ****************
KH 999 REM Kasowanie ekranu i wys
   wietlanie menu
LX 1000 PRINT CHR$(125)
VI 1010 PRINT "Wybierz polecenie
   przy pomocy          manipulato
   ra"
RF 1020 PRINT
GS 1030 PRINT "   DODAJ"
DR 1040 PRINT "   ZMIEN "
JB 1050 PRINT "   USUN"
TQ 1060 PRINT "   DRUKUJ"
ZQ 1070 PRINT "   POMOC"
XY 1080 PRINT "   KONIEC"
```

```
KE 1089 REM
YO 1090 POSITION 4,3
WO 1100 PRINT CHR$(29);
VA 1109 REM Ustawienie szybkosci
    zmian
SB 1110 SC=4:SR=3
JL 1119 REM
GZ 1120 LC=4:RC=4:TR=3:BR=8
JO 1129 REM
QU 1130 DLY1=30
ZF 1140 GOSUB 6000
SB 1150 POSITION 15,15
MM 1160 ON SR-2 GOSUB 2000,2500,3
    000,3500,4000:GOTO 1900
DE 1170 POSITION 2,10
FF 1900 END
KG 2000 PRINT "DODAJ   ":RETURN
CN 2500 PRINT "ZMIEN   ":RETURN
LH 3000 PRINT "USUN    ":RETURN
MN 3500 PRINT "DRUKUJ  ":RETURN
ZO 4000 PRINT "POMOC   ":RETURN
YF 5996 REM ********************
OF 5997 REM *Podprogram kursora *
YN 5998 REM ********************
TK 5999 REM Nacisnij przycisk w c
    elu zatrzymania kursora
YB 6000 IF STRIG(0)=0 THEN RETURN
NF 6009 REM Wolno w dol
KQ 6010 FOR J=1 TO DLY1:NEXT J
QD 6019 REM Sprawdzenie pozycji m
    anipulatora
UQ 6020 IF STICK(0)=7 AND SC<>RC
    THEN SC=SC+1:PRINT CHR$(31);
MX 6030 IF STICK(0)=11 AND SC<>LC
    THEN SC=SC-1:PRINT CHR$(30);
RO 6040 IF STICK(0)=13 AND SR<>BR
    THEN SR=SR+1:PRINT CHR$(29);
NN 6050 IF STICK(0)=14 AND SR<>TR
    THEN SR=SR-1:PRINT CHR$(28);
PO 6060 GOTO 6000
```

```
QO 1 REM ****************
IB 2 REM *              *
TC 3 REM * Program nr 18 *
ID 4 REM *              *
QS 5 REM ****************
RC 10 DIM R(10),R$(10)
QD 59 REM Otwarcie kanalu wejscio
    wego klawiatury
BO 60 OPEN #1,4,0,"K:"
DT 8230 GOSUB 8400:REM ROK
SV 8270 PRINT CHR$(31);
LH 8280 GOSUB 8400:REM MIESIAC
SO 8340 PRINT CHR$(31);
IZ 8350 GOSUB 8400:REM DZIEN
BK 8380 RETURN
GS 8397 REM ******************
NX 8398 REM * POBIERZ 2 CYFRY *
HA 8399 REM ******************
FM 8400 FOR J1=0 TO 1
BI 8410 GET #1,R:R(J1)=R
PF 8440 PRINT CHR$(R);:REM ECHO
CU 8450 NEXT J1
QZ 8460 R$=CHR$(R(0)):R$(2)=CHR$(
    R(1))
BJ 8470 RETURN
```



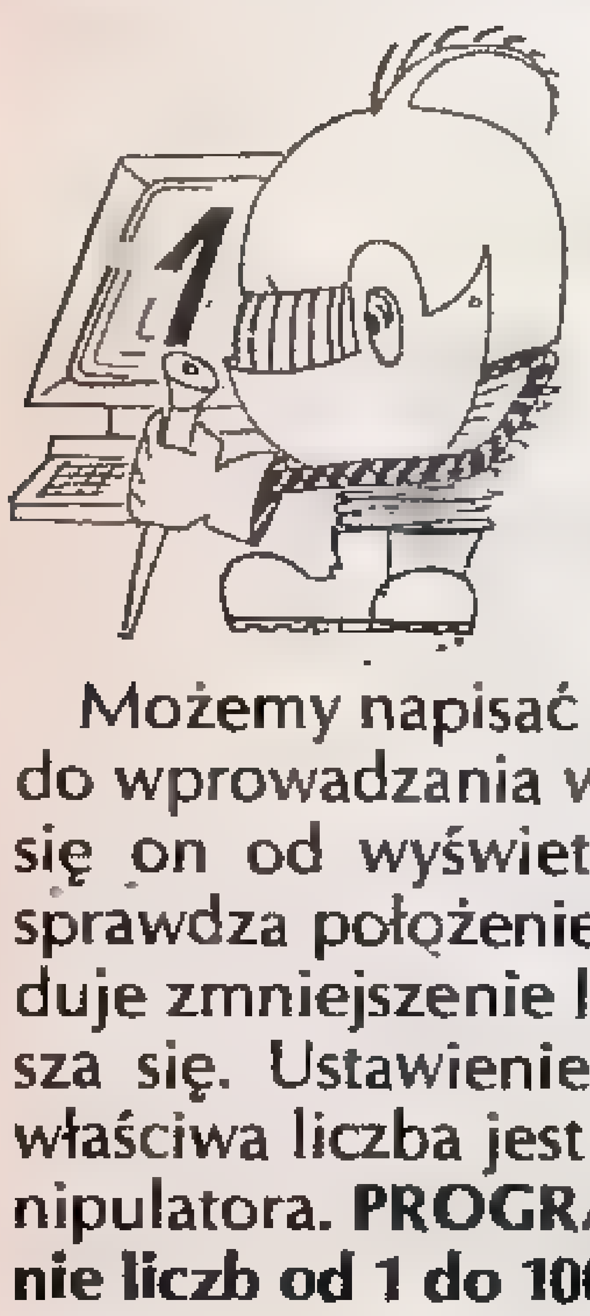
```
QO 1 REM ****************
IB 2 REM *              *
TX 3 REM * Program nr 19 *
ID 4 REM *              *
QS 5 REM ****************
ID 10 DIM MSK$(2),R$(2),DAT$(8),R
    (1)
FC 20 MSK$="__"
KQ 59 REM Otwarcie kanalu klawiat
    ury
BO 60 OPEN #1,4,0,"K:"
YU 1000 PRINT CHR$(125);"Podaj ak
    tualna date w postaci:
              rok-miesiac-dzien."
RW 1010 IC=15:IR=4
ZQ 1020 GOSUB 8200:REM Wprowadzen
    ie daty
IM 1030 POSITION 15,6
EW 1040 PRINT DAT$
FC 1050 END
XW 8197 REM ********************
OF 8198 REM * Wprowadzenie daty *
YE 8199 REM ********************
WG 8200 POSITION IC,IR:PRINT CHR$
    (253);
IA 8210 PRINT MSK$;"-";MSK$;"-";M
    SK$;
WM 8220 POSITION IC,IR:PRINT CHR$
    (253);
DT 8230 GOSUB 8400:REM ROK
BO 8240 Y=VAL(R$)
HB 8250 IF Y<0 OR Y>99 THEN 8200
AO 8260 DAT$(1,2)=R$:DAT$(3,3)="-
    "
SV 8270 PRINT CHR$(31);
LH 8280 GOSUB 8400:REM MIESIAC
ZJ 8290 M=VAL(R$)
KU 8300 IF M<1 OR M>12 THEN 8200
IP 8310 DAT$(4,5)=R$:DAT$(6,6)="-
    "
SI 8320 PRINT CHR$(31);
UL 8325 IF D>31 THEN 8200
IT 8330 GOSUB 8400:REM DZIEN
WU 8340 D=VAL(R$)
ZX 8350 IF D<1 THEN 8200
WW 8360 IF M=2 AND D>29 THEN 8200
SS 8370 IF (M=4 OR M=6 OR M=9 OR
    M=11) AND D>30 THEN 8200
VA 8375 IF D>31 THEN 8200
DV 8380 DAT$(7,8)=R$
CH 8395 RETURN
GS 8397 REM ******************
QB 8398 REM * Pobierz 2 cyfry *
HA 8399 REM ******************
FM 8400 FOR J1=0 TO 1
BI 8410 GET #1,R:R(J1)=R
AB 8419 REM Klawisz <BACK SPACE>
    oznacza restart
AO 8420 IF R=126 THEN POP :GOTO 8
    200
OL 8429 REM Ignorowanie klawiszy
    literowych
PB 8430 IF R<48 OR R>57 THEN 8410
PF 8440 PRINT CHR$(R);:REM ECHO
CU 8450 NEXT J1
QZ 8460 R$=CHR$(R(0)):R$(2)=CHR$(
    R(1))
BJ 8470 RETURN
```

planszy i oglądamy ją przez wizjer kamery. Plansza jest wystarczająco duża, aby nie można było jej objąć w całości, lecz możemy obejrzeć dowolną jej część przez przesuwanie wizjera w górę, w dół, w prawo lub w lewo. Ekran może imitować wizjer, a manipulator może sterować jego ruchem nad polem danych.

Pokażemy, jak zastosować opisaną technikę dla tablicy numerycznej dwuwymiarowej. Jako wartość każdego elementu tablicy przydzielimy liczbę czterocyfrową, która określa numer wiersza i kolumny danego elementu. Na ekranie występują nagłówki wierszy i kolumn. Utworzone okno jest mniejsze niż cały ekran, aby lepiej zilustrować koncepcję okna nad danymi. Nie ma przeszkód, aby utworzyć okno zajmujące cały ekran.

Cały PROGRAM (NR 13) jest stosunkowo prymitywny. Ma tylko jedną szybkość. Zużywa około dwóch sekund na ponowne wyświetlenie okna za każdym razem, gdy zmieni się numer wiersza lub kolumny; daje to 20s na przesunięcie okna o dziesięć wierszy. Można poprawić nieco program i skrócić ten czas dwukrotnie, lecz dziesięć sekund to nadal zbyt długo. Zamiast ciągłego wyświetlania okna tak często jak to możliwe, gdy manipulator wychylony jest w jednym kierunku, program mógłby wyświetlać ponownie dopiero wtedy, gdy przyjmie on położenie środkowe. W ten sposób okno wyświetli się tylko raz dla każdego ciągłego ruchu. Możemy w ten sposób zredukować czas przesuwania okna do dwóch sekund plus czas na odchylenie manipulatora od środka. Oczywiście program będzie aktualizował numery kolumn w miarę przesuwania okna poziomo i numery wierszy w czasie ruchu pionowego, zatem zna on położenie okna. Spróbujcie sami dokonać tych zmian w programie.

Można także przesuwać okno po przekątnej. Spróbujcie rozszerzyć program nr 13, między liniami 300 i 330 aby umożliwić taki ruch okna. Po wykryciu ukośnej pozycji manipulatora program musi zmienić jednocześnie wiersz i kolumnę, zmienne R i C.



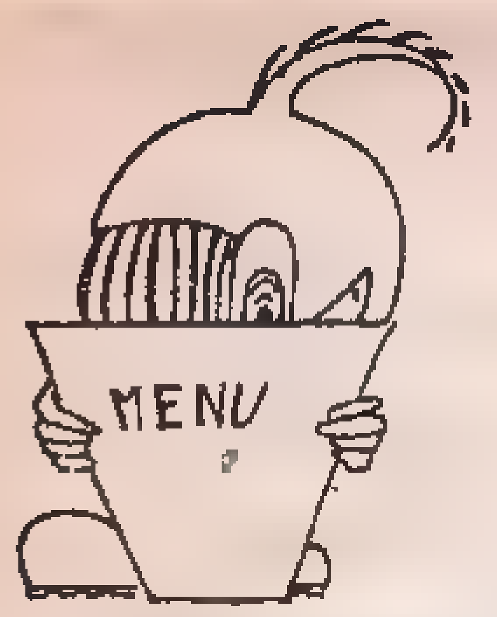
Wprowadzanie liczb manipulatorem

Możemy napisać program, który używa manipulatora do wprowadzania wartości numerycznych. Rozpoczyna się on od wyświetlania liczby na ekranie. Następnie sprawdza położenie manipulatora. Ruch w lewo powoduje zmniejszenie liczby. Ruch w prawo i liczba zwiększa się. Ustawienie na środku przerywa zmiany. Gdy właściwa liczba jest na ekranie, naciskamy przycisk manipulatora. **PROGRAM NR 14 demonstruje wprowadzanie liczb od 1 do 100.**

Program działa poprawnie, lecz niełatwo jest wybrać określoną liczbę. Jest on zbyt wrażliwy na ruch manipulatora oraz zbyt często sprawdza jego położenie. Należy wprowadzić pętlę opóźniającą (linia 6560).

Lepiej wykonany jest PROGRAM NR 15. Używa tych samych zmiennych co ogólny podprogram wprowadzania dla określenia zakresu wprowadzania wartości (LO i HI), położenia kursora (IR i IC) i wielkości pola (IL). Wprowadzoną wartość zwraca w zmiennej R. Podprogram z tego programu używa dwóch pętli opóźniających do sterowania szybkości zmiany liczby (linia 6560). Startując od małego opóźnienia (linia 6570) dla maksymalnej szybkości zmian. Gdy manipulator jest w położeniu centralnym, podprogram przestawia się na małą szybkość zmian (linia 6600). Jeśli zatem przesuniemy manipulator w prawo lub w lewo i przytrzymamy, liczba zmienia się z dużą szybkością. Szybkie przesunięcie w lewo lub prawo i zmiany następują wolniej.

```
QO 1 REM ****************
IB 2 REM *              *
NK 3 REM * Program nr 20 *
ID 4 REM *              *
QS 5 REM ****************
IK 9 REM Inicjowanie zmiennych
DK 10 DIM MSK$(40),R$(20)
WZ 30 MSK$="_":MSK$(40)=MSK$:MSK$(2)=MSK$
KP 49 REM Otwarcie kanalu klawiatury
BN 50 OPEN #1,4,0,"K:"
QB 89 REM Skok do programu slownego
IC 90 GOTO 1000
MF 595 REM ***************
BH 596 REM *  Podprogram  *
FH 597 REM * wprowadzania *
SZ 598 REM * ciagu znakow *
MR 599 REM ***************
AZ 600 R$=""
UL 610 POKE 755,0:REM Wylaczenie kursora
FA 630 POSITION IC,IR
PH 640 PRINT MSK$(1,IL)
SN 649 REM Poczatek pola
FE 650 POSITION IC,IR
BU 659 REM Wczytaj znak
KE 660 GET #1,R
SZ 670 J=LEN(R$)
AM 679 REM Wlaczenie kursora jesli RETURN
LZ 680 IF R=155 THEN POKE 755,2:RETURN
PR 689 REM Jesli znak dobry to dodaj do ciagu
AF 690 IF R>=32 AND R<=95 AND J<IL THEN R$(J+1,J+1)=CHR$(R):PRINT CHR$(R);:GOTO 660
OJ 699 REM Czy znak kasowania
VO 700 IF R<>126 OR J=0 THEN GOTO 660
CN 709 REM Kasowanie ostatniego znaku
VI 710 POSITION IC+J-1,IR
MV 720 PRINT MSK$(1,1);
VM 730 POSITION IC+J-1,IR
UX 740 IF J>1 THEN R$=R$(1,J-1)
RC 750 IF J=1 THEN R$=""
QK 760 GOTO 660
GI 1000 PRINT CHR$(125):REM CLS
XD 1010 POSITION 2,4
LH 1020 PRINT "Imie: "
AV 1030 IC=8:IR=4:IL=20:GOSUB 600
AC 1040 POSITION 2,8
VV 1050 PRINT "WPROWADZONO: ";R$
NB 1060 CLOSE #1
FJ 1070 END
```



Wykorzystanie menu do ograniczenia wybrów

Najłatwiejszym sposobem wyeliminowania błędów jest uważne zaprojektowanie programu w taki sposób, aby użytkownik miał możliwie jak najmniej możliwości wyboru. Istota pytań zadawanych użytkownikowi przez program może uczynić jego pracę łatwą lub trudną. W niektórych programach trzeba wypełnić puste miejsca. Czasami wypełnianie jest jedynym wyborem. Kiedy indziej będzie pytanie o wielu wyborach. Zamiast „Co chcesz zrobić?", program pyta „Którą opcję wybierasz?". Na tym polega istota menu.

Można wrócić do programu nr 6, który wprowadza polecenia. Do wyboru były: A, C, D, E, P lub H. Zamiast tego można przygotować menu realizujące to samo. Korzystając z prezentowanych podprogramów możemy łatwo napisać program do wyświetlenia menu i wprowadzania poleceń.

Wykorzystanie menu jest lepsze dla użytkownika i programisty. Użytkownik nie musi pamiętać lub szukać dostępnych opcji. Programista nie musi pisać skomplikowanych procedur, które wyświetlą rozszerzenia poleceń. Nie ma gwarancji, że użytkownik wprowadzi tylko wyświetlone opcje, zatem program wciąż musi sprawdzać poprawność wprowadzanych informacji.

Prawie wszystkie wejścia mogą być rozłożone na ciąg pytań o wielu wyborach. Każde z nich może być przedstawione jako menu. Użytkownik przechodzi swoją drogą przez menu, aż do udzielenia odpowiedzi na końcowe pytanie.



Używanie manipulatora do wyboru menu

Komputer może być zaprogramowany do przesuwania kursora po ekranie pod kontrolą manipulatora. Jeśli na ekranie wyświetlone jest menu, użytkownik przesuwa kursor, aż osiągnie jeden z wyborów menu. Wtedy naciska przycisk manipulatora dla dokonania wyboru. Program to wykrywa, zna położenie kursora i określa, która pozycja menu odpowiada temu położeniu.

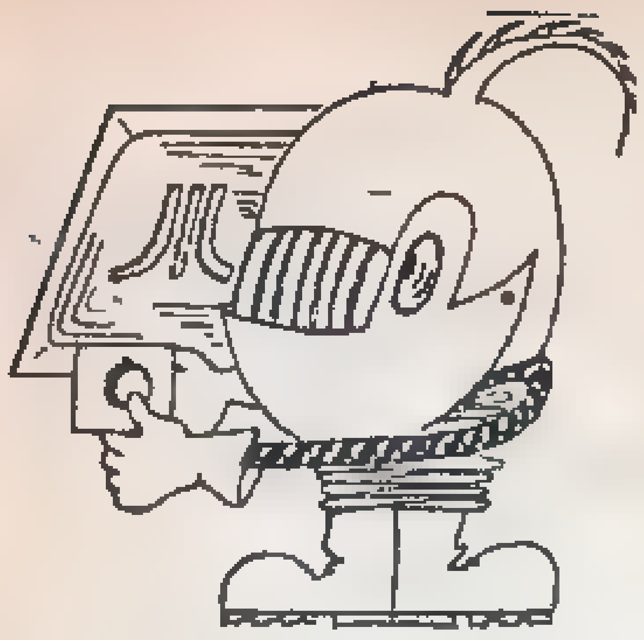
Podprogram kursora z PROGRAMU NR 16 „przywiązuje" manipulator do kursora. Wykrywa położenie manipulatora, wyświetla odpowiednie znaki ruchu kursora i aktualizuje wartości zmiennych przechowujących położenie kursora. **PROGRAM NR 16 demonstruje jego działanie.**

Następny PROGRAM (NR 17) używa opisanej procedury do wprowadzania wyboru z menu. Wyświetla menu i umieszcza kursor obok pierwszego polecenia. Manipulator może przesuwać kursor w górę i dół, nie może jednak przesuwać go w bok. Jak powyżej, naciśnięcie przycisku zatrzymuje kursor; zmienne SC i SR przechowują położenie kursora.

Pozostaje przetłumaczenie pozycji kursora na wybrane polecenie i działanie zgodnie z nim. **W PROGRAMIE NR 17 całe działanie sprowadza się do wyświetlenia nazwy wybranego polecenia.**

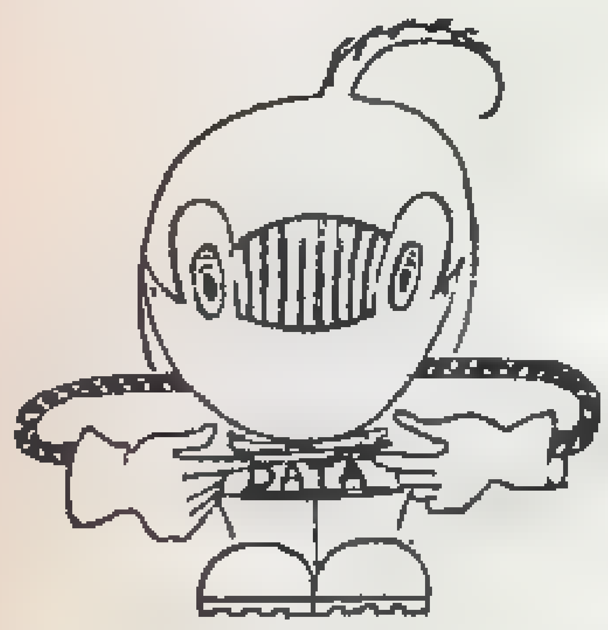
```
QO 1 REM ****************
IB 2 REM *              *
OF 3 REM * Program nr 21 *
ID 4 REM *              *
QS 5 REM ****************
OA 845 REM *************
PN 846 REM * Wylaczenie *
XN 847 REM *  klawisza  *
EN 848 REM *   BREAK    *
OM 849 REM *************
SS 850 J=PEEK(16)-128
SD 860 IF J<0 THEN RETURN
TZ 870 POKE 16,J
KV 880 POKE 53774,J
ZW 890 RETURN
```

```
1 REM ***************
2 REM *             *
3 REM * Program nr 22 *
4 REM *             *
5 REM ***************
19 REM Kody ATASCII symboli uz
ywanych w grach
20 DATA 0,16,19,20,96,123
999 REM Otwarcie kanalu we/wy
ekranu
1000 OPEN #6,12,0,"S:"
1010 PRINT CHR$(125)
1019 REM Wyswietlenie zestawu
symboli
1020 PRINT "Wybierz swoj symbo
l"
1030 RESTORE 20
1040 FOR J=3 TO 8
1050 READ TOKEN
1060 POSITION 4,J
1070 PRINT CHR$(TOKEN)
1080 NEXT J
1089 REM Start kursora w 4,3
1090 POSITION 4,2
1100 PRINT CHR$(29);
1110 SC=4:SR=3
1119 REM Okreslenie zakresu ku
rsora
1120 LC=4:RC=4:TR=3:BR=8
1129 REM Szybkosc kursora
1130 DLY1=30
1140 GOSUB 6000:REM Przesuwani
e kursora
1149 REM Pobranie znaku z ekra
nu
1150 LOCATE SC,SR,TOKEN
1159 REM Zamiana symbolu na ne
gatyw
1160 TOKEN=TOKEN-128*SGN(TOKEN
-127)
1170 POSITION 2,12
1180 PRINT "Wybrales  ";CHR$(T
OKEN)
1900 END
```



Wprowadzanie znaków manipulatorem

Używając funkcji **LOCATE** oraz podprogramu kursora możliwe jest wprowadzenie dowolnego znaku bezpośrednio z ekranu za pomocą manipulatora. Program wyświetla najpierw znaki, z których można wybierać. Użytkownik przesuwa kursor od znaku do znaku manipulatorem. Gdy kursor zatrzyma się na żądanym znaku, trzeba nacisnąć przycisk. Program odczytuje wartość z ekranu (za pomocą LOCATE). **Technikę tę ilustruje PROGRAM NR 22.** Wymaga on podprogramu kursora (z programu nr 16).



Wprowadzanie daty

Aby wprowadzić datę musimy podać rok, miesiąc i dzień jako liczby dwucyfrowe. Program dostarcza separatorów rozdzielających poszczególne pozycje. Separatorem może być kreska ułamkowa lub dowolny inny znak.

Użytkownik musi wiedzieć, gdzie wprowadzić następne dane. Dlatego program będzie używał maski wprowadzania. **Instrukcje z PROGRAMU NR 18 akceptują wejście dwucyfrowe.** Program automatycznie kończy wprowadzanie danych po podaniu dwóch cyfr. Użytkownik nie musi naciskać klawisza **RETURN.**

Istnieją różne sposoby wyeliminowania pewnych błędów podczas wprowadzania daty. Należą do nich:

— przyjmowanie tylko znaków numerycznych (cyfr),

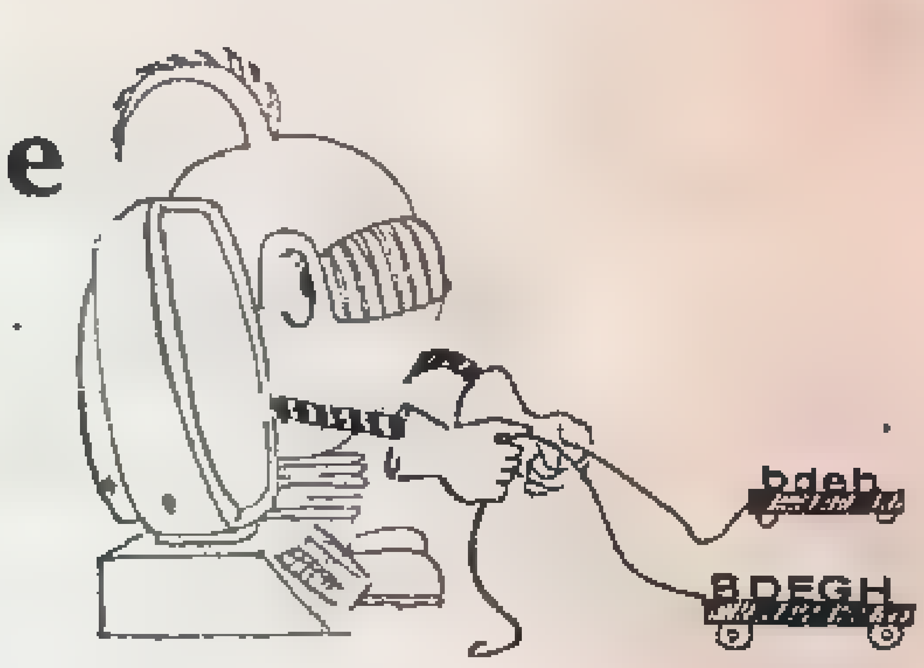
— testowanie poprawności roku, miesiąca i dnia,

— dostarczenie środków ponownego wprowadzania daty.

PROGRAM NR 19 zawiera cały podprogram wprowadzania daty obejmujący wymienione ulepszenia. Dozwolone są tylko dane numeryczne (linia 8430). Miesiąc musi mieć wartość między 1 a 12 (linia 8300). Program nie bierze pod uwagę setek lat, lecz sprawdza maksymalną liczbę dni w określonym miesiącu (linie 8350 do 8375). Dozwolony jest dowolny rok od 00 do 99 (linia 8250). Wprowadzenie niepoprawnej daty wznawia cały ciąg wprowadzania. Jeśli użytkownik naciśnie klawisz **BACK SPACE**, również wznawiany jest ciąg wprowadzania daty (linia 8420).

Zauważmy, że data tworzona jest w postaci ciągu ośmioznakowego **DAT$** jako rok, miesiąc i dzień (linia 8260, 8310 i 8380).

Wprowadzanie ciągu znaków



Uniwersalny PODPROGRAM NR 11 wprowadzania miał znaczne uproszczenia. Umożliwiał użytkownikowi wprowadzanie pozycji, które były dłuższe od maski wejścia. Co gorsze, użytkownik mógł przesuwać kursor przez cały ekran klawiszami strzałek. Wszystko to powodowało, że wcześniej czy później użytkownik mógł zniszczyć obraz. Jeśli użyjemy instrukcji GET zamiast INPUT, możemy dokładniej kontrolować wprowadzanie.

PROGRAM NR 20 zawiera podprogram, który wprowadza ciąg znaków. Zauważmy, że naciśnięcie klawisza RETURN powoduje wygenerowanie znaku EOL, który kończy wprowadzanie. Klawisz BACK'S powoduje cofnięcie o jeden znak.

Istnieje kilka sposobów polepszenia tego podprogramu. Poniżej wymienione są pewne pomysły:

— użycie zmiennej do określania numeru kanału wejściowego,

— zezwolenie na wprowadzanie wielkich i małych liter (kody ATASCII 97 do 122),

— dodanie na początku instrukcji TRAP w celu uaktywnienia procedury obsługi błędu od linii ERRHDL,

— wywołanie podprogramu wyświetlania instrukcji rozszerzonych (od linii **AMPSUB**), jeśli wprowadzony jest określony znak.

W ostatnim kroku przed wyjściem z podprogramu powinniśmy sprawdzić, czy wprowadzone informacje są numeryczne. Jeśli tak, to trzeba przetworzyć ciąg znaków na wartość numeryczną i sprawdzić jej zakres. Możemy polegać na instrukcji **TRAP** i podprogramie obsługi błędu w zakresie wykrywania błędów wprowadzania. Nie próbujmy sprawdzać zakresu liczby przed przetłumaczeniem jej na wartość numeryczną.



Działania na zmiennych tekstowych

Dużą wadą języka Atari BASIC jest brak tablic tekstowych. Można ją jednak wyeliminować przez symulowanie takich tablic za pomocą zmiennych tekstowych. Taki twór nazwiemy pseudotablicą. Nasze działanie polega na podzieleniu ciągu znaków na podciągi o stałej długości (warunek podstawowy) i traktowaniu każdego podciągu jako elementu pseudotablicy. W celu wyznaczenia pozycji początkowej elementu pseudotablicy musimy znać numer i długość każdego elementu. Istnieją dwa podstawowe ograniczenia:

1) wszystkie elementy pseudotablicy muszą mieć taką samą długość. Zwykle określa ją najdłuższy element. Jeśli wprowadzany ciąg jest krótszy, jego nieużywaną część musimy wypełnić spacjami,
2) obliczanie adresu początkowego elementu pseudotablicy jest czasochłonne. Można zauważyć opóźnienia w realizacji programu.

Program „SYMULACJA TABLICY TEKSTOWEJ" wykorzystuje pseudotablicę przechowywaną w A$. Zawiera ona dziesięć elementów o długości dziesięciu znaków każdy. Ele-

```
HQ 1 REM *************
ER 2 REM *           *
QC 3 REM * Symulacja *
FT 4 REM * tablicy   *
SA 5 REM * tekstowej *
EV 6 REM *           *
HW 7 REM *************
HO 10 DIM A$(100),TEMP$(10),BL$(1
   0)
LK 19 REM Dziesiec wartosci ciagu
IL 20 DATA Ryszard,Natalia,Antoni
   ,Marek,,12345678901,+++,A+B=1E
   100,**=,abrakadabra
KK 29 REM zapelnienie BL$ spacjam
   i
MU 30 BL$=" ":BL$(10)=BL$:BL$(2)=
   BL$
QP 39 REM Przydzielenie wartosci
   pseudo-tablicy
HS 40 FOR N=1 TO 10
VW 49 REM Najpierw ustawienie war
   tosci do ciagu przejsciowego
NP 50 READ TEMP$
MZ 59 REM Wyznaczenie dlugosci no
   wej wartosci
SD 60 TL=LEN(TEMP$)
QX 69 REM Uzupelnienie ciagu spac
   jami
VV 70 IF TL<10 THEN TEMP$(TL+1)=B
   L$
IB 79 REM Wyznaczenie adresu elem
   entu tablicy
ZP 80 START=(N-1)*10+1
PK 89 REM Przydzielenie wartosci
   do elementu tablicy
QP 90 A$(START)=TEMP$
HP 100 NEXT N
VV 109 REM Wyswietlenie przydziel
   onych wartosci
SG 110 FOR N=1 TO 10
HI 119 REM Obliczenie poczatku el
   ementu
QN 120 START=(N-1)*10+1
TE 130 PRINT "Element ";N;" to: "
   ;A$(START,START+9)
HX 140 NEXT N
OA 150 END
```

```
NY 1 REM *************
DR 2 REM *           *
XE 3 REM * Podprogram *
KX 4 REM * sortowania *
OC 5 REM * elementow  *
EV 6 REM *  tablicy   *
DW 7 REM *           *
OF 8 REM *************
NO 9 REM
ZS 22050 DATA 104,104,104,133,205
   ,104,104,133,206,104,104,133,2
   03,104,104,133,204
DJ 22060 DATA 104,133,217,104,133
   ,216,104,133,209,104,133,208,1
   69,0,133,218,133,207,162
DS 22070 DATA 1,165,216,133,214,1
   65,217,133,215,24,165,214,133,
   212,101,205,133,214,165,215
GE 22080 DATA 133,213,105,0,133,2
   15,164,203,165,206,240,10,177,
   214,209,212,144,44,240,12
OF 22090 DATA 176,19,177,214,209,
   212,144,13,240,2,176,30,200,19
   6,204,240,227,176,23,144
OI 22100 DATA 223,169,1,133,218,1
   64,205,136,177,214,72,177,212,
   145,214,104,145,212,192,0
ON 22110 DATA 208,241,232,224,0,2
   08,2,230,207,228,208,208,172,1
   65,209,197,207,208,166,165
AM 22120 DATA 218,201,0,208,144,9
   6
YD 22200 W=0:V=1:RESTORE 22050
JX 22210 DIM SO$(142)
GJ 22220 FOR I=1 TO 142:READ A:SO
   $(I,I)=CHR$(A):NEXT I
EP 22225 RETURN
```

```
FK 1 REM ***************
FJ 2 REM *              *
AZ 3 REM *  Podprogram  *
EV 4 REM * wyszukiwania *
JH 5 REM *   elementu   *
AN 6 REM *   tablicy    *
FO 7 REM *              *
FR 8 REM ***************
NO 9 REM
VT 20000 DATA 216,104,104,133,204
   ,104,133,203,104,104,133,205,1
   04,104,133,206,104,133,208,104
   ,133,207,104,104
VO 20010 DATA 133,209,104,104,133
   ,210,169,0,133,212,133,213,24,
   165,203,101,209,133,203,165,20
   4,105,0,133
VH 20020 DATA 204,164,210,136,177
   ,203,209,207,208,4,136,16,247,
   96,24,165,203,101,205,133,203,
   165,204,105
IP 20030 DATA 0,133,204,24,165,21
   2,105,1,133,212,197,206,208,21
   9,96,-1
NG 20040 RESTORE 20000:DIM SZ$(87
   )
BN 20050 FOR I=1 TO 87
EY 20060 READ A:SZ$(I,I)=CHR$(A)
GM 20070 NEXT I
FX 20080 READ A:IF A=-1 THEN RETU
   RN
JO 20090 ? "BLAD DANYCH":STOP
```

menty tablicy pobierane są z instrukcji **DATA**. Włączono do nich ciąg pusty oraz dwa ciągi zbyt długie, aby zmieścić się w jednym elemencie tablicy. Każdy ciąg z **DATA** wstawiany jest do zmiennej tekstowej **TEMP$**. Jest to konieczne, ponieważ Atari BASIC nie zezwala na używanie zmiennych ze wskaźnikami w instrukcji **READ**. Program dołącza spacje do krótkich ciągów w celu usunięcia pozostałości poprzednich wartości (linia 60 i 70). W końcu wyznacza indeks elementu pseudotablicy i przydziela mu wartość utworzoną w zmiennej **TEMP$**. Obejrzyjmy, co stało się z pustym ciągiem i wartościami, które są zbyt długie.

Elementy tablicy tekstowej lub pseudotablicy są zwykle rekordami składającymi się z kilku pól. Większość działań wykonywanych na nich polega na sortowaniu rekordów według określonego klucza, którym jest zwykle jedno z pól oraz wyszukiwaniu rekordu na podstawie zawartości jednego z pól. Atari BASIC, w odróżnieniu od nowszych, bardziej rozbudowanych wersji tego języka, nie ma niestety wbudowanych funkcji sortowania i wyszukiwania. Podprogramy realizujące te funkcje są bardzo łatwe do napisania i zawierają kilka linii programu. Dlatego nie śmielibyśmy nawet prezentować ich czytelnikom. Mają one jednak podstawową wadę: bardzo długi czas działania, rzędu kilku a nawet kilkunastu minut. Zdarzyło się nam stracić cierpliwość podczas oczekiwania na posortowanie kilkudziesięciu rekordów. **Efektem tego są dwa podprogramy w języku maszynowym: SORTOWANIE oraz WYSZUKIWANIE.** Po ich wykorzystaniu czas oczekiwania na zakończenie tych operacji uległ znacznemu skróceniu.

Oba podprogramy dostarczamy w postaci umożliwiającej włączenie ich do własnych programów. Po wykonaniu skoków do odpowiednich podprogramów, za pomocą instrukcji **GOSUB**, interesujące nas podprogramy zostają umieszczone w ciągach. Są to procedury przemieszczalne (relokowalne) i mogą być umieszczone gdziekolwiek w programie.

Sposoby wywołania obu podprogramów są podobne. I tak wywołanie procedury sortowania ma następującą postać:

A=USR (ADR (SO$), DR, TYP, PP, PK, ADR (X$), IR)

gdzie poszczególne parametry mają następujące znaczenie:

- SO$ — nazwa ciągu zawierającego podprogram sortowania,
- DR — długość rekordu (element pseudotablicy),
- TYP — rodzaj sortowania: 0 — wzrastający, 1 — malejący,
- PP — pozycja początkowa klucza sortowania w rekordzie,
- PK — pozycja końcowa klucza sortowania w rekordzie
- X$ — nazwa pseudotablicy zawierającej sortowane elementy,
- IR — ilość elementów pseudotablicy (rekordów), które należy posortować.

Pozycje początkowe i końcowe klucza sortowania powinny być podane w postaci liczby bajtów od początku rekordu. Na przykład, pierwsza pozycja rekordu ma przesunięcie 0, druga —1, a setna —99.

Podprogram ten sortuje rekordy o stałej długości od 2 do 250 bajtów. Klucz sortowania może być umieszczony gdziekolwiek w rekordzie i może być dowolnej długości. Rekordy muszą być w postaci znaków ATASCII.

Wywołanie podprogramu wyszukiwania rekordu na podstawie zawartości jednego z jego pól ma następującą postać:

K=USR (ADR (SZ$), ADR (D$), DR, IR-1, ADR (P$), KP, DP)

gdzie poszczególne parametry mają następujące znaczenie:

- SZ$ — nazwa ciągu zawierającego podprogram wyszukiwania,
- D$ — nazwa ciągu zawierającego elementy pseudotablicy,
- DR — długość rekordu
- IR — ilość rekordów (elementów pseudotablicy),
- P$ — nazwa ciągu zawierającego poszukiwane pole,
- KP — przesunięcie poszukiwanego pola względem początku rekordu,
- DP — długość poszukiwanego pola.

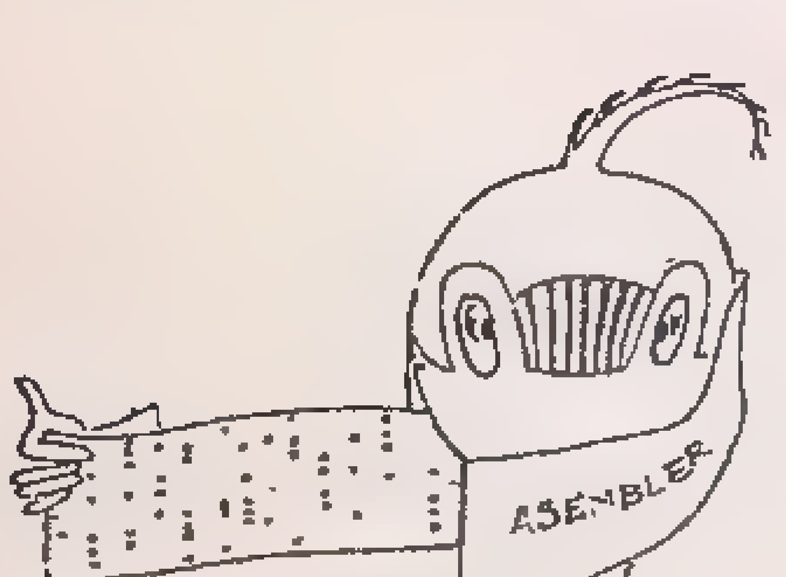
Po zakończeniu działania zmienna K będzie zawierać numer rekordu zawierającego poszukiwane pole (pierwszego jaki wystąpił).



Unieszkodliwienie klawisza BREAK

Nawet najlepiej zaprojektowany program wciąż ma słaby punkt. Można zatrzymać jego realizację (czasami przypadkowo) poprzez naciśnięcie klawisza **BREAK**. Automatycznie na ekranie pojawia się komunikat, który rujnuje pieczołowicie utworzony obraz. Prawdopodobnie nie będzie możliwe poprawne kontynuowanie programu po takim przerwaniu, ponieważ instrukcja **CONT** wznawia działanie od początku linii programu, w której nastąpiło przerwanie. Jeśli zdarzy się, że jest to linia o wielu instrukcjach, pewne z nich będą wykonane ponownie.

Istnieje sposób unieszkodliwienia klawisza BREAK (PROGRAM NR 21). Niestety jest on wrażliwy na wiele zdarzeń, które przywracają normalne działanie klawisza. Należą do nich: naciśnięcie klawisza RESET, pierwsza instrukcja **PRINT**, dowolna instrukcja **OPEN** dotycząca ekranu (urządzenia o kodach E: lub S:), pierwsza instrukcja **PRINT** po takim **OPEN** oraz **GRAPHICS**. Najprostszym sposobem pokonania tych ograniczeń jest częste wykonywanie podprogramu wyłączenia klawisza **BREAK**. Dobrym miejscem wywoływania go jest podprogram wprowadzania.



Asembler

Wykorzystanie firmowego programu Assembler-Editor przez posiadaczy komputera Atari z pamięcią zewnętrzną w postaci magnetofonu odznacza się wieloma niedogodnościami. Należą do nich;

— długi czas wprowadzania programu (ok. 15 minut) spowodowany tym, że wymaga uprzedniego załadowania Translatora, czyli starej wersji systemu operacyjnego (opracowanej dla komputerów Atari 400 i 800),
— konieczność przestudiowania kilkudziesięciostronicowej instrukcji obsługi programu,
— program wynikowy, przechowany na kasecie magnetycznej i wprowadzony do komputera, nie chce działać.

W celu uniknięcia opisanych trudności można wykorzystać prezentowany program, który udaje asembler (to znaczy symuluje pewne jego funkcje). Umożliwia on przygotowanie krótkich podprogramów w języku maszynowym i przechowanie ich w wybranym obszarze pamięci (najczęściej na stronie 6 RAM). W programie tym nie ma odrębnych faz edycji i translacji, wprowadzane rozkazy wpisywane są (POKE) bezpośrednio do wybranego obszaru pamięci. Rozkazy podawane są w postaci symbolicznej. Wartości argumentów mogą być wprowadzone dziesiętnie lub szesnastkowo, zgodnie z konwencją przyjętą po uruchomieniu programu, która może być zmieniona w każdej chwili. Istnieje możliwość podania skoku do przodu bez znajomości aktualnego adresu przeznaczenia. W tym celu w charakterze argumentu podajemy słowo **WPRZÓD**. Program wstawia wstępnie wartość zerową do bajtu argument. Aktualny adres skoku zostanie wpisany po rozpoznaniu słowa **ETYKIETA**. Wtedy do zaznaczonej komórki wprowadzany jest adres bieżącej komórki (a dokładniej jego odległość od rozkazu skoku).

Komenda **LIST** umożliwia wyświetlenie adresów i zawartości poszczególnych komórek. Możliwe jest również umieszczenie dowolnej liczby, z przedziału 0—255, w wybranej komórce pamięci. Służy do tego funkcja **WSTAW**, po której należy podać adres i zawartość komórki, obie wartości koniecznie dziesiętnie.

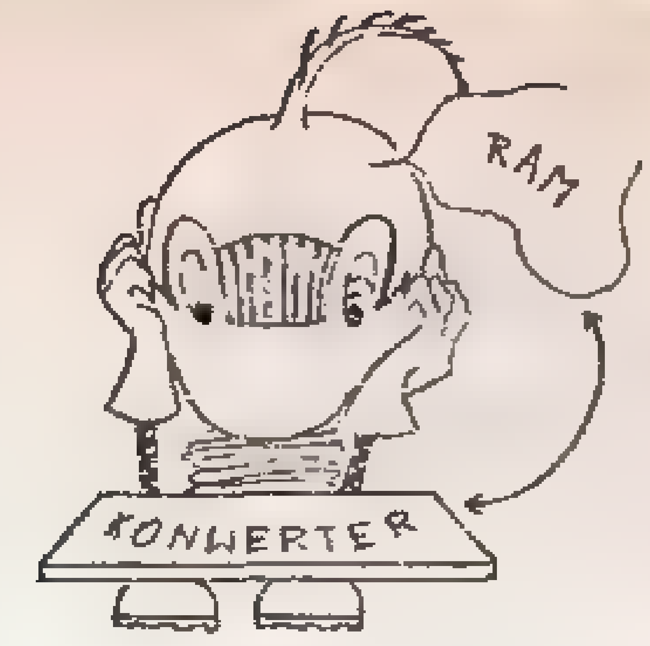
Za pomocą poleceń **HEX** lub **DZIES** w każdej chwili możemy przejść z jednego systemu liczenia do drugiego. Zakończenie działania programu uzyskamy po napisaniu **KONIEC**. Otrzymamy wówczas adresy początku i końca procedury w języku maszynowym.

Prezentowany program nie pretenduje do roli prawdziwego asemblera, brakuje mu wielu funkcji oraz możliwości podawania nazw symbolicznych komórek. Pomimo to może ułatwić początkową naukę języka asemblera. Zamiast wstawiać kody rozkazów za pomocą **POKE**, możemy pisać je w postaci symbolicznej, jak w prawdziwym asemblerze. Przydaje się on do pisania prostych i krótkich procedur w języku maszynowym.

```
HQ 1 REM *************
ER 2 REM *           *
OB 3 REM * ASSEMBLER *
ET 4 REM *           *
HU 5 REM *************
HM 10 HX=0:REM IF HX=0 TO ASEMBLE
   R DZIALA NA LICZBACH DZIESIETN
   YCH
CT 20 DIM HE$(16),ZO$(3),R$(10),M
   N$(12),ZA$(1),AZ$(1),L$(3),SA$
   (4),H$(4),LR$(1)
HW 30 OPEN #1,12,0,"E:"
DY 50 HE$="0123456789ABCDEF":SZ=1
   :ZO$="000"
EU 100 ? "   KONWENCJA STOSOWANA 
   W ASSEMBLERZE"
LA 110 DIM M$(56*3),TY(56),OP(56)
CU 120 FOR I=1 TO 56:READ MN$:M$(
   I*3-2,I*3)=MN$(1,3)
MB 122 TY(I)=VAL(MN$(4,4)):OP(I)=
   VAL(MN$(5))
KI 130 NEXT I:? :? :?
ZH 140 ? "Sposob adresowania":?
XE 150 ? "Bezposredni          LDA
    #15"
SG 155 ? "Absolutny            LDA
    1500"
GE 160 ? "Strony zerowej       LDA
    15"
NQ 165 ? "Akumulator           ASL
   "
QN 170 ? "Posredni X           LDA
    (15X)"
UY 175 ? "Posredni Y           LDA
    (15)Y"
TD 177 ? "Strony zerowej  X    LDA
    15X"
WD 179 ? "Strony zerowej  Y    LDX
    15Y"
ZZ 180 ? "Absolutny  X         LDA
    1500X"
CR 185 ? "Absolutny  Y         LDA
    1500Y"
GN 190 ? "Liczby dziesietne D czy
    szesnastkowe S";:INPUT MN$
KW 192 HX=0:IF MN$(1,1)="S" THEN
   HX=1
JV 194 POSITION 0,13:? CHR$(156);
   CHR$(156);:IF HX=1 THEN ? "SZE
   SNASTKOWO"
XO 195 IF HX=0 THEN ? "DZIESIETNI
   E"
BU 197 ? :? "Adresy: stosuj 1536-
   1791 ($0600-$06FF)":? :?
DA 200 ? CHR$(156);CHR$(156);"Pod
   aj adres poczatku ":? "  twoj
   ego programu";:INPUT SA$
WX 205 IF SA$="" THEN ? CHR$(28);
   CHR$(28);:GOTO 200
VR 210 IF HX=1 THEN H$=SA$:GOSUB
   5000:SA=DE:GOTO 217
NC 215 SA=VAL(SA$)
JQ 217 IF SA<256 OR SA>=40960 THE
   N ? CHR$(28);CHR$(28);CHR$(28)
   ;CHR$(28);"Zly adres !!!":? :G
   OTO 200
GE 220 TA=SA:? CHR$(12):GOTO 230
AI 225 ? :? CHR$(253);"BLAD WEJSC
   IA":? :IF HX=1 THEN ? "(n.p. #
   5 winno byc #05)":?
JY 230 IF HX=1 THEN DE=SA:SZ=3:GO
   SUB 4000:? H$;": ";:GOTO 240
SI 235 ? SA;": ";
CG 240 TRAP 225:INPUT #1;MN$:? CH
   R$(28);:POKE 85,20:IF MN$="" T
   HEN ? CHR$(156);:GOTO 230
OF 241 REM NOWE OPERACJE
IP 242 IF LEN(MN$)>6 THEN IF MN$(
   LEN(MN$)-5)="WPRZOD" THEN FB=S
   A:GOTO 260
SD 243 IF MN$="ETYKIETA" THEN FR=
   SA-FB:POKE FB+1,FR-2:? "  OK":
   GOTO 230
VY 244 IF MN$="WSTAW" THEN ? :? "
   ADRES,LICZBA(DZIS.)",:INPUT AD
   DR,NUM:POKE ADDR,NUM:GOTO 230
FG 246 IF MN$="LIST" THEN GOSUB 9
   000:GOTO 230
VZ 248 IF MN$="HEX" THEN HX=1:?
   GOTO 230
QG 249 IF MN$="DZIES" THEN HX=0:?
    :GOTO 230
NX 250 IF MN$="KONIEC" THEN 8000
JE 260 L=LEN(MN$):L$=MN$(1,3)
MN 270 FOR I=1 TO 56:IF L$=M$(I*3
   -2,I*3) THEN 300
GI 280 NEXT I
QU 290 GOTO 850
TP 300 REM ROZKAZY JEDNOBAJTOWE
ZZ 301 TY=TY(I):OP=OP(I)
```

```
SC 305 IF FB=SA THEN TN=0 GOTO 20
    10
XX 310 IF TY=0 THEN GOTO 1000
VI 320 IF TY=3 THEN TY=1:IF L=3 T
    HEN OP=OP+8:GOTO 1090
IG 330 R$=MN$(5):IF HX=1 THEN GOS
    UB 6000
WH 340 LR$=R$(1,1):LL=LEN(R$):IF
    LR$="#" THEN 480
CY 350 IF LR$="(" THEN 520
EW 360 IF TY=8 THEN 600
EG 370 IF TY=3 THEN OP=OP+8:GOTO
    1000
SN 380 IF R$(LL)="X" OR R$(LL)="Y
    " THEN 630
ZH 390 IF L$(1,1)="J" THEN 820
VD 400 TN=VAL(R$):IF TN>255 THEN
    430
VK 410 IF TY=1 OR TY=3 OR TY=4 OR
     TY=5 THEN OP=OP+4
JX 420 GOTO 2000
HO 430 H=INT(TN/256):L=(TN-256*H)
    :IF TY=2 OR TY=7 THEN OP=OP+8:
    GOTO 470
NU 440 IF TY=1 OR TY=3 OR TY=4 OR
     TY=5 THEN OP=OP+12:GOTO 470
KZ 450 IF TY=6 OR TY=9 THEN 470
QQ 460 GOTO 850
KR 470 GOTO 3000
IT 480 TN=VAL(R$(2))
EU 490 IF TY=1 THEN OP=OP+8:GOTO
    2000
OZ 500 IF TY=4 OR TY=5 THEN GOTO
    2000
QH 510 GOTO 850
NG 520 IF R$(LL-1)=")Y" THEN 540
OA 530 IF R$(LL-1)="X)" THEN 570
QF 540 TN=VAL(R$(2,LL-1))
UY 550 IF TY=1 THEN OP=OP+16:GOTO
     2000
QR 560 GOTO 850
QL 570 TN=VAL(R$(2,LL-1))
ZV 580 IF TY=1 THEN GOTO 2000
QX 590 GOTO 850
GH 600 TN=VAL(R$):TN=TN-SA-2:IF T
    N<-128 OR TN>127 THEN ? "ZA DA
    LEKO";:GOTO 850
IS 610 IF TN<0 THEN TN=TN+256
JZ 620 GOTO 2000
NJ 630 IF R$(LL-1)=")Y" THEN 540
GV 640 IF R$(LL)="X" THEN 720
RE 650 REM *ZERO Y
ZW 660 TN=VAL(R$(1,LL-1)):IF TN>2
    55 THEN 680
FL 670 IF TY=2 OR TY=5 THEN 730
HT 675 IF TY=1 THEN 760
ET 680 GOSUB 770:IF TY=1 THEN OP=
    OP+24:GOTO 710
UQ 690 IF TY=5 THEN OP=OP+28:GOTO
     710
QH 700 GOTO 850
KI 710 GOTO 3000
SL 720 TN=VAL(R$(1,LL-1)):IF TN>2
    55 THEN GOSUB 770:GOTO 780
WL 730 IF TY=2 THEN OP=OP+16:GOTO
     760
PG 740 IF TY=1 OR TY=3 OR TY=5 TH
    EN OP=OP+20:GOTO 760
QR 750 GOTO 850
KI 760 GOTO 2000
NP 770 H=INT(TN/256):L=TN-256*H:R
    ETURN
QS 780 IF TY=2 THEN OP=OP+24:GOTO
     810
UR 790 IF TY=1 OR TY=3 OR TY=5 TH
    EN OP=OP+28:GOTO 810
QI 800 GOTO 850
KJ 810 GOTO 3000
GC 820 TN=VAL(R$)
WT 830 GOSUB 770
OO 840 GOTO 710
WN 850 ? CHR$(253);"BLAD":GOTO 23
    0
QQ 1000 REM ROZKAZY 1 BAJTOWE
PA 1010 POKE SA,OP:SA=SA+1:IF HX=
    1 THEN 1030
ZF 1020 ? OP:GOTO 230
UR 1030 DE=OP:GOSUB 4000:? H$:GOT
    O 230
RJ 2000 REM ROZKAZY 2 BAJTOWE
UO 2005 IF TN>256 THEN ? :? "Erro
    r--";TN;">256 ($100)":GOTO 230
DC 2010 POKE SA,OP:POKE SA+1,TN:S
    A=SA+2:IF HX=1 THEN 2030
GY 2020 ? OP;" ";TN:GOTO 230
DU 2030 DE=OP:GOSUB 4000:? H$;" "
    ;
VU 2040 DE=TN:GOSUB 4000:? H$:GOT
    O 230
SC 3000 REM ROZKAZY 3 BAJTOWE
EF 3010 POKE SA,OP:POKE SA+1,L:PO
    KE SA+2,H:SA=SA+3:IF HX=1 THEN
     3030
JP 3020 ? OP;" ";L;" ";H:GOTO 230
DV 3030 DE=OP:GOSUB 4000:? H$;" "
    ;
FY 3040 DE=L:GOSUB 4000:? H$;" ";
IT 3050 DE=H:GOSUB 4000:? H$:GOTO
     230
ZV 4000 REM DZIESIETNE NA HEX
UK 4010 H$="":A=INT(DE/256):IF A>
    0 THEN AH=INT(A/16):AL=A-AH*16
    :H$=HE$(AH+1,AH+1):H$(2)=HE$(A
    L+1,AL+1)
LS 4020 A=DE-A*256:AH=INT(A/16):A
    L=A-AH*16:H$(LEN(H$)+1)=HE$(AH
    +1,AH+1):H$(LEN(H$)+1)=HE$(AL+
    1,AL+1):SZ=1:RETURN
YJ 5000 REM HEX NA DZIESIETNE
LU 5010 D=0:0=3:FOR M=1 TO 4:W=AS
    C(H$(M))-48:IF W>9 THEN W=W-7
YQ 5030 D=D*16+W:NEXT M:DE=INT(D)
    :RETURN
CP 6000 REM PRZYJECIE SZESNASTKOW
    EGO KODU OPERACJI I TLUMACZENI
    E GO NA DZIESIETNY
YD 6010 IF R$(1,1)="#" THEN H$="0
    0":H$(3)=R$(2):GOSUB 5000:R$="
    #":R$(2)=STR$(DE):RETURN
FR 6020 LS=LEN(R$):AZ$=R$(1,1):ZA
    $=R$(LS):IF AZ$<>"(" THEN 6050
GZ 6030 IF ZA$="Y" THEN H$="00":H
    $(3)=R$(2,4):GOSUB 5000:R$="("
    :R$(2)=STR$(DE):R$(LEN(R$)+1)=
    ")Y":RETURN
BW 6040 IF ZA$=")" THEN H$="00":H
    $(3)=R$(2,4):GOSUB 5000:R$="("
    :R$(2)=STR$(DE):R$(LEN(R$)+1)=
    "X)":RETURN


UY 6050 IF ZA$="X" OR ZA$="Y" THE
    N 6070
MA 6060 H$="":IF LS<4 THEN H$=ZO$
    (1,4-LS)
QY 6065 H$(LEN(H$)+1)=R$:GOSUB 50
    00:R$=STR$(DE):RETURN
TV 6070 IF LS=5 THEN H$=R$(1,4):G
    OTO 6090
FP 6080 H$="00":H$(3)=R$(1,2)
IY 6090 GOSUB 5000:R$=STR$(DE):R$
    (LEN(R$)+1)=ZA$:RETURN
KD 8000 ? :? "POCZATEK  ";TA;:SZ=
    3:DE=TA:GOSUB 4000:? " ($";H$;
    ")"
LE 8010 ? "KONIEC     ";SA-1;:DE=
    SA-1:SZ=3:GOSUB 4000:? " ($";H
    $;")":END
UJ 9000 ? :FOR I=TA TO SA-1
IK 9010 IF HX=1 THEN DE=I:GOSUB 4
    000:? H$,:DE=PEEK(I):GOSUB 400
    0:? H$
QK 9020 IF HX=0 THEN ? I,PEEK(I)
FG 9030 NEXT I
AT 9040 RETURN
AW 20000 DATA ADC1097,AND1033,ASL
    3002,BCC8144,BCS8176,BEQ8240,B
    IT7036,BMI8048
VX 20010 DATA BNE8208,BPL8016,BRK
    0000,BVC8080,BVS8112,CLC0024,C
    LD0216,CLI0088
FZ 20020 DATA CLV0184,CMP1193,CPX
    4224,CPY4192,DEC2198,DEX0202,D
    EY0136,EOR1065
VI 20030 DATA INC2230,INX0232,INY
    0200,JMP6076,JSR9032,LDA1161,L
    DX5162,LDY5160
TR 20040 DATA LSR3066,NOP0234,ORA
    1001,PHA0072,PHP0008,PLA0104,P
    LP0040,ROL3034
JU 20050 DATA ROR3098,RTI0064,RTS
    0096,SBC1225,SEC0056,SED0248,S
    EI0120,STA1129
WE 20060 DATA STX2134,STY2132,TAX
    0170,TAY0168,TSX0186,TXA0138,T
    XS0154,TYA0152
```



Konwerter języka maszynowego

Podprogram w języku maszynowym, otrzymany w wyniku działania poprzedniego programu i przechowywany w pamięci RAM nie nadaje się do włączenia go do programu w BASIC'u. Należy go przedstawić w postaci zrozumiałej przez BASIC i przechować w pamięci zewnętrznej. Funkcje te realizuje konwerter języka maszynowego, który automatycznie tłumaczy kod maszynowy (przechowywany w pamięci RAM) na pełny podprogram w języku BASIC, który może być łatwo dołączony do naszych programów. Otrzymany podprogram jest kompletny; wymaga tylko spełnienia następujących warunków:

- numery linii programu nie powinny być wyższe niż 31000,
- powinien być on wywołany możliwie jak najwcześniej w programie.

Prezentowany program jest dosyć elastyczny. Można wybrać kilka sposobów przechowywania procedury w języku maszynowym:

- w postaci ciągu (prawdopodobnie najbezpieczniejsza i najbardziej uniwersalna metoda, program będzie automatycznie generował ciąg znaków plus niezbędne instrukcje DIM).
- przechowanie w określonych miejscach pamięci. Adresy mogą być takie same jak dotychczas lub zmienione. Program będzie generował ciąg instrukcji DATA. Dostarczy także krótką procedurę, która pobiera te dane (READ) i wprowadza je (POKE) do pamięci.
- może być używana dowolna kombinacja ciągu i komórek pamięci. Program połączy je w podprogram.

Po zdefiniowaniu tablic i ciągów opisany program zajmuje mniej niż 5KB pamięci.

W celu konwersji języka maszynowego na postać akceptowalną przez BASIC należy wykonać następujące kroki:

1. Zakładamy, że program w języku

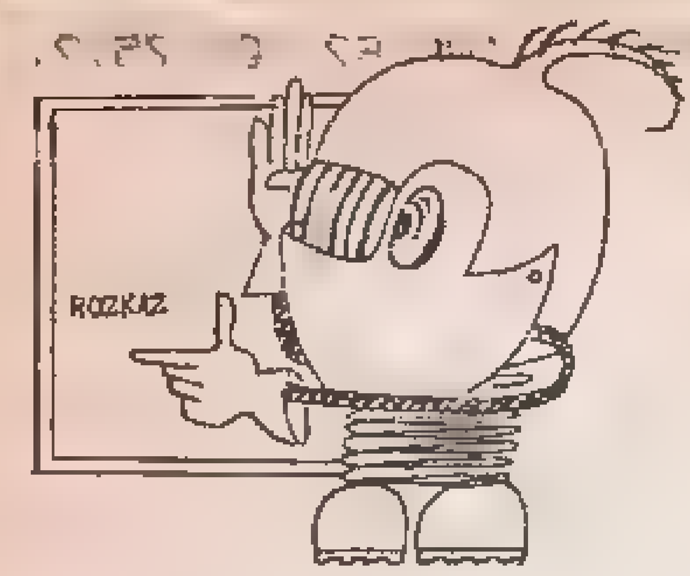
maszynowym umieszczony jest w obszarze pamięci.

2. Wprowadzamy, za pomocą **ENTER**, konwerter języka maszynowego, który wcześniej został przechowany za pomocą **LIST**.
3. Uruchamiamy program. Jesteśmy proszeni o podanie adresów początku i końca procedury maszynowej w RAM. Odpowiedzi należy udzielać w postaci dziesiętnej.
4. Musimy wybrać metodę przechowania procedury. Po wybraniu ciągu należy podać jego nazwę.
5. W wypadku przechowywania w określonym obszarze pamięci musimy podać, czy zmieniamy adresy tego obszaru.
6. Na końcu jesteśmy pytani, czy chcemy dokonać dodatkowych konwersji (na przykład innej procedury). Jeśli tak, to program przejdzie na początek. Jeśli nie, komputer zamyka wszystkie zbiory i kończy działanie.
7. Można teraz wykasować program konwerter (za pomocą NEW), wprowadzić nasz program w BASIC'u, dołączyć procedurę w języku maszynowym za pomocą ENTER „C:" (należy wykasować linię nr 1).
8. Wprowadzamy teraz instrukcję **GOSUB** z podaniem numeru pierwszej linii procedury.

Przechowywanie procedur maszynowych w postaci ciągu wymaga pewnego komentarza. Po pierwsze, aby działały one poprawnie, musimy je pisać w postaci przemieszczalnej. Oznacza to, że nie mogą zawierać żadnych instrukcji **JMP** lub **JSR** do określonych komórek pamięci wewnątrz programu. Ponieważ ciąg może być umieszczony w dowolnym miejscu pamięci, nieprzemieszczalny kod prawie na pewno spowoduje awarię. Po drugie, poważnym problemem jest występowanie znaków o kodach 34 lub 155. Ich obecność powoduje, że edytor ekranowy przedwcześnie obcina nasz ciąg i daje komunikat błędu. Dlatego program wykonuje następujące czynności po napotkaniu jednego z tych znaków:

1) Wstawia znak spacji i oznacza jego pozycję w ciągu.
2) Zapisuje instrukcje podprogramu tak, że wartości te są wstawiane do ciągu bez przejścia przez edytor ekranowy. W tym celu używa on funkcji CHR$.
3) Program może manipulować maksymalnie piętnastoma znakami o kodach 34 i 155. Sprawdza ich ilość i ostrzega, gdy jest zbyt duża.

```
FK 1 REM ****************
WC 2 REM *  KONWERTER    *
PB 3 REM *    JEZYKA     *
VV 4 REM * MASZYNOWEGO   *
FO 5 REM ****************
MX 10 CLR :GRAPHICS 0:POKE 752,1:
   POKE 756,206:? "   KONWERTER 
   JEZYKA MASZYNOWEGO ":GOSUB 600
   :POKE 756,224
YC 20 DIM A$(1),D0$(3),QUOTE(14),
   RETRN(14)
KR 30 D0$="   ":TRAP 580:GOSUB 74
   0:V=0:LNO=32050:LN=31000:F=0
JL 35 OPEN #3,8,0,"C:":? #3;"1 DA
   TA ";:FOR I=0 TO 59:? #3;"0,";
   :NEXT I:? #3;"0";:? #3
VU 40 ? :? :? "WPROWADZ ADRES POC
   ZATKOWY KODU":POKE 752,0:GOSUB
    590:INPUT S:SO=S:SF=S
RE 50 ? "WPROWADZ ADRES KONCOWY K
   ODU":GOSUB 590:INPUT S:FO=S:FF
   =S:GOSUB 640
BK 60 ? "SPOSOB PRZECHOWANIA PROC
   EDURY?"
IZ 65 ? :? "     1.W TYCH SAMYCH AD
   RESACH?":? "     2.WEWNATRZ CIAG
   U?":? "    3.W NOWYCH ADRESACH?
   "
PE 80 GOSUB 590:? :? "PROSZE WPRO
   WADZIC LICZBE I <RETURN>!":INP
   UT N
NG 90 IF (N<>1 AND N<>2 AND N<>3)
    THEN ? CHR$(253);"ZLA ODPOWIE
   DZ! SPROBUJ JESZCZE RAZ!":GOSU
   B 600:GOTO 60
LU 100 IF N=3 THEN ? :? "NOWY ADR
   ES POCZATKOWY PROCEDURY?":GOSU
   B 590:INPUT S:SF=S
HX 110 IF N=3 THEN ? "NOWY ADRES
   KONCOWY PROCEDURY!":GOSUB 590:
   INPUT S:FF=S
SF 120 IF N=3 THEN IF FF-SF<>FO-S
   O THEN ? CHR$(253);" NIEPOPRAW
   NY ADRES KONCOWY! SPROBUJ JESZ
   CZE RAZ!":? :GOTO 110
RT 130 IF N=1 OR N=3 THEN F=1:V=V
   +1:GOTO 180
CK 140 L=FO-SO+1:GOSUB 680:GOSUB
   610
AI 150 ? "CZY CHCESZ OTRZYMAC WYD
   RUK ":? "CIAGU DANYCH W ASCII
   !":GOSUB 590:INPUT A$
EI 160 IF A$="T" THEN N=4:? "NACI
   SNIJ <RETURN> PO WLACZENIU DRU
   KARKI!":GOSUB 590:INPUT A$:OPE
   N #2,8,0,"P:"
OS 170 GOTO 260
WB 180 ? #3;LNO;" DATA ";SF;",";F
   F:LNO=LNO+10
ST 190 ? #3;LNO;" DATA ";
VY 200 FOR I=0 TO 19
OD 210 IF SO+I=FO+1 THEN POP :IF
   I THEN ? #3;",";-1:? #3:LNO=LN
   O+10:GOTO 490
ZP 215 IF SO+I=FO+1 THEN IF I=0 T
   HEN ? #3;-1:? #3:LNO=LNO+10:GO
   TO 490
TL 220 IF I THEN ? #3;",";
OS 230 ? #3;PEEK(SO+I);
CJ 240 NEXT I:? #3:LNO=LNO+10:SO=
   SO+20:GOTO 190
UL 260 IF N=4 THEN ? #2:? #2;"**D
   ANE DLA ";D0$;"**"
HX 270 LS=1:Z=0:W=0:? #3;LN;" DIM
    ";D0$;"(";L;"):";
II 280 IF N=4 THEN FOR I=0 TO L-1
   :? #2;PEEK(SO+I);:IF I<L-1 THE
   N ? #2;",";
CE 290 IF N=4 THEN NEXT I
WG 300 LR=L-80:IF LR<=0 THEN LF=L
   S+L-1
EL 310 IF LR>0 THEN LF=LS+80-1:L=
   LR
YI 320 ? #3;D0$;"(";LS;",";LF;")=
   ";:? #3;CHR$(34);:FOR I=LS TO
   LF
WG 330 IF PEEK(SO+I-1)=34 THEN ?
   #3;" ";:QUOTE(Z)=I:Z=Z+1:GOTO
   360
MP 340 IF PEEK(SO+I-1)=155 THEN ?
    #3;" ";:RETRN(W)=I:W=W+1:GOTO
    360
KL 350 ? #3;CHR$(PEEK(SO+I-1));
YQ 360 NEXT I:IF LR>0 THEN LS=LS+
   80:? #3;CHR$(34):? #3:LN=LN+10
   :? #3;LN;" ";:GOTO 300
BX 370 ? #3;CHR$(34):? #3:LN=LN+1
   0
DL 380 QT=0:RT=0:FOR X=0 TO 14:IF
    QUOTE(X) THEN QT=QT+1
FU 390 IF RETRN(X) THEN RT=RT+1
UK 400 NEXT X:IF QT=0 AND RT=0 TH
   EN 490
RF 410 ? #3;LN;" RESTORE ";LN+20:
   LN=LN+10
BR 420 IF QT THEN ? #3;LN;" FOR X
   =1 TO ";QT;":READ Z:";D0$;"(Z,
   Z)=CHR$(34):NEXT X":LN=LN+10
SJ 430 IF QT THEN ? #3;LN;" DATA
   ";:FOR Y=0 TO QT-1:? #3;QUOTE(
   Y);:IF Y AND Y<QT-1 THEN ? #3;
   ",";
DP 440 IF QT THEN NEXT Y:? #3:LN=
   LN+10
HQ 450 IF RT THEN ? #3;LN;" FOR X
   =1 TO ";RT;":READ Z:";D0$;"(Z,
   Z)=CHR$(155):NEXT X":LN=LN+10
PF 460 IF RT THEN ? #3;LN;" DATA
   ";:FOR Y=0 TO RT-1:? #3;RETRN(
   Y);:IF Y AND Y<QT-1 THEN ? #3;
   ",";
ED 470 IF RT THEN NEXT Y:? #3:LN=
   LN+10
MZ 490 GOSUB 740:? "GOTOWE":GOSUB
    590:INPUT A$
ZI 500 IF A$="N" THEN D0$="   ":C
   LOSE #2:GOTO 40
YE 510 IF F=0 THEN 570
GE 520 ? #3;"32000 W=0:V=";V;":RE
   STORE 32050"
AO 530 ? #3;"32010 READ X,Y:FOR I
   =X TO Y:READ Z:POKE I,Z:NEXT I
   "
ZS 540 ? #3;"32020 READ Z:IF Z<>-
   1 THEN ?";CHR$(34);"BLAD KODU!
    SPRAWDZ INSTRUKCJE DATA!";CHR
   $(34);":END"
CW 550 ? #3;"32030 W=W+1:IF W<V T
   HEN 32010"
FH 560 ? #3;"32040 RETURN"
MQ 570 CLOSE #2:CLOSE #3:END
DL 580 CLOSE #2:CLOSE #3:TRAP 400
   00:? CHR$(253);"BLAD ";PEEK(19
   5);" W LINII ";PEEK(186)+256*P
   EEK(187);"!":END
AD 590 FOR T=10 TO 6 STEP -1:FOR
   S=8 TO 0 STEP -1:SOUND 0,15-S,
   10,T:NEXT S:NEXT T:SOUND 0,0,0
   ,0:RETURN
JS 600 FOR T=1 TO 400:NEXT T:RETU
   RN
RH 610 ? :? "WPROWADZ DWUZNAKOWA
   NAZWE CIAGU":? "PLUS ZNAK $":G
   OSUB 590:INPUT D0$
RF 615 IF LEN(D0$)<3 THEN GOSUB 7
   50:GOTO 610
ZC 620 IF ASC(D0$(1,1))>90 OR ASC
   (D0$(1,1))<65 OR D0$(2,2)="$"
   OR D0$(3,3)<>"$" THEN GOSUB 75
   0:GOTO 610
ZI 630 RETURN
DS 640 IF SO<1792 THEN RETURN
XV 645 IF FO>(256*PEEK(106)-1000)
    THEN ? CHR$(253);"NIE MAM TYL
   E PAMIECI!":POP :GOTO 40
QK 650 BATOP=PEEK(144)+256*PEEK(1
   45):IF BATOP>SO-100 THEN ? CHR
   $(253);"UWAGA! TEN PROGRAM MOZ
   E ZNISZCZYC ":GOTO 670
ZO 660 RETURN
GP 670 ? "TWOJ KOD! SPRAWDZ WYNIK
   I!":GOSUB 600:RETURN
LP 680 QT=0:RT=0:FOR I=0 TO L-1:I
   F PEEK(SO+I)=34 THEN QT=QT+1
KU 690 IF PEEK(SO+I)=155 THEN RT=
   RT+1
NM 700 NEXT I:IF RT<16 AND QT<16
   THEN RETURN
VZ 710 ? CHR$(253);"UWAGA! TWOJ K
   OD ZAWIERA":? "WIECEJ NIZ 15 W
   ARTOSCI ATASCII":? "DLA RETURN
    LUB NAWIASOW"
JN 720 ? "DLATEGO NIE MOGE GO PRZ
   ETWORZYC!":? "PROSZE DOKONAC I
   NNEGO WYBORU":GOSUB 600:GOSUB
    600
GN 730 POP :GOTO 60
VX 740 FOR I=0 TO 14:QUOTE(I)=0:R
   ETRN(I)=0:NEXT I:RETURN
JK 750 ? CHR$(253);"ZLA ODPOWIEDZ
   ! SPROBUJ JESZCZE RAZ!":RETURN
```



Disasembler

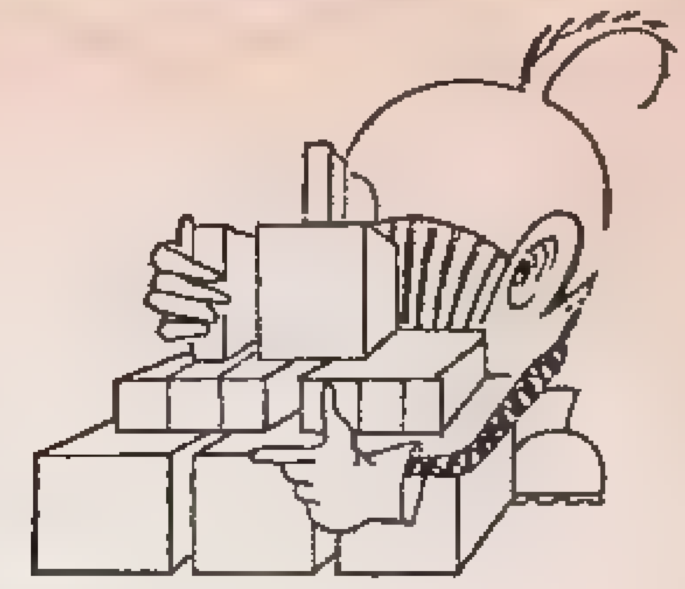
Trzecim programem tej gry jest disasembler realizujący funkcję odwrotną do asemblera: liczby przechowywane w komórkach pamięci przetwarza na postać rozkazową. Prezentowany program umożliwia obejrzenie kodu maszynowego dowolnego programu binarnego, w szczególnym przypadku gry. W aktualnej wersji automatycznie przechodzi do wprowadzenia programu z taśmy magnetycznej. Następnie sam odnajduje adres początku programu i rozpoczyna wyświetlanie rozkazów.

```
VC 98 REM ****************
PS 99 REM *              *
MK 100 REM * DISASSEMBLER *
AB 101 REM *              *
LA 102 REM ****************
WF 105 GRAPHICS 0:POSITION 11,0:?
    "[DISASSEMBLER Z TM]":? :? "P
    rosze czekac....."
QI 110 DIM OPCODE$(256*10),LN(255
    ),NB(255),T$(10),D$(5),B$(28)
ET 120 FOR I=0 TO 255
GJ 125 READ T$,NB
DT 130 LN(I)=LEN(T$)
RA 140 OPCODE$(I*10+1,I*10+LN(I))
    =T$
ZA 150 NB(I)=NB
GD 160 NEXT I
NO 170 GRAPHICS 0:POSITION 11,0:?
    "[DISASSEMBLER Z TM]"
TE 180 ? :?
UQ 185 GOSUB 800
HV [illegible] TRAP 190:? CHR$(156);CHR$(
    28);"Adres poczatkowy (Dziesie
    tnie)",:ADDR=A:? A:TRAP 40000
UN 195 ADRW=PEEK(A+2)+PEEK(A+3)*2
    56
TY 200 IF ADDR<0 OR ADDR>65535 TH
    EN 190
YO [illegible] OP=PEEK(ADDR):NB=NB(OP)
BR 220 T$=OPCODE$(OP*10+1,OP*10+L
    N(OP))
BV 230 ? ADRW;:POKE 85,10:? OP;:P
    OKE 85,15
BM 240 ON NB+2 GOTO 242,244,250,2
    60,270
TI 242 NB=2:T=PEEK(ADDR+1):IF T>1
    28 THEN T=T-256
EP 243 ? T;:POKE 85,25:? T$;"  ";
    ADRW+2+T:GOTO 300
CR 244 ? "[BRAK KODU]":NB=1:GOTO [illegible]

MZ 246 ? T$;" ";ADRW+2+T:GOTO [illegible]
NJ 250 POKE 85,25:? T$:GOTO [illegible]
ZZ 260 ? PEEK(ADDR+1);:POKE 85,25
    :D$=STR$(PEEK(ADDR+1)):GOSUB [illegible]
    [illegible] GOTO [illegible]
TO 270 ? PEEK(ADDR+1);:POKE 85,20
    :? PEEK(ADDR+2);:POKE 85,25
AU 280 D$=STR$((PEEK(ADDR+1)+256*
    PEEK(ADDR+2))):GOSUB [illegible]
XF [illegible] ADDR=ADDR+NB:IF ADDR<0 THE
    N ADDR=65536-T
BB 310 IF ADDR>65535 THEN ADDR=T
UM 312 ADRW=ADRW+NB
DB 320 IF PEEK(53279)=7 THEN [illegible]
PJ 330 GOTO [illegible]
QK [illegible] ? T$(1,4+(LN(OP)>4));D$;T$
    (4+2*(LN(OP)>5)):RETURN
BW 500 DATA BRK,1,ORA (X),2,?,0,?
    ,0,?,0,ORA  ,2,ASL  ,2,?,0,PHP
    ,1,ORA # ,2

CX 510 DATA ASL A,1,?,0,?,0,ORA
    ,3,ASL  ,3,?,0,BPL,-1,ORA ()Y,
    2,?,0,?,0
FH 520 DATA ?,0,ORA  X,2,ASL  X,2
    ,?,0,CLC,1,ORA  Y,3,?,0,?,0,?,
    0,ORA  X,3
WE 530 DATA ASL ,2,?,0,JSR  ,3,AN
    D (X),2,?,0,?,0,BIT ,2,AND ,2,
    ROL ,2,?,0
WH 540 DATA PLP,1,AND # ,2,ROL A,
    1,?,0,BIT ,3,AND ,3;ROL ,3,?,0
    ,BMI,-1,AND ()Y,2
UE 550 DATA ?,0,?,0,?,0,AND  X,2,
    ROL  X,2,?,0,SEC,1,AND  Y,3,CL
    I,1,?,0
FG 560 DATA ?,0,AND  X,3,ROL  X,3
    ,?,0,RTI,1,EOR (X),2,?,0,?,0,?
    ,0,EOR  ,2
OW 570 DATA LSR  ,2,?,0,PHA,1,EOR
     # ,2,LSR  ,3,?,0,JMP  ,3,EOR
     ,3,LSR  ,3,?,0
PU 580 DATA BVC,-1,EOR ()Y,2,?,0,
    ?,0,?,0,EOR  X,2,LSR  X,2,?,0,
    CLI,1,EOR  Y,2
XP 590 DATA ?,0,?,0,?,0,EOR  X,3,
    LSR  X,3,?,0,RTS,1,ADC (X),2,?
    ,0,?,0
KQ 600 DATA ?,0,ADC  ,2,ROR  ,2,?
    ,0,PLA,1,ADC # ,2,ROR A,1,?,0,
    JMP (),108,ADC  ,3
RY 610 DATA ROR  ,3,?,0,BVS,-1,AD
    C ()Y,2,?,0,?,0,?,0,ADC  X,2,R
    OR  X,2,?,0
KJ 620 DATA SEI,1,ADC  Y,3,?,0,?,
    0,?,0,ADC  X,3,ROR  X,3,?,0,?,
    0,STA (X),2
EL 630 DATA ?,0,?,0,STY  ,2,STA
    ,2,STX  ,2,?,0,DEY,1,?,0,TXA,1
    ,?,0
FZ 640 DATA STY  ,3,STA  ,3,STX
    ,3,?,0,BCC,-1,STA ()Y,2,?,0,?,
    0,STY  X,2,STA  X,2
IN 650 DATA STX  Y,2,?,0,TYA,1,ST
    A  Y,3,TXS,1,?,0,?,0,STA  X,3,
    ?,0,?,0
CI 660 DATA LDY # ,2,LDA (X),2,LD
    X # ,2,?,0,LDY  ,2,LDA  ,2,LDX
     ,2,?,0,TAY,1,LDA # ,2
GN 670 DATA TAX,1,?,0,LDY  ,3,LDA
     ,3,LDX  ,3,?,0,BCS,-1,LDA ()
    Y,2,?,0,?,0
TP 680 DATA LDY  X,2,LDA  X,2,LDX
     Y,2,?,0,CLV,1,LDA  Y,3,TSX,1
    ,?,0,LDY  X,3,LDA  X,3
AP 690 DATA LDX  Y,3,?,0,CPY # ,2
    ,CMP (X),2,?,0,?,0,CPY  ,2,CMP
     ,2,DEC  ,2,?,0
BO 700 DATA INY,1,CMP # ,2,DEX,1,
    ?,0,CPY  ,3,CMP  ,3,DEC  ,3,?,
    0,BNE,-1,CMP ()Y,2

NO 710 DATA ?,0;?,0,?,0,CMP  X,2,
    DEC  X,2,?,0,CLD,1,CMP  Y,3,?,
    0,?,0
LQ 720 DATA ?,0,CMP  X,3,DEC  X,3
    ,?,0,CPX # ,2,SBC (X),2,?,0,?,
    0,CPX  ,2,SBC  ,2
SI 730 DATA INC  ,2,?,0,INX,1,SBC
     # ,2,NOP,1,?,0,CPX  ,3,SBC  ,
    3,INC  ,3,?,0
NO 740 DATA BEQ,-1,SBC (Y),2,?,0,
    ?,0,?,0,SBC  X,2,INC  X,2,?,0,
    SED,1,SBC  Y,3
AU 750 DATA ?,0,?,0,?,0,SBC  X,3,
    INC  X,3,?,0
VW 800 FOR II=1 TO 28:READ B:B$(I
    I,II)=CHR$(B):NEXT II
TJ 810 ? "ILE BAJTOW WCZYTAC":INP
    UT ILOSC
SN 820 OPEN #2,4,128,"C:"
IV 830 A=PEEK(144)+PEEK(145)*256+
    256
OZ 840 U=USR(ADR(B$),32,7,A,ILOSC
    )
CW 845 DATA 104,104,104,170,104,1
    04,157,66,3,104,157,69,3,104,1
    57,68,3,104,157,73,3,104,157,7
    2,3,76,86,228
ED 850 POKE 54018,60:ADRW=PEEK(A+
    2)+PEEK(A+3)*256:RETURN
```



Budowa zamku

Ostatnim z prezentowanych programów jest przykład prostej gry edukacyjnej umożliwiającej poznanie kształtów figur geometrycznych lub wzorów na obliczanie pola powierzchni tych figur. Wariant gry może być wybrany bezpośrednio po jej uruchomieniu.

```
FK 1 REM ****************
FJ 2 REM *              *
WZ 3 REM * Budowa zamku *
FO 5 REM ****************
OL 50 BRK=1:IF PEEK(53279)=5 THEN
     BRK=0
LN 60 GOSUB 3000:GOTO 90
SG 70 POKE 77,0:IF BRK THEN POKE
    16,64:POKE 53774,64
ZY 80 RETURN
TV 90 DIM TP(8),COL(8),ANS$(114),
    INV$(114),FRM$(114),INF$(114)
OS 100 ANS$(1,50)="  PROSTOKAT
    KWADRAT     ROWNOLEGLOBOK    K
    OLO    "
KA 110 ANS$(51,105)="    TRAPEZ
      POLKOLE       TROJKAT
     [ HELP ]   "
CQ 120 FRM$(1,50)="     W * L
      S^2          B * H       PI
    * R^2 "
VI 130 FRM$(51,105)="  1/2 *H*(B+
    T)  1/2 *PI*R^2 1/2 * B * H
     [ HELP ]     "
DV 140 INV$(1,50)="[ PROSTOKAT
    [ KWADRAT   ROWNOLEGLOBOK   K]
    [OLO]"
SR 150 INV$(51,105)="[    TRAPEZ
    [     POLKOLE       TROJKAT  ]
    [ [ HELP ] ]"
VS 160 INF$(1,50)="[    W * L
    [     S^2          B * H     PI]
    [* R^2 ]"
UU 170 INF$(51,105)="[ 1/2 *H*(B+
    [T)  1/2 *PI*R^2 1/2 * B * H  ]
    [ [ HELP ]    ]"
RT 175 ANS$(10,10)=CHR$(1):ANS$(2
    8,28)=CHR$(15):ANS$(35,35)=CHR
    $(12):ANS$(46,46)=CHR$(12):ANS
    $(68,68)=CHR$(15)
TM 176 ANS$(82,82)=CHR$(15):ANS$(
    69,69)=CHR$(12):ANS$(85,85)=CH
    R$(1)
EN 177 INV$(10,10)=CHR$(129):INV$
    (28,28)=CHR$(143):INV$(35,35)=
    CHR$(140):INV$(46,46)=CHR$(140
    ):INV$(68,68)=CHR$(143)
ZV 178 INV$(69,69)=CHR$(140):INV$
    (82,82)=CHR$(143):INV$(85,85)=
    CHR$(129)
HP 180 COL(1)=0:COL(2)=13:COL(3)=
    26:COL(4)=0
EX 190 COL(5)=13:COL(6)=26:COL(7)
    =0:COL(8)=13
ZR 200 TRUE=1:FALSE=0:NONE=FALSE:
    GRAPHICS 0:POKE 756,CH/256
BZ 210 GOSUB 720:REM KRESLENIE
YM 220 REM KRESLENIE ZAMKU
SU 230 GRAPHICS 7:POKE 756,CH/256
    :POKE 708,56:POKE 709,186:POKE
     710,130:POKE 712,150:GOSUB 70
    :COLOR 2:RESTORE 970
```

```
CQ 240 FOR I=1 TO 12:READ X1,Y1,X
   2,Y2:GOSUB 530:NEXT I:REM KRES
   LENIE PROSTOKATOW
DH 250 FOR I=1 TO 4:READ X1,Y1,X2
   ,Y2,X3,X4:GOSUB 600:NEXT I:REM
   TRAPEZY
RH 260 FOR I=1 TO 2:READ X1,Y1,X2
   ,Y2,Y3,Y4:GOSUB 550:NEXT I:REM
   ROWNOLEGLOBOK
GK 270 FOR I=1 TO 4:READ X1,Y1,X2
   ,Y2,X3,X4
MW 280 PLOT X1,Y2:DRAWTO X2,Y1:DR
   AWTO X4,Y1
ZD 290 DRAWTO X3,Y2:DRAWTO X1,Y2:
   NEXT I
OY 300 PLOT 41,10:DRAWTO 41,0:DRA
   WTO 49,3:DRAWTO 41,6
OC 310 FOR I=1 TO 2:READ X1,Y1,X2
   ,Y2:GOSUB 620:NEXT I:REM TROJK
   ATY
BG 320 FOR P=1 TO 7:READ X1,Y1,X2
   ,Y2,E,J:GOSUB 570:NEXT P:REM O
   KREGI
WS 330 ON WYBOR GOSUB 640,680
SD 340 LET TEXT=PEEK(660)+256*PEE
   K(661)
JM 350 TP(1)=TEXT:TP(2)=TEXT+13:T
   P(3)=TEXT+26:TP(4)=TEXT+40
JJ 360 TP(5)=TEXT+53:TP(6)=TEXT+6
   6:TP(7)=TEXT+80:TP(8)=TEXT+93
ZD 370 SP=5:OLDSP=SP:GOSUB 1460:R
   EM GLOWNY PROGRAM
UN 380 FOR DR=1 TO NUMSHAPES-1:CO
   RRECT=FALSE
KI 390 READ SHAPE:GOSUB 1700:COLO
   R 2
WN 400 ON SHAPE GOSUB 1750,1820,1
   840,1980,2080,2190,2210,2280
MQ 410 NEXT DR
BX 420 GOSUB 1700:REM KASOWANIE R
   YSUNKU
NF 430 READ SHAPE,COLR,X1,Y1,X2,Y
   2:CORRECT=FALSE:COLOR 2
PJ 440 GOSUB 620:GOSUB 1600:READ
   X1,Y1,X2,Y2
UC 450 COLOR COLR
QT 460 PLOT X2,Y2:DRAWTO X1,Y1
DW 470 Y1=Y1+1:IF Y1<>(Y2+4) THEN
   460
GI 480 ? CHR$(125):IF NUMWRONG=0
   THEN ? " KA";CHR$(26);"DA TWO
   JA ODPOWIED";CHR$(6);" BY";CHR
   $(12);"A POPRAWNA!!!"
UO 490 IF NUMWRONG>0 THEN ? "OPU"
   ;CHR$(19);CHR$(3);"I";CHR$(12)
   ;"E";CHR$(19);" ";NUMWRONG:?
LD 500 ? "W celu dalszej gry naci
   ";CHR$(23);"nij PRZYCISK...";
RQ 510 IF STRIG(0) THEN 510
NO 520 GRAPHICS 0:POKE 756,CH/256
   :GOSUB 70:GOTO 210
FZ 530 PLOT X1,Y1:DRAWTO X2,Y1:DR
   AWTO X2,Y2:REM KRESLENIE PROST
   OKATA
TZ 540 DRAWTO X1,Y2:DRAWTO X1,Y1:
   RETURN
WG 550 PLOT X1,Y2:DRAWTO X2,Y1:DR
   AWTO X2,Y3:REM KRESLENIE ROWNO
   LEGLOBOKU
WM 560 DRAWTO X1,Y4:DRAWTO X1,Y2:
   RETURN
JB 570 PLOT X1,Y1:FOR I=0 TO E ST
   EP J:REM OKREGI & POLOKREGI
AZ 580 X=6*COS(I)+X2:Y=(-6)*SIN(I
   )+Y2
LK 590 DRAWTO X,Y:NEXT I:RETURN
LB 600 PLOT X1,Y2:DRAWTO X2,Y1:DR
   AWTO X3,Y1:REM KRESLENIE TRAPE
   ZU
WK 610 DRAWTO X4,Y2:DRAWTO X1,Y2:
   RETURN
VY 620 PLOT X1,Y1:DRAWTO X2,Y2:DR
   AWTO X1,Y2:REM TROJKAT
YV 630 DRAWTO X1,Y1:RETURN
OX 640 POKE 82,0:? "":POKE 752,1:
   REM KSZTALTY
PY 650 ? ANS$(1,39):? ANS$(40,78)
   :? ANS$(79,104)
WL 660 IF  NOT HELP THEN ? "U";CH
   R$(24);"yj manipulatora do wyb
   rania KSZTA";CHR$(12);"TU.";
ZQ 670 RETURN
UQ 680 POKE 82,0:? "":POKE 752,1:
   REM WYKONAJ WZORY
WJ 690 ? FRM$(1,39):? FRM$(40,78)
   :? FRM$(79,104)
GK 700 IF  NOT HELP THEN ? " U";C
   HR$(24);"yj manipulatora do wy
   brania WZORU";:POKE 752,2
ZF 710 RETURN
RL 720 NUMSHAPES=32:? CHR$(125):P
   OKE 710,146:POKE 752,1:REM INT
   RO EKRAN
ZY 730 POSITION 2,0:? "
      BUDOWA ZAMKU          ",
AS 740 POSITION 2,3:? " Wzory
   Kszta";CHR$(251);"ty"
SW 750 POSITION 2,5:? "W * L
        = PROSTOK";CHR$(1);"T"
KP 760 POSITION 2,6:? "S^2
        = KWADRAT"
HG 770 POSITION 2,7:? "B * H
        = R";CHR$(15);"WNOLEG";C
   HR$(12);"OBOK"
RG 780 POSITION 2,8:? "PI * R^2
        = KO";CHR$(12);"O "
AN 790 POSITION 2,9:? "1/2 * H *
   (B+T) = TRAPEZ"
PK 800 POSITION 2,10:? "1/2 * PI
   * R^2  = P";CHR$(15);CHR$(12);
   "KOLE "
DS 810 POSITION 2,11:? "1/2 * H *
   B      = TR";CHR$(15);"JK";CHR
   $(1);"T"
ID 820 POSITION 0,13:? "Aby gra";
   CHR$(22);" naci";CHR$(23);"nij
   przycisk manipulatora.":GOSUB
   70:POKE 764,255
WF 830 IF STRIG(0) THEN 830
ZZ 840 S=55:GOSUB 1570:REM BEEP
EV 850 IF  NOT STRIG(0) THEN 850
EF 860 GRAPHICS 2+16:POKE 756,CH/
   256:WYBOR=1:REM START
NS 870 POSITION 0,4:PRINT #6;"CO
   CHCESZ SPR";CHR$(92);"BOWA";CH
   R$(37);":":GOSUB 70
RJ 890 IF  NOT STRIG(0) THEN S=25
   :GOSUB 1570:RETURN
VC 900 IF WYBOR=2 THEN 930
NE 910 POSITION 6,7:PRINT #6;"RYS
   UNKI";
IF 920 POSITION 6,9:PRINT #6;"WZO
   RY";
AR 925 PA=0
NO 930 IF STICK(0)=14 THEN WYBOR=
   1:S=75:GOSUB 1570:GOTO 890
XL 940 IF STICK(0)=13 THEN WYBOR=
   2:GOSUB 1540:S=55:GOSUB 1570
SL 950 GOTO 890
HS 960 REM DANE ZAMKU
QQ 970 DATA 5,37,77,76,39,60,46,7
   6,7,15,17,32,65,15,75,32
PB 980 DATA 10,6,14,10,70,10,80,1
   5,60,10,70,15,68,6,72,10
SM 990 DATA 2,10,12,15,12,10,22,1
   5,41,22,57,37,25,22,41,37
HM 1000 DATA 5,32,7,37,17,20,62,3
   2,65,37,75,77
JN 1010 DATA 46,70,52,76,71,77,5,
   70,11,76,33,39
OC 1020 DATA 10,20,15,17,30,27,67
   ,17,72,20,27,30
CN 1030 DATA 18,6,20,10,22,24,2,6
   ,0,10,7,4
YL 1040 DATA 60,6,57,10,64,62,75,
   6,78,10,80,82
VZ 1050 DATA 41,10,57,22,41,10,25
   ,22
EI 1060 DATA 28,47,22,47,6.6,0.1
VR 1070 DATA 39,30,33,30,6.6,0.1
ZP 1080 DATA 55,30,49,30,6.6,0.1
IZ 1090 DATA 68,47,62,47,6.6,0.1
CY 1100 DATA 16,65,22,65,3.15,0.1
HJ 1110 DATA 56,65,62,65,3.15,0.1
HI 1120 DATA 37,48,42,48,3.15,0.1
IH 1130 REM
EB 1140 DATA 1,3,100,67,120,75,5,
   38,77,75
IG 1150 DATA 4,2,112,65,106,65,6.
   6,0.1,44,96,6.6,0.1
UG 1160 DATA 5,2,100,65,105,75,12
   0,125,48,70,53,75,72,78
TC 1170 DATA 1,1,100,65,120,75,39
   ,60,46,75
WR 1180 DATA 6,1,100,65,106,65,3.
   15,0.1,124,130,3.15,0.1
BQ 1190 DATA 5,2,100,60,110,75,12
   0,130,5,70,10,75,32,38
VI 1200 DATA 4,2,112,65,106,65,6.
   6,0.1,124,96,6.6,0.1
CY 1210 DATA 6,1,105,65,111,65,3.
   15,0.1,44,130,3.15,0.1
CK 1220 DATA 1,1,100,62,110,75,7,
   15,17,32
NH 1230 DATA 3,3,100,65,105,75,55
   ,65,10,20,15,17,30,27
HW 1240 DATA 2,1,100,65,105,70,10
   ,6,15,9
NH 1250 DATA 1,2,100,62,102,75,2,
   10,12,15
TC 1260 DATA 3,1,100,65,105,75,55
   ,65,1,6,0,9,6,7
EP 1270 DATA 5,2,100,65,110,75,12
   0,130,4,32,7,37,18,20
DE 1280 DATA 1,2,100,65,122,75,12
   ,10,22,15
HU 1290 DATA 3,1,100,65,110,75,60
   ,70,18,6,20,9,26,23
IM 1300 DATA 2,2,105,65,115,75,25
   ,22,41,37
PU 1310 DATA 7,1,100,65,120,75,40
   ,10,25,22
IC 1320 DATA 2,2,100,65,110,75,41
   ,22,57,37
AJ 1330 DATA 4,3,112,65,106,65,6.
   6,0.1,66,60,6.6,0.1
KK 1340 DATA 1,1,100,60,105,75,65
   ,16,75,32
NY 1350 DATA 1,2,100,62,102,72,60
   ,10,70,15
UU 1360 DATA 3,1,100,65,110,75,60
   ,70,59,6,56,9,63,68
JL 1370 DATA 5,2,110,75,100,65,13
   0,120,62,32,64,37,75,79
WJ 1380 DATA 1,2,100,67,122,72,70
   ,10,80,15
LV 1390 DATA 7,1,100,62,110,75,41
   ,10,41,22
BI 1400 DATA 3,3,100,65,110,75,60
   ,70,68,17,73,20,28,32
CJ 1410 DATA 2,1,100,70,105,75,68
   ,6,72,9
GN 1420 DATA 3,1,100,65,105,75,55
   ,65,76,6,78,9,84,80
YZ 1430 DATA 4,3,112,65,106,65,6.
   6,0.1,100,60,6.6,0.1
JH 1440 DATA 6,1,100,65,106,65,3.
   15,0.1,84,96,3.15,0.1

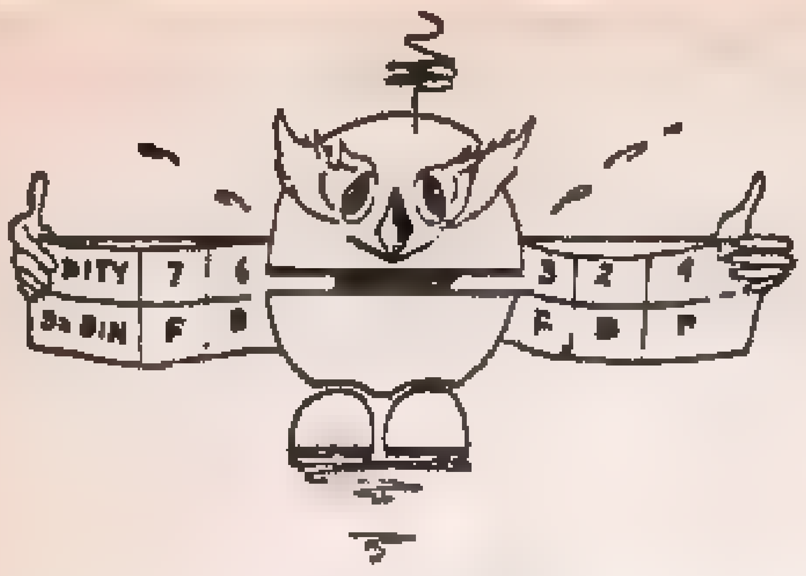
FW 1450 DATA 7,1,102,65,115,75,42
   ,0,50,3
RO 1460 ROW=INT((SP-1)/3):OLDROW=
   INT((OLDSP-1)/3):REM NEGATYW T
   EKSTU
SZ 1470 POKE 656,OLDROW:POKE 657,
   COL(OLDSP):POKE 658,0
ZS 1480 IF WYBOR=1 THEN ? ANS$((O
   LDSP-1)*13+1,(OLDSP-1)*13+13);
CW 1490 IF WYBOR=2 THEN ? FRM$((O
   LDSP-1)*13+1,(OLDSP-1)*13+13);
CN 1500 POKE 656,ROW:POKE 657,COL
   (SP):POKE 658,0
FC 1510 IF WYBOR=1 THEN ? INV$((S
   P-1)*13+1,(SP-1)*13+13);
PU 1520 IF WYBOR=2 THEN ? INF$((S
   P-1)*13+1,(SP-1)*13+13);
AS 1530 RETURN
EB 1540 POSITION 6,7:PRINT #6;"RY
   SUNKI":REM NEGATYW WYBOROW
YC 1550 POSITION 6,9:PRINT #6;"WZ
   ORY"
BB 1560 RETURN
OG 1570 SOUND 0,5,10,10:REM BEEP
ON 1580 FOR D=1 TO 20:NEXT D
YL 1590 SOUND 0,0,0,0:RETURN
JN 1600 GOSUB 70:TR=STRIG(0):ST=S
   TICK(0):OLDSP=SP:IF ST=15 AND
   TR=1 THEN 1600:REM SPRAWDZENIE
    PRZYCISKU
UQ 1610 POKE 53279,7:IF TR=0 THEN
    S=35:GOSUB 1570:ANS=SP:GOSUB
   2320
EM 1620 IF ANS=8 THEN GOSUB 2280
PP 1630 IF CORRECT THEN RETURN
JS 1640 IF ST=7 THEN SP=SP+1:IF (
   SP-1)/3=INT((SP-1)/3) THEN SP=
   SP-3
VG 1650 IF SP=9 THEN SP=7
KZ 1660 IF ST=11 THEN SP=SP-1:IF
   SP/3=INT(SP/3) THEN SP=SP+3:IF
    SP=9 THEN SP=8
DL 1670 IF ST=13 THEN SP=SP+3:IF
   SP>8 THEN SP=SP-9:IF SP=0 THEN
    SP=3
VP 1680 IF ST=14 THEN SP=SP-3:IF
   SP<1 THEN SP=9+SP:IF SP=9 THEN
    SP=6
ZP 1690 GOSUB 1460:GOTO 1600
RI 1700 COLOR 0:X1=85:X2=150:REM
   KASOWANIE RYSUNKU
SU 1710 FOR Y=55 TO 80:SOUND 0,25
   5-(2*Y),10,10
YD 1720 PLOT X1,Y:DRAWTO X2,Y
OW 1730 NEXT Y:SOUND 0,0,0,0
AZ 1740 RETURN
SL 1750 READ COLR,X1,Y1,X2,Y2:REM
    PROSTOKAT
```

```
JY 1760 GOSUB 530:GOSUB 1600
WA 1770 READ X1,Y1,X2,Y2:COLOR CO
   LR
QW 1780 FOR Y=Y1 TO Y2
YY 1790 PLOT X1,Y:DRAWTO X2,Y
LZ 1800 NEXT Y
AS 1810 RETURN
EX 1820 GOSUB 1750:REM KWADRAT
AY 1830 RETURN
MV 1840 READ COLR,X1,Y1,X2,Y2,Y3,
   Y4:REM ROWNOLEGLOBOK
SC 1850 PA=PA+1
LA 1860 GOSUB 550:GOSUB 1600
WU 1870 IF PA=1 OR PA=5 THEN 1930
BZ 1880 READ X1,Y1,X2,Y2,X3,X4:CO
   LOR COLR
IK 1890 PLOT X2,Y1:DRAWTO X1,Y2
DU 1900 X1=X1+1:X2=X2+1
YM 1910 IF X2<>X3 THEN 1890
AX 1920 RETURN
DJ 1930 READ X1,Y1,X2,Y2,Y3,Y4:CO
   LOR COLR
HX 1940 PLOT X1,Y2:DRAWTO X2,Y1
FN 1950 Y2=Y2+1:Y1=Y1+1
WW 1960 IF Y2<>Y4 THEN 1940
BM 1970 RETURN
TN 1980 READ COLR,X1,Y1,X2,Y2,E,J
   :REM KOLO
ML 1990 GOSUB 570:GOSUB 1600
GD 2000 READ XC,YC,E,J:COLOR COLR
RQ 2010 XC=XC/2:YC=YC/2
PG 2020 FOR R=0 TO E STEP J
SQ 2030 X=6*COS(R)+XC
NY 2040 Y=(-6)*SIN(R)+YC
XN 2050 PLOT XC,YC:DRAWTO X,Y
JD 2060 NEXT R
AV 2070 RETURN
YH 2080 READ COLR,X1,Y1,X2,Y2,X3,
   X4:REM TRAPEZ
YB 2090 GOSUB 600
ZM 2100 GOSUB 1600
YZ 2110 READ X1,Y1,X2,Y2,X3,X4
EF 2120 COLOR COLR
HF 2130 PLOT X1,Y2:DRAWTO X2,Y1
DR 2140 X1=X1+1:X2=X2+1
NY 2150 IF X2<>X3 THEN 2130
IA 2160 PLOT X3,Y1:DRAWTO X1,Y2
TT 2170 X1=X1+1:IF X1<>X4 THEN 21
   60
BA 2180 RETURN
NV 2190 GOSUB 1980:REM POLKOLE
AE 2200 RETURN
YM 2210 READ COLR,X1,Y1,X2,Y2:REM
   TROJKAT
YK 2220 GOSUB 620
ZX 2230 GOSUB 1600
BF 2240 READ X1,Y1,X2,Y2:X3=X2:CO
   LOR COLR
NT 2250 PLOT X1,Y1:DRAWTO X2,Y2:X
   2=X2+1
RF 2260 IF X2<>X3+17 THEN 2250
AZ 2270 RETURN
MP 2280 HELP=TRUE:? CHR$(125):ON
   WYBOR GOSUB 690,650
FB 2290 FOR D=1 TO 500:NEXT D:HEL
   P=FALSE:ANS=0
XD 2300 ? CHR$(125):ON WYBOR GOSU
   B 650,690
CU 2310 ANS=0:RETURN
GS 2320 IF ANS=SHAPE THEN CORRECT
   =TRUE
EW 2330 IF ANS=8 THEN RETURN
DX 2340 IF ANS<>SHAPE THEN NUMWRO
   NG=NUMWRONG+1:GOSUB 2360
AV 2350 RETURN
YO 2360 S=150:FOR BUZZ=1 TO 3:GOS
   UB 1570
OU 2370 NEXT BUZZ:RETURN
JC 3000 DIM PGM$(32):CH=(PEEK(106
   )-8)*256:POKE 106,PEEK(106)-16
ZV 3010 RESTORE 3035:FOR I=1 TO 3
   2:READ X:PGM$(I,I)=CHR$(X):NEX
   T I:X=USR(ADR(PGM$),224*256,CH
   )
QO 3020 READ X:IF X=-1 THEN POKE
   756,CH/256:RETURN
ED 3030 FOR I=0 TO 7:READ Y:POKE
   CH+X*8+I,Y:NEXT I:GOTO 3020
AY 3035 DATA 104,104,133,213,104,
   133,212,104,133,215,104,133,21
   4,162,4,160,0,177,212,145,214,
   200,208,249,230
GF 3036 DATA 213,230,215,202,208,
   240,96
KN 3040 DATA 65,0,24,60,102,102,1
   26,102,12
OR 3050 DATA 67,12,60,102,96,96,1
   02,60,0
LN 3080 DATA 70,12,24,126,12,24,4
   8,126,0
LP 3100 DATA 76,0,96,120,112,224,
   96,126,0
EB 3130 DATA 79,12,60,102,102,102
   ,102,60,0
GF 3160 DATA 83,12,60,96,60,6,6,6
   0,0
ZL 3170 DATA 86,12,24,0,60,96,96,
   60,0
JX 3180 DATA 87,12,24,62,96,60,6,
   124,0
DP 3190 DATA 88,0,24,0,126,12,48,
   126,0
SI 3200 DATA 90,24,0,126,12,24,48
   ,126,0
EZ 3210 DATA 123,0,56,24,28,56,24
   ,60,0
ZL 3220 DATA 60,12,60,102,102,102
   ,102,60,0
TX 3230 DATA 5,12,60,102,96,96,10
   2,60,0
EW 3240 DATA -1
```

ATARI

```
10 REM *** SPEKTROGRAF MASOWY ***
20 OPEN #1,4,0,"K:":GRAPHICS 2:POKE 710,0:POKE 752,1:POKE 764,255
30 POSITION 0,5:? #6;" SPEKTROGRAF MASOWY":SETCOLOR 0,1,6
35 ? CHR$(127);"PROSZE CHWILE POCZEKAC":GOSUB 460
40 ? CHR$(28);CHR$(127);CHR$(253);"NACISNIJ DOWOLNY KLAWISZ":GET #1,S
50 REM ________________
60 REM SCHEMAT SPEKTROGRAFU
70 REM ________________
80 GRAPHICS 7:COLOR 2:DL=PEEK(560)+256*PEEK(561)
90 POKE DL+85,13:POKE 660,54:POKE 661,157:POKE 710,0:GOSUB 330
100 PLOT 110,30:DRAWTO 130,30:DRAWTO 130,1:DRAWTO 110,1
110 PLOT 100,1:DRAWTO 80,1:DRAWTO 80,30:DRAWTO 100,30:DRAWTO 30,30
120 FOR T=-1.55 TO 1.6 STEP 0.05:DRAWTO SIN(T)*50+80,COS(T)*50+30:NEXT T
130 PLOT 140,15:DRAWTO 120,15:DRAWTO 120,3:DRAWTO 120,27
140 PLOT 70,15:DRAWTO 90,15:DRAWTO 90,3:DRAWTO 90,27
150 PLOT 135,13:DRAWTO 140,13
160 PLOT 70,12:DRAWTO 76,12:PLOT 73,10:DRAWTO 73,14
170 FOR T=30 TO 80:PLOT T,20:DRAWTO T,30:NEXT T:COLOR 0:POKE 752,1
180 REM ____________
190 REM RUCH CZASTEK
200 REM ____________
210 COLOR 0:PLOT 34,20:DRAWTO 34,30:PLOT 45,20:DRAWTO 45,30
220 PLOT 73,20:DRAWTO 73,30:PLOT 57,20:DRAWTO 57,30:A=1
230 ? CHR$(127);"  W-WOLNO S-SZYBKO R-RESTART":?
240 U=USR(15000):FOR T=1 TO A:NEXT T:B=PEEK(764)
250 IF B=46 THEN A=A+20:POKE 764,255
260 IF B=62 THEN A=A-20:POKE 764,255
270 IF B=40 THEN GOTO 290
280 GOTO 240
290 U=USR(15256):RUN
300 REM ________________
310 REM GRAFIKA PLAJEROW
320 REM ________________
330 POKE 623,8:POKE 559,62:POKE 54279,130:POKE 53277,3
340 POKE 704,8:POKE 705,10:POKE 706,12:POKE 707,15:U=USR(1545)
350 RESTORE 390:FOR T=33794 TO 33806:READ A:POKE T,A:NEXT T
360 RESTORE 400:FOR T=34050 TO 34062:READ A:POKE T,A:NEXT T
370 RESTORE 410:FOR T=34306 TO 34318:READ A:POKE T,A:NEXT T
380 RESTORE 420:FOR T=34562 TO 34574:READ A:POKE T,A:NEXT T:RETURN
390 DATA 0,0,0,0,24,60,60,24,0,0,0,0
400 DATA 0,0,0,8,28,28,8,0,0,0,0,0
410 DATA 0,0,0,0,8,28,8,0,0,0,0,0
420 DATA 0,0,0,0,0,1,1,0,0,0,0,0
430 REM ________________
440 REM PROGRAM MASZYNOWY
450 REM ________________
460 RESTORE 490:FOR T=1545 TO 1639:READ A:POKE T,A:NEXT T
470 RESTORE 580:FOR T=15000 TO 15508:READ A:POKE T,A:NEXT T
480 RESTORE 860:FOR T=1528 TO 1539:READ A:POKE T,A:NEXT T:RETURN
490 DATA 104,160,0,169,0,133,26,169,127,133,27,169,0,145,26,165,26
500 DATA 105,1,133,26,144,244,24,230,27,165,27,201,143,208,235,96,0
520 DATA 170,160,1,177,26,136,145,26,200,200,192,14,208,245
530 DATA 198,26,202,224,0,208,236,234,234,234,234,234,234,24,96,0,0
550 DATA 170,160,12,177,26,200,145,26,136,136,192,0,208,245
560 DATA 230,26,202,224,0,208,236,234,234,234,234,234,234,24,96,0
580 DATA 104,172,250,5,140,0,208,192,76,208,5,160,151,140,250,5
590 DATA 206,250,5,173,248,5,133,26,173,249,5,133,27,185,96,59
600 DATA 201,0,240,26,74,176,5,32,74,6,144,13,24,201,80,208,5
610 DATA 160,0,140,0,208,32,43,6,165,26,141,248,5,172,253,5
620 DATA 140,1,208,192,86,208,5,160,151,140,253,5,206,253,5,173,251,5
630 DATA 133,26,173,252,5,133,27,185,160,59,201,0,240,26,74,176,5
640 DATA 32,74,6,144,13,24,201,80,208,5,160,0,140,1,208,32,43,6
650 DATA 165,26,141,251,5,172,0,6,140,2,208,192,96,208,5,160,151
660 DATA 140,0,6,206,0,6,173,254,5,133,26,173,255,5,133,27,185,214,59
670 DATA 201,0,240,26,74,176,5,32,74,6,144,13,24,201,80,208,5,160,0
680 DATA 140,2,208,32,43,6,165,26,141,254,5,172,3,6,140,3,208,192,111
690 DATA 208,5,160,151,140,3,6,206,3,6,173,1,6,133,26,173,2,6,133,27
700 DATA 185,254,59,201,0,240,26,74,176,5,32,74,6,144,13,24,201,80,208,5
710 DATA 160,0,140,3,208,32,43,6,165,26,141,1,6,173,252,2,201,255
720 DATA 96,0,0,0,0,0,104,169,0,141,0,208,141,1,208,141,2
730 DATA 208,141,3,208,141,111,2,96,0,0,0
750 DATA 161,17,17,17,7,7,7,7,7,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,3,3
760 DATA 3,3,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4
770 DATA 4,4,4,6,6,6,6,6,6,16,16,16,160,0
780 DATA 161,17,17,11,11,7,7,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,3,0,0,0,0,0
790 DATA 0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,6,6,10,10
800 DATA 16,16,160,0
810 DATA 161,15,15,5,5,5,5,5,5,3,3,3,3,3,3,3,3,3,3,0,0,0,0,0,0,0
820 DATA 0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,14,14,160,0
830 DATA 161,17,13,5,5,3,3,3,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
840 DATA 2,2,2,2,2,2,4,4,12,16,160
860 DATA 1,132,150,1,133,150,1,134,150,1,135,150
```

JAN GOLLA



DYNAMICZNE BAJTY

Wprowadzenie dowolnej procedury w kodzie maszynowym pociąga za sobą konieczność deklarowania obszaru pamięci (np. instrukcją CLEAR), a natomiast przesyłanie pewnych wartości zmusza do POKE-wania. W numerze 5/1986 „IKS-a" pojawiła się propozycja przesyłania parametrów za pomocą funkcji użytkownika (FN).

Obecna propozycja dotyczy wprowadzania kodu maszynowego do... tablicy tekstowej.

FN przy wywołaniu znajduje w pamięci adresy przesyłanych tablic tekstowych, co w łatwy sposób pozwala wywołać USR. Dużym udogodnieniem jest możliwość przesyłania tylko fragmentu tablicy tekstowej zawierającej odpowiednią procedurę. Pozwala to na utworzenie całych zestawów (np. graficznych, edycyjnych, matematycznych) trzymanych w jednej tablicy tekstowej.

Oto niektóre wady i zalety tej metody:

Wady:

1) konieczność stosowania DEF FN — należy pamiętać o zgodności parametrów i ich liczby między DEF FN i FN, co ewentualnie pociąga za sobą definiowanie kilku DEF FN. 2) konieczność rezerwowania określonej tablicy tekstowej na kod maszynowy (przy znacznej liczbie tablic, jakie można zadeklarować, nie jest to zbyt trudne. 3) nie wolno używać RUN i CLEAR — powodują one kasowanie obszaru zmiennych, czyli niszczą również procedury (program można z równym powodzeniem uruchomić rozkazami GO TO i GO SUB). 4) procedury muszą być całkowicie relokowalne i wymagają stosowania skoków względnych (skoki te mają jednak ograniczony zasięg).

Zalety:

1) umieszczenie procedury w tablicy tekstowej pozwala zachować maksymalny obszar pamięci dla Basic'a. Nie trzeba się też martwić o miejsce (tablica tekstowa jest automatycznie umieszczona w obszarze zmiennych). 2) eleganckie i czytelne wprowadzenie parametrów przy użyciu FN (interpreter dodatkowo sprawdza, czy przesyłane dane istnieją i czy struktura danych jest taka sama, jak w DEF FN — w razie stwierdzenia błędu, Spectrum przerywa działanie programu i wyświetla odpowiedni komunikat). 3) zestawianie i porządkowanie procedur odbywa się tak jak składanie i kasowanie ciągów tekstowych (trzeba to jednak wykonywać bardzo ostrożnie, aby nie skasować np. części procedury — uruchomienie jej spowoduje najprawdopodobniej reset systemu). 4) zapis, odczyt, weryfikację procedur przeprowadzamy identycznie, jak dla innych tablic tekstowych. 5) kod maszynowy możemy wprowadzić do dowolnej tablicy tekstowej. 6) DEF FN można umieścić w każdym miejscu programu (ze względu na szybkość odszukania przez interpreter DEF FN powinno być raczej na początku programu). 7) procedury są nieliczne i mogą być łączone w dowolne zestawy. 8) procedury są całkowicie zabezpieczone przed błędami — każdy błąd powoduje awaryjne przerwanie programu i wyświetlanie odpowiedniego komunikatu.

Opisana wyżej metoda i jej własności nasunęły mi pewien pomysł: INDYWIDUALNY BANK PROCEDUR. Osoby umiejące programować w kodzie maszynowym mogłyby tworzyć różnorodne procedury spełniające wymienione wymagania. Użytkownik mógłby wykorzystywać je w dowolnych zestawach, które sam by złożył. Wymiana jednej procedury na inną (np. lepszą i szybszą) polegałaby na wymianie odpowiedniego fragmentu tablicy tekstowej. Proponuję również nadsyłać propozycje procedur, byłoby to źródło pomysłów dla twórców oprogramowania.

Aby przybliżyć omawiany problem zachęcam do przeanalizowania procedury, która spełnia wszystkie wymagane kryteria (jest jednocześnie udoskonaleniem programu zamieszczonego w nr. 5/1986 „IKS-a"). Dotyczy zmiany atrybutów koloru w zadanym obszarze ekranu. Wywołuje się ją przez:

FN f(a$,x,y,dx,dy,,w,p,s)

gdzie: a$ — tablica tekstowa zawierająca procedurę maszynową
x — numer kolumny, jak w AT ($\emptyset \le x \le 31$)
y — numer wiersza, jak w AT ($\emptyset \le y \le 21$)
dx — długość obszaru ($1 \le dx \le 32$)
dy — wysokość obszaru ($1 \le dy \le 22$)

Uwaga: ponieważ obszar nie może wykroczyć poza ekran, powinny być spełnione dodatkowe warunki:

$$1 \le x + dx \le 32$$
$$1 \le y + dy \le 22$$

w — atrybut wzorcowy (w = 128 * flash + 64 * bright + 8 * paper + ink) jest on porównywany z atrybutami zadanego obszaru i ich częściowa lub całkowita zgodność (o czym decyduje zmienna „s") może spowodować określane zmiany w atrybucie ekranu.

p — atrybut podstawiany (struktura taka sama jak „w") jest on częściowo lub całkowicie wstawiany w miejsce badanego atrybutu ekranu (o czym decyduje również zmienna „s") wtedy, gdy zostaną spełnione warunki dotyczące zmiennej „w".

s — zmienna sterująca ($\emptyset \le s \le 255$)

	7	6	5	4	3	2	1	Ø
BITY	7	6	5	4	3	2	1	Ø
s = BIN	F	B	P	I	F	B	P	I
	DOTYCZY ZMIENNEJ „W"				DOTYCZY ZMIENNEJ „P"			

gdzie: F — flash
B — bright
P — paper
I — ink
F,B,P,I = {Ø,1}

dla zmiennej „w": jest kontrolowana zgodność tylko tego fragmentu atrybutu ekranu ze zmienną „w", dla którego odpowiednio F,B,P,I przyjmują wartość 1.

dla zmiennej „p": jest wymieniany tylko ten fragment atrybutu ekranu na odpowiadający mu fragment zmiennej „p", dla którego F,B,P,I są równe 1.

Funkcje dodatkowe:

s = BIN ØØØØØØØØ — procedura nie działa
s = BIN ØØØØ1111 — wymiana bezwzględna
s = BIN 1111ØØØØ — FN przyjmuje adres pierwszego atrybutu spełniającego zadane warunki (jeśli taki atrybut istnieje).

Przykład: chcemy na całym obszarze ekranu wymienić ink na czerwony w każdym atrybucie, który ma zie-

lony paper, włączony flash i wyłączony bright:
FN f(a$, 0, 0, 22, 32, BIN 10100XXX, BIN XXXXX010, BIN 11100001), gdzie X może przyjąć wartość „0" albo „1" (przecież zmienna „s" nie narzuca warunku na ink w porównaniu zę zmienną „w", a w czasie wymiany należy wymienić tylko ink).

Ważne wyjaśnienie: kod maszynowy z równym powodzeniem możemy trzymać w zmiennej tekstowej. Można ją w prosty sposób sumować i kasować. Nie można jej niestety zapisywać na taśmie i dlatego przed zapisem tekst ze zmiennej tekstowej powinien być zapisany do tablicy tekstowej.

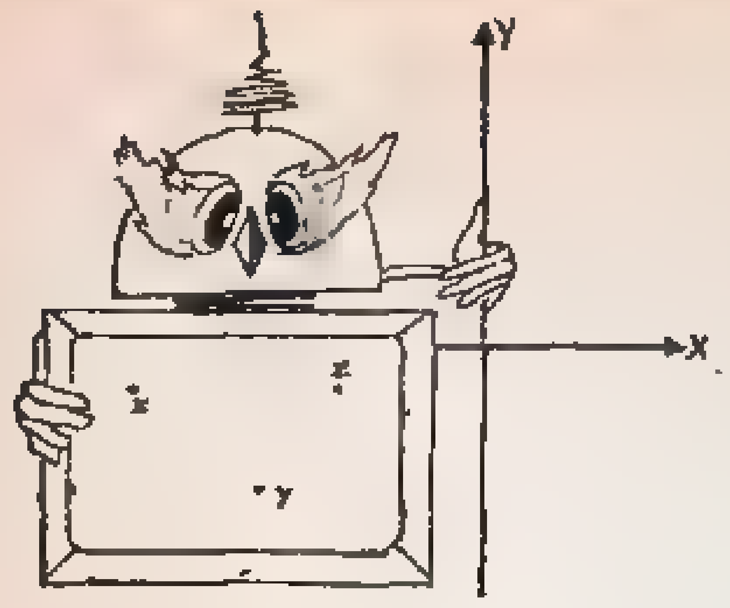
```
   1 REM
********************************
*                              *
*      Krzysztof  Pozniak      *
*                              *
*    Klub Mikrokomputerowy     *
*          HOBBYTE    © 1987   *
*                              *
********************************
   4 RESTORE : DIM a$(233): LET
l=1: LET d=100
   5 LET s1=0: LET s2=0: LET z=1
: FOR k=1 TO 20: READ x: LET s1=
s1+x: LET s2=s2+z*x: LET a$(l)=C
HR$ x: IF l=233 THEN GO TO 7
   6 LET l=l+1: LET z=-z: NEXT k
   7 READ s01: READ s02: IF s1<>
s01 OR s2<>s02 THEN PRINT FLASH
1,"blad w linii ";d: STOP
   8 LET d=d+10: IF l<233 THEN G
O TO 5
   9 CLS
  10 FOR k=1 TO 21: PRINT TAB 10
;"tekst": NEXT k
  20 DEF FN f(a$,a,b,c,d,e,f,g)=
USR (PEEK (PEEK 23563+256*PEEK 2
3564+4)+256*PEEK (PEEK 23563+256
*PEEK 23564+5))
  30 LET s=2: FOR x=19 TO 1 STEP
 -3
  31 IF FN f(a$,s+4,s,x+INT (2*R
ND),x+INT (2*RND),8*INT (7*RND),
INT (7*RND)*8,INT (256*RND)) THE
N
  33 LET s=s+1. NEXT x
  40 FOR p=6 TO 1 STEP -1: IF FN
 f(a$,3+INT (3*RND),3+INT (3*RND
),15,15,p*7,INT (7*RND)*7,INT (2
55*RND)) THEN
  45 NEXT p
  50 GO TO 30
 100 DATA 221,42,11,92,17,8,0,6,
7,221,126,11,221,182,12,221,182,
14,221,182,1997,39
 110 DATA 15,32,45,221,25,16,238
,221,42,11,92,221,182,29,40,32,1
75,221,182,37,2077,-5
 120 DATA 40,26,221,126,13,221,1
34,29,56,18,254,33,48,14,221,126
,21,221,134,37,1993,201
 130 DATA 56,6,254,23,48,2,24,5,
46,25,195,85,0,175,221,182,61,20
0,175,221,2004,155
 140 DATA 102,61,203,124,40,2,24
6,128,203,116,40,2,246,64,203,10
8,40,2,246,56,2232,506
 150 DATA 203,100,40,2,246,7,87,
175,203,92,40,2,246,126,203,84,4
0,2,246,64,2210,898
 160 DATA 203,76,40,2,246,56,203
,68,40,2,246,7,95,221,166,53,111
,123,47,95,2100,694
 170 DATA 122,221,166,45,103,253
,229,213,17,32,0,253,33,0,88,221
,70,21,175,176,2438,-432
 180 DATA 40,4,253,25,16,252,22,
0,221,94,13,253,25,209,221,78,37
,221,70,29,2083,-247
 190 DATA 253,229,175,178,40,12,
253,126,0,162,188,32,13,62,255,1
87,40,28,253,126,2612,328
 200 DATA 0,163,181,253,119,0,25
3,35,16,228,13,175,177,40,11,253
,225,213,17,32,2404,-360
 210 DATA 0,253,25,209,24,207,19
3,253,229,193,253,225,201,2265,-
415
```

```
00001   ; procedura ustawia atrybuty
00002   ; koloru w/g okreslonych warun-
00003   ; kow przekazanych przez para-
00004   ; metry z Basic'a.
00005   ; program calkowicie relokowalny
00006   defadd equ  23563
00007   odleg  equ  8 ;odleglosc miedzy
00008   ;                kolejnymi danymi
00009   ilosc  equ  7 ;ilosc zmiennych
00010   dlug   equ  32 ;ilosc bajtow w
00011   ;                wierszu ekranu
00012   atryb  equ  22528 ;adres pier-
00013   ;                  wszego atry-
00014   ;                  butu koloru
00015          ld   ix,(defadd)
00016          ld   de,odleg
00017          ld   b,ilosc
00018   ;kontrola poprawnosci danych
00019   popr   ld   a,(ix+11)
00020          or   (ix+12)
00021          or   (ix+14)
00022          or   (ix+15)
00023          jr   nz,error
00024          add  ix,de
00025          djnz popr
00026          ld   ix,(defadd)
00027          or   (ix+29)
00028          jr   z,error
00029          xor  a
00030          or   (ix+37)
00031          jr   z,error
00032          ld   a,(ix+13)
00033          add  (ix+29)
00034          jr   c,error
00035          cp   33
00036          jr   nc,error
00037          ld   a,(ix+21)
00038          add  (ix+37)
00039          jr   c,error
00040          cp   23
00041          jr   nc,error
00042          jr   prog
00043   ; obsluga bledu
00044   ;komunikat: Q Parametr error
00045   error  ld   l,25
00046          jp   55h
00047   ; program wlasciwy wymiany
00048   ; tworzenie masek dla porownania
00049   ; i ewentualnej wymiany w/g "s"
00050   prog   xor  a
00051          or   (ix+61)
00052          ret  z
00053          xor  a
00054          ld   h,(ix+61)
00055          bit  7,h
00056          jr   z,skok1
00057          or   a,10000000b
00058   skok1  bit  6,h
00059          jr   z,skok2
00060          or   a,01000000b
00061   skok2  bit  5,h
00062          jr   z,skok3
00063          or   a,00111000b
00064   skok3  bit  4,h
00065          jr   z,skok4
00066          or   a,00000111b
00067   skok4  ld   d,a
00068          xor  a
00069          bit  3,h
00070          jr   z,skok5
00071          or   a,10000000b
00072   skok5  bit  2,h
00073          jr   z,skok6
00074          or   a,01000000b
00075   skok6  bit  1,h
00076          jr   z,skok7
00077          or   a,00111000b
00078   skok7  bit  0,h
00079          jr   z,skok8
00080          or   a,00000111b
00081   skok8  ld   e,a
00082          and  (ix+53)
00083          ld   l,a
00084          ld   a,e
00085          cpl
00086          ld   e,a
00087          ld   a,d
00088          and  (ix+45)
00089          ld   h,a
00090   ; obliczenie w iy adresu pocza-
00091   ; tku okna na ekranie
00092          push iy
00093          push de
00094          ld   de,dlug
00095          ld   iy,atryb
00096          ld   b,(ix+21)
00097          xor  a
00098          or   b
00099          jr   z,dod
00100   mnoz   add  iy,de
00101          djnz mnoz
00102   dod    ld   d,0
00103          ld   e,(ix+13)
00104          add  iy,de
00105          pop  de
00106   ; wymiana atrybutow
00107          ld   c,(ix+37)
00108   petla1 ld   b,(ix+29)
00109          push iy
00110   petla2 xor  a
00111          or   d
00112          jr   z,podst
00113          ld   a,(iy+0)
00114          and  d
00115          cp   h
00116          jr   nz,next
00117          ld   a,255
00118          cp   e
00119          jr   z,pozyc
00120   podst  ld   a,(iy+0)
00121          and  e
00122          or   l
00123          ld   (iy+0),a
00124   next   inc  iy
00125          djnz petla2
00126          dec  c
00127          xor  a
00128          or   c
00129          jr   z,pozyc
00130          pop  iy
00131          push de
00132          ld   de,32
00133          add  iy,de
00134          pop  de
00135          jr   petla1
00136   pozyc  pop  bc
00137          push iy
00138          pop  bc
00139          pop  iy
00140          ret
```

SPECTRUM

```
 10 REM Ruch jednostajnie przyspieszony,autor JAN GOLLA
 20 DIM x(300): DIM y(300)
 30 INPUT "Podaj przyspieszenie a=";a: REM aby wydruk zmiescil sie na ekranieprzyjac a<=5
 40 PRINT AT 1,15;"a=";a: PLOT 110,155: DRAW 50,0
 50 LET t=1
 60 PRINT AT 3+t,0;"v";t;"=";a*t;AT 3+t,8;"s";t;"=";a*t^2/2;AT 3+t,18;"delta s";t;"=";a*t^2/2-a*(t-1)^2/2
 70 LET t=t+1
 80 IF t<=17 THEN GO TO 60
 90 INPUT "Czy sporzadzic wykres v=f(t)";a$
100 IF a$="tak" THEN GO TO 120
110 IF a$="nie" THEN STOP
120 CLS
130 PLOT 0,0: DRAW 250,0: PLOT 0,0: DRAW 0,160: PRINT AT 1,0;"v";AT 21,31;"t"
140 FOR t=1 TO 255
145 IF 10*t>=250 OR a*t>=170 THEN GO TO 165
150 CIRCLE 10*t,a*t,1
160 NEXT t
165 FOR t=1 TO 255
166 IF a*(t/10)>=170 THEN GO TO 170
168 PLOT t,a*(t/10)
169 NEXT t
170 INPUT "czy sporzadzic wykres s=f(t)";b$
180 IF b$="tak" THEN GO TO 200
190 IF b$="nie" THEN STOP
200 CLS
210 PLOT 0,0: DRAW 250,0: PLOT 0,0: DRAW 0,165: PRINT AT 1,0;"S";AT 21,31;"t"
220 FOR t=1 TO 255
225 IF a*t^2/2>=170 OR 10*t>=250 THEN GO TO 250
230 CIRCLE 10*t,a*t^2/2,1
240 NEXT t
250 FOR t=1 TO 255
255 IF a*(t/10)^2/2>=170 THEN STOP
260 PLOT t,a*(t/10)^2/2
270 NEXT t
```

J. GOLLA



SPECTRUM

PRYWATNY JĘZYK PROGRAMOWANIA

Zapewniam cię, drogi czytelniku, że po przeczytaniu tego artykułu i „wpalcowaniu” kilkuset cyferek kodu maszynowego, będziesz mógł stworzyć swój własny język programowania.

Zasada jest prosta: procedura w kodzie maszynowym porównuje wzorce rozkazów z rozkazem rzeczywistym. Jeżeli nastąpi zgodność, odpowiednie parametry porównania zostaną przekazane do Basicu.

Procedurę wywołuje się za pomocą funkcji użytkownika (FN). DEF FN znajduje się w linii piątej programu. Ma pięć parametrów:

- n$ — tablica tekstowa, w której trzymana jest procedura maszynowa. Opis tej metody znajduje się w opisie „Dynamicznych bajtów” (obok),
- t$ — tekst rozkazu rzeczywistego, który będzie porównywany ze wzorcami,
- o$ — tablica bądź zmienna tekstowa przeznaczona na adresy początku i końca parametrów rozkazu,
- r$ — zestaw rozkazów wzorcowych utworzonych z tekstów stałych i symboli zawartych w zmiennej w$,
- w$ — zestaw znaków wzorcowych definiujący poszczególne elementy wzorca plus znak-separator, rozdzielający kolejne wzorce.

Proponuję porównać teraz linię 2Ø1 programu, gdzie definiuję r$, z programem „Grafika przestrzenna”. Oprócz paru dziwnych znaczków teksty stałe wzorców i programu są identyczne. Teraz parę słów o dziwnych znaczkach: opisują one poszczególne elementy wzorca. W nawiasach podaję sposób zapisu. Używam znaków podanych w FN (linia 21Ø). Odpowiadają one pozycji w$ w DEF FN. Uwaga: □ oznacza dowolny znak.

Elementami wzorca są:

- dowolny tekst stały zapisany jako zestaw znaków,
- (£□) tekst wstawiony jako parametr: za tekst uważa się zestaw znaków do napotkania znaku umieszczonego za £,
- (#□) to samo co powyżej, ale łącznie ze znakiem końcowym,
- (@□) pozwala opuścić dowolną ilość zadanego znaku,
- ($) tekst bez narzuconego znaku końcowego,
- (&) dowolna liczba akceptowana przez ZX Spectrum,
- ”/” separator poszczególnych wzorców.

Znaki definiujące poszczególne elementy możemy określić sami. Kolejność znaków przekazywanych do w$ jest następująca: £ # @ $ / & (porównaj linię 21Ø).

Najlepiej prześledzić działanie procedury na przykładzie. Niech rozkazem będzie szósta linia programu „Grafika komputerowa”:

pozycja:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	29	30	31	32
znak:				ś	r	o	d	e	k		o	s	i		1	Ø	Ø	,	B	Ø								

Procedura porównuje wzorce po kolei. Zgodność nastąpi dla wzorca:

pozycja:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
znak:	@		ś	r	o	d	e	k		o	s	l		&	,	&	@	

Działanie analizy jest następujące:

- znak @, a po nim spacja oznacza, że w rozkazie może wystąpić dowolnie dużo spacji — analiza zatrzyma się na znaku „s” (czwarta pozycja rozkazu) gdyż jest on różny od spacji),
- następuje porównanie tekstu stałego aż do symbolu & i pozycji 15 rozkazu,
- symbol & oznacza, że musi wystąpić teraz liczba — pozycja liczby zostaje przekazana do zmiennej o$,
- po liczbie musi wystąpić przecinek,
- potem & narzuca kolejną liczbę — jej parametry miejsca zostają przekazane do zmiennej o$,
- może wystąpić dowolna ilość spacji.

Zmienna o$ wygląda następująco:

pozycja:	1	2	3	4	pozostałe znaki
wartość:	15	18	19	21	nie są ustawione

Za koniec parametru jest uważany pierwszy znak nie należący do danego parametru. Odczyt wartości ze zmiennej o$ uzyskujemy za pomocą CODE. Zgodność wzorca i rozkazu powoduje przesłanie do Basica numeru wzorca (w przypadku, gdy żaden wzorzec nie odpowiada rozkazowi, FN przyjmuje wartość zero). W ten sposób za pomocą FN można skakać do odpowiedniej linii obsługi rozkazu (porównaj linie 21Ø). W przedstawionym przykładzie FN=3, stąd linią obsługi jest linia 214.

Oto pozostałe formaty pozycji:

- £□ zostaje podany adres pierwszego znaku i adresu zadanego znaku,
- #□ zostaje podany adres pierwszego znaku tekstu i adres znaku zadanego zwiększony o jeden,
- $ zostaje podany tylko adres pierwszego znaku.

Procedura jest całkowicie zabezpieczona przed błędami. Złe dane wywołują komunikat systemu:

6 Number too big — gdy podana liczba przekracza zakres arytmetyki ZX Spectrum,

C Nonsense in Basic — gdy wzorzec ma niekompletne dane,

E Out of Data — gdy zmienna odpowiedzi nie może pomieścić wszystkich pozycji parametrów lub gdy rozkaz przekracza 255 znaków,

Q Parametr error — gdy nie ma zgodności parametrów lub gdy wzorzec znaków elementów (w$) nie zawiera dokładnie 6 znaków.

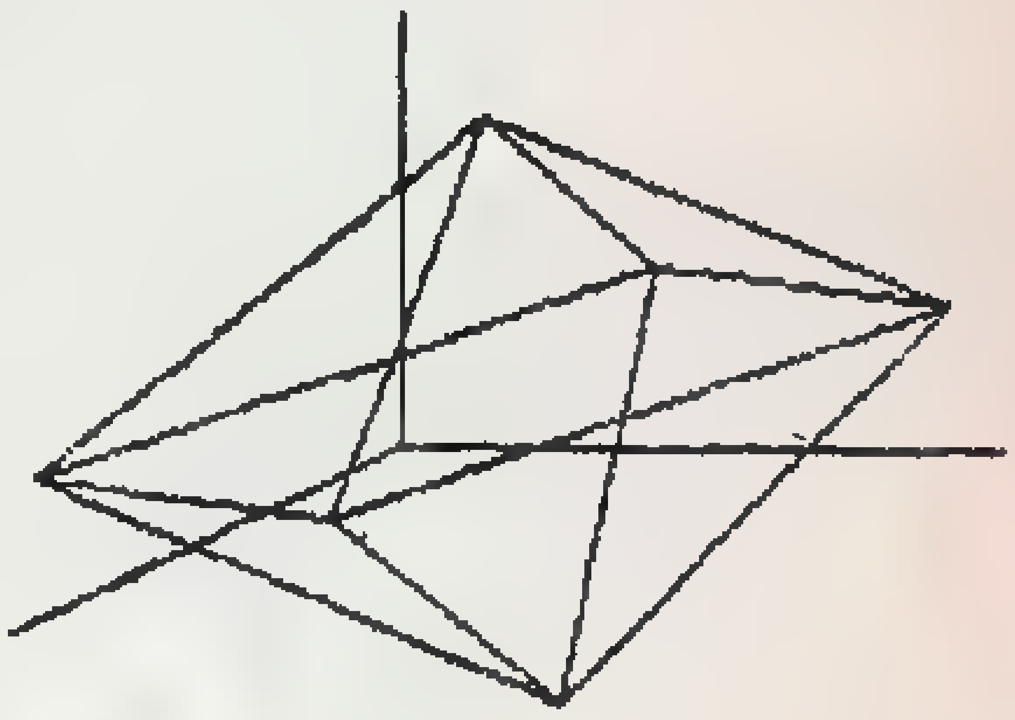
Pojawia się kolejny problem: jak wprowadzić program? W liniach 1Ø—113 znajduje się prosty edytor tekstu, pozwalający na wprowadzenie do 22 linii programu.

Ma on następujące opcje:

- znacznik można poruszać za pomocą kursorów,
- EDIT ustawia znacznik na początku linii,
- CAPS LOOK ustawia znacznik na początku kolejnej linii.

```
  REM [illegible]
********************************
*                              *
*    KRZYSZTOF   POZNIAK       *
*                              *
*    Klub Mikrokomputerowy     *
*                              *
*    [illegible]               *
*                     © 1987   *
*                              *
********************************
   5 DEF FN d(n$,t$,o$,r$,w$)=US
R (PEEK (PEEK 23563+256*PEEK 235
64+4)+256*PEEK (PEEK 23563+256*P
EEK 23564+5))
   8 DIM n$(426): GO TO 9100
  10 REM [illegible]
  15 INK 7: PAPER 0: BORDER 0: C
LS
  20 DIM t$(22*32): DIM i$(32):
LET kl=0: LET kk=0
  25 PRINT AT 0,0;t$: LET inv=1:
 LET dk=0: LET dl=0
  30 GO SUB 100
  31 LET i=CODE INKEY$: IF NOT i
 THEN GO TO 31
  35 BEEP .02,30
  40 IF i=8 AND kk THEN LET dk=-
1: LET dl=0: GO TO 30
  41 IF i=9 AND kk<31 THEN LET d
k=1: LET dl=0: GO TO 30
  42 IF i=10 AND kl<21 THEN LET
dk=0: LET dl=1: GO TO 30
  43 IF i=11 AND kl THEN LET dk=
0: LET dl=-1: GO TO 30
  44 IF i=13 THEN GO TO 200
  45 IF i=12 AND (inv OR kk) THE
N LET i=32*kl+1: LET t$(i TO i+
31)=t$(i TO i+kk-(NOT inv)-1)+t$
(i+kk+1-(NOT inv) TO i+31): PRIN
T AT kl,0;t$(i TO i+31): LET dk=
-(NOT inv)*(NOT NOT kk): LET dl=
0: GO TO 30
  46 IF i=14 THEN LET i=32*kl+1:
 LET t$(i TO i+31)=i$: LET kk=0:
 GO TO 25
  47 IF i=6 THEN LET dl=NOT NOT
(kl<21): LET dk=-kk: GO TO 30
  48 IF i=7 THEN LET kk=0: GO TO
 25
  49 IF i=15 THEN LET inv=NOT in
v: LET dk=0: LET dl=0: GO TO 30
  50 IF i=4 THEN LET i=32*kl: LE
T t$=t$( TO i)+t$(i+33 TO ): GO
TO 25
  51 IF i=5 THEN LET i=32*kl: LE
T t$=t$( TO i)+i$+t$(i+1 TO ): G
O TO 25
  52 IF i>31 AND i<128 AND inv T
HEN LET t$(kl*32+kk+1)=CHR$ i: P
RINT AT kl,kk;CHR$ i: LET dk=(kk
<31): LET dl=0: GO TO 30
  53 IF i>31 AND i<128 AND NOT i
nv THEN LET k=32*kl+1: LET t$(k
TO k+31)=t$(k TO k+kk-1)+CHR$ i+
t$(k+kk TO ): PRINT AT kl,kk;t$(
k+kk TO k+31): LET dk=(kk<31): L
ET dl=0: GO TO 30
  55 LET dk=0: LET dl=0: GO TO 3
0
 100 REM [illegible]
 110 PRINT INVERSE 0; FLASH 0;AT
 kl,kk;SCREEN$ (kl,kk)
 111 LET kk=kk+dk: LET kl=kl+dl
 112 PRINT INVERSE inv; FLASH NO
T inv;AT kl,kk;SCREEN$ (kl,kk)
 113 RETURN
 200 REM [illegible]
 201 LET o$="[illegible]": LET
r$="@ ,&/@ czysc ekran@ /@ srode
k osi &,&@ /@ wymiary osi &,&,&,
&@ /@ rysuj osie@ /@ punkt &,&,&
@ /@ prosta &,&,&@ /@ polacz &,&
,&@ /@ /@ koniec@ "
 205 REM parametry wstepne
 206 LET px=0: LET py=0: LET pz=
0: LET sx=120: LET sy=80: LET wx
=15: LET wy=15: LET wz=15: LET w
xz=-10: LET wyz=-11
 210 FOR k=0 TO 21: LET i=32*k:
GO TO 211+FN d(n$,t$(i+1 TO i+32
),o$,r$,"£#@$/&")
 211 BEEP .2,20: PRINT #1;AT 0,0
;"blad w linii :" INVERSE 1;i$(
k*32+1 TO k*32+31): PAUSE 0: INP
UT "": GO TO 25
 212 GO TO 1000
 213 CLS : GO TO 1000
 214 LET sx=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET sy=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): GO TO 1000
 215 LET wx=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET wy=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
) LET wz=VAL t$(i+CODE o$(5) TO
 i+CODE o$(6)-1): LET wyz=SIN (P
I/180)*VAL t$(i+CODE o$(7) TO i+
CODE o$(8)-1): LET wxz=INT (wz*C
OS wyz): LET wyz=-INT (wz*SIN wy
z): GO TO 1000
 216 PLOT sx,sy  DRAW 255-sx,0.:
 PLOT sx,sy  DRAW 0,175-sy: PLOT
 sx,sy. LET z1=sx wxz: LET z2=sy
/wyz: LET z1=z1*(z1>z2)+z2*(z2>z
1). DRAW -z1*wxz,-z1*wyz: GO TO
1000
 217 LET px=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET py=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): LET pz=VAL t$(i+CODE o$(5) TO
 i+CODE o$(6)-1). PLOT sx+px*wx+
wxz*pz,sy+py*wy+wyz*pz: GO TO 10
00
 218 LET z1=VAL t$(i+CODE o$(1)
TO i+CODE o$(2)-1): LET z2=VAL t
$(i+CODE o$(3) TO i+CODE o$(4)-1
): LET z3=VAL t$(i+CODE o$(5) TO
 i+CODE o$(6)-1): DRAW z1*wx+z3*
wxz,z2*wy+z3*wyz: LET px=px+z1:
LET py=py+z2: LET pz=pz+z3: GO T
O 1000
 219 LET z4=PEEK 23677: LET z5=P
EEK 23678: LET z1=VAL t$(i+CODE
o$(1) TO i+CODE o$(2)-1): LET z2
=VAL t$(i+CODE o$(3) TO i+CODE o
$(4)-1): LET z3=VAL t$(i+CODE o$
(5) TO i+CODE o$(6)-1): DRAW (z1
-px)*wx+(z3-pz)*wxz,(z2-py)*wy+(
z3-pz)*wyz: POKE 23677,z4: POKE
23678,z5: GO TO 1000
 220 GO TO 1000
 221 GO TO 9000
1000 NEXT k
9000 PRINT #1;AT 0,0; INVERSE 1;
"program wykonany": PAUSE 0: INP
UT "": GO TO 25
9100 RESTORE 9950: LET data=9950
: LET z3=1: LET z1=LEN n$
9102 LET s1=0: LET s2=0: LET r=1
: FOR k=1 TO 20: READ z2: LET n$
(z3)=CHR$ z2: LET s1=s1+z2: LET
s2=s2+r*z2: LET r=-r: IF z3=z1 T
HEN GO TO 9105
9103 LET z3=z3+1: NEXT k
9105 READ ss1,ss2: IF ss1<>s1 OR
 ss2<>s2 THEN PRINT "blad w lini
i ";data: STOP
9106 IF z3=z1 THEN GO TO 10
9107 LET data=data+1: GO TO 9102
9950 DATA 221,42,11,92,221,126,4
3,183,32,7,221,126,42,254,6,40,9
,33,66,39,1536,-48
9951 DATA 217,46,25,195,85,0,42,
93,92,229,217,221,110,31,221,102
,32,221,94,33,2306,-36
9952 DATA 221,86,34,1,1,0,221,11
7,4,221,116,5,221,116,6,221,114,
7,217,221,2149,161
9953 DATA 78,40,221,70,41,197,22
1,110,13,221,102,14,221,94,15,22
1,86,16,221,126,2487,11
9954 DATA 217,122,179,40,93,125,
38,87,221,190,0,40,78,221,190,1,
40,72,221,190,2300,242
9955 DATA 2,40,69,221,190,3,40,6
6,217,221,190,4,40,27,221,190,5,
40,64,8,1658,216
9956 DATA 122,179,40,52,8,190,32
,46,35,27,24,204,33,68,38,217,46
,11,195,85,1673,-527
9957 DATA 6,122,179,217,40,28,3,
221,229,221,42,11,92,221,117,4,2
21,116,[illegible],-464
9958 DATA [illegible],7,217,24
,[illegible],24,116,2
4,116,217,[illegible],-414
9959 DATA [illegible],179,40,
[illegible],27,0,122,
[illegible]
9960 DATA [illegible],56,107,0
[illegible],213,197,
[illegible],-205
9961 DATA [illegible],217,209
,[illegible],225,237
[illegible]
9962 DATA [illegible],179,4
0,140,[illegible],40,56,6,
[illegible]
9963 DATA [illegible],137,122,17
[illegible]
9964 DATA [illegible],84,243,24
[illegible],40,256,1
[illegible],-767
9965 DATA [illegible],190,0,[illegible]
,[illegible],217,221,[illegible]
[illegible]
9966 DATA [illegible],38,87,40,19
[illegible],38,197,217,
[illegible],-647
9967 DATA [illegible],201,217,221,8
[illegible],4,221,18
[illegible]
9968 DATA [illegible],113,4,221
[illegible],23,217,2
[illegible],-161
9969 DATA [illegible],84,221,7
[illegible],217,124,179,
[illegible],-67
9970 DATA [illegible],221,190,4
,[illegible],221,190,3
[illegible],-397
9971 DATA 190,[illegible],40,110,221,190,2
,40,113,221,190,0,40,114,35,28,4
0,8,24,[illegible],1816,-26
9972 DATA 33,85,39,217,46,13,195
,85,0,33,89,39,221,42,11,92,217,
221,70,4,1768,94
9973 DATA 221,70,5,226,34,93,92,
201,24,100,120,177,40,222,11,123
,217,2,3,217,2277,-743
9974 DATA 126,254,45,32,5,35,126
,88,40,208,34,93,92,213,229,217,
229,213,197,205,2619,-373
9975 DATA 155,44,205,213,45,193,
209,225,217,209,42,93,92,183,237
,82,235,225,25,235,3164,-240
9976 DATA 122,163,32,172,42,93,9
2,120,177,40,165,11,123,217,2,3,
24,186,24,165,1993,-387
9977 DATA 24,26,24,46,24,71,24,1
4,122,177,40,144,11,123,217,2,3
,126,217,190,1755,-347
9978 DATA 40,221,35,28,40,130,24
,247,120,177,40,245,11,123,217,2
,3,126,35,217,1994,-764
9979 DATA 217,28,40,236,190,35,4
0,135,24,247,217,126,35,87,217,1
90,32,204,35,24,2353,-269
9980 DATA 40,2,24,247,217,122,17
9,40,185,126,221,190,4,40,179,24
,203,120,177,40,2380,478
9981 DATA 199,123,217,2,24,188,7
33,147
```

```
; grafika przestrzenna
; rysowanie osmioscianu
;
  czysc ekran
; parametry ekranu
  srodek osi 100,80
  wymiary osi 20,20,20,155
  rysuj osie
; osmioscian
  punkt 1,0,2
  polacz 4,-2,2
  polacz 2,4,1
  prosta 0,2,4
  polacz 4,-2,2
  polacz 2,4,1
  prosta 7,2,-1
  polacz 4,-2,2
  polacz 2,4,1
  prosta 0,-2,-4
  polacz 4,-2,2
  polacz 2,4,1
  prosta -7,-2,1
```



— TRUE VIDEO kasuje linię, w której znajduje się znacznik,
— INV VIDEO tworzy nową linię nad linią ze znacznikiem (zniszczeniu ulega linia 22),
— ENTER uruchamia program,
— EXTEND MODE kasuje treść linii, w której znajduje się znacznik i ustawia znacznik na początku tej linii,
— GRAPHICS ustawia tryb wprowadzania i kasowania:

a) znacznik INVERSE
— znak w znaczniku jest wymieniany,
— kasowanie niszczy znak w znaczniku,

b) znacznik FLASH:
— znak jest dopisywany przed znacznikiem (ginie ostatnia pozycja linii),
— kasowanie niszczy znak przed znacznikiem,

— DELETE kasuje znaki zgodnie z trybem GRAPHICS,
— mogą być wprowadzane znaki pojedyncze o kodach od 32 do 127.

Mniej doświadczonym programistom zwracam uwagę na identyfikację opcji za pomocą CODE INKEY$.

Uruchamianie edytora:
od linii 10 — kasowania tekstu programu,
od linii 25 — program nie zostaje zniszczony.

I ostatni problem: dotyczy programu-przykładu „Grafika przestrzenna". Należy przedstawiony program wprowadzić za pomocą edytora i uruchomić przez ENTER. Na ekranie powinien pojawić się rysunek pokazany poniżej. Życzę ciekawych wyni-

ków graficznych dzięki zmianie parametrów w rozkazach. Zwracam jednak uwagę, że ze względu na prostotę programu, nie ma tu zabezpieczeń przed błędnymi danymi. Z tego powodu rysunek może „wyjść" poza ekran i wywołać błąd systemu. Należy wtedy wykonać GO TO 25.

Przedstawiam dostępne rozkazy grafiki przestrzennej:
— ;dowolny tekst komentarza,
— czyść ekran — odpowiada CLS,
— środek osi x, y — ustawia środek osi w punkcie odpowiadającym PLOT x, y,
— rysuj osie — rysuje osie układu przestrzennego,
— punkt x, y, z — stawia na ekranie punkt o podanych współrzędnych,
— polacz x, y, z — łączy ostatnio narysowany punkt z punktem o zadanych współrzędnych,

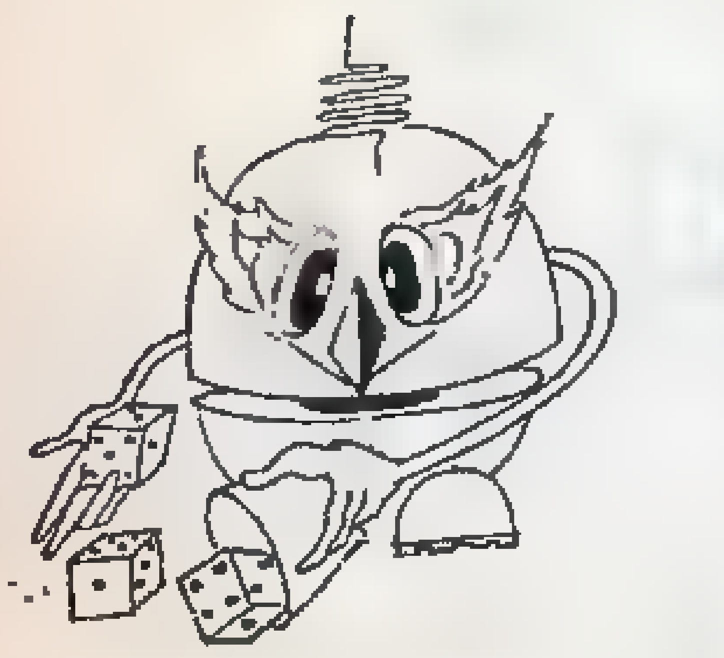
— prosta dx, dy, dz — rysuje prostą analogicznie do DRAW, ale w przestrzeni,

— koniec — powoduje pominięcie linii poniżej i zakończenie programu.
Uwagi dodatkowe:
— program pomija linię złożoną z samych spacji,
— program wypisuje linię nie odpowiadającą żadnemu wzorcowi,
— wymiary osi wx, wy, wz, kąt — określa liczbę pikseli przypadających na odległość jednostkową oraz ustala kąt rzutu osi z,
— prawidłowe wykonanie programu powoduje komunikat: „program wykonany"; naciśnięcie dowolnego klawisza spowoduje powrót do edytora.

Zamiast przedstawionego programu w liniach 200—9000 można umieścić swój prywatny język programowania.

Krzysztof POŹNIAK

SPECTRUM



KATALOG GIER

Program powstał z myślą o ułatwieniu życia zwolennikom gier, którzy wymieniają je między sobą. Jest on specjalizowaną bazą danych, umożliwiającą zapis do 300 rekordów. Każdy z nich zawiera 4 pola (zmienne N$, R$, V$, T$). Na każdym polu można zapisać do 25 symboli (liter lub cyfr). Program przeznaczony jest głównie do wykonywania wykazów gier w alfabetycznej kolejności tytułów, wzbogaconych dodatkowo o charakterystykę gry wg symboliki zamieszczonej na planszy „CHARAKTERYSTYKA GIER". Program ładuje się normalnie, tzn. LOAD"" i obsługuje zgodnie z instrukcjami wyświetlanymi na ekranie. Menu zawiera 6 opcji:

1) Opcja A — umożliwia, po podłączeniu drukarki (ZX-Printer, Seikosha GP-50S), automatyczne uzyskanie wykazu gier. Poza tytułem i charakterystyką gry drukowany i wyświetlany jest również numer, pod którym gra jest zapisana w „KATALOGU GIER".

2) Opcja S — umożliwia szybkie sprawdzenie, jakie tytuły gier, zaczynające się na daną literę, są zapisane w „KATALOGU GIER". Ułatwia ona właściwe wpisanie tytułu gry po wybraniu opcji P.

3) Opcja P — umożliwia dostęp do informacji dotyczących gry. Wyświetlany jest tytuł gry, jej producent, krótki opis gry oraz rok produkcji i nr taśmy, na której jest nagrana. Opcja P pozwala również na zmodyfikowanie zapisów na poszczególnych polach, bez kasowania wszystkich danych opcją K.

4) Opcja W — umożliwia wpisanie kolejnego rekordu. Jeżeli nie chcesz lub nie możesz zapisać danego pola, to wciśnij ENTER. Dwie pierwsze litery na polu o nazwie „Rodzaj gry" zapisz w trybie C, zgodnie z symboliką wyświetlaną na planszy „CHARAKTERYSTYKA GIER", dostępnej po wybraniu opcji A*. Resztę pola możesz zapisać dowolnie, np. dla tytułu ANT ATTACK: ZP Królewny i mrówki, co charakteryzuje grę jako zręcznościowo-przygodową i informuje, co cię czeka podczas gry. Tytuły gier zawsze wpisuj w trybie C, gdyż klawiatura automatycznie do niego przechodzi po wybraniu opcji P lub S.

5) Opcja K — umożliwia skasowanie dowolnego zapisu. Jeżeli wybrałeś ją przez pomyłkę, to wpisz O i wciśnij ENTER, a program wróci do menu. Jeśli pomyliłeś się przy wpisywaniu numeru zapisu do skasowania, to dopisz kilka cyfr, aby liczba która powstanie, była większa od 300 i wciśnij ENTER, co pozwoli ci ponownie wpisać numer zapisu, który chcesz skasować.

6) Opcja Z — umożliwia zapis wprowadzonych danych na taśmie. Po zakończeniu nagrywania nie jest wyświetlany komunikat O.K., przestaje jedynie migać border.

I jeszcze parę uwag praktycznych. Program wgrywa się około 3 minut. Korzystnie jest 2 egzemplarze programu uaktualniać na zmianę, co zabezpiecza przed utratą danych i programu. Nic nie stoi na przeszkodzie, aby użytkownik „KATALOGU GIER" zmienił nazwy pól w rekordzie, jeżeli chce wprowadzić inne dane niż zaproponowane, lub też rozszerzyć o swoje dane personalne i adres nagłówek drukowany automatycznie w opcji A. Po wykonaniu wykazu gier, program daje możliwość wydrukowania dowolnego tekstu, stosownie do potrzeb.

Inspiracją do opracowania „KATALOGU GIER" był zamieszczony w nr. 811/85 francuskiego miesięcznika „Science et Vie" program pt. „Fichier confidentiel". Usunąłem z niego błędy, zaadaptowałem oraz rozszerzyłem.

* P — przygodowa
S — sportowa
Ł — losowa
Y — symulacyjna
T — tekstowa
Z — zręcznościowa
W — wojenna
R — strategiczna
K — kosmiczna
U — pr. użytkowy

A. URBANKOWSKI

```
  10 REM KATALOG
 100 DIM N$(300,25)
 110 DIM R$(300,25)
 120 DIM V$(300,25)
 130 DIM T$(300,25)
 140 DIM X$(1,25)
 141 BORDER 7: PAPER 7: INK 0: C
LS
 142 BORDER 1: PAPER 1: INK 7: P
RINT AT 8,10;"KATALOG GIER": INV
ERSE 1: PRINT AT 0,0;" © 1987 A.
Urbankowski": INVERSE 0: BEEP .1
,20: FLASH 1: PRINT AT 19,8;"ZAT
RZYMAJ  TASME": INVERSE 1: PRINT
 AT 21,5;"WCISNIJ DOWOLNY KLAWIS
Z": INVERSE 0: FLASH 0
 144 PRINT AT 5,7;"*************
*****";AT 11,7;"****************
**"
 146 PRINT AT 6,7;"*": PRINT AT
6,24;"*"
 148 PRINT AT 7,7;"*": PRINT AT
7,24;"*"
 150 PRINT AT 8,7;"*": PRINT AT
8,24;"*"
 152 PRINT AT 9,7;"*": PRINT AT
9,24;"*"
 154 PRINT AT 10,7;"*": PRINT AT
10,24;"*"
 156 IF INKEY$="" THEN GO TO 156
 157 CLS : BEEP .1,20
 158 BORDER 0: PAPER 0: INK 6
 160 FLASH 1: INK 6: PRINT AT 1,
13;"UWAGA": FLASH 0
 164 PRINT AT 3,0;"Program ""KAT
ALOG"" umozliwia ska-talogowanie
 300 gier.Program ob-sluguje sie
 wedlug instrukcji   wyswietlany
ch na ekranie.Pod za-dnym pozore
m nie nalezy uzywac  rozkazu RUN
, ktory niszczy dane  zgromadzon
e w pamieci! Program zatrzymany
przez C.SH+SPACE moz-na uruchomi
c rozkazem GO TO 141,GO TO 200 l
ub CONTINUE.Tytul na-lezy wpisyw
ac w trybie C (duze  litery).Pod
laczenie drukarki po-zwala uzysk
ac wykaz posiadanych gier w porz
adku alfabetycznym.  Tytulu gier
 majace na poczatku  takie same
wyrazy zapisz laczac je znakiem"
"-"",np:JET-PAC,JET-MAN"

 166 INVERSE 1: INK 6: PRINT AT
21,5;"WCISNIJ DOWOLNY KLAWISZ":
INVERSE 0: INK 6
 170 IF INKEY$="" THEN GO TO 170
 180 BEEP .1,20: CLS
 200 OUT 254,0: PAPER 0: INK 6:
PRINT AT 9,1;"Poszukiwanie danyc
h-wcisnij: P";AT 13,3;"Wpisanie
danych-wcisnij: W";AT 5,2;"Spraw
dzenie tytulu-wcisnij:S"
 210 PRINT AT 0,7;"Automatyczny
wydruk";AT 1,8;"tytulow-wcisnij:
A";AT 17,3;"Kasowanie zapisu-wci
snij: K";AT 21,0;"Zapis danych n
a tasme-wcisnij: Z"
 215 LET M=0
 220 LET D$=INKEY$
 250 IF D$="p" THEN GO SUB 1000
 260 IF D$="w" THEN GO SUB 2000
 270 IF D$="k" THEN GO SUB 6000
 280 IF D$="z" THEN GO SUB 7000
 282 IF D$="s" THEN GO SUB 8000
 285 IF D$="a" THEN GO SUB 9000
 290 IF M=1 THEN GO SUB 2210
 300 GO TO 200
1000 CLS : BEEP .1,20
1010 PRINT AT 0,7;"POSZUKIWANIE
DANYCH"
1020 PRINT AT 5,3;"Wpisz tytul,w
cisnij ENTER"
1030 FOR B=1 TO 25
1040 LET X$(1,B)=" "
1050 NEXT B
1100 LET C=0
1110 IF INKEY$="" THEN GO TO 111
0
1120 LET C$=INKEY$
1130 IF INKEY$<>"" THEN GO TO 11
30
1135 IF CODE C$>90 THEN LET C$=C
HR$ (CODE C$-32)
1140 IF CODE C$<>13 THEN LET X$(
1,C+1)=C$
1150 LET C=C+1-2*((CODE C$=8)*(C
<>0))
1160 INK 2: PRINT AT 7,1+C;"-":
BEEP .01,50
1170 INK 4: PRINT AT 7,C;C$: INK
 6
1180 IF C=25 OR CODE C$=13 THEN
GO TO 1200
1190 GO TO 1110
1200 LET R=1
1205 PRINT AT 2,10;"TYTUL Nr."
1210 LET T=0
1230 FOR I=1 TO 25
1235 IF R=301 THEN GO TO 1400
1240 IF X$(1,I)<>N$(R,I) THEN LE
T T=1
1245 IF N$(R,I)=" " OR T=1 THEN
LET I=25
1250 NEXT I
1265 INK 2: PRINT AT 2,21;R
1275 LET R=R+1
1280 IF T=1 THEN GO TO 1210
1290 BEEP .1,20
1310 INK 7: PRINT AT 7,0;N$(R-1)
;AT 10,0;R$(R-1);AT 13,0;V$(R-1)
;AT 16,0;T$(R-1)
1330 INK 6
1340 PRINT AT 5,3;" Informacje d
otyczace gry:";AT 19,1;"NASTEPNY
 TYTUL-WCISNIJ SPACE";AT 21,1;"M
ODYFIKACJE ZAPISOW-WCISNIJ M"

1350 IF INKEY$="m" THEN GO SUB 7
500
1370 IF INKEY$<>" " AND M=0 THEN
 GO TO 1350
1380 CLS
1390 RETURN
1400 FLASH 1: INK 4: PRINT AT 15
,9;"TYTUL NIEZNANY": FLASH 0: IN
K 6
1410 BEEP 1,0: BEEP 1,12: BEEP 1
,24: CLS
1420 RETURN
2000 CLS : BEEP .1,20
2010 PRINT AT 0,8;"WPISANIE DANY
CH"
2020 INK 2: FLASH 1: PRINT AT 2,
8;"   POCZEKAJ   ": FLASH 0
2100 LET R=1
2110 IF N$(R,1)=" " OR R>300 THE
N GO TO 2200
2120 LET R=R+1
2130 IF R<>301 THEN GO TO 2110
2140 PRINT AT 2,8;"  BRAK MIEJSC
A  "
2145 RETURN
2200 PRINT AT 2,9;"ZAPIS  Nr:";R

2210 INK 6: PRINT AT 5,3;"Wpisz
tytul,wcisnij ENTER"
2211 IF M=1 THEN PRINT AT 10,0;"
Tytul aktualny:";AT 11,0;N$(R)
2215 LET C=0
2220 GO SUB 5000
2230 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET N$(R,C)=C$
2240 IF CODE C$=13 OR C=25 THEN
GO TO 2300
2250 GO TO 2220
2300 CLS
2310 INK 6: PRINT AT 5,1;"Wpisz
Producenta,wcisnij ENTER"
2311 IF M=1 THEN PRINT AT 10,0;"
Producent aktualny:";AT 11,0;R$(
R)
2315 LET C=0
2320 GO SUB 5000
2330 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET R$(R,C)=C$
2340 IF CODE C$=13 OR C=25 THEN
GO TO 2400
2350 GO TO 2320
2400 CLS
2410 INK 6: PRINT AT 5,1;"Wpisz
rodzaj gry,wcisnij ENTER"
2411 IF M=1 THEN PRINT AT 10,0;"
Rodzaj gry aktualny:";AT 11,0;V$
(R)
2415 LET C=0
2420 GO SUB 5000
2430 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET V$(R,C)=C$
2440 IF CODE C$=13 OR C=25 THEN
GO TO 2500
2450 GO TO 2420
2500 CLS
2510 INK 6: PRINT AT 3,1;"Wpisz
rok produkcji i nr tasmy,";AT 5,
9;"wcisnij ENTER"
2511 IF M=1 THEN PRINT AT 10,0;"
Rok produkcji,nr tasmy aktualny:
";AT 11,0;T$(R)
2515 LET C=0
2520 GO SUB 5000
2530 IF CODE C$<>13 AND CODE C$<
>8 AND C<>0 THEN LET T$(R,C)=C$
2540 IF CODE C$=13 OR C=25 THEN
GO TO 2600
2550 GO TO 2520
2600 CLS
2610 GO TO 200
5000 IF M=1 THEN PRINT AT 19,2;"
Wcisnij ENTER po modyfikacji";AT
 20,2;"danych,lub aby je pozosta
wic";AT 21,11;"bez zmian"
5005 IF INKEY$="" THEN GO TO 500
5
5010 LET C$=INKEY$
5020 IF INKEY$<>"" THEN GO TO 50
20
5030 LET C=C+1-2*((CODE C$=8)*(C
<>0))
5040 INK 2: PRINT AT 7,1+C;"-":
BEEP .01,50
5050 INK 4: PRINT AT 7,C;C$: INK
 6
5060 RETURN
6000 INK 2: PRINT AT 18,4;"NR ZA
PISU DO SKASOWANIA?";AT 20,0;"[illegible]
[illegible]"
: INK 6: BEEP 1,-24
6010 INPUT F
6020 IF F=0 THEN GO TO 6100
6030 IF F<1 OR F>300 THEN GO TO
6010
6040 FOR I=1 TO 25
6050 LET N$(F,I)=""
6060 LET R$(F,I)=""
6070 LET V$(F,I)=""
6080 LET T$(F,I)=""
6090 NEXT I
6095 BEEP .1,20
6100 CLS
6110 RETURN
7000 SAVE "KATALOG" LINE 141
7100 RETURN
7500 LET M=1
7510 LET R=R-1
7520 RETURN
8000 CLS : PRINT AT 0,2;"PODAJ P
IERWSZA LITERE TYTULU";AT 1,11;"
I POCZEKAJ": PRINT : PRINT : BEE
P .1,20
8005 IF INKEY$<>"" THEN GO TO 80
05
8010 IF INKEY$="" THEN GO TO 801
0
8020 LET K$=INKEY$
8030 IF CODE K$>90 THEN LET K$=C
HR$ (CODE K$-32)
8100 FOR K=1 TO 300
8110 IF N$(K,1)=K$ THEN PRINT K;
TAB 4;N$(K);TAB 30;V$(K,1);TAB 3
1;V$(K,2)
8120 NEXT K
8130 BEEP .1,20
8140 FLASH 1: INVERSE 1: PRINT A
T 21,0;"POWROT DO MENU:WCISNIJ C
OKOLWIEK": INVERSE 0: FLASH 0
8150 IF INKEY$="" THEN GO TO 815
0
8160 CLS : BEEP .1,20: RETURN
9000 CLS : PAUSE 25: BEEP .1,20
9001 PRINT AT 0,6;"CHARAKTERYSTY
KA GIER";AT 1,5;"...............
.......": PRINT : PRINT : PRINT
"P-przygodowa","Z-zrecznosciowa"
: PRINT : PRINT "S-sportowa","W-
wojenna": PRINT : PRINT "L-losow
a","R-strategiczna": PRINT : PRI
NT "Y-symulacyjna","K-kosmiczna"
: PRINT : PRINT "T-tekstowa","U-
pr. uzytkowy": FLASH 1: INVERSE
1: PRINT AT 21,5;"WCISNIJ DOWOLN
Y KLAWISZ": INVERSE 0: FLASH 0
9002 IF INKEY$="" THEN GO TO 900
2
9003 BEEP .1,20: CLS
9004 PRINT " WYKAZ GIER    ZX-SP
ECTRUM 48kb ": PRINT "==========
=======================": PRINT :
 PRINT
9005 LPRINT " WYKAZ GIER    ZX-S
PECTRUM 48kb ": LPRINT "========
=========================": LPRIN
T : LPRINT : LPRINT
9006 POKE 23692,255
9008 FOR L=65 TO 90
9010 LET K$=CHR$ L: GO SUB 9100
9012 NEXT L
9065 LPRINT : LPRINT : LPRINT "
      CHARAKTERYSTYKA GIER      "
: LPRINT "   ...................
.....    ": LPRINT : LPRINT : L
PRINT "P-przygodowa","Z-zrecznos
ciowa": LPRINT : LPRINT "S-sport
owa","W-wojenna": LPRINT : LPRIN
T "L-losowa","R-strategiczna": L
PRINT : LPRINT "Y-symulacyjna","
K-kosmiczna": LPRINT : LPRINT "T
-tekstowa","U-pr. uzytkowy": LPR
INT
9070 IF K$=CHR$ 90 THEN INPUT "W
prowadz tekst do wydruku i nacis
nij ENTER";Y$: LPRINT : LPRINT :
 LPRINT : LPRINT Y$
9080 CLS : FLASH 1: INVERSE 1: P
RINT AT 10,10;"WYKAZ GOTOWY": BE
EP 1,0: BEEP 1,12: BEEP 1,24: IN
VERSE 0: FLASH 0: CLS : RETURN
9100 FOR K=1 TO 300
9110 IF N$(K,1)=K$ THEN PRINT K;
TAB 4;N$(K);TAB 30;V$(K,1);TAB 3
1;V$(K,2)
9120 IF N$(K,1)=K$ THEN LPRINT K
;TAB 4;N$(K);TAB 30;V$(K,1);TAB
31;V$(K,2)
9130 NEXT K
9140 BEEP .1,20: BEEP .1,20: BEE
P .1,20: BEEP .1,20: PAUSE 50: P
RINT : LPRINT : RETURN
```

PROSTOPADŁOŚCIAN

```
 210 LET wskrys=1
 220 GO SUB rysowanie
 230 LET wskrys=0
 240 LET dlbloku=dlbloku+DELTA
 250 GO SUB rysowanie
 260 GO TO wczytaniezn
 900 REM

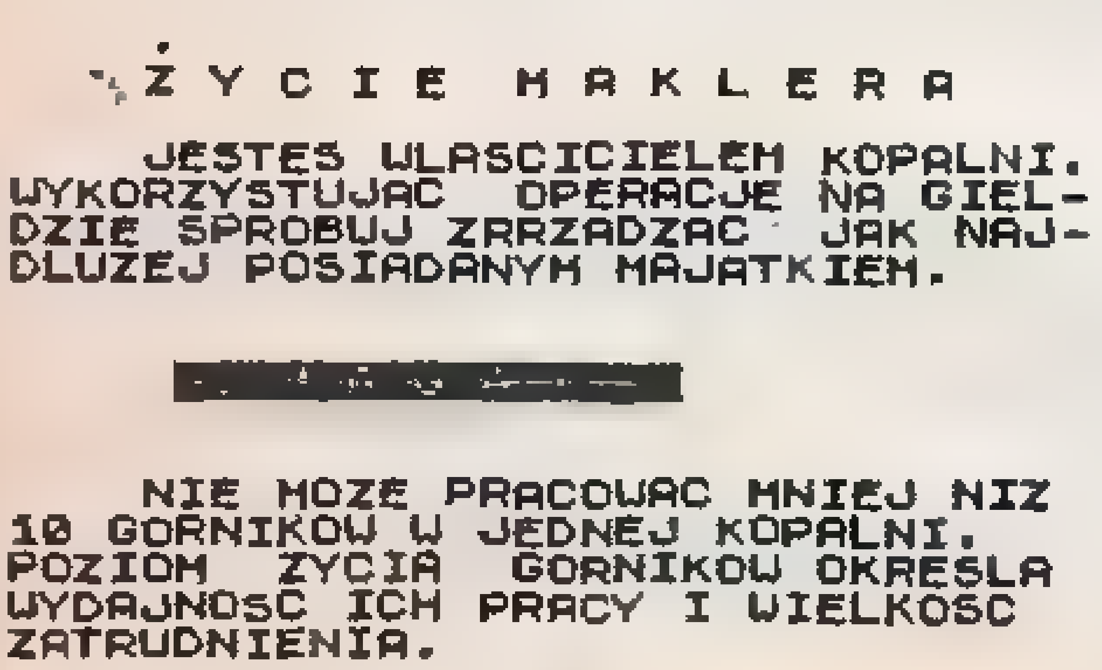
1000 REM rysowanie
         prostopadloscianu
1010 PLOT INVERSE wskrys;
     110-dlbloku,64+dlbloku
1020 DRAW INVERSE wskrys;
     2*dlbloku,0
1030 DRAW INVERSE wskrys;
     0,-2*dlbloku
1040 DRAW INVERSE wskrys;
     dlbloku/2,dlbloku
1050 DRAW INVERSE wskrys;
     0,1.8*dlbloku
1060 DRAW INVERSE wskrys;
     -dlbloku/2,-0.8*dlbloku
1070 PLOT INVERSE wskrys;
     110+1.5*dlbloku,
     64+1.8*dlbloku
1080 DRAW INVERSE wskrys;
     -1.8*dlbloku,0
1090 DRAW INVERSE wskrys;
     -0.7*dlbloku,-0.8*dlbloku
1100 DRAW INVERSE wskrys;
     0,-2*dlbloku
1110 DRAW INVERSE wskrys;
     2*dlbloku,0
1120 RETURN

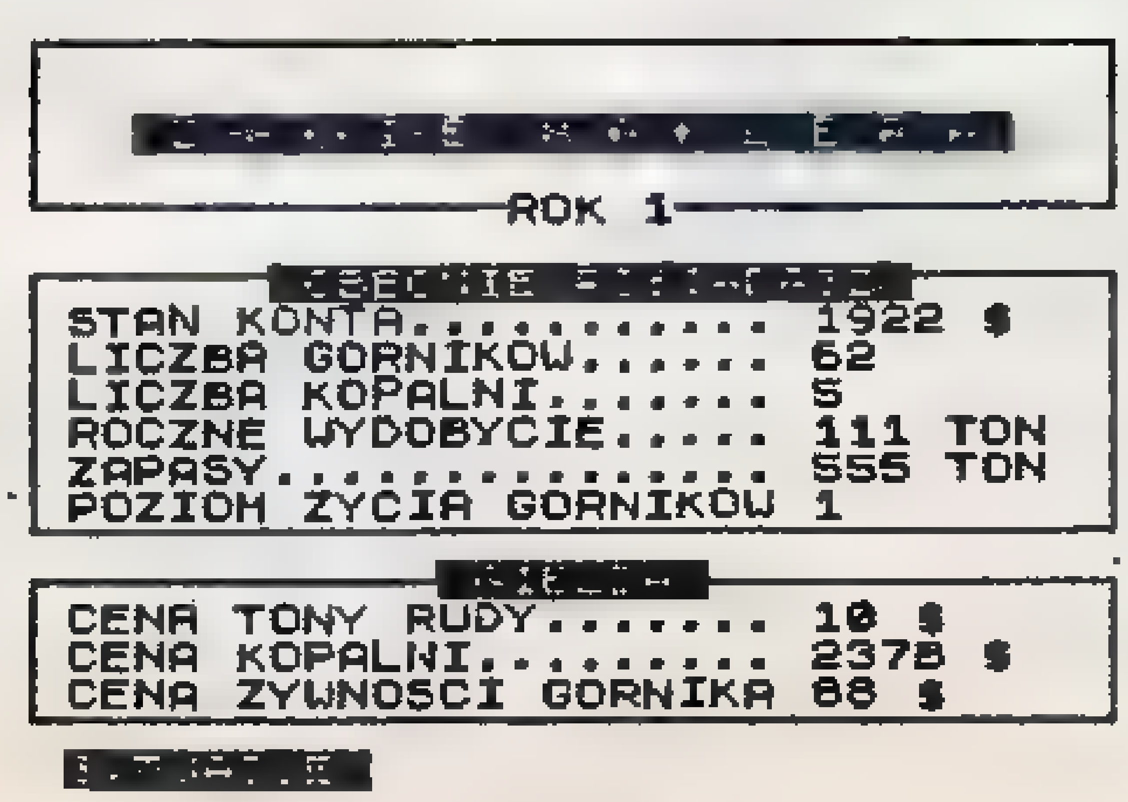
  20 LET rysowanie=1000:
     LET wczytaniezn=160
  40 BORDER 0: PAPER 6:
     INK 9: BRIGHT 1: CLS
  50 PRINT PAPER 1;AT 2,0;  "
Powiekszanie i zmniejszanie
     prostopadloscianu       "
  60 PRINT AT 10,3;"Nacisniecie"
  70 PRINT AT 12,1;  "P lub p -
powoduje powiekszenie"
  80 PRINT AT 14,1;  "Z lub z -
powoduje zmniejszenie"
  90 PRINT FLASH 1;AT 19,4;
     "nacisnij dowolny klawisz"
 100 PAUSE 0
 110 REM
 120 CLS
 130 LET dlbloku=25
 140 LET wskrys=0
 150 GO SUB rysowanie
 155 REM

 160 REM wczytanie znaku
 170 LET DELTA=(CODE INKEY$=80)
              +(CODE INKEY$=112)
              -(CODE INKEY$=90)
              -(CODE INKEY$=122)
 180 LET DELTA=DELTA*3
 190 IF dlbloku+DELTA>63 OR
        dlbloku+DELTA<4 THEN
        LET DELTA=0
 200 IF DELTA=0 THEN
        GO TO wczytaniezn
```

„Życie maklera" jest przetłumaczonym programem z Commodore C-64, opublikowanym w „IKS-e" nr 2/86, na komputer Spectrum, Timex lub ELWRO 800 Junior. Tak więc już nie tylko comodorowcy będą mogli dzięki operacjom giełdowym zostać milionerami lub... zbankrutować.

```
  10 REM     PRZETLUMACZYL
             P. WASKIEWICZ
  15 PAPER 6: BORDER 6: INK 2: C
LS : PRINT AT 2,3; FLASH 1;" Z Y
 C I E  M A K L E R A "
  20 PRINT AT 4,4;"JESTES WLASCI
CIELEM KOPALNI.WYKORZYSTUJAC  OP
ERACJE NA GIEL-DZIE SPROBUJ ZARZ
ADZAC  JAK NAJ-DLUZEJ POSIADANYM
 MAJATKIEM."
  25 PRINT AT 10,5; INVERSE 1;"
U W A G A ! ! "
  30 PRINT AT 13,4;"NIE MOZE PRA
COWAC MNIEJ NIZ 10 GORNIKOW W JE
DNEJ KOPALNI."
  40 PRINT AT 15,0;"POZIOM  ZYCI
A  GORNIKOW OKRESLA WYDAJNOSC IC
H PRACY I WIELKOSC  ZATRUDNIENIA
."
  50 PAUSE 700
 100 LET LK=INT (RND*4+5)
 110 LET LG=INT (RND*30+60)
 120 LET SK=INT (RND*50+10)*LG
 130 LET RW=INT (RND*40+80)
 140 LET Z=0: LET PZG=1: LET R=1
: LET ZN=0
 200 PAPER 1: INK 5: BORDER 1: C
S : PRINT AT 2,3; INVERSE 1;" Z
Y C I E  M A K L E R A "
 203 PLOT 0,175: DRAW 255,0: DRA
 0,-35: DRAW -255,0: DRAW 0,35
 205 PRINT AT 4,14;"ROK ";R
 207 PLOT 0,125: DRAW 255,0: DRA
 0,-55: DRAW -255,0: DRAW 0,55
 210 PRINT AT 6,7; INVERSE 1;" O
BECNIE POSIADASZ "
 215 PRINT AT 7,1;"STAN KONTA...
........ ";SK;" $"
 220 PRINT AT 8,1;"LICZBA GORNIK
OW...... ";LG
 225 PRINT AT 9,1;"LICZBA KOPALN
I....... ";LK
 230 PRINT AT 10,1;"ROCZNE WYDOB
YCIE..... ";RW;" TON"
 235 LET Z=Z+RW*LK
 240 PRINT AT 11,1;"ZAPASY......
......... ";Z;" TON"
 245 PRINT AT 12,1;"POZIOM ZYCIA
 GORNIKOW ";PZG
 247 PLOT 0,59: DRAW 255,0: DRAW
 0,-29: DRAW -255,0: DRAW 0,29
 250 PRINT AT 14,12; INVERSE 1;"
 GIELDA "
 255 LET CR=INT (RND*12+7)
 260 LET CK=INT (RND*2000+2000)
 265 LET CZ=INT (RND*40+60)
 270 PRINT AT 15,1;"CENA TONY RU
DY....... ";CR;" $"
 275 PRINT AT 16,1;"CENA KOPALNI
......... ";CK;" $"
 280 PRINT AT 17.1;"CENA ZYWNOSC
 270 PRINT AT 15,1;"CENA TONY RU
DY....... ";CR;" $"
 275 PRINT AT 16,1;"CENA KOPALNI
......... ";CK;" $"
 280 PRINT AT 17,1;"CENA ZYWNOSC
I GORNIKA ";CZ;" $"
 283 PLOT 0,4: DRAW 255,0
 285 PRINT AT 21,8; INVERSE 1;"
KUPNO/SPRZEDAZ "
 288 PRINT AT 19,1; INVERSE 1;"S
YTUACJE:"
 290 GO SUB 1000
 295 INPUT "ILE TON RUDY SPRZEDA
JESZ ? ";SR
 300 IF SR<0 OR SR>Z THEN  GO TO
 295
 305 IF SR<>0 THEN  LET Z=Z-SR:
LET SK=SK+SR*CR: GO SUB 1010: GO
 SUB 1030
 310 GO SUB 1000
 315 INPUT "ILE KOPALNI SPRZEDAJ
ESZ ? ";KS
 320 IF KS<0 OR KS>LK THEN  GO T
O 315
 325 IF KS<>0 THEN  LET LK=LK-KS
: LET SK=SK+KS*CK: GO SUB 1010:
GO SUB 1020
 330 IF LK=0 THEN  GO TO 2000
 335 GO SUB 1000
 337 IF KS<>0 THEN  GO TO 360
 340 INPUT "ILE KOPALNI KUPUJESZ
? ";KK
 345 IF KK<0 OR KK*CK>SK THEN  G
O TO 340
 350 IF KK<>0 THEN  LET LK=LK+KK
: LET SK=SK-KK*CK: GO SUB 1010:
GO SUB 1020
 355 GO SUB 1000
 360 INPUT "ILE $ WYPLACASZ GORN
IKOM ";WG
 365 IF WG<0 OR WG>SK THEN  GO T
O 360
 370 IF WG<>0 THEN  LET SK=SK-WG
: GO SUB 1010
 400 IF WG/LG>CZ+20 THEN  LET PZ
G=PZG+.1: GO SUB 1040
 405 IF WG/LG<CZ-20 THEN  LET PZ
G=PZG-.1: GO SUB 1040
 410 IF PZG<.6 THEN  GO TO 2010
 415 IF PZG>1.1 THEN  LET RW=RW+
INT (RND*20+5)
 420 IF PZG>=1 THEN  GO TO 435
 425 IF RW>25 THEN  LET RW=RW-IN
T (RND*20+5): IF RW<0 THEN  LET
RW=0
 430 GO SUB 1050
 435 IF LG/LK<10 THEN  GO TO 202
0
 440 IF PZG>1.1 THEN  LET LG=LG+
 440>IF PZG>1.1THEN L
ET LG=LG+INT (RND*10+1):GO SUB 1
060
 445 IF PZG<.9 THEN  LET LG=LG-I
NT (RND*10+1): GO SUB 1060
 450 IF RND*1>.2 THEN  GO TO 465
 455 PRINT AT 19,10; FLASH 1;"RA
DIOAKTYWNE MINERALY....WIELU ZGI
NELO"
 460 LET LG=INT (LG/2): GO SUB 1
060: LET ZN=1
 465 IF RW<150 THEN  GO TO 500
 470 IF ZN=1 THEN  LET ZN=0
 475 PRINT AT 19,10; FLASH 1;"KR
YZYS   NADPRODUKCJI ....WYDOBYCI
E SPADA"
 480 LET RW=INT (RW/2): GO SUB 1
050
 500 LET R=R+1
 505 IF R=100 THEN  GO TO 3000
 510 PAUSE 200
 520 GO TO 200
1000 PAUSE 100
1001 RETURN
1010 PAUSE 50: BEEP .2,20: PRINT
 AT 7,23; INVERSE 1;SK;" $ ": PA
USE 20: PRINT AT 7,23; INVERSE 0
;SK;" $ ": RETURN
1020 PAUSE 50: BEEP .2,20: PRINT
 AT 9,23; INVERSE 1;LK;" ": PAUS
E 20: PRINT AT 9,23; INVERSE 0;L
K;" ": RETURN
1030 PAUSE 50: BEEP .2,20: PRINT
 AT 11,23; INVERSE 1;Z;" TON ":
PAUSE 20: PRINT AT 11,23; INVERS
E 0;Z;" TON ": RETURN
1040 PAUSE 50: BEEP .2,20: PRINT
 AT 12,23; INVERSE 1;PZG;"   ": P
AUSE 20: PRINT AT 12,23; INVERSE
 0;PZG;"   ": RETURN
1050 PAUSE 50: BEEP .2,20: PRINT
 AT 10,23; INVERSE 1;RW;"TON ":
PAUSE 20: PRINT AT 10,23; INVERS
E 0;RW;" TON ": RETURN
1060 PAUSE 50: BEEP .2,20: PRINT
 AT 8,23; INVERSE 1;LG;"   ": PAU
SE 20: PRINT AT 8,23; INVERSE 0;
LG;"   ": RETURN
2000 PRINT AT 19,10; FLASH 1;"NI
E MASZ  KOPALNI.....KONIEC GRY":
 GO TO 3000
2010 PRINT AT 19,10; FLASH 1;"ST
RAJKI  GORNIKOW.....KONIEC GRY":
 GO TO 3000
2020 PRINT AT 19,10; FLASH 1;"ZA
  MALO GOR-NIKOW....KONIEC GRY":
 GO TO 3000
3000 PAUSE 300: INK 2: PAPER 6:
BORDER 6: CLS
3010 PRINT AT 9,10; FLASH 1;" KO
NIEC GRY!! "
3020 PRINT AT 11,1;"BYLES MAKLER
EM ";R;" LAT. "
3025 PRINT AT 12,0;"SKONCZYLES Z
 KASA ";SK;" $"
3030 PRINT AT 13,0;"CHCESZ GRAC
DALEJ ? (T/N) "
3040 INPUT R$
3050 IF R$="N" OR R$="n" THEN  S
TOP
3060 IF R$="T" OR R$="t" THEN  R
UN 100
3070 GO TO 3040
```













KAREN — to właśnie tu można naprawić każdy typ Atari. Tu trafiają uszkodzone komputery z całej Polski, bowiem właśnie KAREN prowadzi naprawy gwarancyjne tych mikrokomputerów (ale najczęściej psują się wrażliwe magnetofony). Ludzie z tej firmy dbają jednak nie tylko o zepsuty sprzęt, ale zabiegają, aby był przede wszystkim dobrze eksploatowany.

To jeden z nowszych modeli Atari. Podstawą tej konstrukcji jest mikroprocesor Motorola 68 000. Pamięć RAM ma 1024 Kb, ROM 192 Kb. Wśród wielu zalet Atari wymienia się znakomitą grafikę. Atari 1040 STF dysponuje 512 kolorami. Duża rozdzielczość to 640×400 punktów na monochromatycznym monitorze; średnia rozdzielczość to punktów w czterech kolorach.

INFORMATYKA
KOMPUTERY
SYSTEMY

zeszyty programów komputerowych nr 1/88

Dodatek „Żołnierza Wolności" Cena 200 zł



SPIS TREŚCI

Wprowadzenie — str. 2

Podstawowe instrukcje i typy danych — str. 2

Złożone typy danych — str. 10

Procedury i funkcje — str. 15

Elementy grafiki komputerowej — str. 19

Dodatek (Jak uruchomić program?) — str. 23

Programy: I „Gra Hamiltona" — str. 25

„Wieża Hanoi" — str. 29

Opisy gier komputerowych:
Vegas Jackpot — str. 30
Alley Cat — str. 31

UWAGA! Następny zeszyt zawiera opis języka C — rewelacyjne narzędzie w rękach użytkowników mikrokomputerów.

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Pieła, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego.

U-40

Nr zam. 1117. Nr ind. 361682.

Wprowadzenie

Niniejsze opracowanie przeznaczone jest dla użytkowników mikrokomputerów, którzy pragną poszerzyć swoją wiedzę w zakresie samodzielnego programowania w języku PASCAL.

Wprowadzane wiadomości teoretyczne autorzy ilustrują prostymi przykładami zadań, które zawierają rozwiązania praktyczne. Przykłady te decydują o właściwym opanowaniu przedstawionych zagadnień i dostosowane są do możliwości percepcyjnych uczniów szkół średnich oraz wyższych klas szkół podstawowych.

Opracowanie składa się z 4 rozdziałów:

1. Podstawowe instrukcje i typy danych
Rozdział obejmuje przykłady rozwiązań prostych zadań obliczeniowych, uwzględniających przede wszystkim materiał możliwy do opanowania przez uczniów szkół podstawowych.

2. Złożone typy danych
Zawiera przykłady zapisu zbiorów danych z uwzględnieniem możliwości ich przychowywania w pamięci zewnętrznej.

3. Procedury i funkcje
Jest to wstęp do programowania zaawansowanego, w którym wykorzystuje się logiczne wyodrębnione fragmenty programów.

Elementy grafiki komputerowej
Obejmuje przykłady wyświetlania tekstów i obrazów.

Wszystkie zaprezentowane programy zostały uruchomione w systemie TURBO PASCAL na mikrokomputerze AMSTRAD CPC 61128, pracującym pod nadzorem CP/M PLUS. Podstawowe zasady pracy w systemie TURBO PASCAL zawiera dodatek. Przedstawione programy, poza tymi, które obejmują elementy grafiki, mogą być uruchamiane także na mikrokomputerach serii IBM PC. Systematyczne ujęcie materiału powinno sprzyjać opanowaniu podstawowych właściwości języka PASCAL bez względu na komputer, z którym jest on związany.

PODSTAWOWE INSTRUKCJE I TYPY DANYCH

1

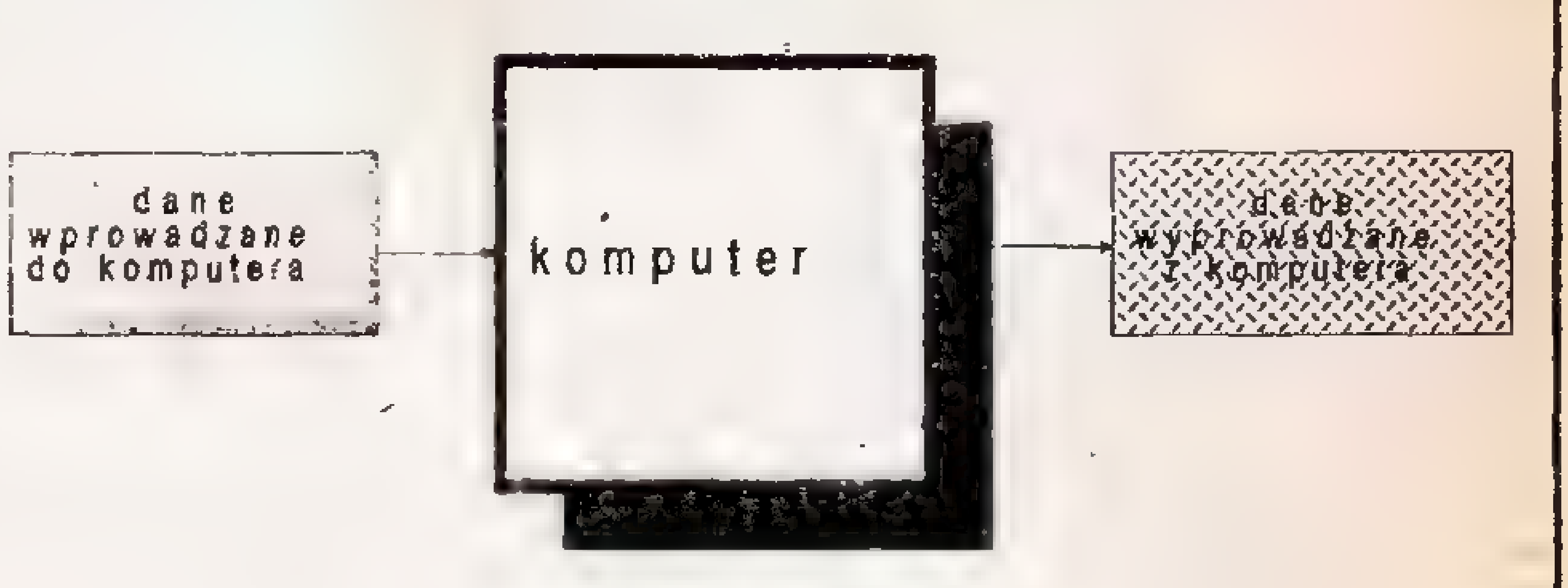


Komputer jest narzędziem przystosowanym do przetwarzania informacji. Informacje umieszczone na odpowiednich nośnikach zwane są danymi. Stąd też często mówi się o tym, że komputer umożliwia przetwarzanie danych. Proces użytkowania komputera przedstawia rysunek 1.

Dane mogą być wprowadzane do komputera jako tekst w języku angielskim, natomiast wyprowadzane z komputera jako ten sam tekst przetłumaczony na język polski.

W tym przypadku przetwarzanie danych przez komputer polega na tłumaczeniu tekstu. Ten sam komputer może być wykorzystywany w innych celach np. do obliczania pola kwadratu na podstawie wyprowadzonej długości boku. W każdym z obydwu przypadków komputer realizuje inne czynności. Sposób postępowania komputera określa przepis zawarty w tzw. programie. Program ten musi być wprowadzony do komputera przed rozpoczęciem wymaganych czynności. Takim programem może być dowolna gra komputerowa znajdująca się na kasecie magnetofonowej lub dyskietce. Użytkownik może również sporządzić program samodzielnie, wykorzystując w tym celu specjalne narzędzia zwane językami programowania.

PASCAL jest jednym z wielu języków programowania charakteryzującym się łatwością zapisu programów obejmujących liczną klasę problemów. Najprostszy program zapisany w języku PASCAL wyprowadzający tekst:



RYS.1. Proces użytkowania komputera

rozpoczynamy naukę języka PASCAL na monitor zamieszczono w przykładzie 1.

Przykład 1

```
PROGRAM prog1;
 BEGIN
  WRITE('rozpoczynamy naukę języka PASCAL');
 END.
```

Program posiada strukturę przedstawioną na rysunku 2.

Wszystkie wyróżnione słowa (zapisane wielkimi literami) zwane są **słowami kluczowymi.** Każde z nich ma ściśle określone znaczenie.

Podstawowym elementem programu o nazwie **prog1** jest instrukcja **WRITE** powodująca wyprowadzenie na monitor tekstu ujętego w apostrofy. Program ten opisuje szczególny przypadek przetwarzania danych, nie wymaga bowiem wyprowadzania żadnych danych wejściowych.

Najprostszy program uwzględniający wprowadzanie i wyprowadzanie danych w niezmienionej postaci, musi zawierać co najmniej dwie instrukcje, z których jedna wprowadza dane, a druga powoduje ich wyprowadzenie. Pomiędzy wprowadzeniem a wyprowadzeniem dane muszą być gdzieś przechowywane. Struktura programu powinna zatem uwzględniać istnienie obiektów umożliwiających przechowanie danych. Obiekty te zwane są zmiennymi. Zasadę konstruowania takiego programu przedstawia rysunek 3, a sam program został zamieszczony w przykładzie 2.

Przykład 2

```
PROGRAM prog2;
VAR
  tekst:string[5];
 BEGIN
  READ(tekst);
   WRITE(tekst);
 END.
```

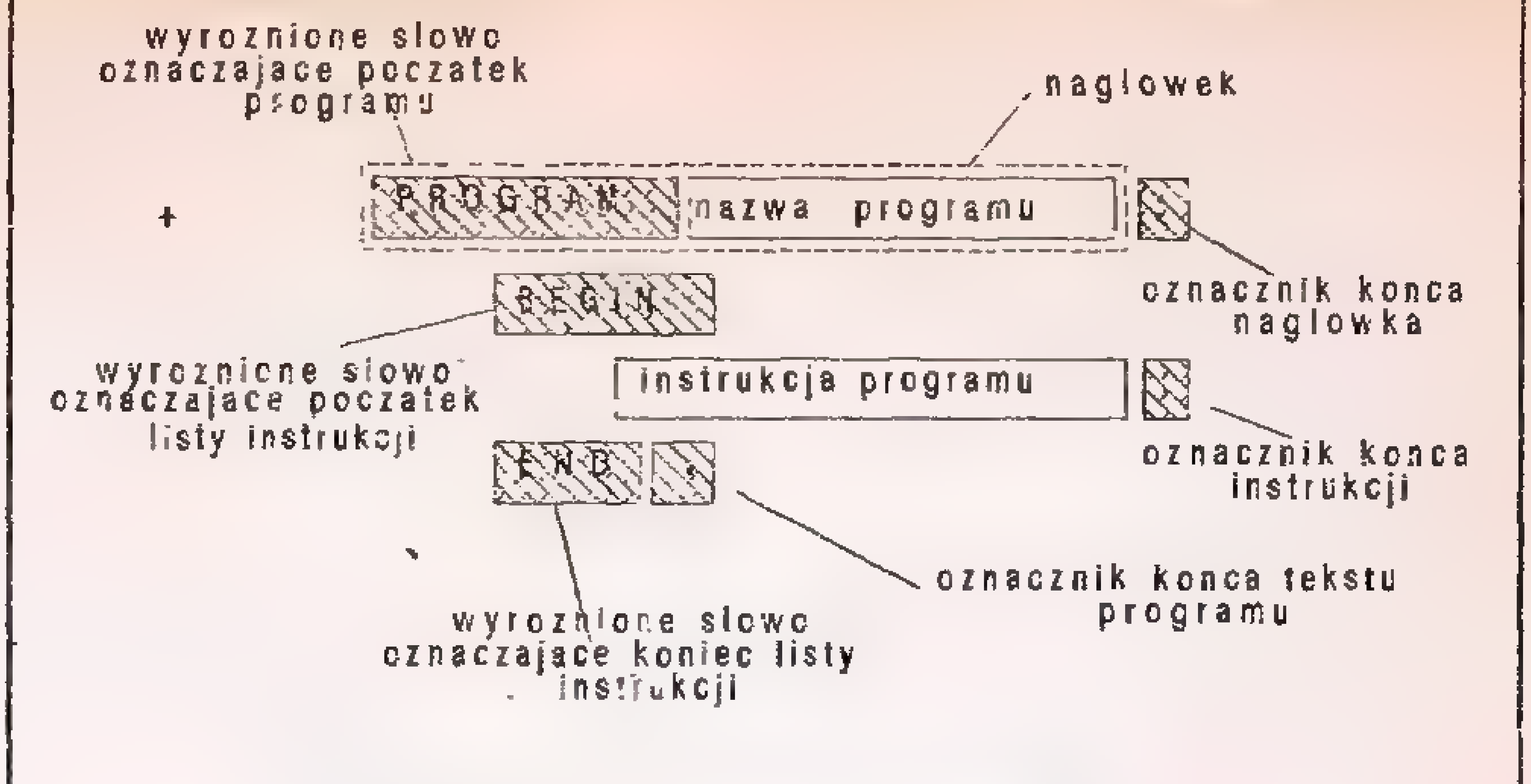
Strukturę programu przedstawiono na rys. 4.

W programie występuje deklaracja zmiennej, tj. zapis określający:

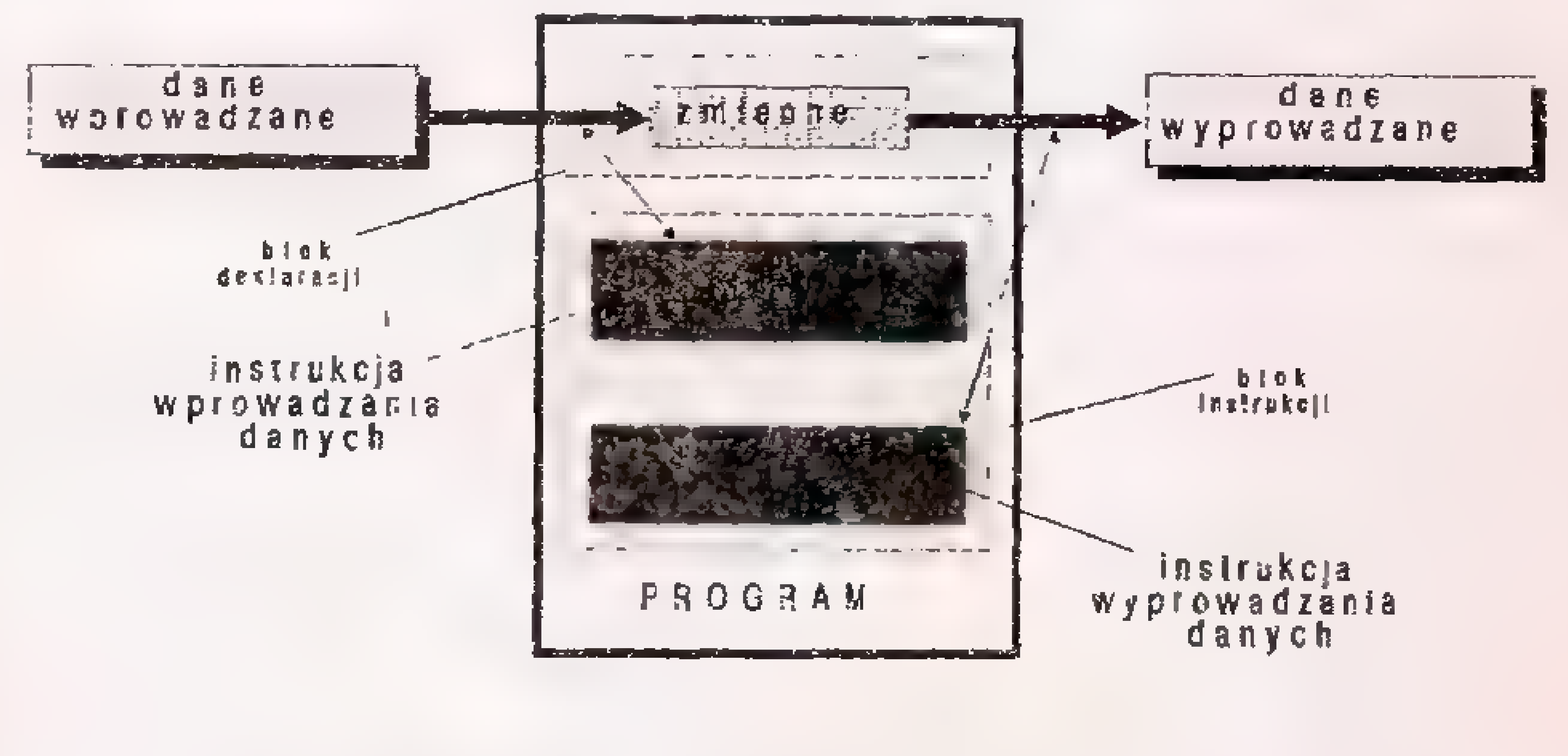
— nazwę zmiennej (*tekst*),

— typ zmiennej (*string [5]*) identyfikujący zakres dopuszczalnych wartości, jakie może przyjmować zmienna.

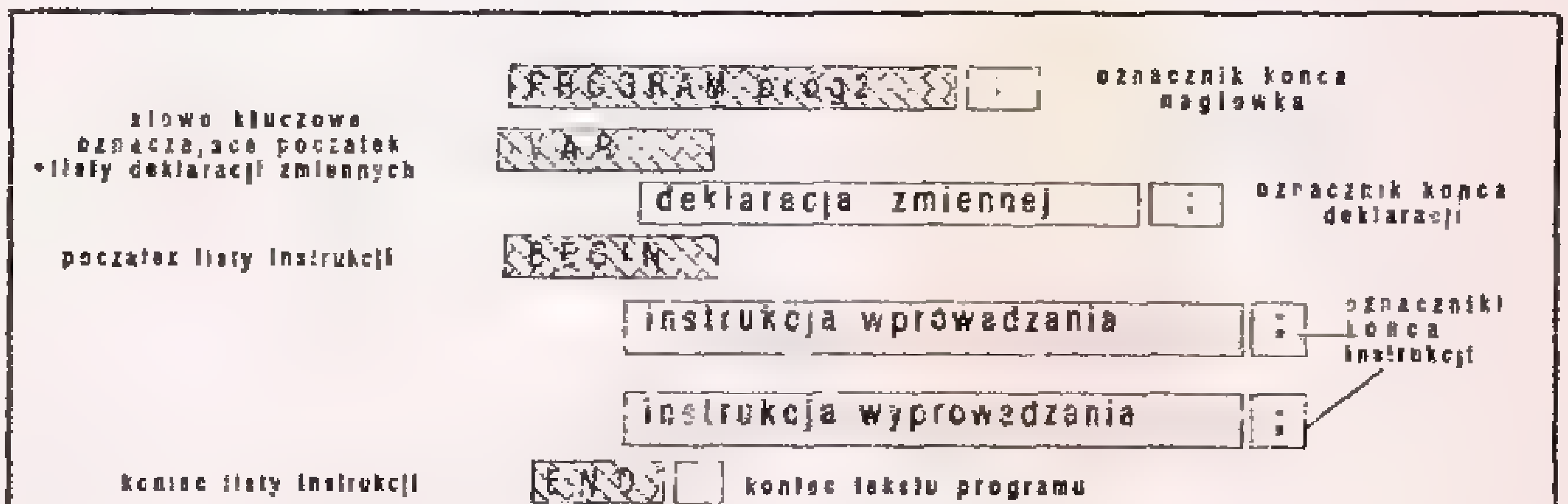
W programie **prog2** wartością zmiennej *tekst* może być ciąg dowolnych znaków o długości nie większej niż 5. Przykładowo ciągi *abcde*, *xy*, 1234 są poprawnymi wartościami zmiennej *tekst* w przeciwieństwie do ciągu 123456, który jest zbyt długi. W liście instrukcji program prog. 2 pierwszą instrukcją jest **READ** z parametrem *tekst*. W wyniku jej wykonania następuje wczytanie ciągu znaków (o długości nie przekraczającej 5) do zmiennej *tekst*. Wyprowadzenie wczytanego tekstu realizuje znana już instrukcja WRITE, której parametrem jest w tym przypadku zmienna *tekst*.



RYS.2. Struktura programu z przykladu 1



RYS.3. Ilustracja zasady konstrukcji programu wprowadzania i wyprowadzania danych



Rys.4. Struktura programu z przykladu 2

Program **prog2** wykonany dla ciągu *abcd* (zakończenie wprowadzania ciągu następuje po wciśnięciu klawisza **ENTER**) powoduje wyświetlenie tekstu *abcd*. Jest on wyświetlany bezpośrednio po ciągu wejściowym. W efekcie na ekranie monitora pozostaje ciąg *abcdabcd*. Chcąc oddzielić te ciągi możemy wyświetlić ciąg wyprowadzany w nowej linii. Efekt ten uzyskamy wstawiając instrukcję **WRITELN** pomiędzy instrukcje READ i WRITE (przykład 3).

Przykład 3

```
PROGRAM prog3;
 VAR
  tekst:string[5];
 BEGIN
  READ(tekst);
  WRITELN;
  WRITE(tekst);
 END.
```

Podane przykłady dotyczyły wprowadzania i wyprowadzania łańcuchów znaków, PASCAL umożliwia również wprowadzanie danych traktowanych jako liczby. Program zamieszczony w przykładzie 4 pozwala na wprowadzenie dwóch liczb rzeczywistych (liczby dziesiętne, w których część ułamkową — jeśli występuje — oddziela się od części całkowitej kropką zamiast przecinka) i ich wyprowadzenie.

W programie wykorzystano instrukcje z przykładu 3. Programy różnią się:

— deklaracjami zmiennych — zmienne x i y są zadeklarowane jako liczby rzeczywiste **(real),**

— postacią instrukcji WRITE, w której oprócz nazwy wyprowadzanej zmiennej podano sumaryczną ilość znaków przeznaczonych na zapis liczby *(6)* oraz ilość znaków przeznaczonych na część ułamkową *(2).*

W uzupełnieniu należy dodać, że sumaryczna ilość znaków liczby obejmuje kropkę dziesiętną oraz znak liczby (wyprowadzany tylko w przypadku liczb ujemnych).

Przykład 4

```
PROGRAM prog4;
 VAR
  x:real;
  y:real;
 BEGIN
  READ(x);
  READ(y);
  WRITELN;
  WRITE('x=');
  WRITE(x:6:2);
  WRITELN;
  WRITE('y=');
  WRITE(y:6:2);
 END.
```

Program **prog4** wykonano dla liczb x = 210,85 i y = —20 uzyskując wyniki:

x = 210.85
y = 20.00

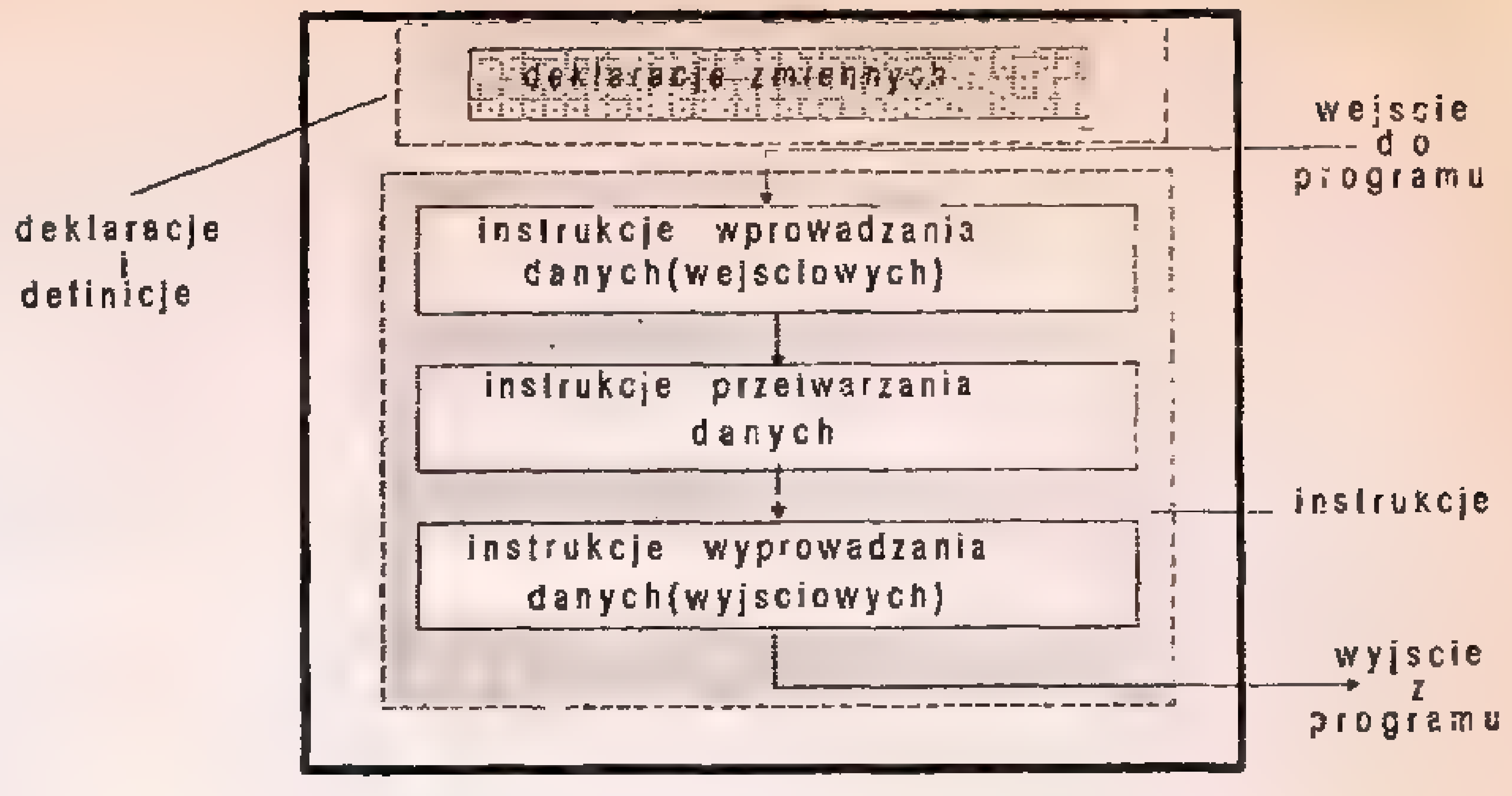
Program **prog4** pomimo iż realizuje bardzo proste funkcje, jest stosunkowo obszerny. W celu skrócenia zapisu można wykorzystać następujące właściwości języka:

— dla zmiennych jednakowego typu, nazwa typu może być podana po wyliczeniu nazw tych zmiennych,

— w ramach jednej instrukcji READ można wprowadzić więcej niż jedną daną, przy czym nie muszą być one tego samego typu,

— w ramach jednej instrukcji WRITE można wyprowadzić więcej niż jedną daną (bez względu na jej typ).

Właściwości te wykorzystano w programie **prog5** (przykład 5) realizującym te same funkcje co program **prog4**.



Rys.5.Struktura prostego programu

Przykład 5

```
PROGRAM prog5;
 VAR
  x,y:real;
 BEGIN
  READ(x,y);
  WRITELN;
  WRITELN('x=',x:6:2);
  WRITE('y=',y:6:2);
 END.
```

W dotychczas podanych przykładach działanie programów ograniczało się do wprowadzania i wyprowadzania danych. Podstawowym zadaniem komputera jest jednak przetwarzanie danych realizowanych za pomocą odpowiednich instrukcji języka PASCAL.

Typową strukturę programu uwzględniającą wprowadzenie, przetwarzanie i wyprowadzanie danych przedstawiono na rysunku 5.

Podstawową instrukcją przetwarzania danych jest instrukcja przypisania przyjmująca formę:

nazwa zmiennej *wyrażenie*

:=

identyfikator instrukcji

Instrukcja ta powoduje przypisanie zmiennej, występującej po lewej stronie identyfikatora instrukcji, nowej wartości, będącej wynikiem obliczenia wartości wyrażenia występującego po prawej stronie znaku identyfikatora instrukcji.

W odniesieniu do zmiennych rzeczywistych wyrażenia w języku PASCAL konstruowane są tak jak wyrażenia arytmetyczne.

Różnice dotyczą wyłącznie postaci niektórych operatorów oraz zapisu ułamków. Sposób wyliczania wartości wyrażenia określony jest wagą operatorów:

— wyrażenia w nawiasach obliczane są w pierwszej kolejności,

— mnożenie (* zamiast .) oraz dzielenie (/ zamiast :) wykonywane są w kolejności ich występowania przed wykonaniem dodawania i odejmowania,

— dodawanie (+) i odejmowanie (—) wykonywane są w kolejności ich występowania.

Ułamki traktowane są jako operacje dzielenia licznika przez mianownik.

Proste programy zamieszczone w przykładach 6—11 obejmują kolejno obliczenie:

— sumy, różnicy, iloczynu i ilorazu dwóch liczb rzeczywistych,

— wartości sinusa, cosinusa i tangensa jednego z kątów trójkąta prostokątnego o podanych bokach,

— pola prostokąta o podanych bokach,

— pola trapezu o podanych podstawach i wysokości,

— objętości kuli o podanym promieniu,

— objętości stożka o podanym promieniu podstawy i zadanej wysokości.

Przykład 6

```
PROGRAM prog6;
 VAR
  x,y,suma,roznica,
  iloczyn,iloraz:real;
 BEGIN
  READ(x,y);
  WRITELN;
  WRITELN('pierwsza liczba =',x:8:4);
  WRITELN('druga liczba =',y:8:4);
  suma:=x+y;
  roznica:=x-y;
  iloczyn:=x*y;
  iloraz:=x/y;
  WRITELN('suma =',suma:8:4);
  WRITELN('roznica =',roznica:8:4);
  WRITELN('iloczyn =',iloczyn:8:4);
  WRITE('iloraz =',iloraz:8:4);
 END.
```

Przykład 7

```
PROGRAM prog7;
 VAR
  a,b,c,sinus,cosinus,tangens:real;
 BEGIN
  READ(a,b,c);
  WRITELN;
  WRITELN('wczytane boki trojkata',
          ' prostokatnego:');
  WRITELN('a =',a:3:0,', b =',b:3:0,
          ', c =',c:3:0);
```

```
  sinus:=b/c;
  cosinus:=a/c;
  tangens:=b/a;
  WRITELN;
  WRITELN('sinus kata alfa = ',sinus:6:4);
  WRITELN('cosinus kata alfa = ',cosinus:6:4);
  WRITE('tangens kata alfa = ',tangens:6:4);
 END.
```

Przykład 8

```
PROGRAM prog8;
 VAR
  a,b,pole:real;
 BEGIN
  WRITELN;
  WRITELN('OBLICZENIE POLA PROSTOKATA:');
  WRITELN('podaj dlugosci bokow');
  READ(a,b);
  pole:=a*b;
  WRITELN;
  WRITELN('pole prostokata o bokach');
  WRITELN(' a =',a:6:3,', b =',b:6:3);
  WRITE('wynosi ',pole:8:3);
 END.
```

Przykład 9

```
PROGRAM prog9;
 VAR
  a,b,h,pole:real;
 BEGIN
  WRITELN;
  WRITELN('OBLICZENIE POLA TRAPEZU:');
  WRITELN('podaj dlugosci:');
  WRITELN('   -dolnej podstawy,');
  WRITELN('   -gornej podstawy,');
  WRITELN('   -wysokosc.');
  READ(a,b,h);
  pole:=0.5*(a+b)*h;
  WRITELN;
  WRITELN('pole trapezu o podstawach');
  WRITELN(' a =',a:6:3,', b =',b:6:3);
  WRITELN('i wysokosci h =',h:6:3);
  WRITE('wynosi ',pole:8:3);
 END.
```

Przykład 10

```
PROGRAM prog10;
 VAR
  r,objetosc:real;
 BEGIN
  WRITELN;
  WRITELN('OBLICZENIE OBJETOSCI KULI:');
  WRITELN('podaj promien kuli');
  READ(r);
  objetosc:=4/3*pi*r*r*r;
       {pi - stala dostepna w PASCAL-u}
  WRITELN;
  WRITELN('objetosc kuli o promieniu');
  WRITELN(' r =',r:6:3);
  WRITE('wynosi ',objetosc:10:3);
 END.
```

Przykład 11

```
PROGRAM prog11;
 VAR
  r,h,objetosc:real;
 BEGIN
  WRITELN;
  WRITELN('OBLICZENIE OBJETOSCI STOZKA:');
  WRITELN('podaj promien podstawy');
  WRITELN('i wysokosc stozka');
  READ(r,h);
  objetosc:=pi*r*r/3;
  WRITELN;
  WRITELN('objetosc stozka o promieniu');
  WRITELN('podstawy r =',r:6:3);
  WRITELN('i wysokosci h =',h:6:3);
  WRITE('wynosi ',objetosc:8:3);
 END.
```

Dotychczas poznane instrukcje umożliwiają już wykonanie prostych obliczeń na wprowadzonych danych oraz prezentację wyników. Na ogół jednak to nie wystarcza, ponieważ wykonanie takich działań często zależy do spełnienia określonych warunków. Na przykład wyznaczenie wartości bezwzględnej liczby zgodnie z definicją:

$$|x| = \begin{cases} x \text{ dla } x \geq 0 \\ -x \text{ dla } x < 0 \end{cases}$$

wymaga porównania wartości liczby z zerem.

Podstawową instrukcją umożliwiającą wykonanie działań uzależnionych od spełnienia określonych warunków jest instrukcja alternatywy przyjmująca postać:

IF *warunek* **THEN**

instrukcja 1

ELSE

instrukcja 2

Warunek jest wyrażeniem logicznym tzn. takim, które w danej sytuacji (dla określonych wartości zmiennych w nim występujących) może być spełnione lub nie. Jeżeli warunek jest spełniony to następuje wykonanie *instrukcji 1* (a *instrukcja 2* nie jest w ogóle wykonywana). W przypadku przeciwnym wykonywana jest *instrukcja 2* (*instrukcja 1* nie jest wtedy wykonywana).

Program obliczania wartości bezwzględnej zadanej liczby zamieszczono w przykładzie 12.

Przykład 12

```
PROGRAM prog12;
 VAR
  x:real;
 BEGIN
  READ(x);
  WRITELN;
  WRITELN('wartosc bezwzgledna liczby');
  WRITELN('        ',x:8:4);
  IF x>=0 THEN
   x:=x
  ELSE
   x:=-x;
  WRITE('wynosi',x:8:4);
 END.
```

Występujący w programie warunek

$$x >= 0$$

jest spełniony tylko wtedy, gdy x ma wartość nieujemną. Taką interpretację warunku wskazuje operator relacyjny >= (większy lub równy). W warunkach mogą występować ponadto następujące operatory relacyjne: < (mniejszy), <= (mniejszy lub równy), = (równy), > (większy), <> (różny).

Wyrażenia występujące w przykładzie 12 zawierają nazwy tych samych zmiennych po obu stronach operatora np:

$$x: = -x$$

W takim przypadku przy obliczaniu wartości wyrażenia brana jest pod uwagę wartość zmiennej x przed wykonaniem instrukcji przypisania. Wyznaczona wartość wyrażenia jest następnie przypisywana zmiennej x jako nowa wartość (poprzednia wartość zostaje zniszczona).

Ze składni instrukcji alternatywy wynika, że zarówno w przypadku spełnienia, jak i niespełnienia warunku wykonywana jest pojedyncza instrukcja. W języku PASCAL za pojedynczą instrukcję przyjmuje się także instrukcję złożoną — sekwencję instrukcji ograniczoną słowami kluczowymi **BEGIN—END:**

BEGIN
instrukcja 1;
instrukcja 2;
instrukcja 3;

END

W przykładzie 13 zamieszczono zmodyfikowaną wersję programu z przykładu 7, w którym wyznaczanie sinusa, cosinusa i tangensa (zapisane w postaci instrukcji złożonej) jest wykonywane tylko wtedy, gdy trójkąt o podanych bokach jest trójkątem prostokątnym.

Przykład 13

```
PROGRAM prog13;
 VAR
  a,b,c,sinus,cosinus,tangens:real;
 BEGIN
  READ(a,b,c);
  WRITELN;
  WRITELN('wczytane boki trojkata');
  WRITELN('a =',a:3:0,', b =',b:3:0,
        ', c =',c:3:0);
  IF (a*a+b*b)=(c*c) THEN
   BEGIN
    sinus:=b/c;
    cosinus:=a/c;
    tangens:=b/a;
    WRITELN;
    WRITELN('sinus kata alfa = ',
            sinus:6:4);
    WRITELN('cosinus kata alfa = ',
            cosinus:6:4);
    WRITE('tangens kata alfa = ',
            tangens:6:4);
   END
  ELSE
   WRITE('nie tworza trojkata ',
          'prostokatnego');
 END.
```

Bardziej złożony program, ilustrujący wykorzystanie instrukcji alternatywy, zamieszczono w przykładzie 14. Program ten umożliwia obliczanie pierwiastków rzeczywistych trójmianu kwadratowego. W poprzednich przykładach programista określał postać wyprowadzanych liczb. W przypadku, gdy programiście nie zależy na narzuceniu formatu wyprowadzanych liczb może wykorzystać postać standardową, która dla liczb rzeczywistych jest następująca:

uu#.#########E☆## *dla liczb większych lub równych zero,*

u — #.#########E☆## *dla liczb mniejszych od zera.*

gdzie:

u — *reprezentuje spację,*
\# — *reprezentuje cyfrę dziesiętną,*
☆ — *reprezentuje znak* + lub —

Wartość tak wprowadzonej liczby należy odczytać jako c. 10^m

gdzie:

c — *liczba znajdująca się przed literą E* (cecha),
m — *liczba znajdująca się za literą E* (mantysa).

Wartość zmiennej wyprowadzona jest w formacie standardowym, gdy w instrukcji WRITE występuje wyłącznie nazwa zmiennej.

Przykład 14

```
PROGRAM prog14;
 VAR
  a,b,c,
  x1,x2,
  delta:real;
 BEGIN
  WRITELN;
  WRITELN('podaj wspolczynniki trojmianu');
  READ(a,b,c);
  IF a=0 THEN
   WRITELN(' niedopuszczalna wartosc ',
           'wspolczynnika a')
  ELSE
   BEGIN
    WRITELN;
    WRITELN('dla wspolczynnikow: a =',a);
    WRITELN('                    b =',b);
    WRITELN('                    c =',c);
    delta:=b*b-4*a*c;
    IF delta<0 THEN
     WRITELN('brak pierwiastkow rzeczywi',
             'stych')
    ELSE
     IF delta=0 THEN
      BEGIN
       x1:=-b/(2*a);
       WRITELN('trojmian ma jeden pierwi',
               'astek');
       WRITE('podwojny x =',x1);
      END
     ELSE
      BEGIN
       x1:=(-b-SQRT(delta))/(2*a);
       x2:=(-b+SQRT(delta))/(2*a);
       WRITELN('trojmian ma dwa pierwia',
               'stki: x1 =',x1);
       WRITE(' ':29,'x2 =',x2);
      END;
   END;
 END.
```

Należy dodać, że wszystkie wprowadzane liczby rzeczywiste mogą być podane w postaci liczb dziesiętnych z kropką (np. 10.35) lub w postaci standardowej (np. 10.35E+0, 1.035E+1, 0.11035E+2, 1035E—2).

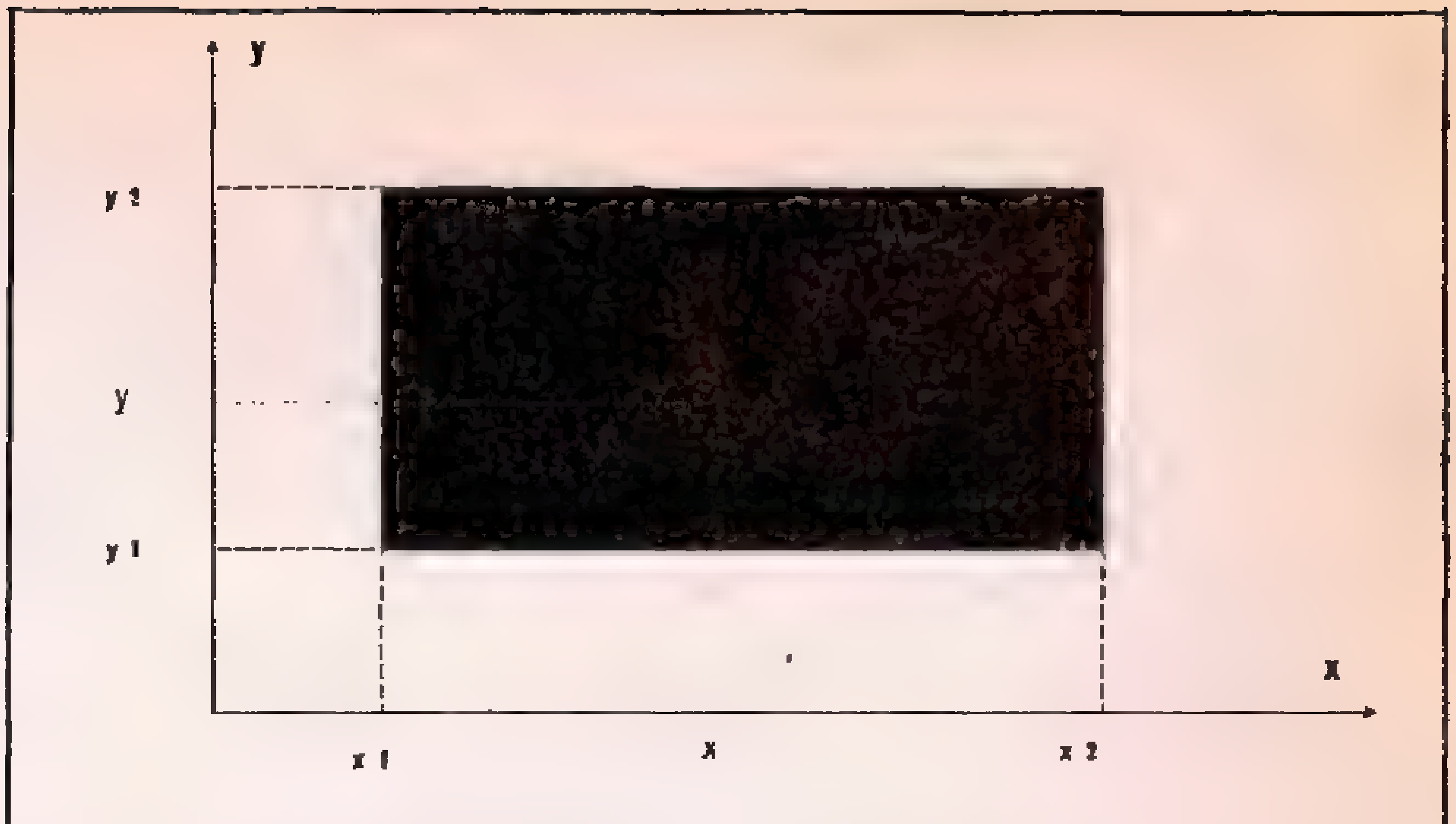
Instrukcja alternatywy zakłada wykonanie różnych akcji w przypadku spełnienia i niespełnienia warunku. Niejednokrotnie jednak konstrukcja programu wymaga wykonania jakiejś akcji tylko w przypadku spełnienia warunku. Wygodniej jest wtedy wykorzystać uproszczoną postać instrukcji alternatywy, zwaną instrukcją warunkową:

IF *warunek* **THEN** *instrukcja 1*

Instrukcja 1 wykonywana jest tylko w przypadku spełnienia warunku. Przykłady wykorzystania instrukcji warunkowej zamieszczono w programach **prog15** i **prog16** (przykłady 15 i 16).

Przykład 15

```
PROGRAM prog15;
 VAR
  x:real;
 BEGIN
  READ(x);
  WRITELN;
  WRITELN('wartosc bezwzgledna liczby');
  WRITELN('          ',x:8:4);
  IF  x<0 THEN
    x:=-x;
  WRITE('wynosi',x:8:4);
 END.
```



Rys.6.Ilustracja do przykladu 17

Przykład 16

```
PROGRAM prog16:
 VAR
  a,b,h,r,pole,objetosc:real;
  nrw:integer;
 BEGIN
  WRITELN;
  WRITELN('PROGRAM UMOZLIWIA OBLICZENIE:');
  WRITELN(' 1 - POLA PROSTOKATA,');
  WRITELN(' 2 - POLA TRAPEZU,');
  WRITELN(' 3 - OBJETOSCI KULI,');
  WRITELN(' 4 - OBJETOSCI STOZKA,');
  WRITELN('podaj numer wariantu');
  WRITELN;
  READLN(nrw);
  IF nrw=1 THEN
   BEGIN
    WRITELN;
    WRITELN('OBLICZENIE POLA PROSTOKATA:');
    WRITELN('podaj dlugosci bokow');
    READ(a,b);
    pole:=a*b;
    WRITELN;
    WRITELN('pole prostokata o bokach');
    WRITELN(' a =',a:6:3,', b =',b:6:3);
    WRITE('wynosi ',pole:8:3);
   END;
  IF nrw=2 THEN
   BEGIN
    WRITELN;
    WRITELN('OBLICZENIE POLA TRAPEZU:');
    WRITELN('podaj dlugosci:');
    WRITELN('   -dolnej podstawy,');
    WRITELN('   -gornej podstawy,');
    WRITELN('   -wysokosc:');
    READ(a,b,h);
    pole:=0.5*(a+b)*h;
    WRITELN;
    WRITELN('pole trapezu o podstawach');
    WRITELN(' a =',a:6:3,', b =',b:6:3);
    WRITELN('i wysokosci h =',h:6:3);
    WRITE('wynosi ',pole:8:3);
   END;
  IF nrw=3 THEN
   BEGIN
    WRITELN;
    WRITELN('OBLICZENIE OBJETOSCI KULI:');
    WRITELN('podaj promien kuli');
    READ(r);
    objetosc:=4/3*pi*r*r*r;
    WRITELN;
    WRITELN('objetosc kuli o promieniu');
    WRITELN(' r =',r:6:3);
    WRITE('wynosi ',objetosc:10:3);
   END;
  IF nrw=4 THEN
   BEGIN
    WRITELN;
    WRITELN('OBLICZENIE OBJETOSCI STOZKA:');
    WRITELN('podaj promien podstawy');
    WRITELN('i wysokosc stozka');
    READ(r,h);
    objetosc:=pi*r*r/3;
    WRITELN;
    WRITELN('objetosc stozka o promieniu');
    WRITELN('podstawy r =',r:6:3);
    WRITELN('i wysokosci h =',h:6:3);
    WRITE('wynosi ',objetosc:8:3);
   END;
 END.
```

W przykładzie 15 wystąpiła potrzeba zmiany operatora relacyjnego, zastosowanego w programie z przykładu 12. Operator >= zastąpiono operatorem < w celu umieszczenia instrukcji

x : = —x

w odpowiednim miejscu. Ten sam efekt można uzyskać stosując jeden z operatorów logicznych tj. operator negacji **NOT.** Instrukcja warunkowa w programie z przykładu 15 przyjęłaby wówczas postać:

IF NOT *x* = 0 **THEN** x: = —x

Oprócz operatora NOT w PASCAL-u dostępne są ponadto operatory logiczne **OR** (logiczne *lub*) oraz **AND** (logiczne *i*) umożliwiające składanie warunków złożonych z warunków prostszych. Sposób wykorzystania operatora logicznego AND przedstawiono w przykładzie 17 zawierającym program sprawdzania czy punkt o zadanych współrzędnych leży w polu prostokąta określonego jak na rysunku 6.

Przykład 17

```
PROGRAM prog17;
 VAR
  x1,y1,x2,y2,
  x,y:real;
 BEGIN
  WRITELN;
  WRITELN('PODAJ WSPOLRZEDNE PUNKTOW:');
  WRITELN(' (x1,y1), (x2,y2), (x,y)');
  READ(x1,y1,x2,y2,x,y);
  WRITELN;
  WRITELN('PUNKT O WSPOLRZEDNYCH');
  WRITELN('x=',x,' y=',y);
  IF (x>=x1) AND (x<=x2) AND (y>=y1)
     AND (y<=y2) THEN
   WRITE('LEZY W POLU PROSTAKATA')
  ELSE
   WRITE('NIE LEZY W POLU PROSTOKATA');
 END.
```

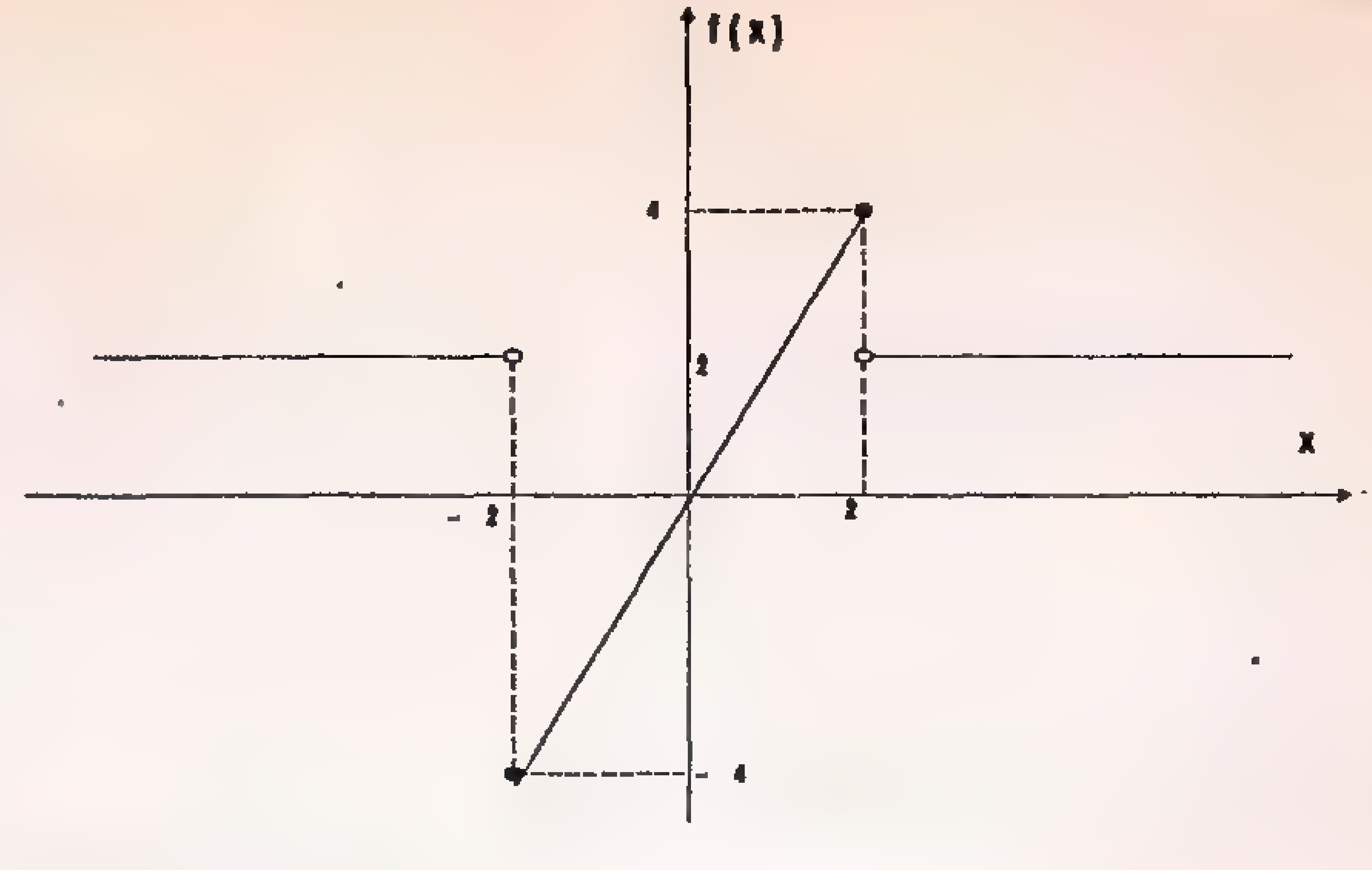
Sposób wykorzystania operatora logicznego OR zamieszczono w przykładzie 18, zawierającym program obliczania wartości funkcji

$$f(x) = \begin{cases} 2 \text{ dla } x<-2 \text{ lub } x>2 \\ 2x \text{ dla } -2<x<2 \end{cases}$$

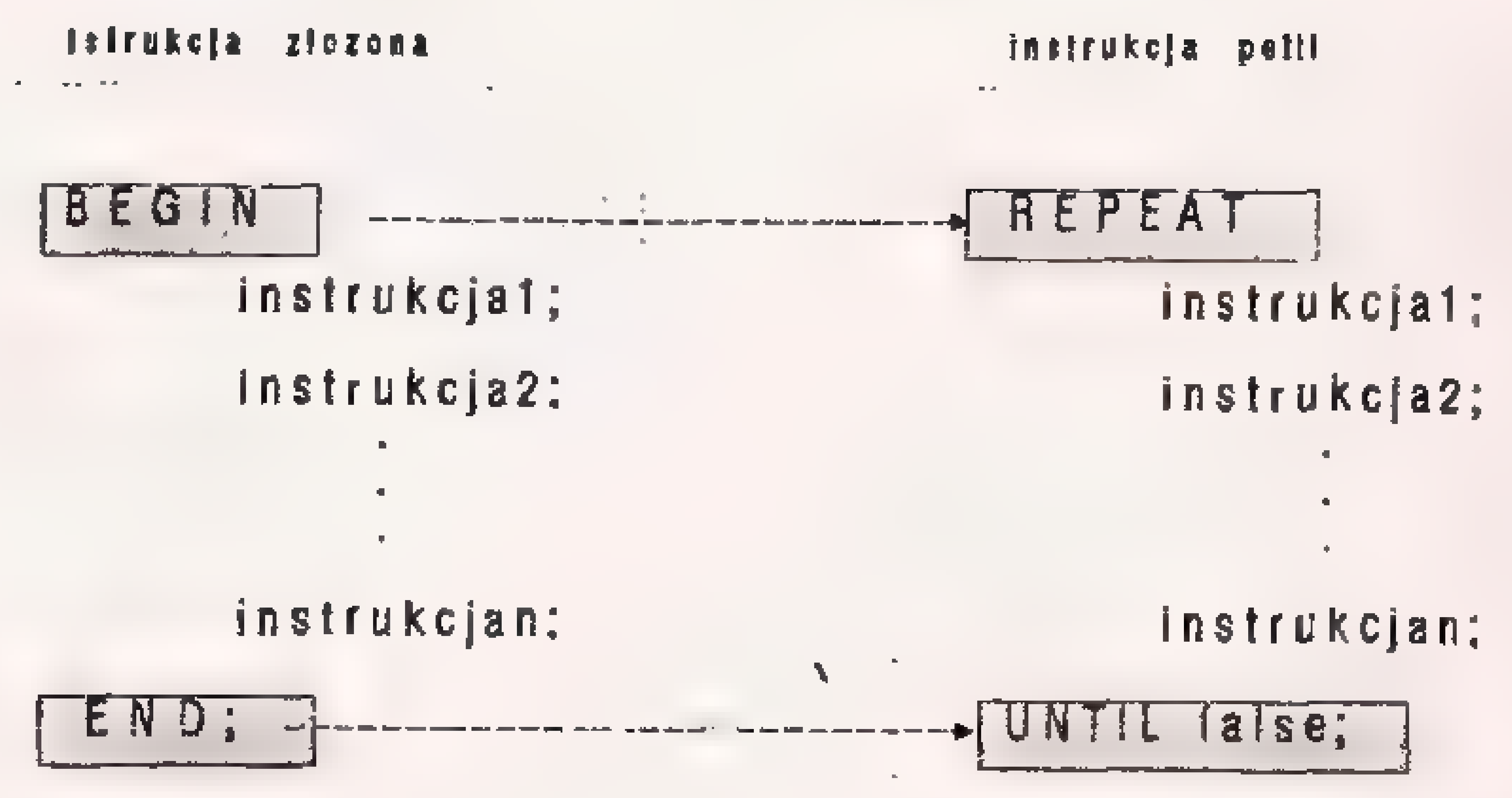
której przebieg zobrazowano na rysunku 7.

Przykład 18

```
PROGRAM prog18;
 VAR
  x,w:real;
 BEGIN
  WRITELN;
  WRITELN('PODAJ WARTOSC x');
  READ(x);
  WRITELN;
  WRITELN('WARTOSC FUNKCJI DLA x=',x);
  IF (x<-2) OR (x>2) THEN
   w:=5
  ELSE
   w:=2*x;
  WRITE(' WYNOSI ',w);
 END.
```



Rys.7.Ilustracja do przykladu 18



Rys.8.Instrukcja zlozona i iteracji

Podobnie jak w przypadku wyrażeń arytmetycznych przy obliczaniu warunków (stwierdzaniu, czy warunki są spełnione) obowiązują reguły pierwszeństwa realizacji. Są one następujące:

- *warunki w nawiasach obliczane są w pierwszej kolejności,*
- *najwyższy priorytet spośród operatorów logicznych ma operator NOT,*
- *operator AND ma pierwszeństwo przed operatorem OR.*

Wszystkie przytoczone dotychczas programy charakteryzują się możliwością jednorazowego wykonania opisanych w nich akcji. Powtarzanie tych akcji jest możliwe jedynie poprzez kolejne wykonania programów. Powtórzenie tych samych akcji przy jednokrotnym uruchomieniu programu byłoby możliwe poprzez powielenie wszystkich instrukcji opisujących daną akcję. Jest to działanie nieefektywne, ponieważ liczba żądanych powtórzeń często nie jest znana.

W języku PASCAL istnieje możliwość opisania w bardzo prosty sposób nieskończonego powtarzania czynności. Na takich samych zasadach, jak dla instrukcji złożonej można zbudować powtarzalną sekwencję instrukcji (rys. 8).

W odróżnieniu od instrukcji złożonej, instrukcje występujące pomiędzy słowami kluczowymi **REPEAT—UNTIL** *false* powtarzane są nieskończoną ilość razy.

W przykładzie 19 zamieszczono program naśladujący działanie prostego kalkulatora umożliwiającego wykonanie dodawania, odejmowania, mnożenia i dzielenia dwóch liczb.

Przykład 19

```
PROGRAM prog19;
 VAR
  znak1,znak2:string[2];
  liczba1,liczba2,
  wynik:real;
 BEGIN
  WRITELN;
  REPEAT
   READ(liczba1,znak1,liczba2,znak2);
   IF znak1=' +' THEN
    wynik:=liczba1+liczba2;
   IF znak1=' -' THEN
    wynik:=liczba1-liczba2;
   IF znak1=' *' THEN
    wynik:=liczba1*liczba2;
   IF znak1=' /' THEN
    wynik:=liczba1/liczba2;
   IF ((znak1=' +') OR (znak1=' -') OR
       (znak1=' *') OR (znak1=' /'))
        AND (znak2=' =') THEN
    WRITE(wynik)
   ELSE
    IF ((znak1<>' +') AND (znak1<>' -') AND
      (znak1<>' *') AND (znak1<>' /'))
       OR ((znak2<>' =') AND (znak2<>' K')) THEN
     BEGIN
      WRITELN;
      WRITE('nierozpoznana operacja');
     END;
   WRITELN;
  UNTIL false;
END.
```

Z konstrukcji programu wynika, że akcje naśladowanego kalkulatora będą wykonywane wielokrotnie. Każda akcja polega na wprowadzeniu *liczby, operatora, drugiej liczby* oraz *znaku równości.* Ze względu na właściwości dotychczas poznanych instrukcji wprowadzania danych po każdej liczbie i operatorze konieczne jest wprowadzenie spacji.

Nietrudno zauważyć, że nasz kalkulator nie da się wyłączyć — działania programu nie możemy bowiem przerwać.

Wykorzystana instrukcja **REPEAT — UNTIL** *false* jest szczególnym przypadkiem instrukcji iteracji przyjmującej postać:

REPEAT
instrukcja 1;
instrukcja 2;

instrukcja n;
UNTIL *warunek*

Sekwencja instrukcji zawarta między słowami kluczowymi REPEAT — UNTIL jest powtarzana tak długo, jak długo warunek występujący po UNTIL nie jest spełniony. Spełnienie warunku — sprawdzane każdorazowo po wykonaniu ostatniej instrukcji sekwencji — powoduje zakończenie realizacji instrukcji iteracji. Wynika stąd, że sekwencja instrukcji:

instrukcja 1;
instrukcja 2;
.
.
.
instrukcja n;

zostanie wykonana co najmniej raz. W szczególnym przypadku warunek może być określony jako wartość logiczna:
— *false* — oznaczająca niespełnienie warunku,
— *true* — oznaczająca spełnienie warunku.

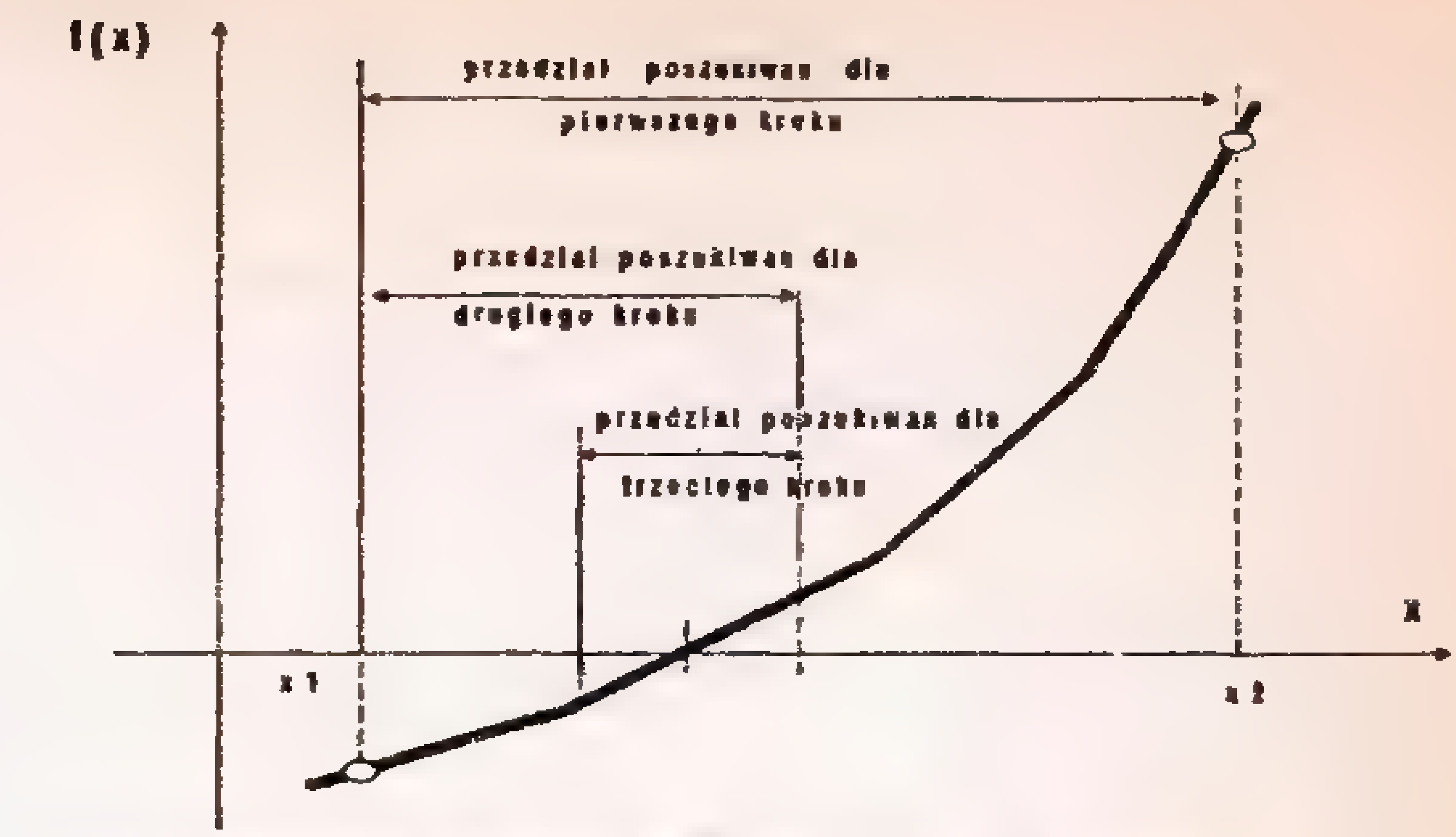
Zatem instrukcja:
REPEAT ... UNTIL *false*
nie zakończy się nigdy, a instrukcja
REPEAT ... UNTIL *true*
jest równoważna instrukcji złożonej.

Instrukcję iteracji wykorzystano w programie z przykładu 20 rozszerzając program naśladujący działanie kalkulatora o możliwość wyłączenia kalkulatora w przypadku wciśnięcia klawisza *k* zamiast dowolnego operatora.

Przykład 20

```
PROGRAM prog20;
 VAR
  znak1,znak2:string[2];
  liczba1,liczba2,
  wynik:real;
 BEGIN
  WRITELN;
  REPEAT
   READ(liczba1,znak1,liczba2,znak2);
   IF znak1=' +' THEN
    wynik:=liczba1+liczba2;
   IF znak1=' -' THEN
    wynik:=liczba1-liczba2;
   IF znak1=' *' THEN
    wynik:=liczba1*liczba2;
   IF znak1=' /' THEN
    wynik:=liczba1/liczba2;
   IF ((znak1=' +') OR (znak1=' -') OR
       (znak1=' *') OR (znak1=' /'))
        AND (znak2=' =') THEN
    WRITE(wynik)
   ELSE
    IF (znak1<>' +') AND (znak1<>' -') AND
       (znak1<>' *') AND (znak1<>' /') AND
       (znak1<>' k') AND (znak1<>' K') AND
       (znak2<>' k') AND (znak2<>' K') THEN
     BEGIN
      WRITELN;
      WRITE('nierozpoznana operacja');
     END;
   WRITELN;
  UNTIL (znak1=' k') OR (znak1=' K')
     OR (znak2=' k') OR (znak2=' K');
  WRITE('WYLACZENIE KALKULATORA');
 END.
```

Instrukcja iteracji jest bardzo ważnym elementem języka programowania. Dlatego też kolejne cztery przykłady poświęcone są jej wykorzystaniu. Dotyczą one znanych zagadnień matematycznych. Rozważmy problem obliczania wartości szeregu:



Rys.9.Schemat okreslania przedzialu , w ktorym lezy pierwiastek funkcji f(x) metoda polowienia przedziale

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

z zadaną dokładnością. Można go rozwiązać przez dodawanie do siebie kolejno obliczonych wartości wyrazów szeregu. Należy przy tym zauważyć, że:
— wartości kolejnych wyrazów szeregu można wyznaczyć z zależności:

$$a_{n+1} = a_n \frac{x}{n+1}$$

przyjmując, że $a_0 = 1$ (a_n — n-ty wyraz szeregu),
— o dokładności obliczeń decyduje wartość ostatnio obliczonego wyrazu szeregu.
Program obliczania wartości podanego szeregu zamieszczono w przykładzie 21.

Przykład 21

```
PROGRAM prog21;
 VAR
  n,       {numer wyrazu szeregu}
  s,       {wartosc szeregu}
  p,       {wartosc kolejnego wyrazu}
  eps,     {dokladnosc obliczen}
  x:real;
 BEGIN
  s:=0;
  p:=1;
  n:=0;
  READ(x,eps);
  REPEAT
   s:=s+p;
   n:=n+1;
   p:=p*x/(n+1);
  UNTIL abs(p)<=eps;
  WRITELN;
  WRITELN('wartosc szeregu dla x=',x);
  WRITELN('obliczona z dokladnoscia eps=',
          eps);
  WRITE('wynosi s=',s);
 END.
```

Równie popularnym zadaniem jest wyznaczenie miejsca zerowego funkcji f (x) w zadanym przedziale. W praktyce najczęściej kreśli się funkcję

$$y = f(x)$$

i szacuje w przybliżeniu jej miejsce zerowe. Niżej przedstawiono dwie metody wyznaczania miejsca zerowego funkcji w zadanym przedziale wykorzystując twierdzenie:

„jeżeli f(x) jest funkcją ciągłą w przedziale $[x_1, x_2]$ *i jeśli* $f(x_1)$ *i* $f(x_2)$ *są przeciwnego znaku, to w przedziale* $[x_1, x_2]$ *istnieje co najmniej jeden pierwiastek rzeczywisty".*

Pierwsza metoda, tzw. metoda połowienia przedziałów, pozwala przy zadanym przedziale początkowym $[x_1, x_2]$ określić przedział, w którym znajduje się miejsce zerowe funkcji *f(x)* z dokładnością *eps*. Postępując zgodnie z tą metodą (jej ideę ilustruje rysunek 9) dzielimy przedział $[x_1, x_2]$ na połowy i w wyznaczonym środku przedziału obliczamy wartość funkcji f_s

$$f_S = f\left(\frac{x_1 + x_2}{2}\right)$$

Następnie badamy czy wartość funkcji w wyznaczonym punkcie zmieniła znak względem $f(x_1)$. Jeżeli tak, to zawężamy przedział do $[x_1, x_S]$, jeżeli nie, to zawężamy do $[x_S, x_2]$

$$x_S = \frac{x_1 + x_2}{2}$$

Po zbadaniu warunków dokładności wyprowadzamy wyniki (jeśli są spełnione) albo powyższe czynności powtarzamy. W przykładzie 22 zamieszczono program określenia przedziału, wykorzystujący metodę połowienia przedziałów, w którym znajduje się miejsce zerowe funkcji:

$$f(x) = -x^3 + 2x^2 + 9$$

z żądaną dokładnością, przy zadanym przedziale początkowym.

Przykład 22

```
PROGRAM prog22;
 VAR
  x1, {dolna granica przedzialu}
  x2, {gorna granica przedzialu}
  xs, {srodek przedzialu}
  f1, {wartosc funkcji w x1}
  fs, {wartosc funkcji w xs}
  eps:real;  {dokladnosc obliczen}
 BEGIN
  WRITELN;
  WRITELN('podaj granice przedzialu',
         ' i dokladnosc obliczen');
  READ(x1,x2,eps);
  REPEAT
   xs:=(x1+x2)/2;
   fs:=-SQR(xs)*xs+2*SQR(xs)+9;
   f1:=-SQR(x1)*x1+2*SQR(x1)+9;
   IF f1*fs<>0 THEN
    IF f1*fs<0 THEN
     x2:=xs
    ELSE
     BEGIN
      f1:=fs;
      x1:=xs;
     END;
  UNTIL (ABS(x1-x2)<=eps) OR (f1*fs=0);
  WRITELN;
  WRITELN('znalezione granice przedzialu:');
  WRITELN('x1 = ',x1);
  WRITE('x2 = ',x2);
 END.
```

Wykorzystana w programie funkcja **SQR** oblicza kwadrat podanego argumentu (x_S lub x_1), a *funkcja* **ABS** — wartość bezwzględną argumentu. Idea drugiej metody została zilustrowana na rysunku 10 (jest to tzw. metoda Newtona).

Rozpoczynając od zerowego przybliżenia x_0, kolejne x_1 wyznaczamy przez wykreślenie:

a) linii pionowej $x = x_0$ aż do przecięcia z krzywą $y = f(x)$ — w ten sposób wyznaczony został punkt p_0,

b) stycznej do krzywej $y = f(x)$ w punkcie p_0 aż do przecięcia z osią x w punkcie x_1.

Powtarzając opisane czynności, otrzymujemy ciąg przybliżeń x_0, x_1, x_2,..., którego granica

$$\lim_{i \to \infty} x_i = x$$

jest poszukiwanym miejscem zerowym funkcji *f(x)*. Z rysunku 10 wynika, że tangens kąta nachylenia stycznej poprowadzonej przez punkt p_0 jest równy

$$tg\alpha = \frac{f(x_0)}{x_0 - x_1} = f'(x_0)$$

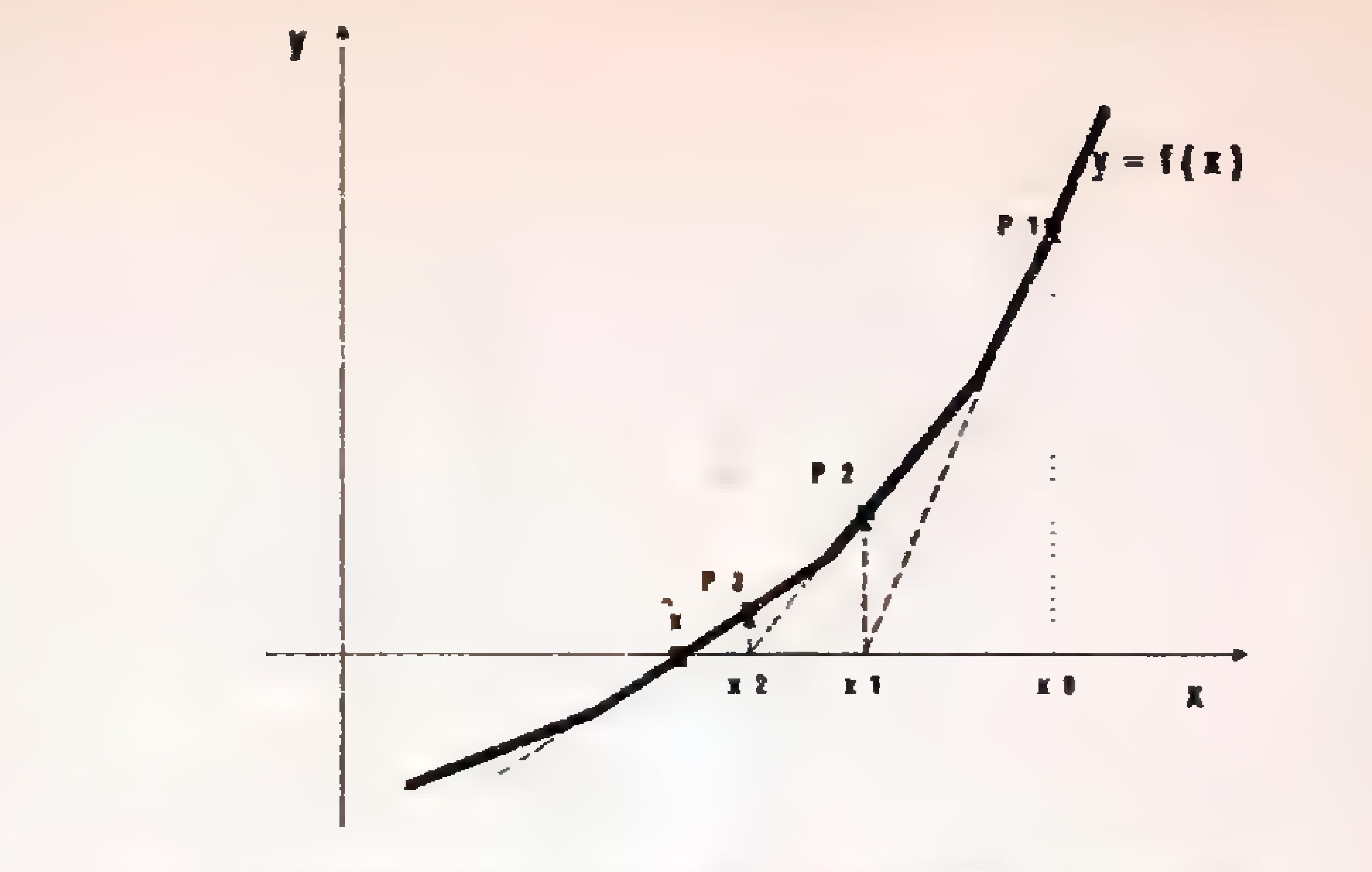
Jeżeli został spełniony warunek $f'(x_0) \neq 0$, to otrzymujemy

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Ogólny wzór na kolejne przybliżenie miejsca zerowego funkcji *f(x)* ma następującą postać:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad dla\ i = 0,1,2...$$

Wzór ten jest słuszny przy założeniu, że $f'(x_i) \neq 0$, a zatem w niektórych przypadkach metoda Newtona zawodzi (należy więc wybrać nowy punkt startu).



Rys.10. Graficzna interpretacja metody Newtona dla przypadku gdy f'(x)≠0

Metodę Newtona wykorzystano do wyznaczenia miejsca zerowego funkcji

$$f(x) = x^4 - 0{,}5\, x^3 - 30$$

z zadaną dokładnością i punktem startu (przykład 23).

Przykład 23

```
PROGRAM prog23;
 VAR
  x,           {punkt aktualny}
  xk,          {punkt kolejny}
  eps,         {zadana dokladnosc obliczen}
  kwx,         {kwadrat x}
  doklb:real;  {biezaca dokladnosc obliczen}
  n,           {maksymalna liczba krokow}
  i:integer;   {licznik krokow}
 BEGIN
  WRITELN;
  WRITELN('podaj zerowe przyblizenie, ',
         'dokladnosc obliczen');
  WRITELN('i maksymalna ilosc krokow');
  READ(x,eps,n);
  i:=0;
  REPEAT
   kwx:=x*x;
   xk:=x-(kwx*(kwx-0.5*x)-30)/
      (kwx*(4*x-1.5));
   doklb:=ABS((xk-x)/xk);
   IF doklb>eps THEN
                BEGIN
                 x:=xk;
                 i:=i+1;
                END;
  UNTIL (doklb<=eps) OR (i=n);
  WRITELN;
  WRITELN('miejsce zerowe funkcji x=',x);
  WRITE('wyznaczone w i=',i,' krokach');
 END.
```

W przypadku żądania dużej dokładności obliczanie przedstawionego programu może trwać zbyt długo, a dla niektórych przedziałów zmienności funkcji może się w ogóle nie zakończyć (uwaga ta dotyczy wszystkich problemów tego typu).

Dobrze skonstruowany program powinien zawierać górne ograniczenie liczby wykonywanych powtórzeń. Liczba powtórzeń (maksymalna liczba powtórzeń) jest liczbą całkowitą. W języku PASCAL istnieje możliwość deklarowania zmiennych przyjmujących wartości całkowite. W programie **prog23** zostało to wykorzystane do deklaracji licznika kroków *(i)* oraz maksymalnej liczby kroków *(n)*.

Rozważmy teraz problem obliczania całki oznaczonej

$$\int_a^b f(x)\,dx$$

Obliczenie takiej całki jest proste, gdyż *f(x)* jest funkcją elementarną. W przypadkach, gdy nie udaje się łatwo wyznaczyć funkcji pierwotnej, stosujemy metody obliczania przybliżonej wartości całki oznaczonej. Jedną z takich metod jest metoda prostokątów. Ideę metody prostokątów ilustruje rysunek 11.

W metodzie tej przedział całkowania od $a = x_0$ do $b = x_n$ dzieli się na n równych części punktami x_1, x_2,..., x_n. Następnie oblicza się wartość funkcji podcałkowej w punktach

$\frac{x_i - x_{i-1}}{2}$ dla $i \in [1,n]$. Wartość całki jest sumą pól prostokątów utworzonych jak na rysunku 11. Można ją zatem wyrazić wzorem:

$$\int_a^b f(x)\,dx \approx h \cdot \sum_{i=1}^{n} f(x_i)$$

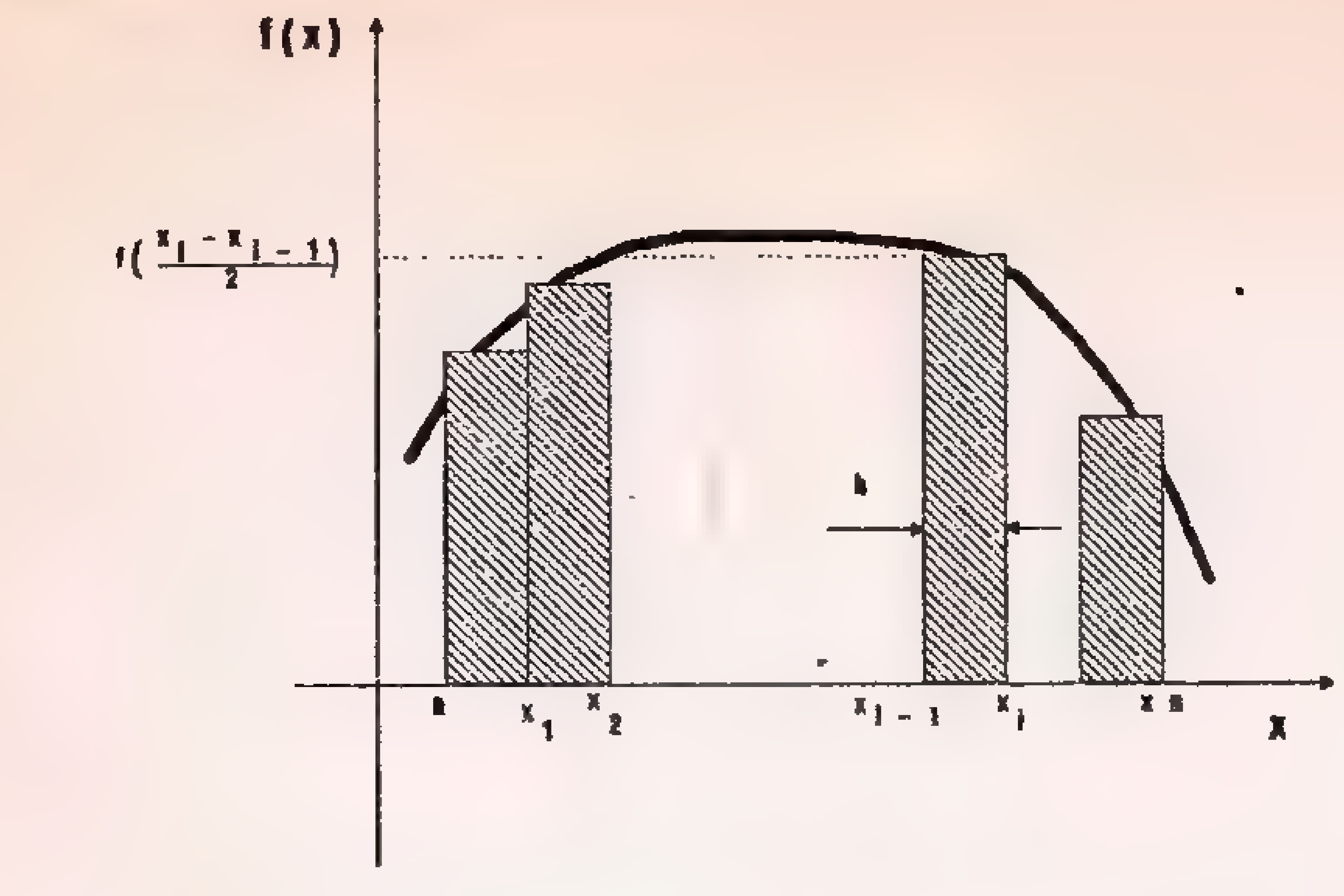
gdzie:

n — liczba podprzedziałów,

$$h = \frac{b-a}{n}$$

$$x_j = a + (j - \frac{1}{2})\,h$$

Przykład 24 przedstawia program obliczania całki



Rys.11.Ilustracja metody prostokatow

$$\int_a^b (\ln x + 5x)\,dx$$

metodą prostokątów. Przyjęto w nim, że kolejne przybliżenia całki są obliczane (przy podwojonej każdorazowo liczbie podprzedziałów) dotąd, aż bezwzględna wartość różnicy między dwoma kolejnymi przybliżeniami będzie co najmniej równa wymaganej dokładności obliczeń. Wprowadzono również ograniczenie ilości wykonywanych powtórzeń.

Przykład 24

```
PROGRAM prog24;
 VAR
  a, {dolna granica przedzialu calkowania}
  b, {gorna granica przedzialu calkowania}
  eps:real; {dokladnosc obliczen}
  n:integer; {liczba podprzedzialow}
  h, {szerokosc podprzedzialu}
  w1, {wartosc calki z kroku poprzedniego}
  w2, {aktualna wartosc calki}
  x,      { zmienne }
  i:real; { robocze }
 BEGIN
  WRITELN;
  WRITELN('PODAJ:-dolna granice przedzi',
          'alu calkowania');
  WRITELN(' ':6,'-gorna granice przedzi',
          'alu calkowania');
  WRITELN(' ':6,'-dokladnosc obliczen');
  READ(a,b,eps);
  n:=1;
  h:=b-a;
  x:=a+h/2;
  w2:=LN(x)+5*x;
  REPEAT
   n:=2*n;
   h:=h/2;
   w1:=w2;
   w2:=0;
   x:=a+h/2;
   i:=1;
   REPEAT
    w2:=w2+LN(x)+5*x;
    x:=x+h;
    i:=i+1;
   UNTIL i>n;
   w2:=w2*h;
  UNTIL ABS(w2-w1)<=eps;
  WRITELN;
  WRITELN('wartosc calki dla funkcji ',
          'F(X)=LN(X)+5*X');
  WRITELN('w przedziale ',a,'  -',b);
  WRITELN('wyliczona z dokladnoscia ',eps);
  WRITE('wynosi ',w2);
 END.
```

Obok instrukcji REPEAT w języku PASCAL występuje druga instrukcja iteracji przyjmująca postać:

WHILE *warunek* **DO** *instrukcja 1*

Powoduje ona, że *instrukcja 1* wykonywana jest tylko wtedy, gdy warunek jest spełniony i tak długo, jak długo jest on spełniony. Oznacza to, że przed każdym wykonaniem *instrukcji 1* następuje sprawdzenie warunku. Niespełnienie warunku powoduje zakończenie powtarzania realizacji *instrukcji 1*. Przykład wykorzystania instrukcji **WHILE** zamieszczono w programie **prog25**, wyznaczającym największy wspólny podzielnik dwóch całkowitych liczb dodatnich.

Przykład 25

```
PROGRAM prog25;
 VAR
  x,y:real;
 BEGIN
  WRITELN('PODAJ x I y');
  READ(x,y);
  WRITELN;
  WRITELN('NAJWIEKSZY WSPOLNY',
          'PODZIELNIK LICZB');
  WRITELN('x =',x:5:0,' y =',y:5:0);
  WHILE x<>y DO
   IF x>y THEN
    x:=x-y
   ELSE
    y:=y-x;
  WRITE('WYNOSI ',x:5:0);
 END.
```

Uwaga! Jeśli przed wykonaniem instrukcji WHILE warunek nie jest spełniony, to *instrukcja 1* nie jest w ogóle wykonywana.

ZŁOŻONE TYPY DANYCH

2



Dotychczas poznane instrukcje wyczerpują zbiór najczęściej wykorzystywanych instrukcji języka PASCAL. Jednak o użyteczności języka decydują nie tylko dostępne instrukcje, ale także dostępne dla programisty typy danych. Występujące w przedstawionych wcześniej przykładach typy *string*, *real* i *integer* należą do grupy typów prostych. Oprócz nich w języku PASCAL istnieją również typy strukturalne, umożliwiające opis obiektów, traktowanych jako zbiory wartości typów prostych i strukturalnych. Typy strukturalne są szczególnie przydatne w systemach gromadzenia, przechowywania i przetwarzania dużych ilości informacji. Podstawową cechą takich systemów jest możliwość logicznego grupowania danych. Najprostszą konstrukcją języka PASCAL, pozwalającą na grupowanie danych jest *rekord*. Rekordy umożliwiają opisywanie obiektów, które można scharakteryzować wieloma cechami. Przykładem takiego obiektu może być opis mikrokomputera dostępnego na rynku krajowym. Przyjmując dla uproszczenia, że zawiera on 20-znakową nazwę mikrokomputera, 12-znakową nazwę firmy produkującej mikrokomputer oraz minimalną cenę mikrokomputera (w tys. zł), „rynkowy" mikrokomputer można opisać następująco:

RECORD
  nazmik: string [20];
  firma: string [12];
  cenamin: integer
END

gdzie:

nazmik, firma, cenamin — nazwy składowych rekordu zwanych polami,
RECORD, END — para słów kluczowych, oznaczających definicję rekordu.

Definicja rekordu może wystąpić jako element definicji typu zmiennych przyjmującej postać:

TYPE *nazwa typu = określenie typu*

Jeżeli określenie typu będzie definicją powyższego rekordu, a nazwą typu będzie *opmikrok*, to definicja typu przyjmie postać:

TYPE *opmikrok* = **RECORD**
nazmik: string [20];
[12];
cenamin: integer
END

Definicja ta umożliwia deklarowanie zmiennych np. według formuły

VAR *r1, r2, r3: opmikrok;*

W tym przypadku każda zmienna opisuje inny mikrokomputer. Sposób odwoływania się do pól zmiennej rekordowej przez tzw. nazwy kwalifikowane wyjaśnia przykład 26.

Przykład 26

```
...
TYPE opmikrok=RECORD
               nazmik:string[20];
               firma:string[12];
               cenamin:integer
              END;
...
VAR
 rek:opmikrok;
 x:string[20];
...
BEGIN
...
 rek.nazmik:='AMSTRAD';
 rek.firma:='BEAB';
 rek.cenamin:=750;
...
 x:=rek.nazmik;
...
```

Po wykonaniu przedstawionej sekwencji instrukcji poszczególnym polom zmiennej *rek* zostaną przypisane wartości. Zmienna x (typu takiego jak pole *nazmik*) przyjmie wartość AMSTRAD.

Przedstawiony sposób odwoływania się do pól zmiennej rekordowej może być uciążliwy (zwłaszcza, gdy jest ich dużo lub są to pola typu strukturalnego), gdyż każde pole musi być poprzedzone co najmniej nazwą zmiennej. Wygodniejszy dostęp do pól zmiennej rekordowej umożliwia instrukcja **WITH**, zwana instrukcją wiążącą lub instrukcją dostępu. Pozwala ona na odwoływanie się do pól zmiennej rekordowej bezpośrednio przez nazwy pól, przy jednokrotnym podaniu nazwy zmiennej rekordowej. Wykorzystując instrukcję **WITH**, fragment programu z przykładu 26 można zmodyfikować tak jak podano w przykładzie 27.

Przykład 27

```
...
TYPE opmikrok=RECORD
               nazmik:string[20];
               firma:string[12];
               cenamin:integer
              END;
...
VAR
 rek:opmikrok;
 x:string[20];
...
BEGIN
...
 WITH rek DO
  BEGIN
...
   nazmik:='AMSTRAD';
   firma:='BEAB';
   cenamin:=750;
...
   x:=nazmik;
...
  END;
...
```

W praktyce występuje konieczność gromadzenia danych o identycznej strukturze, których liczba nie jest z góry określona i może się zmieniać w czasie. W języku PASCAL podstawową konstrukcją o takich właściwościach jest plik — sekwencja danych o nieokreślonej długości, przechowywanych w pamięci zewnętrznej (np. dyskowej).

Definicja pliku przyjmuje postać:

TYPE *nazwa pliku* = **FILE OF** *nazwa typu elementu,*

Przykładowa definicja pliku zawierającego rekordy opisujące mikrokomputer jest następująca:

TYPE *zbopmk* = **FILE OF** *opmikrok;*

Z definicją typu nie wiąże się żadna fizyczna reprezentacja. Jest to bowiem informacja o strukturze zmiennych, którym przypisano w deklaracji ten typ. W deklaracji:

VAR *zb1, zb2,: zbopmk;*

powołuje się dwie zmienne *zb1, zb2* reprezentowane jako pliki — zbiory rekordów typu *opmikrok*.

Z plikami związane są operacje:
— tworzenia pliku,
— otwarcia pliku,
— zamknięcia pliku,
— dopisania nowego elementu do pliku,
— odczytu bieżącego elementu z pliku,
— zmiany bieżącego elementu pliku,
— kasowania pliku,
— zmiany nazwy pliku,

Operacje te zostaną omówione w trakcie rozwijania programów wkomponowanych w prosty system obsługi domowej biblioteki obejmujący:
— zakładanie kartoteki (pliku) opisów pozycji,
— dopisywanie nowych opisów pozycji do kartoteki,
— wypisywanie zawartości kartoteki,
— kasowanie wskazanych opisów pozycji z kartoteki.

Przed omówieniem poszczególnych programów niezbędne jest krótkie wyjaśnienie niektórych terminów związanych z operacjami na plikach. W systemie TURBO PASCAL przyjęto, że nazwa pliku nie jest identyfikatorem zbioru danych w pamięci zewnętrznej. Zatem przed wykonaniem jakiejkolwiek operacji na pliku niezbędne jest dowiązanie do nazwy pliku nazwy zbioru w pamięci zewnętrznej z zawartością pliku. Rozwiązanie to realizuje instrukcja

ASSIGN *(nazwa pliku, 'nazwa zbioru')*

gdzie:
nazwa pliku — nazwa zmiennej plikowej (np. zb1),
nazwa zbioru — nazwa zbioru w pamięci zewnętrznej z zawartością pliku (nazwa ta jest łańcuchem znaków o postaci przyjętej dla zbiorów roboczych edytora — patrz „Dodatek").

Po wykonaniu **ASSIGN** wszystkie operacje na pliku są wykonywane na przypisanym mu zbiorze.
Przykładowo:

ASSIGN *(księgozbiór, 'bibldom')*

powoduje dowiązanie zbioru o nazwie *bibldom* do pliku o nazwie *księgozbiór.*

Tworzenie nowego pliku (utworzenie pustego zbioru w pamięci zewnętrznej) realizowane jest za pomocą instrukcji

REWRITE *(nazwa pliku)*

REWRITE tworzy w pamięci zewnętrznej zbiór o nazwie podanej w ASSIGN i udostępnia do zapisu pierwszy element pliku. Jeżeli zbiór o podanej nazwie już istnieje w pamięci zewnętrznej, to jest on kasowany. Po wykonaniu **REWRITE** plik jest pusty, tzn. nie zawiera żadnych elementów.

Po wykorzystaniu pliku należy go zamknąć (przygotować do przechowywania i ponownego wykorzystania) za pomocą instrukcji:

CLOSE *(nazwa pliku)*

Program zakładania pustej kartoteki opisów pozycji biblioteki domowej przedstawiono w przykładzie 28.

Przykład 28

```
PROGRAM prog26;
 TYPE
  pozbibl=RECORD
           autor:string[35];
           tytul:string[50];
           rokwyd,
           nrkat:integer
          END;
  zbbibl=FILE OF pozbibl;
 VAR
  ksiegozbior:zbbibl;
 BEGIN
  ASSIGN(ksiegozbior,'bibldom');
  REWRITE(ksiegozbior);
  CLOSE(ksiegozbior);
 END.
```

W kolejnym przykładzie zamieszczono program dopisywania rekordów na koniec założonej kartoteki.

Wykorzystano w nim instrukcje umożliwiające zmianę bieżącego (zapisywanego lub odczytywanego) elementu pliku. Instrukcja:

SEEK *(nazwa pliku, numer elementu)*

powoduje, że element o wskazanym numerze staje się elementem bieżącym (numer elementu liczony jest od zera). W celu uniknięcia konieczności poszukiwania ostatniego elementu pliku można wykorzystać funkcję

FILESIZE *(nazwa pliku),*

której wartością jest liczba rekordów pliku.
Instrukcja:

SEEK *(nazwa pliku,* **FILESIZE** *(nazwa pliku))*

spowoduje, że elementem bieżącym staje się nowy, pusty element umieszczony na końcu pliku. Zapis do bieżącego elementu pliku realizuje instrukcja

WRITE *(nazwa pliku, nazwa zmiennej)*

przy czym *nazwa zmiennej* określa zmienną, której wartość jest zapisywana do bieżącego elementu pliku. Typ zmiennej musi być zgodny z typem elementów pliku.

Otwarcie istniejącego pliku jest wykonywane przez instrukcję.

RESET *(nazwa pliku)*

Należy przypomnieć, że instrukcja **REWRITE** nie może tu być wykorzystywana, bowiem niszczy ona dotychczasową zawartość pliku. Instrukcja **RESET** może być używana tylko w odniesieniu do już istniejących plików. W wyniku wykonania **RESET** elementem bieżącym staje się pierwszy element pliku.

W szczególnym przypadku jest on pusty.

Przykład 29

```
PROGRAM prog27;
 TYPE
  pozbibl=RECORD
           autor:string[35];
           tytul:string[50];
           rokwyd,
           nrkat:integer
          END;
  zbbibl=FILE OF pozbibl;
 VAR
  ksiegozbior:zbbibl;
  rek:pozbibl;
  znkon:string[1];
 BEGIN
  ASSIGN(ksiegozbior,'bibldom');
  RESET(ksiegozbior);
  SEEK(ksiegozbior,FILESIZE(ksiegozbior));
  WRITELN;
  REPEAT
   WITH rek DO
    BEGIN
     WRITELN('podaj nazwisko i imie autora',
             ' (do 35 znakow)');
     READLN(autor);
     WRITELN('podaj tytul ksiazki',
             ' (do 50 znakow)');
     READLN(tytul);
     WRITELN('podaj rok wydania');
     READLN(rokwyd);
     WRITELN('nadaj numer katalogowy');
     READLN(nrkat);
    END;
   WRITE(ksiegozbior,rek);
   WRITELN('czy wprowadzasz nastepna ',
           'pozycje? (T/N)');
   READLN(znkon);
  UNTIL (znkon='N') OR (znkon='n');
  CLOSE(ksiegozbior);
 END.
```

W uzupełnieniu do przykładu 29 należy dodać, że program **prog27** może być wykorzystywany wielokrotnie bez względu na zawartość kartoteki. Przy jednokrotnym uruchomieniu programu można dopisywać do kartoteki dowolną liczbę opisów pozycji.

Istotnym elementem systemu gromadzenia informacji jest możliwość przeglądania zawartości kartotek. Program przeglądania naszej kartoteki zamieszczono w przykładzie 30. Wykorzystano w nim instrukcję:

READ *(nazwa pliku, nazwa zmiennej)*

powodującą przypisanie zmiennej *nazwa zmiennej* wartości bieżącego elementu. Podobnie, jak dla instrukcji **WRITE**, powinien być spełniony warunek zgodności typów. Ponadto wykorzystano funkcję logiczną:

EOF *(nazwa pliku)*

której wartością jest *true*, jeżeli bieżący element jest ostatnim elementem pliku, *false* — w przypadku przeciwnym.

Przykład 30

```
PROGRAM prog28;
 TYPE
  pozbibl=RECORD
           autor:string[35];
           tytul:string[50];
           rokwyd,
           nrkat:integer
          END;
  zbbibl=FILE OF pozbibl;
 VAR
  ksiegozbior:zbbibl;
  rek:pozbibl;
  lpom:string[1];
 BEGIN
  ASSIGN(ksiegozbior,'bibldom');
  RESET(ksiegozbior);
  CLRSCR;
  WHILE NOT EOF(ksiegozbior) DO
   BEGIN
    READ(ksiegozbior,rek);
    WITH rek DO
     BEGIN
      WRITELN('AUTOR: ',autor);
      WRITELN('TYTUL: ',tytul);
      WRITELN('ROK WYDANIA: ',rokwyd);
      WRITELN('NUMER KATALOGOWY: ',nrkat);
     END;
    WRITELN('========================',
            '========================');
    WRITELN;
    WRITELN(' ':20,'- VERTE -');
    READ(lpom);
    CLRSCR;
   END;
  CLOSE(ksiegozbior);
 END.
```

Ponieważ zawartość całego pliku może się nie zmieścić na ekranie, program **prog28** został tak skonstruowany, że wyświetlenie kolejnej pozycji uwarunkowane jest wciśnięciem klawisza **ENTER**.

W kolejnym przykładzie zamieszczono program umożliwiający kasowanie wskazanych opisów pozycji kartoteki.

W programie zastosowano bardzo prostą metodę usuwania opisów. Usunięcie jednego opisu powoduje przepisanie wszystkich pozostałych opisów do pomocniczego zbioru *zbpom* (rysunek 12).

Następnie kasowany jest zbiór *bibldom*, a zbiór *zbpom* przyjmuje nazwę *bibldom*. W programie wykorzystano dwie nowe instrukcje:

ERASE *(nazwa pliku)*

powodującą skasowanie zbioru związanego z plikiem oraz:

RENAME *(nazwa pliku, 'nazwa zbioru')*

zmieniającą nazwę zbioru związanego z plikiem na nazwę *nazwa zbioru*. Obie powyższe instrukcje mogą być wykonywane tylko na plikach, które zostały uprzednio zamknięte za pomocą instrukcji **CLOSE**.

Przykład 31

```
PROGRAM prog29;
 TYPE
  pozbibl=RECORD
           autor:string[35];
           tytul:string[50];
           rokwyd,
           nrkat:integer
          END;
  zbbibl=FILE OF pozbibl;
 VAR
  ksiegozbior,
  zbpom:zbbibl;
  rek:pozbibl;
  idpozkas:string[55];
  znkon:string[1];
 BEGIN
  REPEAT
   ASSIGN(ksiegozbior,'bibldom');
   ASSIGN(zbpom,'pom.dta');
   RESET(ksiegozbior);
   REWRITE(zbpom);
   WRITELN;
   WRITELN('podaj tytul kasowanej pozycji');
   READLN(idpozkas);
   WHILE NOT EOF(ksiegozbior) DO
    BEGIN
     READ(ksiegozbior,rek);
     IF rek.tytul<>idpozkas THEN
        WRITE(zbpom,rek);
    END;
   CLOSE(ksiegozbior);
   CLOSE(zbpom);
   ERASE(ksiegozbior);
   RENAME(zbpom,'bibldom');
   WRITELN;
   WRITELN('czy kasujesz nastepna pozycje?',
           ' (T/N)');
   READ(znkon);
  UNTIL (znkon='N') OR (znkon='n');
 END.
```

Pliki umożliwiają gromadzenie i przechowywanie danych w pamięciach zewnętrznych. Jest to ich niewątpliwą zaletą. Do wad plików można zaliczyć długi czas dostępu do ich elementów (ze względu na wymagany kontakt z pamięcią zewnętrzną).

Alternatywną formą deklarowania zbiorów danych w języku PASCAL są zmienne tablicowe. Nie mają one swojej reprezentacji w pamięciach zewnętrznych, przechowywane są w pamięci komputera i „giną" wraz z

zakończeniem programu. Liczba ich elementów jest z góry określona. Dane przechowywane są w postaci tablic jedno-, dwu- lub wielowymiarowych.

Jednowymiarowa tablica, to ciąg ponumerowanych elementów, których wartości są tego samego typu. Dwuwymiarowa tablica, to ciąg wierszy lub kolumn, z których każda liczy tyle samo elementów tego samego typu. Rozmiary dwuwymiarowej tablicy, to liczba wierszy i liczba elementów w wierszu czyli liczba kolumn. Za pomocą jednowymiarowych tablic opisujemy zwykle takie obiekty jak wektory, natomiast za pomocą tablic dwuwymiarowych — macierze. Ponieważ nie przewidujemy przedstawiania problemów wymagających tablic o wymiarach większych niż 2, poprzestaniemy na omówieniu tablic co najwyżej dwuwymiarowych.

Naturalnym sposobem opisu tablic jest notacja:

a) **ARRAY** [*1.. n*] **OF** *integer*
dla opisu tablicy jednowymiarowej o n elementach typu integer,

b) **ARRAY** [*1.. m, 1.. n*] **OF** *real*
dla opisu tablicy dwuwymiarowej o rozmiarach m x n i elementach typu real.

W przykładzie 32 zamieszczono prosty program nadawania wartości elementom jednowymiarowej tablicy a.

Przykład 32

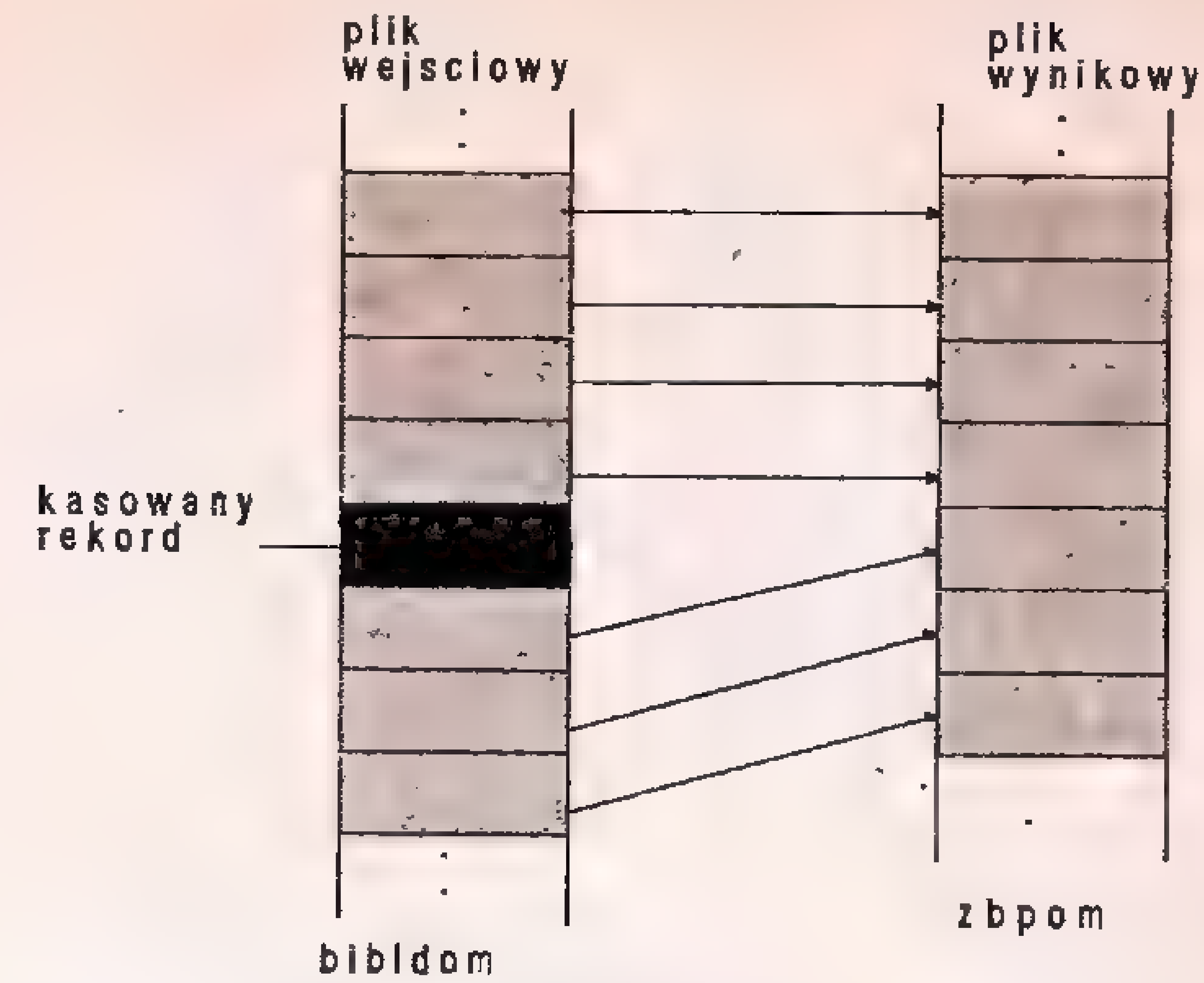
```
PROGRAM prog30;
 VAR
  a:ARRAY[1..6] OF integer;
  i:integer;
 BEGIN
  WRITELN;
  i:=1;
  WHILE i<=6 DO
   BEGIN
    a[i]:=i;
    WRITELN(a[i]);
    i:=i+1;
   END;
 END.
```

Działania na elementach tablicy przeprowadza się tak jak na zmiennych typu zgodnego z typem elementów tablicy. Każdy element tablicy jest identyfikowany jednoznacznie przez nazwę tablicy oraz tzw. indeks, określający położenie elementów w tablicy, np.:

a [*i*] — wskazywany przez wartość zmiennej *i* element jednowymiarowej tablicy a.

a [*i — 1*] — wskazywany przez wartość wyrażenia *(i — 1)*, element jednowymiarowej tablicy a.

W przypadku, gdy indeks jest wyrażeniem, wartość wyrażenia jest obliczana przed udostępnieniem elementu tablicy. Należy podkreślić, że bieżąca wartość indeksu musi mieścić



Rys.12.Ilustracja metody kasowania rekordu z pliku

się w przedziale wartości dopuszczalnych dla tego indeksu (w programie **prog30** — [1,10]).

W przykładzie 32 do opisu powtarzających się czynności wykorzystano znaną już instrukcję **WHILE**. Nietrudno zauważyć, że w tym przypadku stosowanie tej instrukcji nie jest zbyt wygodne. Konieczne jest bowiem ustalenie liczby powtórzeń operacji wykonanych w instrukcji **WHILE** oraz określenie warunku kończącego jej wykonanie. Warunek powinien być spełniony wtedy, gdy osiągnie się wymaganą liczbę powtórzeń.

Nieprecyzyjne określenie warunku prowadzi do niepoprawnej liczby wykonanych powtórzeń (najczęściej o 1 mniejszej lub o 1 większej od wymaganej).

Do opisu czynności powtarzanych określoną ilość razy wygodniejsza jest instrukcja o postaci:

FOR *I*: = e_1 **TO** e_2 **DO** *instrukcja 1,*

gdzie:

I — zmienna kontrolna (licznik powtórzeń),

e_1 — wyrażenie określające początkową wartość zmiennej kontrolnej,

e_2 — wyrażenie określające końcową wartość zmiennej kontrolnej.

Zapis ten oznacza, że *instrukcja 1* wykonywana jest dla kolejnych wartości zmiennej kontrolnej *I* od wartości początkowej e_1 z krokiem 1 do wartości końcowej e_2. Wartość zmiennej kontrolnej jest zmieniana automatycznie, zatem zmiana taka nie jest dopuszczalna w ramach *instrukcji 1*. Zmienna kontrolna może być natomiast wykorzystywana jako argument w wyrażeniach, jeżeli takie występują w *instrukcji 1*. Przyjmuje się, że po wykonaniu instrukcji **FOR** wartość zmiennej kontrolnej jest nieokreślona.

W przykładzie 33 zamieszczono zmodyfikowany program **prog31**, w którym zamiast instrukcji **WHILE** wykorzystano instrukcję **FOR**.

Przykład 33

```
PROGRAM prog31;
 VAR
  a:ARRAY[1..6] OF integer;
  i:integer;
 BEGIN
  WRITELN;
  FOR i:=1 TO 6 DO
   BEGIN
    a[i]:=i;
    WRITELN(a[i]);
   END;
 END.
```

Aby zamknąć listę instrukcji iteracyjnych języka PASCAL opiszemy drugą postać instrukcji FOR. Jest ona następująca:

FOR *I*: = e_1 **DOWNTO** e_2 **DO** *instrukcja 1*

gdzie:

I, e_1, e_2, *instrukcja 1* — mają identyczne znaczenie, jak w poprzednio omówionej postaci instrukcji FOR.

Zapis ten oznacza, że *instrukcja 1* wykonywana jest dla kolejnych wartości zmiennej kontrolnej *I* od wartości początkowej e_1 do wartości końcowej e_2, przy czym wartość zmiennej kontrolnej jest zmniejszona o 1 po każdym wykonaniu *instrukcji 1*. Przedstawiona postać instrukcji występuje w programie porządkowania sześciu zadanych liczb całkowitych w sposób rosnący (przykład 34) metodą tzw. sortowania bąbelkowego. Metoda ta opiera się na zasadzie porównywania i zamiany par sąsiadujących

ze sobą liczb, do chwili ustawienia ich w kolejności od najmniejszej do największej. Wynika stąd, że warunkiem uporządkowania jest jednoczesne udostępnienie wszystkich sześciu liczb. Jeżeli potraktujemy zadane liczby jako elementy „pionowej” tablicy jednowymiarowej, to przy pewnej dozie wyobraźni możemy je uważać za znajdujące się w naczyniu z wodą bąbelki o „wadze” proporcjonalnej do ich wielkości. Rezultatem każdego przejścia przez tablicę będzie wypchnięcie bąbelka na poziom odpowiadający jego wadze. Przebieg sortowania bąbelkowego ilustruje tabela 1.

TABELA 1

Ustawienie początkowe	Pierwsze przejście	Drugie przejście	Trzecie przejście	Czwarte przejście	Piąte przejście
574	8	8	8	8	8
303	574	23	23	23	23
34	303	574	34	34	34
125	34	303	574	125	125
8	125	34	303	574	303
23	23	125	125	303	574

Przykład 34

```
PROGRAM prog32;
 VAR
  liczby:ARRAY[1..6] OF integer;
  x,i,j:integer;
 BEGIN
  FOR i:=1 TO 6 DO
   READ(liczby[i]);
  WRITELN;
  WRITELN('wprowadzone liczby:');
  FOR i:=1 TO 6 DO
   WRITE(liczby[i]:5,' ':3);
  FOR i:=2 TO 6 DO
   FOR j:=6 DOWNTO i DO
    IF liczby[j-1]>liczby[j] THEN
     BEGIN
      x:=liczby[j-1];
      liczby[j-1]:=liczby[j];
      liczby[j]:=x;
     END;
  WRITELN;
  WRITELN('liczby uporzadkowane:');
  FOR i:=1 TO 6 DO
   WRITE(liczby[i]:5,' ':3);
 END.
```

Kolejne dwa przykłady dotyczą działań na macierzach. Przykład 35 przedstawia program dodawania dwóch macierzy o ustalonych wymiarach. Przypomnijmy, że sumą dwóch macierzy **A** i **B** o tych samych wymiarach jest trzecia macierz **C**, której elementy wyznaczane są wg zależności:

$$c_{ij} = a_{ij} + b_{ij}$$

gdzie:

i — indeks wiersza,
j — indeks kolumny.

W programie, w deklaracji zmiennych tablicowych *(a, b, c)*, wykorzystano typ *macierz* opisujący macierze o wymiarach 3x5 i elementach typu *real*. Wprowadzenie elementów macierzy *a* i *b* następuje wierszami.

Przykład 35

```
PROGRAM prog33;
 TYPE
  macierz=ARRAY[1..3,1..5] OF real;
 VAR
  a,b,         {macierze wejsciowe}
  c:macierz;   {macierz wyjsciowa}
  i,j:integer; {zmienne pomocnicze}
 BEGIN
  WRITELN;
  WRITELN('wprowadz elementy macierz  A');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     READ(a[i,j]);
    READLN;
   END;
  WRITELN('wprowadz elementy macierzy B');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     READ(b[i,j]);
    READLN;
   END;
  FOR i:=1 TO 3 DO
   FOR j:=1 TO 5 DO
    c[i,j]:=a[i,j]+b[i,j];
  WRITELN;
  WRITELN('dane wejsciowe:');
  WRITELN(' ':10,'macierz A');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     WRITE(' ':3,a[i,j]:10);
    WRITELN;
   END;
  WRITELN(' ':10,'macierz B');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     WRITE(' ':3,b[i,j]:10);
    WRITELN;
   END;
  WRITELN('wynik:');
  WRITELN(' ':10,'macierz C=A+B');
  FOR i:=1 TO 3 DO
   BEGIN
    FOR j:=1 TO 5 DO
     WRITE(' ':3,c[i,j]:10);
    WRITELN;
   END;
 END.
```

Zastanówmy się, co należałoby zmienić w programach z przykładów 34 i 35 w przypadku innych wymiarów tablic.

Nietrudno zauważyć, że zmiana wymiarów „rozprzestrzeni” się na cały program. Należy bowiem zmodyfikować również wszystkie instrukcje programu, w których występują wartości graniczne wymiarów tablic. Zmiana wymiarów tablicy *liczby* z przykładu 34 wymaga wprowadzenia zmian w sześciu wierszach programu, a zmiana wymiarów tablicy *macierz* z przykładu 35 — aż w trzynastu wierszach.

Wad tych jest pozbawiony program mnożenia macierzy przedstawiony w przykładzie 36. Przypomnijmy, że iloczyn **AxB** dwóch macierzy **A** i **B** jest taką macierzą **C**, w której element wiersza *i* i kolumny *j* jest sumą iloczynów elementów wiersza *i* macierzy **A** przez odpowiednie elementy kolumny *j* macierzy **B**. Liczba kolumn macierzy **A** musi być przy tym równa liczbie wierszy macierzy **B**. Jeżeli więc macierz **A** jest stopnia *p x q* i macierz **B** — stopnia *q x r*, to macierz **C** jest stopnia *p x r*.

$$\mathbf{C}_{pxr} = \mathbf{A}_{pxq} \cdot \mathbf{B}_{qxr}$$

gdzie:

$$c_{ij} = \sum_{k=1}^{q} a_{ik} \cdot b_{kj}$$

Wymiary tablic zostały określone przez nazwy stałych *(dw, gw, dk, gk)*, zdefiniowanych po słowie kluczowym **CONST**. Ponieważ w instrukcjach programu użyto również zdefiniowanych stałych, nie jest on „wrażliwy” na zmiany wymiarów tablic. Ewentualna zmiana wymiarów tablic wymaga jedynie przedefiniowania odpowiednich stałych.

Przykład 36

```
PROGRAM prog34;
 CONST
  dw=1; {dolna wartosc indeksu wierszy}
  gw=7; {gorna wartosc indeksu wierszy}
  dk=1; {dolna wartosc indeksu kolumn}
  gk=7; {gorna wartosc indeksu kolumn}
 TYPE
  tablica=ARRAY[dw..gw,dk..gk] OF real;
 VAR
  p,           {liczba wierszy macierzy A}
  q1,          {liczba kolumn macierzy A}
  q2,          {liczba wierszy macierzy B}
  r:integer;   {liczba kolumn macierzy B}
  a,b,           {macierze wejsciowe}
  c:tablica;     {macierz wynikowa}
  i,j,k:integer; { zmienne }
  s:real;        { robocze }
 BEGIN
  WRITELN;
  WRITELN('podaj wymiary macierzy A');
  READLN(p,q1);
  IF (p>gw) OR (q1>gk) THEN
   BEGIN
    WRITELN;
    WRITELN('niepoprawne wymiary macierzy A');
   END
  ELSE
   BEGIN
    WRITELN('podaj wymiary macierzy B');
    READLN(q2,r);
    IF (q2>gw) OR (r>gk) THEN
     BEGIN
      WRITELN;
      WRITELN('niepoprawne wymiary macierzy B');
     END
    ELSE
     IF q1<>q2 THEN
      BEGIN
       WRITELN;
       WRITELN('niezgodnosc wymiarow macie',
               'rzy A i B');
      END
     ELSE
      BEGIN
       WRITELN('wprowadz elementy macierzy A');
       FOR i:=dw TO p DO
        BEGIN
         FOR j:=dk TO q1 DO
          READ(a[i,j]);
         READLN;
        END;
       WRITELN('wprowadz elementy macierzy B');
       FOR i:=dw TO q2 DO
        BEGIN
         FOR j:=dk TO r DO
          READ(b[i,j]);
         READLN;
        END;
```

```
        FOR i:=dw TO q2 DO
         FOR j:=dk TO r DO
          BEGIN
           s:=0;
           FOR k:=dk TO q1 DO
            s:=s+a[i,k]*b[k,j];
           c[i,j]:=s;
          END;
        WRITELN;
        WRITELN(' ':5,'macierz A');
        FOR i:=dw TO p DO
         BEGIN
          FOR j:=dk TO q1 DO
           WRITE(a[i,j]:10,' ');
          WRITELN;
         END;
        WRITELN(' ':5,'macierz B');
        FOR i:=dw TO q2 DO
         BEGIN
          FOR j:=dk TO r DO
           WRITE(b[i,j]:10,' ');
          WRITELN;
         END;
        WRITELN(' ':5,'macierz C=A*B');
        FOR i:=dw TO p DO
         BEGIN
          FOR j:=dk TO r DO
           WRITE(c[i,j]:10,' ');
          WRITELN;
         END;
       END;
   END;
 END.
```

Analizując program **prog34** można zauważyć, że aktualnie wymagane wymiary tablic są wczytywane w czasie wykonywania programu. Pozwala to na zmianę wymiarów tablic bez konieczności modyfikacji tekstu programu. Oczywiście zakres zmian jest ograniczony zdefiniowanymi wymiarami tablic. Po wczytaniu bieżących wartości wymiarów tablic należy sprawdzić czy mieszczą się one w przedziale wartości dopuszczalnych.

W dotychczasowych przykładach zaprezentowano dwa sposoby opisu zbiorów danych: w postaci plików oraz tablic. W przykładzie 37 zamieszczono program sortowania zawartości pliku znaną już metodą sortowania bąbelkowego. Ponieważ metoda ta wymaga jednoczesnego udostępnienia wszystkich sortowanych elementów, przed sortowaniem wszystkie elementy pliku zostały umieszczone w tablicy. Zwiększa się w ten sposób szybkość sortowania, ale dla większych plików metody tej zastosować nie można (ze względu na ograniczoną pojemność pamięci komputera). W tych przypadkach trzeba stosować inne metody, których ze względu na ich złożoność nie prezentujemy.

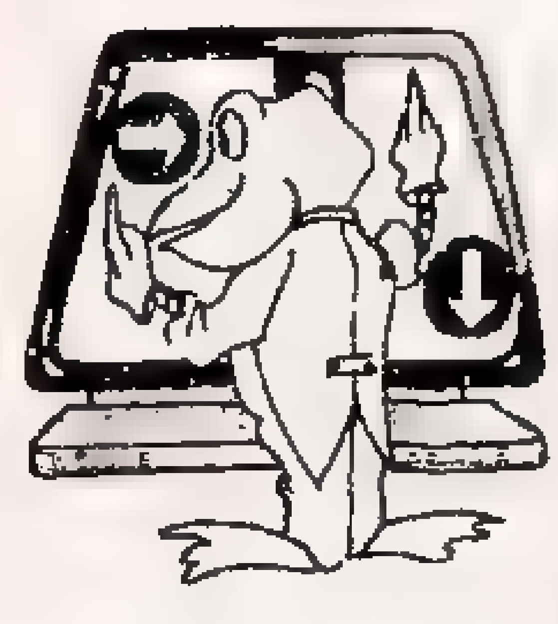
Przykład 37

```
PROGRAM prog35;
 CONST
  n=100;
 TYPE
  pozbibl=RECORD
          autor:string[35];
          tytul:string[50];
          rokwyd,
          nrkat:integer
         END;
  zbbibl=FILE OF pozbibl;
  zbpao=ARRAY[1..100] OF pozbibl;
 VAR
  ksiegozbior:zbbibl;
  tabpom:zbpao;
  rek:pozbibl;
  i,j,ilop:integer;
 BEGIN
  ASSIGN(ksiegozbior,'bibldom');
  RESET(ksiegozbior);
  ilop:=0;
  WHILE NOT EOF(ksiegozbior) DO
   BEGIN
    ilop:=ilop+1;
    READ(ksiegozbior,tabpom[ilop]);
   END;
  FOR i:=2 TO ilop DO
   FOR j:=ilop DOWNTO i DO
    IF tabpom[j-1].autor >
       tabpom[j].autor THEN
     BEGIN
      rek:=tabpom[j-1];
      tabpom[j-1]:=tabpom[j];
      tabpom[j]:=rek;
     END;
  RESET(ksiegozbior);
  FOR i:=1 TO ilop DO
   WRITE(ksiegozbior,tabpom[i]);
  CLOSE(ksiegozbior);
 END.
```

Należy zwrócić uwagę na typ elementów tablicy, w której przeprowadza się sortowanie — elementami tymi są rekordy odpowiadające elementom pliku. Po przesortowaniu elementy tablicy są zapisywane do tego samego pliku.

PROCEDURY I FUNKCJE

3



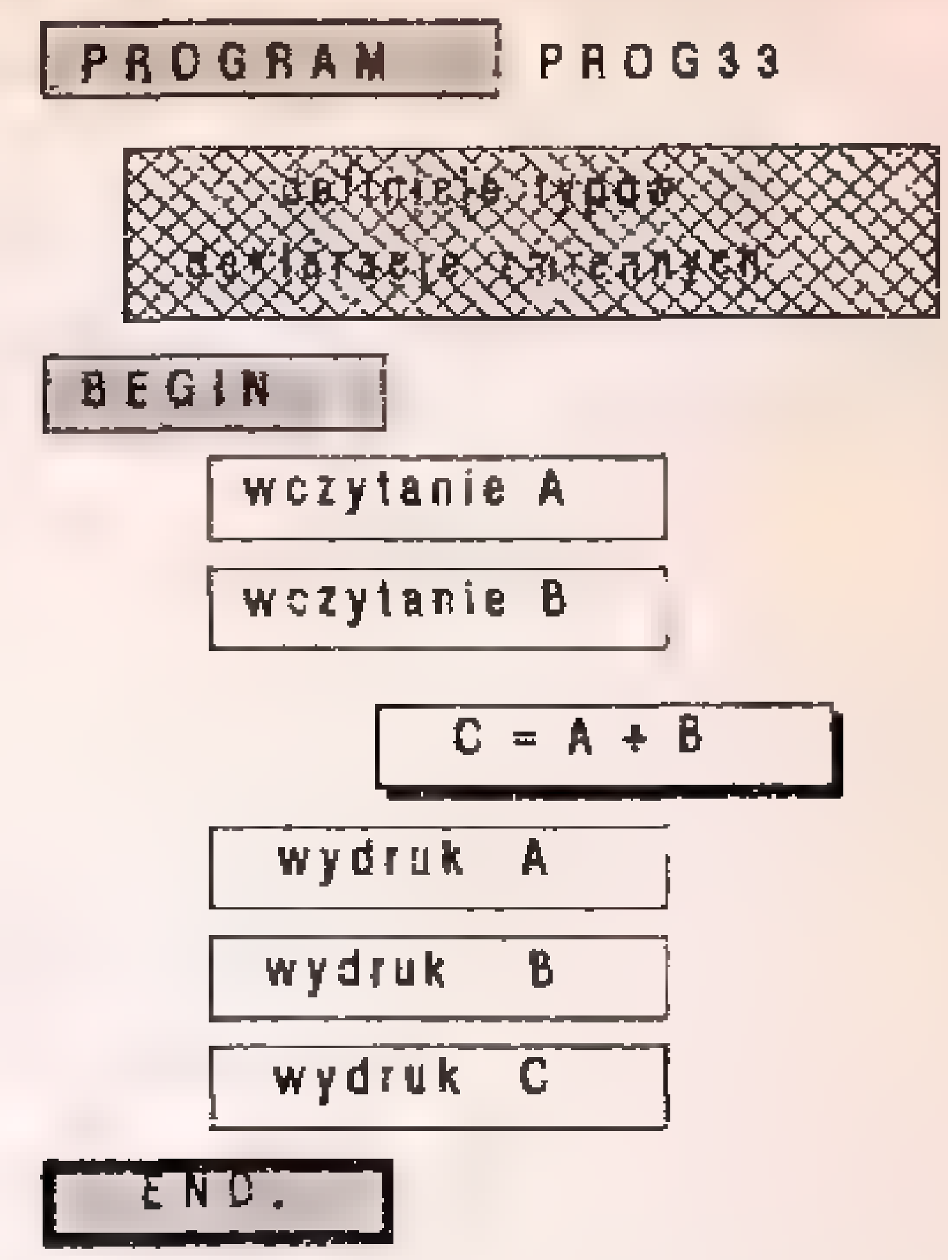
Programy przedstawione w przykładach 35 i 36 należą do najdłuższych spośród dotychczas prezentowanych. Istotny wpływ na ich długość mają powtarzające się sekwencje instrukcji wprowadzania i wyprowadzania elementów macierzy. Strukturę programu z przykładu 35 zilustrowano rysunkiem 13.

Sytuacja taka występuje w wielu programach. Często bowiem w różnych miejscach programu występuje potrzeba wykonywania niemal identycznych sekwencji instrukcji. Dla programisty wygodne jest więc wyodrębnienie takich sekwencji i traktowanie ich jako swego rodzaju wzorców, zwanych w językach programowania **procedurami.** Wzorcem takim może być np. sekwencja wprowadzania elementów macierzy:

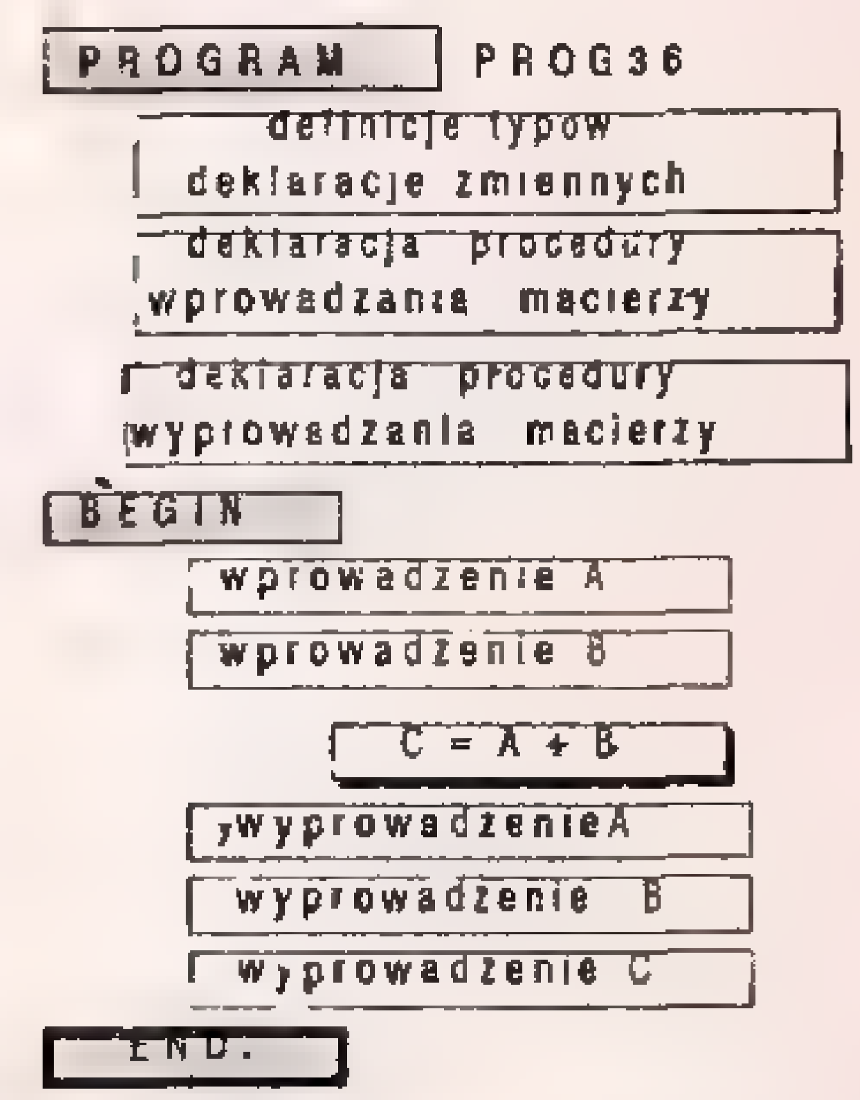
FOR *j*: = *1* **TO** 3 **DO**
BEGIN;
FOR *j*:=*1* **TO** *5* **DO**
READ (*x* [*i*, *j*])];
READLN;
END;

W miejscu wystąpienia nazwy macierzy wprowadzono w niej znak x symbolizujący tę nazwę. Symbol x może być traktowany tak jak traktowany jest argument funkcji, gdyż wynik wykonania procedury jest uwarunkowany nazwą konkretnej macierzy.

W języku PASCAL istnieje możli-



Rys.13.Struktura programu PROG33

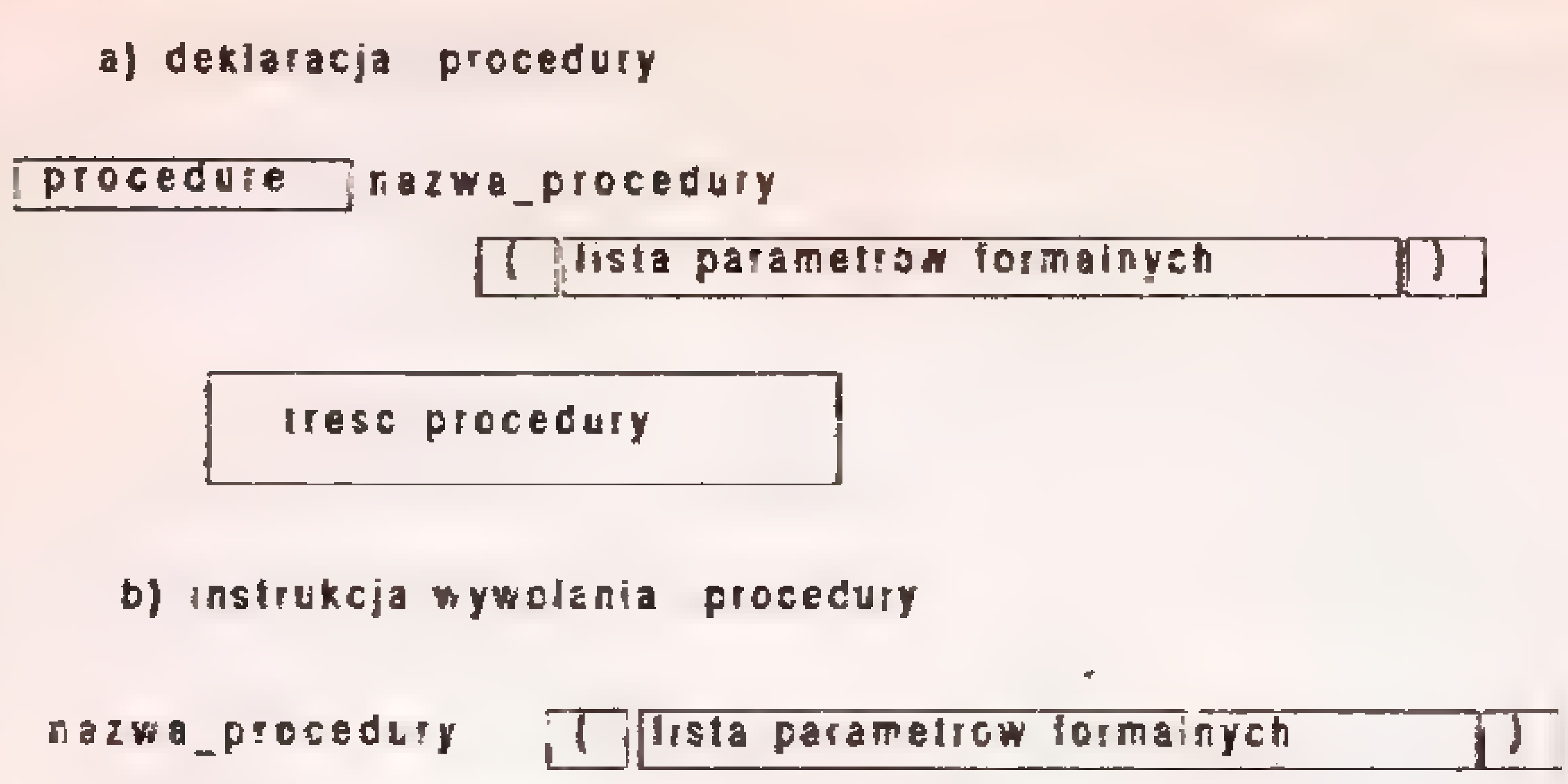


Rys.14.Struktura programu z przykładu 35

wość deklarowania procedur (wzorców sekwencji instrukcji wraz z listą argumentów, zwanych parametrami formalnymi) oraz możliwość wykonywania procedur z konkretnymi wartościami parametrów (zwanych parametrami aktualnymi), drogą wywołania specjalnej instrukcji zwanej instrukcją procedury. Uwzględniając powyższe możliwości struktura programu z rysunku 13 przyjmie postać, jak na rysunku 14.

Z powyższego rysunku wynika, że logika programu nie uległa zmianie. Dodatkowymi jego elementami są deklaracje procedur, zawierające powtarzalne sekwencje instrukcji. Strukturę deklaracji i wywołania procedury zilustrowano rysunkiem 15.

Z rysunku wynika, że treść procedury opatrzona jest nazwą (*nazwa procedury*) wraz z listą parametrów formalnych, a instrukcja procedury zawiera jedynie nazwę procedury (a nie treść) oraz listę parametrów aktualnych. W odniesieniu do wcześniej przedstawionej sekwencji wprowadzania elementów macierzy, deklara-



Rys.15. Struktura deklaracji procedury i instrukcji wywolania procedury

cja odpowiadającej procedury przyjmie postać:

PROCEDURE *czytmac*
(**VAR** *x : macierz*);
VAR *i, j: integer;*
BEGIN
FOR *i:* = *1* **TO** *3* **DO**
BEGIN
FOR *j:* = *1* **TO** *5* **DO**
READ *(x [i, j]);*
READLN*;*
END

a odpowiadająca jej instrukcja wywołania przyjmie postać:
czytmac(a)

dla wprowadzania elementów macierzy a i postać:
czytmac(b)

dla wprowadzania elementów macierzy *b*.

W liście parametrów formalnych procedury *czytmac* zadeklarowano obiekt x typu *macierz* z programu **prog33**.

Należy dodać, że:

a) jeżeli parametr formalny jest poprzedzony słowem kluczowym **VAR**, to wskazuje ono, że wartość parametru aktualnego może ulec zmianie w wyniku wykonania procedury (brak słowa **VAR** zapewnia niezmienność wartości parametru aktualnego, bez względu na treść procedury);
b) treść procedury ma budowę analogiczną do budowy programu i obejmuje:
   - definicje stałych,
   - definicje typów,
   - deklaracje zmiennych,
   - deklaracje procedur,
   - listę instrukcji (zawartą pomiędzy **BEGIN** i **END**);
c) deklaracja procedury zakończona jest znakiem „;", a instrukcja wywołania procedury traktowana jest tak jak każda instrukcja.

Program **prog36** (dodawania macierzy z uwzględnieniem procedur) zamieszczono w przykładzie 38.

Przykład 38

```
PROGRAM prog36;
 TYPE
  macierz=ARRAY[1..3,1..5] OF real;
 VAR
  a,b,          {macierze wejsciowe}
  c:macierz;    {macierz wyjsciowa}
  i,j:integer;  {zmienne pomocnicze}
 PROCEDURE czytmac(VAR x:macierz);
  VAR
   i,j:integer;
  BEGIN
   FOR i:=1 TO 3 DO
    BEGIN
     FOR j:=1 TO 5 DO
      READ(x[i,j]);
     READLN;
    END;
  END;
 PROCEDURE piszmac(VAR x:macierz);
  VAR
   i,j:integer;
  BEGIN
   FOR i:=1 TO 3 DO
    BEGIN
     FOR j:=1 TO 5 DO
      WRITE(' ':3,x[i,j]:10);
     WRITELN;
    END;
  END;
 BEGIN
  WRITELN;
  WRITELN('wprowadz elementy macierzy A');
  czytmac(a);
  WRITELN('wprowadz elementy macierzy B');
  czytmac(b);
  FOR i:=1 TO 3 DO
   FOR j:=1 TO 5 DO
    c[i,j]:=a[i,j]+b[i,j];
  WRITELN;
  WRITELN('dane wejsciowe:');
  WRITELN(' ':10,'macierz A');
  piszmac(a);
  WRITELN(' ':10,'macierz B');
  piszmac(b);
  WRITELN('wynik:');
  WRITELN(' ':10,'macierz C=A+B');
  piszmac(c);
 END.
```

Z przykładu 38 wynika, że celem wyodrębnienia procedur jest skracanie tekstu programu. W rzeczywistości jednak podstawowym celem wyodrębniania procedur jest dekompozycja programu na fragmenty, z których każdy stanowi logicznie zamkniętą całość. Ma to szczególne znaczenie w przypadku projektowania dużych programów, ale efekty konsekwentnego stosowania podziału programu na jednostki logiczne dają się zauważyć w każdym programie. Uzyskuje się bowiem większą przejrzystość programu i znacznie zmniejsza możliwość popełnienia błędu. Ilustracją sposobu dekompozycji jest przykład 39. Zamieszczono w nim program obliczania sumy, różnicy, iloczynu i ilorazu dwóch liczb zespolonych, w którym każde działanie zostało potraktowane jako oddzielna procedura.

Przykład 39

```
PROGRAM prog37;
 TYPE
  liczbazesp=RECORD
              re,       {czesc rzeczywista}
              im:real   {czesc urojona    }
             END;
 VAR
  l1,l2,wyn:liczbazesp;
 PROCEDURE sumazesp(x,y:liczbazesp;
                    VAR z:liczbazesp);
  BEGIN
   z.re:=x.re+y.re;
   z.im:=x.im+y.im;
  END;
 PROCEDURE roznicazesp(x,y:liczbazesp;
                       VAR z:liczbazesp);
  BEGIN
   z.re:=x.re-y.re;
   z.im:=x.im-y.im;
  END;
 PROCEDURE iloczynzesp(x,y:liczbazesp;
                       VAR z:liczbazesp);
  BEGIN
   z.re:=x.re*y.re-x.im*y.im;
   z.im:=x.re*y.im+x.im*y.re;
  END;
 PROCEDURE ilorazzesp(x,y:liczbazesp;
                      VAR z:liczbazesp);
  VAR
   pom:real;
  BEGIN
   pom:=y.re*y.re+y.im*y.im;
   z.re:=(x.re*y.re+x.im*y.im)/pom;
   z.im:=(x.im*y.re-x.re*y.im)/pom;
  END;
  PROCEDURE wypiszzesp(l:liczbazesp);
   BEGIN
    WRITE(' ':5,l.re);
    IF l.im>=0 THEN WRITE(' +');
    writeln(l.im,'i');
   END;
 BEGIN
  WRITELN;
  WRITELN('PODAJ CZESC RZECZYWISTA ',
          'PIERWSZEJ LICZBY');
  READLN(l1.re);
  WRITELN('PODAJ CZESC UROJONA ',
          'PIERWSZEJ LICZBY');
  READLN(l1.im);
  WRITELN('PODAJ CZESC RZECZYWISTA ',
          'DRUGIEJ LICZBY');
  READLN(l2.re);
  WRITELN('PODAJ CZESC UROJONA ',
          'DRUGIEJ LICZBY');
  READ(l2.im);
  WRITELN;
  WRITELN('suma podanych liczb:');
  sumazesp(l1,l2,wyn);
  wypiszzesp(wyn);
  WRITELN('roznica podanych liczb:');
  roznicazesp(l1,l2,wyn);
  wypiszzesp(wyn);
  WRITELN('iloczyn podanych liczb:');
  iloczynzesp(l1,l2,wyn);
  wypiszzesp(wyn);
  IF (l2.re=0) AND (l2.im=0) THEN
   WRITE('iloraz podanych liczb nie ',
         'moze byc wyznaczony')
  ELSE
   BEGIN
    WRITELN('iloraz podanych liczb:');
    ilorazzesp(l1,l2,wyn);
    wypiszzesp(wyn);
   END;
 END.
```

W przypadku bardziej złożonych programów ich teksty mimo stosowania procedur są bardzo obszerne. TURBO PASCAL umożliwia zwiększenie przejrzystości poprzez włączenie we wskazane miejsca fragmentów tekstu programu wcześniej

przygotowanego. Są one wskazywane w specjalnej dyrektywie kompilatora o strukturze:

{*$I nazwa zbioru*}

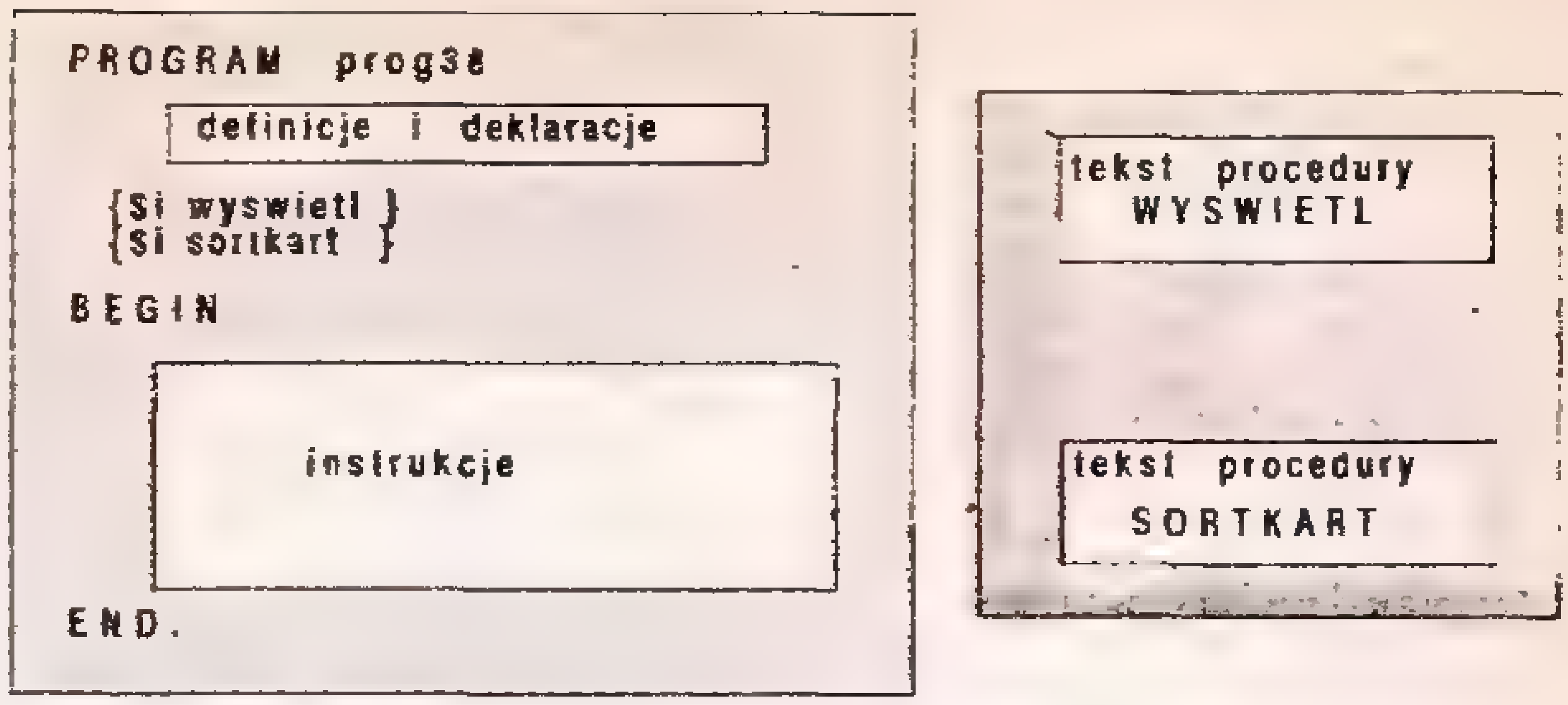
Dyrektywa ta zawiera nazwę zbioru w pamięci dyskowej, w której przechowywany jest odpowiedni fragment tekstu programu. Sposób użytkowania tej przydatnej dyrektywy został zilustrowany przykładem 40. Zamieszczono w nim program wyprowadzający zawartość założonej wcześniej kartoteki (przykład 29) przed i po uporządkowaniu jej elementów wg klucza, którym jest nazwisko autora. Przyjęto, że teksty dwóch wyodrębnionych procedur: wyprowadzania zawartości kartoteki (przykład 41) i porządkowania elementów kartoteki (przykład 42), znajdują się w oddzielnych zbiorach o nazwach: **WYSWIETL.PAS** i **SORTKART.PAS.**

Przykład 40

```
PROGRAM prog38;
 TYPE
  pozbibl=RECORD
           autor:string[35];
           tytul:string[50];
           rokwyd,
           nrkat:integer
          END;
  zbbibl=FILE OF pozbibl;
 VAR
  ksiegozbior:zbbibl;
  nazwazb:string[12];
  zn:string[1];
  {$i wyswietl.pas}
  {$i sortkart.pas}
 BEGIN
  WRITELN;
  WRITELN('PODAJ NAZWE ZBIORU');
  READ(nazwazb);
  ASSIGN(ksiegozbior,nazwazb);
  wyswkart(ksiegozbior);
  WRITELN('CZY WYPROWADZASZ KARTOTEKE',
          ' UPORZADKOWANA (T/N)?');
  READ(zn);
  IF (zn='t') OR (zn='T') THEN
   BEGIN
    sortkart(ksiegozbior);
    wyswkart(ksiegozbior);
   END;
 END.
```

Przykład 41

```
PROCEDURE wyswkart(VAR zb:zbbibl);
 VAR
  i:integer;
  rek:pozbibl;
  lpom:string[1];
 BEGIN
  RESET(zb);
  CLRSCR;
  WHILE NOT EOF(zb) DO
   BEGIN
    READ(zb,rek);
    WITH rek DO
     BEGIN
      WRITELN('AUTOR: ',autor);
      WRITELN('TYTUL: ',tytul);
      WRITELN('ROK WYDANIA: ',rokwyd);
      WRITELN('NUMER KATALOGOWY: ',nrkat);
     END;
    WRITELN('========================',
            '========================');
    WRITELN;
    WRITELN(' ':20,'- VERTE -');
    READ(lpom);
    CLRSCR;
   END;
  CLOSE(zb);
 END;
```



Rys.16.Struktura programu z wlaczonymi zbiorami

Przykład 42

```
PROCEDURE sortkart(VAR zb:zbbibl);
 CONST
  n=100;
 TYPE
  zbpam=ARRAY[1..100] OF pozbibl;
 VAR
  tabpom:zbpam;
  rek:pozbibl;
  i,j,ilop:integer;
 BEGIN
  RESET(zb);
  ilop:=0;
  WHILE NOT EOF(zb) DO
   BEGIN
    ilop:=ilop+1;
    READ(zb,tabpom[ilop]);
   END;
  FOR i:=2 TO ilop DO
   FOR j:=ilop DOWNTO i DO
    IF tabpom[j-1].autor >
       tabpom[j].autor THEN
     BEGIN
      rek:=tabpom[j-1];
      tabpom[j-1]:=tabpom[j];
      tabpom[j]:=rek;
     END;
  RESET(zb);
  FOR i:=1 TO ilop DO
   WRITE(zb,tabpom[i]);
  CLOSE(zb);
 END;
```

Zbiory **WYSWIETL.PAS** oraz **SORTKART.PAS** tworzone są tak, jak gdyby stanowiły oddzielne programy. Dodajmy, że tekst całego programu (tzn. z włączonymi zbiorami) musi mieć strukturę akceptowaną przez kompilator (rys. 16).

W przytoczonych wyżej przykładach logicznie wyodrębnionym fragmentem programu była procedura traktowana jako sekwencja instrukcji. Jej wywołanie jest więc instrukcją. Inne właściwości mają procedury funkcyjne, zwane dalej **funkcjami. Funkcja,** podobnie jak procedura, jest wyodrębnionym fragmentem tekstu programu. Wynikiem jej wywołania jest wartość (typu określonego w nagłówku funkcji) obliczona w treści funkcji. Wywołanie funkcji nie jest więc instrukcją, lecz elementem wyrażenia lub warunku.

Struktura deklaracji funkcji jest następująca:

FUNCTION *nazwa funkcji* (*lista parametrów formalnych*) **:** *typ wyniku* **;** *treść funkcji*

W treści funkcji musi wystąpić co najmniej jedna instrukcja przypisania o strukturze:

nazwa *funkcji* **:** = *wyrażenie*

która wartość przypisywanego wyrażenia pozwala traktować jako wynik funkcji. Jeżeli taka instrukcja nie wystąpi w treści funkcji lub nie zostanie wykonana, to wartość funkcji będzie nieokreślona.

Wywołanie funkcji ma taką samą postać, jak wywołanie procedury.

W przykładzie 43 zamieszczono program obliczania ilości *k*-elementowych kombinacji w zbiorze *n*-elementowym wg wzoru:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Wydzielono tam funkcję obliczającą wartość silni wskazanej liczby.

Przykład 43

```
PROGRAM prog39;
 VAR
  n,k:integer;
  ilkomb:real;
 FUNCTION silnia(x:integer):real;
  VAR
   i:integer;
   pom:real;
  BEGIN
   IF x=0 THEN silnia:=1
   ELSE
    BEGIN
     pom:=1;
     FOR i:=1 TO x DO
      pom:=pom*i;
     silnia:=pom;
    END;
  END;
 BEGIN
  WRITELN;
  WRITELN('podaj N i K');
  READ(n,k);
  ilkomb:=silnia(n)/(silnia(k)*silnia(n-k));
  WRITELN;
  WRITELN('ilosc kombinacji z ',n,' po ',k);
  WRITE('wynosi ',ilkomb);
 END.
```

W przykładzie 43 liczby naturalne deklarowane są jako rzeczywiste. Wynika to z bardzo ograniczonego (do 32767) zakresu liczb typu *integer*.

W kolejnym przykładzie zamieszczono program obliczania wartości wyrażenia:

$$x^n \cdot sinh\,(x)$$

W programie wyodrębniono funkcję podnoszenia liczby *x* do potęgi *n*(n — liczba całkowita) oraz funkcję obliczania *sinh (x)* wg zależności.

$$sinh\,(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots$$

Przykład 44

```
PROGRAM prog40;
 VAR
  n:integer;
  x,wynik:real;
 FUNCTION potega(x:real;n:integer):real;
  VAR
   i:integer;
   pom:real;
  BEGIN
   pom:=1;
   IF n<>0 THEN
    FOR i:=1 TO ABS(n) DO
     pom:=pom*x;
   IF n<0 THEN potega:=1/pom
   ELSE potega:=pom;
  END;
 FUNCTION sh(x:real):real;
  VAR
   n,s,p:real;
  BEGIN
   s:=0;
   p:=x;
   n:=0;
   REPEAT
    s:=s+p;
    n:=n+1;
    p:=p*x*x/(2*n*(2*n+1));
   UNTIL abs(p)<=0.001;
   sh:=s;
  END;
 BEGIN
  WRITELN;
  WRITELN('podaj X i N');
  READ(x,n);
  wynik:=potega(x,n)*sh(x);
  WRITELN;
  WRITELN('obliczona wartosc wyrazenia');
  WRITELN('dla X = ',x,' i N = ',n);
  WRITE('wynosi ',wynik);
 END.
```

Poznany zakres właściwości języka PASCAL umożliwia rozwiązywanie zadań poprzez sekwencyjne wykonywanie instrukcji lub poprzez wielokrotne wykonywanie tych samych instrukcji aż do spełnienia pożądanych warunków.

Konstruując programy główną uwagę należy zwrócić na poprawność logiczną oraz warunki zakończenia programu (brak nieskończonych pętli).

Istnieje szereg problemów, które dają się opisać metodami rekursji. W metodach tych zakłada się wykorzystanie tej samej funkcji (procedury) z innymi argumentami do obliczenia jej wartości. Przykładem może tu być definicja funkcji silni:

$$n! = \begin{cases} 1 & dla\ n = 0 \\ n\,(n-1)! & dla\ n > 0 \end{cases}$$

w której do obliczenia jej wartości dla argumentu *n* wykorzystuje się wartość funkcji dla argumentu n — *1*. Definicję taką można zapisać wykorzystując deklarację funkcji rekurencyjnej języka PASCAL (przykład 45).

Przykład 45

```
FUNCTION silnia(n:integer):integer;
 BEGIN
  IF n=0 THEN
       silnia:=1
  ELSE
   silnia:=n*silnia(n-1);
 END;
```

Szczególną własnością takich funkcji jest umieszczenie wywołania funkcji w jej treści. Podobną regułę można zastosować w stosunku do procedur rekurencyjnych. W przykładzie 46 zamieszczono procedurę porządkowania elementów jednowymiarowej tablicy liczb całkowitych. Porządkowanie odbywa się wg następującej reguły.

1) wyszukuje się element najmniejszy,
2) zamienia się miejscami element najmniejszy z pierwszym (bieżącym) elementem tablicy,
3) kroki 1,2 i 3 są powtarzane z pominięciem pierwszego (bieżącego) elementu tablicy.

W kroku 3 powtarza się więc czynności 1,2 i 3, zawężając pole poszukiwań elementu minimalnego.

Przykład 46

```
PROCEDURE sortuj(m,n:integer;
                 VAR tab:ciag);
 PROCEDURE zmienmin(k,l:integer);
  VAR
   elmin,nrelmin,
   i,pom:integer;
  BEGIN
   nrelmin:=k;
   elmin:=tab[k];
   FOR i:=k+1 TO l DO
    IF tab[i]<elmin THEN
      BEGIN
       nrelmin:=i;
       elmin:=tab[i];
      END;
   pom:=tab[nrelmin];
   tab[nrelmin]:=tab[k];
   tab[k]:=pom;
  END;
 BEGIN
  IF m<>n THEN
   BEGIN
    zmienmin(m,n);
    sortuj(m+1,n,tab);
   END;
 END;
```

Podobnie jak dla funkcji, w deklaracji procedury rekurencyjnej występuje jej wywołanie.

Programowanie z wykorzystaniem rekurencji jest zadaniem trudnym. Niżej zamieszczono dwa przykłady ilustrujące zastosowanie rekurencji do rozwiązywania zagadnień z dziedziny kombinatoryki.

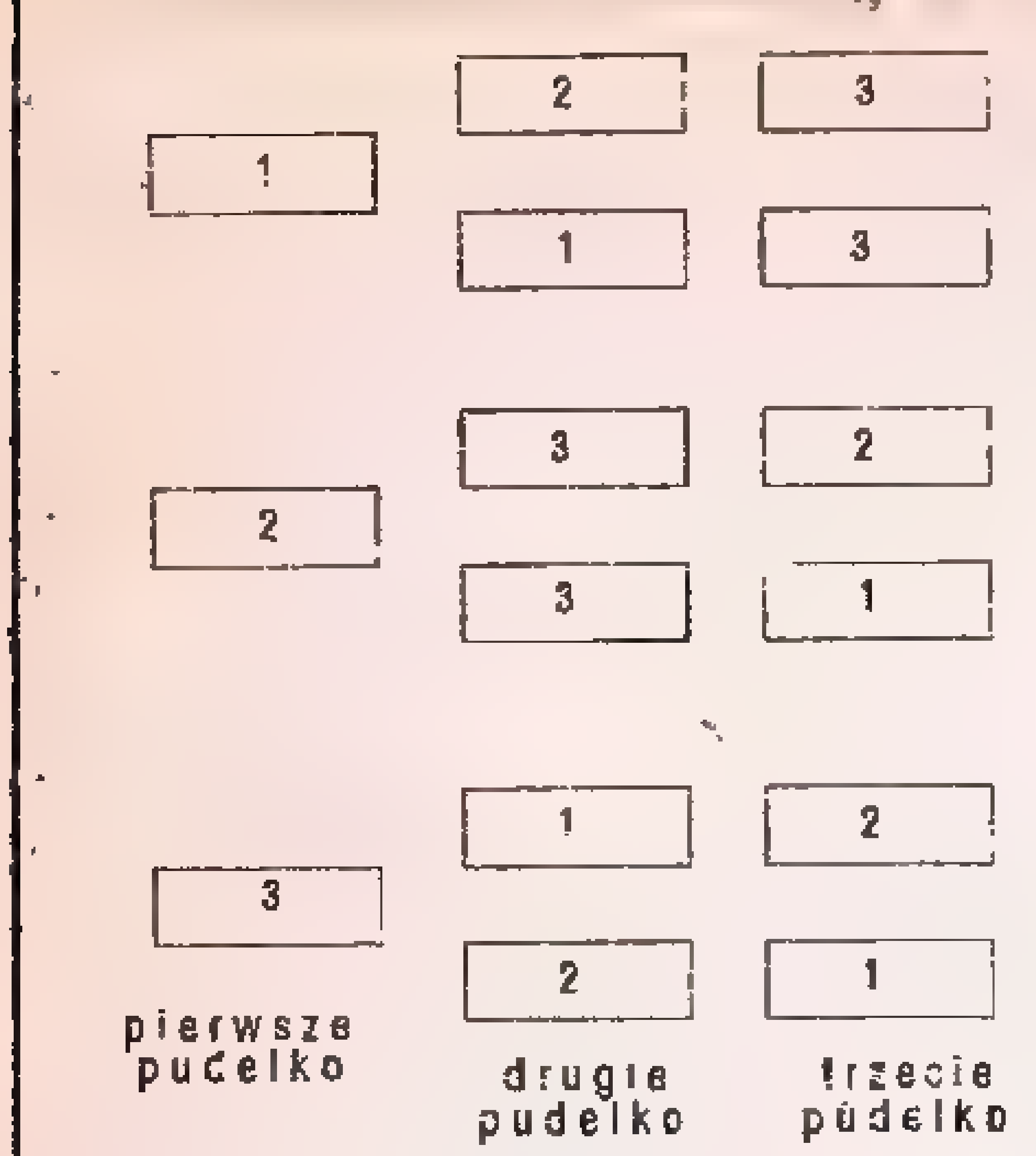
W przykładzie 47 zamieszczono program generowania wszystkich permutacji zbioru $\{1,...,n\}$. Permutacją zbioru n-elementowego nazywamy każdy różnowartościowy ciąg elementów zbioru **X** o długości *n*. W programie wykorzystano metodę generowania permutacji w tzw. porządku antyleksykograficznym, w którym ciąg permutacji zbioru $\{1,...,n\}$ ma następujące własności:

a) w pierwszej permutacji elementy występują w rosnącej kolejności, a w ostatniej — w malejącej, czyli ostatnia permutacja jest odwróceniem pierwszej,

b) podany ciąg permutacji można podzielić na *n* bloków o długości $(n-1)!$. Odpowiadają one malejącym wartościom elementu w ostatniej pozycji. Pierwsze $n-1$ pozycji bloku zawierającego element *p* w ostatniej pozycji określa ciąg permutacji zbioru $\{1,...,n\} \setminus \{p\}$ w porządku antyleksykograficznym.

Przykład 47

```
PROGRAM prog41;
 {$a-}
 CONST
  maxil=10;
 TYPE
  ciag=ARRAY[1..maxil] OF integer;
 VAR
  perm:ciag;
  n,il,i:integer;
 PROCEDURE przestaw(a,b:integer);
  VAR
   elp:integer;
  BEGIN
   elp:=perm[a];
   perm[a]:=perm[b];
   perm[b]:=elp;
  END;
 PROCEDURE drukujperm;
  VAR
   i:integer;
   zn:string[1];
  BEGIN
   IF il>20 THEN
    BEGIN
     READ(zn);
     CLRSCR;
     il:=1;
    END;
   FOR i:=1 TO n DO
    WRITE(perm[i],' ');
   WRITELN;
   il:=il+1;
  END;
 PROCEDURE odwrocenie(m:integer);
  VAR
   i,j:integer;
  BEGIN
   i:=1;
   j:=m;
   WHILE i<j DO
    BEGIN
     przestaw(i,j);
     i:=i+1;
     j:=j-1;
    END;
  END;
 PROCEDURE nowaperm(m:integer);
  VAR
   i:integer;
  BEGIN
   IF m=1 THEN drukujperm
   ELSE
    FOR i:=1 TO m DO
     BEGIN
      nowaperm(m-1);
      IF i<m THEN
       BEGIN
        przestaw(i,m);
        odwrocenie(m-1);
       END;
     END;
  END;
BEGIN
 WRITELN;
 WRITELN('PODAJ WARTOSC n<=',maxil);
 REPEAT
  READ(n);
  IF n>maxil THEN
   WRITELN('NIEPOPRAWNA WARTOSC n -',
           ' PODAJ NOWA WARTOSC n')
 UNTIL n<=maxil;
 CLRSCR;
 il:=1;
 FOR i:=1 TO n DO
  perm[i]:=i;
 nowaperm(n);
END.
```



Rys.17. Ilustracja do przykladu 48

W programie **prog41** wyodrębniono cztery procedury:

— przestaw *(a,b)* — zamiana miejscami elementów a i *b* w tablicy *perm* zawierającej permutację;
— odwrócenie *(m)* — zamiana kolejności *m* pierwszych elementów tablicy *perm;*
— *nowaperm (m)* — rekurencyjne utworzenie kolejnej *m*-elementowej permutacji;
— *drukujperm* — wyświetlenie bieżącej permutacji.

Po wyświetleniu kolejnych 20 permutacji program oczekuje na wciśnięcie klawisza **ENTER** w celu wyświetlenia dalszych permutacji.

W procedurach tych wykorzystano tzw. **zmienną globalną** *perm,* a w procedurze *drukujperm* także zmienną *il,* zadeklarowaną na poziomie programu (poza procedurami). Deklaracje zmiennych globalnych muszą wystąpić przed deklaracjami procedur, w których są wykorzystywane. Występujący w programie zapis **{$a —}** jest niezbędny dla programów rekurencyjnych.

W kolejnym przykładzie zamieszczono program rozwiązujący zadanie wyznaczania wszystkich możliwych rozmieszczeń *k* obiektów w *n* pudełkach, tak aby w kazdym pudełku znalazł się jeden obiekt ($k \leq n$). Zauważmy, że dla $k = n$ zadanie sprowadza się do wyznaczania n-elementowych permutacji. Dla $k < n$ zadanie jest bardziej złożone. Metodę wyznaczania takich rozmieszczeń prześledzimy dla $k = 2$ *i* $n = 3$. Można przyjąć następujący sposób postępowania:

a) spośród wszystkich pudełek wybieramy jedno (zwane dalej pierwszym) i wkładamy do niego pierwszy obiekt.
b) spośród dwóch pozostałych pudełek wybieramy jedno (zwane dalej drugim) i wkładamy do niego drugi obiekt,
c) nie zmieniając pierwszego pudełka, przekładamy drugi obiekt do następnego pudełka.

Wszystkie możliwe rozmieszczenia uzyskamy powtarzając czynności a — c dla każdego z możliwych sposobów wyboru pierwszego pudełka. Numerując pudełka od 1 do 3 i przyjmując zasadę, że spośród wolnych pudełek wybieramy pudełko o najmniejszym numerze, uzyskamy kolejne rozmieszczenia obiektów jak na rysunku 17.

Jeżeli przez **P** oznaczymy zbiór numerów pudełek — $P = \{1,...n\}$ — to nasze zadanie sprowadza się do znalezienia wszystkich ciągów róznowartościowych $< p_1, p_2 ... p_k >$ o wyrazach ze zbioru **P** (są to tzw. k-elementowe wariacje bez powtórzeń ze zbioru n-elementowego). Wyraz p_1 takiego ciągu możemy wybrać na *n* sposobów, wyraz p_2 na $n - 1$ sposobów. Przy ustalonych wyrazach $p_1, p_2 ... p_{i-1}$ możemy jako p_i wybrać dowolny spośród $n - i + 1$ elementów zbioru $\mathbf{P} \setminus \{ p_1, p_2, ... p_{i-1}\}$

Przykład 48

```
PROGRAM prog42;
 {$a-}
 CONST
  nn=9;
  kk=7;
 TYPE
  elem=ARRAY[1..nn] OF char;
  kontr=ARRAY[1..nn] OF boolean;
  sukc=ARRAY[1..kk] OF char;
 VAR
  dowvcr:elem;
  praw:kontr;
  wynik:sukc;
  i,n,k,liczw:integer;
 PROCEDURE wyprowadz;
  VAR
   m:integer;
   zn:char;
  BEGIN
   IF liczw>20 THEN
    BEGIN
     READ(zn);
     clrscr;
     liczw:=2;
    END
   ELSE
    liczw:=liczw+1;
   FOR m:=1 TO k-1 DO
    WRITE(wynik[m]);
   WRITELN(wynik[k]);
  END;
 PROCEDURE wybierz(i,n,k:integer; dowybr:elem;
                    praw:kontr; VAR wynik:sukc);
  VAR
   l:integer;
  BEGIN
   IF i<k THEN
    BEGIN
     FOR l:=1 TO n DO
      IF praw[l] THEN
       BEGIN
        wynik[i]:=dowybr[l];
        praw[l]:=false;
        wybierz(i+1,n,k,dowybr,praw,wynik);
        praw[l]:=true;
       END;
    END
   ELSE
    FOR l:=1 TO n DO
     IF praw[l] THEN
      BEGIN
       wynik[k]:=dowybr[l];
       wyprowadz;
      END;
  END;
BEGIN
 REPEAT
  WRITELN;
  WRITELN('podaj K i N (K<=N)');
  READ(k,n);
 UNTIL k<=n;
 WRITE('podaj elementy zbioru : ');
 FOR i:=1 TO n DO
  BEGIN
   praw[i]:=true;
   READ(dowybr[i]);
  END;
 clrscr;
 WRITELN;
 WRITE(' ',k,'-elementowe wariacje z ',
       n,'-elementowego zbioru:');
 FOR i:=1 TO n DO
  WRITE(' ',dowybr[i]);
 WRITELN;
 WRITELN;
 liczw:=1;
 wybierz(1,n,k,dowybr,praw,wynik);
END.
```

W przykładzie wykorzystano dwa nowe typy zmiennych:

— *char* — którego wartościami są wszystkie reprezentacje znaków oraz kodów sterujących występujących na klawiaturze mikrokomputera; dla naszych celów można uznać, że jest on równoważny deklaracji *string* [1];
— *boolean* — którego wartościami są stałe logiczne *true i false.*

ELEMENTY GRAFIKI KOMPUTEROWEJ

4



Grafika komputerowa jest bardzo ważną dziedziną informatyki rozwijaną w wielu liczących się ośrodkach. Jej celem jest wypracowanie efektywnych programów (standardów) umożliwiających łatwe i szybkie generowanie obrazów figur i brył z uwzględnieniem punktu widzenia i napotykanych przeszkód.

W ramach niniejszego opracowania nie jest możliwe przedstawienie żadnego znanego standardu graficznego, a nawet przedstawienie zasad generowania prostych obrazów. Naszym celem jest jedynie zaprezentowanie koncepcji kreślenia kolorowych obrazów w komputerze AMSTRAD CPC 6128 w oparciu o pakiet graficzny CPC 6128 — Grafik Version 1.0 firmy Heinsoeth Software Munich (wybrane problemy grafiki komputerowej prezentowane są w numerach „IKS-a", w cyklu „Grafika na ekranie").

Ekran monitora dzieli się na dwie odrębnie wysterowane, niezależne od siebie części:

— obszar główny, wykorzystywany do zapisu tekstów i określenia obrazów, zwany dalej kartką;
— pobocze.

Pobocze nie jest wykorzystywane do zapisu tekstów i kreślenia obrazów,

można natomiast zmieniać jego kolor. Kolor pobocza stanowi tło dla kartki. Do zmiany koloru pobocza wykorzystywana jest procedura.

border *(i,j)*

której parametrami są identyfikatory kolorów. Programista dysponuje 27 kolorami, oznaczanymi numerami od 0 do 26. Kolory przyporządkowane poszczególnym numerom zamieszczono w tabeli 2.

W procedurze **border** podaje się numery dwóch kolorów zmienianych kilka razy w ciągu sekundy. Jeżeli oba numery są jednakowe, to efekt migotania jest niedostrzegalny.

Program umożliwiający zmianę koloru pobocza (bez migotania) zamieszczono w przykładzie 49.

Przykład 49

```
PROGRAM prog43;
 VAR
  i:integer;
  ch:char;
  {$i grafik3.inc}
 BEGIN
  FOR i:=0 TO 26 DO
   BEGIN
    border(i,i);
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
   END;
  border(1,1);
 END.
```

Program ten po każdorazowym naciśnięciu dowolnego klawisza zmienia kolor pobocza na odpowiadający kolejnemu numerowi. Do sprawdzenia czy klawisz został wciśnięty wykorzystano funkcję standardową **keypressed,** przyjmującą wartość *true* po wciśnięciu klawisza i *false* po odczytaniu kodu wciśniętego klawisza (**READ** (*kbd,...*). Po wyczerpaniu wszystkich możliwych kolorów, kolor pobocza zmienia się na niebieski, przyjmowany standardowo przez system. Efekt migotania można uzyskać realizując program z przykładu 50, w którym kolejne pary kolorów pobocza są ustalane przez użytkownika.

Przykład 50

```
PROGRAM prog44;
 VAR
  k1,k2:integer;
  ch:char;
  {$i grafik3.inc}
 BEGIN
  REPEAT
   WRITELN('zmiana koloru ramki (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('podaj nr pierwszego koloru');
     READLN(k1);
     WRITELN('podaj nr drugiego koloru');
     READLN(k2);
     border(k1,k2);
    END;
   UNTIL (ch='N') OR (ch='n');
  border(1,1);
 END.
```

Bardziej skomplikowane są zasady wykorzystywania kartki. Można na niej zapisywać teksty **piórem tekstowym** oraz kreślić obrazy **piórem graficznym.** Przed użyciem pióra należy wskazać **kałamarz,** z którego czerpany będzie tusz. W każdym kałamarzu można umieścić jednocześnie — w dwóch odrębnych jego częściach — dwa tusze (różnokolorowe lub tego samego koloru). Jeżeli tusz będzie czerpany z kałamarza, w którym umieszczono różnokolorowe tusze, uzyskamy efekt migotania jak w przypadku pobocza.

Liczba dostępnych kałamarzy uzależniona jest od trybu wykorzystania ekranu:
— 16, w trybie 0 (numerowanych od 0 do 15),
— 4, w trybie 1 (numerowanych od 0 do 3),
— 2, w trybie 2 (numerowanych od 0 do 1).

W miarę wzrostu numeru trybu wykorzystania ekranu rośnie jakość kreślonego obrazu (rozdzielczość) kosztem zmniejszającej się liczby kałamarzy możliwych do jednoczesnego wykorzystania, a tym samym liczby kolorów.

Do napełniania kałamarzy wykorzystuje się procedurę

ink *(nrk, i, j)*

gdzie:
nrk — numer kałamarza (który powinien uwzględnić tryb wykorzystania ekranu),
i,j — numery kolorów tuszu (wg tabeli zamieszczonej na obudowie każdego CPC 6128).

W przykładzie 51 przedstawiono procedurę napełniania kałamarzy wg żądań użytkownika. Procedura ta pozwala na wypełnienie tylko takiej ilości kałamarzy, jaka wynika z trybu wykorzystania ekranu.

Przykład 51

```
PROCEDURE napelnij(tryb:integer);
 VAR
  il,k1,k2,
  i:integer;
  ch:char;
 BEGIN
  IF tryb=0 THEN il:=16
  ELSE
   IF tryb=1 THEN il:=4
   ELSE il:=2;
  i:=0;
  REPEAT
   WRITELN('zmiana tuszu w ');
   WRITELN('kalamarzu nr',i);
   WRITELN('(T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('nr pierwszego koloru');
     READLN(k1);
     WRITELN('nr drugiego koloru');
     READLN(k2);
     ink(i,k1,k2);
    END;
   i:=i+1;
  UNTIL i=il;
 END;
```

Jak już podano, do zapisu tekstów oraz kreślenia obrazów wykorzystywane są dwa różne pióra: tekstowe i graficzne. Z każdym z nich związane są oddzielne procedury.

Ich omawianie rozpoczniemy od procedur związanych z piórem tekstowym.

Użycie pióra tekstowego powinno być poprzedzone wyborem kałamarza, którego tusze będą wykorzystywane do zapisu tekstu. Służy do tego procedura.

pen *(nrk)*

gdzie: *nrk* — numer kałamarza.

Wszystkie litery zapisywane są na tle, którego kolor wybierany jest przez wykonanie procedury

paper *(nrk)*

gdzie: *nrk* — numer kałamarza.

Jeżeli procedury te nie zostaną użyte przed wykorzystaniem pióra to dla tła przydzielany jest standardowo kałamarz 0, a dla pióra — kałamarz 1.

Dla ilustracji działania procedur **paper** i **pen** w przykładzie 52, po ustaleniu trybu wykorzystania ekranu i napełnieniu kałamarzy, następuje wybieranie kolejnych kałamarzy dla tła. W przykładzie 53 — po czynnościach wstępnych — następuje wybieranie kolejnych kałamarzy dla pióra.

Przykład 52

```
PROGRAM prog45;
 VAR
  tryb,ilosc,nrk:integer;
  ch:char;
  {$i grafik3.inc}
  {$i p51}
 BEGIN
  WRITELN('podaj tryb');
  READLN(tryb);
  napelnij(tryb);
  IF tryb=0 THEN ilosc:=16
  ELSE
   IF tryb=1 THEN ilosc:=4
   ELSE ilosc:=2;
  mode(tryb);
  FOR nrk:=0 TO ilosc-1 DO
   BEGIN
    paper(nrk);
    WRITELN;
    WRITELN('01234567890123456789');
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
   END;
  mode(2);
  ink(0,1,1);
  paper(0);
  ink(1,26,26);
  pen(1);
 END.
```

Przykład 53

```
PROGRAM prog46;
 VAR
  tryb,ilosc,nrk:integer;
  ch:char;
  {$i grafik3.inc}
  {$i p51}
 BEGIN
  WRITELN('podaj tryb');
  READLN(tryb);
  napelnij(tryb);
  IF tryb=0 THEN ilosc:=16
  ELSE
   IF tryb=1 THEN ilosc:=4
   ELSE ilosc:=2;
  mode(tryb);
  FOR nrk:=0 TO ilosc-1 DO
   BEGIN
    pen(nrk);
    WRITELN;
    WRITELN('01234567890123456789');
    REPEAT UNTIL keypressed;
    READ(kbd,ch);
   END;
```

```
mode(2);
ink(0,1,1);
paper(0);
ink(1,26,26);
pen(1);
END.
```

W obu podanych przykładach do ustalenia żądanego trybu wykorzystania ekranu zastosowano procedurę:

mode *(nrtryb)*

gdzie

nrtryb — przyjmuje wartości 0,1,2

Należy wyjaśnić, że z trybem wykorzystania ekranu wiąże się również liczba znaków tekstu, które mogą być wyświetlane w jednym wierszu. Wynosi ona:

— 20, dla trybu 0,
— 40, dla trybu 1,
— 80, dla trybu 2.

Przed zakończeniem każdego z programów przywracany jest standardowy tryb wykorzystania ekranu (2), standardowe numery kałamarzy tła (0) i pióra (1) oraz standardowe numery tuszów w tych kałamarzach.

Program z przykładu 54 umożliwia prześledzenie wszystkich przedstawionych właściwości pióra tekstowego.

Przykład 54

```
PROGRAM prog47;
 VAR
  tryb,nrk:integer;
  ch:char;
  {$i grafik3.inc}
  {$i p51}
 BEGIN
  WRITELN('podaj tryb');
  READLN(tryb);
  mode(tryb);
  WRITELN;
  REPEAT
   WRITELN('zmiana tuszu w ');
   WRITELN('kalamarzach (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
     napelnij(tryb);
   WRITELN('zmiana tla (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('podaj nr kalamarza');
     READLN(nrk);
     paper(nrk);
    END;
   WRITELN('zmiana piora (T/N)?');
   READ(kbd,ch);
   IF (ch='T') OR (ch='t') THEN
    BEGIN
     WRITELN('podaj nr kalamarza');
     READLN(nrk);
     pen(nrk);
    END;
  UNTIL ch=' ';
  mode(2);
  ink(0,1,1);
  paper(0);
  ink(1,26,26);
  pen(1);
 END.
```

Z piórem graficznym związanych jest znacznie więcej procedur.

Przed użyciem pióra graficznego niezbędne jest wywołanie bezparametrowej procedury

initgrafik

ustalającej początkowe, standardowe warunki pracy. Po zakończeniu wykorzystywania pióra graficznego należy wywołać bezparametrową procedurę

leavegrafik

Odpowiednikiem procedury *pen* dla pióra graficznego jest procedura

grafpen *(nrk)*

a odpowiednikiem procedury **paper** — procedura

grafpaper *(nrk)*

Zasady wykorzystywania kałamarzy przez pióra graficzne są identyczne, jak dla pióra tekstowego.

Kartka podzielona jest na 400x640 punktów (niezależnie od trybu wykorzystania ekranu), tj. 400 wierszy (numerowanych od 0 do 399) po 640 punktów (numerowanych od 0 do 639) w każdym z nich. Punkt można zakreślić za pomocą procedury.

plot (x,y)

gdzie:
x — nr wiersza,
y — nr punktu w wierszu

Od ostatnio ustalonego położenia pióra można wykreślić odcinek do punktu wskazanego w procedurze

draw (x,y)

gdzie:
x,y — współrzędne punktu, do którego będzie kreślony odcinek.

Dodajmy, że punkt wskazany w draw staje się ostatnio ustalonym położeniem pióra graficznego. W przykładzie 55 zamieszczono program kreślenia prostokąta wyznaczającego ramy kartki.

Przykład 55

```
PROGRAM prog48;
 {$i grafik3.inc}
 BEGIN
  ink(0,0,0);
  ink(1,19,19);
  initgrafik;
  grafpaper(0);
  grafpen(1);
  plot(0,0);
  draw(0,399);
  draw(639,399);
  draw(639,0);
  draw(0,0);
  REPEAT UNTIL keypressed;
  leavegrafik;
  ink(0,1,1);
  ink(1,26,26);
 END.
```

W kolejnym przykładzie zamieszczono prosty program kreślenia prostokąta o współrzędnych wierzchołków (100,100), (100,250), (550,250) (550,100) wraz z przekątnymi, kreślonymi innymi kolorami.

Przykład 56

```
PROGRAM prog49;
 {$i grafik3.inc}
 BEGIN
  initgrafik;
  mode(0);
  ink(2,0,0);
  ink(3,12,26);
  ink(4,16,16);
  ink(5,11,11);
  grafpaper(2);
  clg(2);
  grafpen(5);
  plot(100,100);
  draw(100,250);
  draw(550,250);
  draw(550,100);
  draw(100,100);
  grafpen(3);
  draw(550,250);
  grafpen(4);
  plot(100,250);
  draw(550,100);
  REPEAT UNTIL keypressed;
  leavegrafik;
 END.
```

W programie **prog49** wykorzystano procedurę

clg *(nrk)*

„czyszczącą" kartkę w kolorze ustalonym w ostatnio wykonanej procedurze **grafpaper** i ustawiającą nowy numer kałamarza dla tła pióra graficznego.

Następne przykłady ilustrują sposób kreślenia figur graficznych oraz niektóre problemy związane z projektowaniem bardziej złożonych obrazów (ograniczając się do płaszczyzny).

W przykładzie 57 przedstawiono jeden ze sposobów kreślenia okręgu. Wykorzystano w nim metodę polegającą na kolorowaniu punktów położonych na okręgu w odległości kątowej co jeden stopień.

Przykład 57

```
PROGRAM prog50;
{$i grafik3.inc}
 PROCEDURE okrag(xs,ys,r:integer);
  VAR
   x,y,i:integer;
   wsp:real;
  BEGIN
   wsp:=pi/180;
   FOR i:=0 TO 359 DO
    BEGIN
     x:=round(r*sin(i*wsp));
     y:=round(r*cos(i*wsp));
     plot(xs+x,ys+y);
    END;
  END;
 BEGIN
  initgrafik;
  ink(0,0,0);
  grafpaper(0);
  clg(0);
  grafpen(1);
  okrag(320,200,100);
  REPEAT UNTIL keypressed;
  leavegrafik;
  ink(0,1,1);
 END.
```

Wrażenie obracania się koła uzyskano w programie z przykładu 58 poprzez zmianę kolorów szprych koła. Efekt ten jest namiastką bardzo dynamicznie rozwijającej się problematyki animacji obrazów.

Przykład 58

```
PROGRAM prog51;
{$i grafik3.inc}
 VAR
  i,j:integer;
  ch:char;
  wsp:real;
 PROCEDURE okrag(xs,ys,r:integer);
  VAR
   x,y,i:integer;
   wsp:real;
  BEGIN
   wsp:=pi/180;
   FOR i:=0 TO 359 DO
    BEGIN
     x:=round(r*sin(i*wsp));
     y:=round(r*cos(i*wsp));
     plot(xs+x,ys+y);
    END;
  END;
 BEGIN
  initgrafik;
  mode(1);
  ink(2,0,0);
  grafpaper(2);
  clg(2);
  ink(0,12,12);
  ink(3,10,10);
  grafpen(0);
  okrag(200,200,150);
  wsp:=pi/180;
  j:=0;
  FOR i:=0 TO 29 DO
   BEGIN
    grafpen(3);
    plot(200,200);
    draw(200+round(149*sin(6*j*wsp)),
         200+round(149*cos(6*j*wsp)));
    j:=j+1;
    grafpen(1);
    plot(200,200);
    draw(200+round(149*sin(6*j*wsp)),
         200+round(149*cos(6*j*wsp)));
    j:=j+1;
   END;
  WHILE NOT keypressed DO
   BEGIN
    ink(1,10,10);
    ink(3,26,26);
    FOR i:=1 TO 4000 DO;
    ink(3,10,10);
    ink(1,26,26);
    FOR i:=1 TO 4000 DO;
   END;
  leavegrafik;
  ink(1,26,26);
  ink(0,1,1);
 END.
```

Inny problem to kolorowanie figur. W programie **prog52** z przykładu 59 zilustrowano jeden ze sposobów kolorowania pola prostokąta.

Przykład 59

```
PROGRAM prog52;
 {$i grafik3.inc}
  VAR
   i,nr:integer;
   ch:char;
 BEGIN
  initgrafik;
  mode(0);
  ink(0,0,0);
  grafpaper(0);
  clg(0);
  grafpen(1);
  plot(100,100);
  draw(100,250);
  draw(550,250);
  draw(550,100);
  draw(100,100);
  REPEAT
   gotoxy(0,23);
   WRITE('podaj nr kalamarza  ');
   gotoxy(19,23);
   READ(nr);
   grafpen(nr MOD 16);
   FOR i:=100 TO 250 DO
    BEGIN
     plot(100,i);
     draw(550,i);
    END;
   REPEAT UNTIL keypressed;
   READ(kbd,ch);
  UNTIL ch=' ';
  ink(0,1,1);
  mode(2);
  leavegrafik;
 END.
```

W programie wykorzystano instrukcję

gotoxy *(x,y)*

umożliwiającą ustawienie położenia pióra tekstowego na określonym miejscu ekranu. Ekran widziany jest tu jako tablica znaków o wymiarach 25 wierszy i liczbie kolumn zależnej od trybu wykorzystania ekranu.

Wszędzie tam, gdzie zamieszcza się obrazy obiektów w przestrzeni trójwymiarowej pojawia się problem likwidowania rysunków niewidocznych (przesłoniętych) części obiektów, ścian i krawędzi.

Najprostszą ilustracją tego problemu jest kreślenie nakładających się częściowo prostokątów, przedstawione w przykładzie 60.

Przykład 60

```
PROGRAM prog53;
 {$i grafik3.inc}
 VAR
  x11,y11,x12,y12,
  x21,y21,x22,y22:integer;
 BEGIN
  WRITELN('podaj wspolrzedne dolnego ',
          'prostokata');
  READLN(x11,y11,x12,y12);
  WRITELN('podaj wspolrzedne gornego ',
          'prostokata');
  READLN(x21,y21,x22,y22);
  initgrafik;
  mode(0);
  ink(0,0,0);
  clg(0);
  grafwindow(x11,y11,x12,y12);
  grafpaper(4);
  clg(4);
  grafwindow(x21,y21,x22,y22);
  grafpaper(8);
  clg(8);
  REPEAT UNTIL keypressed;
  mode(2);
  ink(0,1,1);
  leavegrafik;
 END.
```

W programie wykorzystano właściwości tzw. okien graficznych wyznaczanych za pomocą procedury

grafwindow (x_l, y_l, x_p, y_p)

gdzie:

x_l, y_l — współrzędne lewego dolnego wierzchołka okna,

x_p, y_p, — współrzędne prawego górnego wierzchołka okna.

Okno graficzne jest prostokątem. Przyjmuje się, że dla pióra graficznego kartka jest początkowym, standardowym oknem graficznym.

W podanym przykładzie każdy prostokąt reprezentowany jest w postaci okna, a do ich kolorowania wykorzystano znaną już instrukcję **clg.** Dotyczy ona ostatnio określonego okna. Efekt przesłaniania uzyskano sterując kolejnością kolorowania prostokątów. W praktycznie spotykanych zadaniach uzyskanie tego efektu wymaga skomplikowanych obliczeń i należy do najbardziej złożonych problemów grafiki komputerowej.

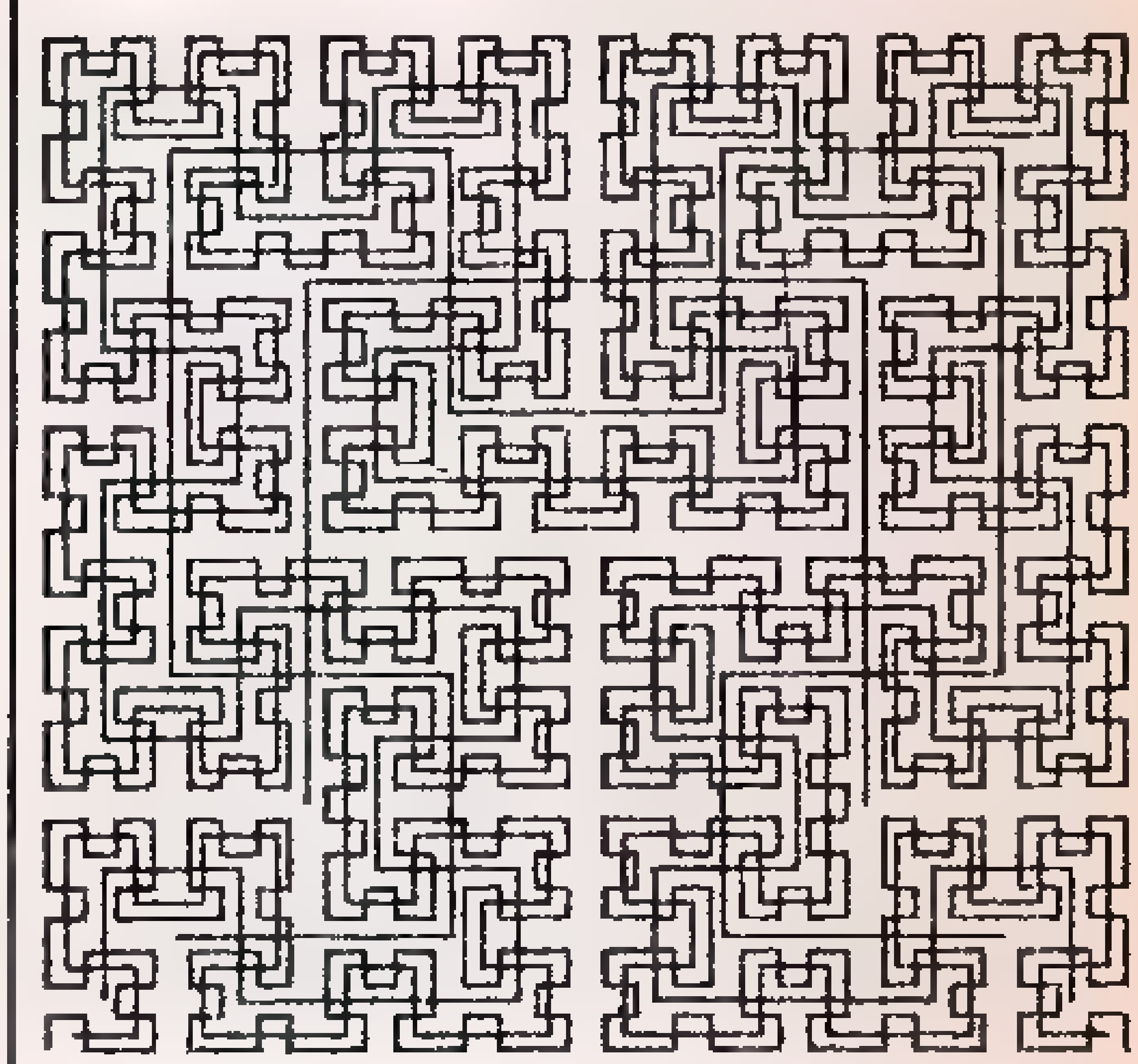
W przykładzie 61 okna graficzne wykorzystano do wyświetlenia kolorów tuszów, którymi standardowo napełnione są poszczególne kałamarze.

Przykład 61

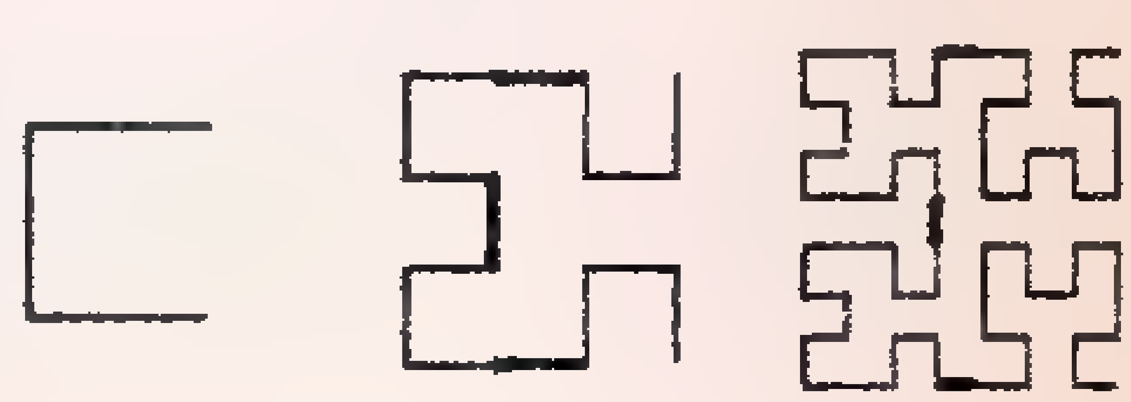
```
PROGRAM prog54;
 {$i grafik3.inc}
VAR
 x1,x2,y1,y2,
 i,j,k:integer;
BEGIN
 x1:=10;
 x2:=60;
 y1:=50;
 y2:=150;
 initgrafik;
 mode(0);
 ink(0,0,0);
 grafpaper(0);
 clg(0);
 k:=1;
 FOR i:=0 TO 1 DO
  FOR j:=0 TO 7 DO
   BEGIN
    grafwindow(x1+j*80,y1+i*125,
               x2+j*80,y2+i*125);
    clg(k);
    k:=k+1;
   END;
 REPEAT UNTIL keypressed;
 ink(0,1,1);
 ink(1,26,26);
 mode(2);
 leavegrafik;
END.
```

Podane przykłady na pewno nie oddają w pełni złożoności problematyki grafiki komputerowej. Ilustrują one jedynie wybrane, najczęściej spotykane problemy.

Na zakończenie zamieszczamy przykład ilustrujący wykorzystanie rekurencji do kreślenia krzywych, zwanych krzywymi Hilberta. (poniżej)



Atrakcyjny układ graficzny pokazany na poniższym rysunku powstał ze złożenia pięciu krzywych. (poniżej)



Procedura **A** jest wywoływana w programie głównym tylko raz dla każdej z nakładanych na siebie krzywych Hilberta. Program główny określa również punkt początkowy krzywej,

tj. początkowe wartości zmiennych x i y oraz przyrost jednostkowy *h*. Przez *h* oznaczono tu całą szerokość strony, na której będą kreślone krzywe.

Ponieważ w języku PASCAL obowiązuje zasada, że pierwsze wystąpienie obiektu powinno być jego definicją, zastosowano tzw. deklaracje wyprzedzające procedur B,C i D. W deklaracji takiej podaje się pełny nagłówek procedury, a blok procedury zastępuje się słowem kluczowym **FORWARD.** W dalszej części programu deklaruje się blok procedury poprzedzony jej identyfikatorem (parametrów nie podaje się).

Przykład 62

```
PROGRAM prog55;
   {program kresli krzywe Hilberta }
   {zadanego rzedu                 }
{$a-}
{$i grafik3.inc}
 CONST
   h0=256;
 VAR
   nrk,i,h,x,y,x0,y0,n:integer;
 PROCEDURE b(i:integer); FORWARD;
 PROCEDURE c(i:integer); FORWARD;
 PROCEDURE d(i:integer); FORWARD;
 PROCEDURE ustawpioro;
  BEGIN
   plot(x,y);
  END;
 PROCEDURE kresl;
  BEGIN
   draw(x,y);
  END;
 PROCEDURE a(i:integer);
  BEGIN
   IF i>0 THEN
    BEGIN
     d(i-1); x:=x-h; kresl;
     a(i-1); y:=y h; kresl;
     a(i-1); x:=x+h; kresl;
     b(i-1)
    END
  END;
 PROCEDURE b;
  BEGIN
   IF i>0 THEN
    BEGIN
     c(i-1); y:=y+h; kresl;
     b(i-1); x:=x+h; kresl;
     b(i-1); y:=y-h; kresl;
     a(i-1)
    END
  END;
 PROCEDURE c;
  BEGIN
   IF i>0 THEN
    BEGIN
     b(i-1); x:=x+h; kresl;
     c(i-1); y:=y+h; kresl;
     c(i-1); x:=x-h; kresl;
     d(i-1)
    END
  END;
 PROCEDURE d;
  BEGIN
   IF i>0 THEN
    BEGIN
     a(i-1); y:=y-h; kresl;
     d(i-1); x:=x-h; kresl;
     d(i-1); y:=y+h; kresl;
     c(i-1)
    END
  END;
BEGIN
 initgrafik;
 i:=0; h:=h0;
 x0:=(h DIV 2)+75;
 y0:=x0+50;
 READ(n);
 mode(0);
 ink(3,0,0);
 grafpaper(3);
 clg(3);
 ink(4,11,11);
 ink(5,7,7);
 ink(6,12,12);
 ink(7,19,19);
 ink(8,16,16);
 nrk:=3;
 REPEAT
  nrk:=nrk+1;
  i:=i+1; h:=h DIV 2;
  x0:=x0+(h DIV 2);
  y0:=y0+(h DIV 2);
  x:=x0; y:=y0;
  grafpen(nrk);
  ustawpioro;
  a(i);
 UNTIL i=n;
 REPEAT UNTIL keypressed;
 mode(2);
 leavegrafik
END.
```

Zakończenie

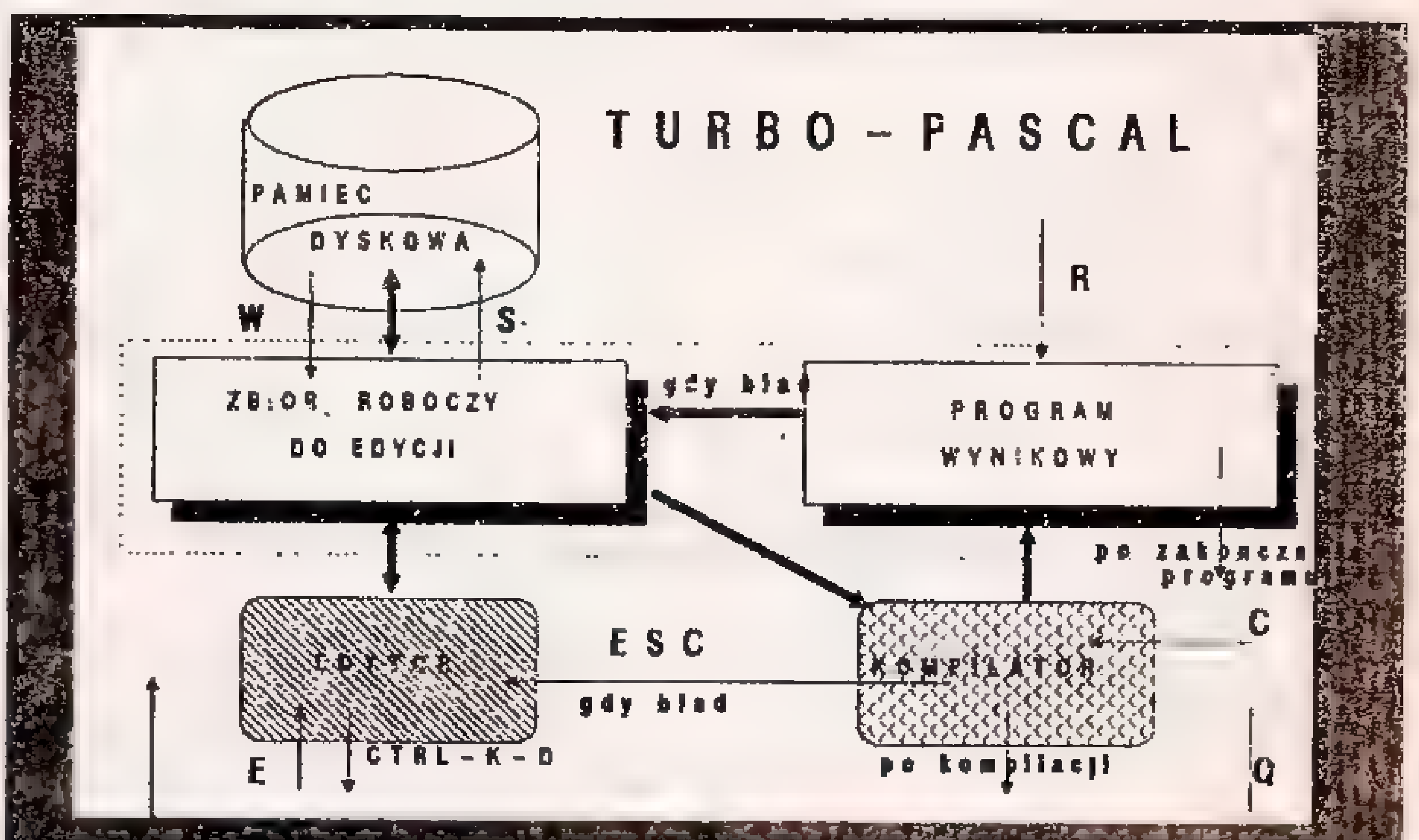
W niniejszym opracowaniu zostały omówione jedynie niektóre właściwości języka PASCAL. Autorzy sądzą, że pozwoli to tworzyć nieskomplikowane programy osobom, które opanowały matematykę na poziomie szkoły średniej, a w wielu przypadkach — tylko podstawowej.

Adam KAPICA
Stefan ROZMUS

Dodatek

TURBO PASCAL jest systemem programowania obejmującym:
— edytor programów źródłowych, umożliwiających przygotowanie tekstu programu,
— kompilator, tłumaczący tekst programu na postać zrozumiałą przez komputer,
— programy pomocnicze, realizujące między innymi uruchamianie programu, składowanie tekstu programu w pamięci dyskowej oraz ładowanie tekstu programu z pamięci dyskowej do systemu.

Strukturę TURBO PASCAL — zilustrowano na rys. 18



Rys. 18. Organizacja systemu TURBO-PASCAL:
⟶ - przepływy danych ⟶ - przepływy sterowania (komendy)

Uwidoczniono tu podstawowe komendy systemu (akcje wywoływane przez użytkownika) przypisując je odpowiednim elementom. Wejście do systemu TURBO PASCAL następuje w wyniku wywołania komendy TURBO. Z poziomu systemu CP/M PLUS (lub CP/M-80 2.2). Powrót do systemu CP/M jest efektem komendy Q wywołanej z poziomu TURBO PASCAL. Na poziomie systemu TURBO PASCAL możliwe jest:
— wejście do edytora (komenda **E**);
— wywołanie kompilatora (komenda **C**, która powinna być podana po uprzednim przygotowaniu tekstu programu);
— uruchomienie programu, po zakończeniu kompilacji (komenda **R**);
— przesłanie tekstu programu do pamięci dyskowej (komenda **S**), lub
— załadowanie tekstu programu z pamięci dyskowej do zbioru.

Wyjście z edytora następuje po jednoczesnym wciśnięciu klawiszy **<CTRL>** i **K** a następnie **D.** Wyjście z kompilatora następuje po zakończeniu kompilacji programu lub po wykryciu błędu. Jeżeli kompilator wykrył błąd w programie źródłowym (sygnalizując go użytkownikowi), po wciśnięciu klawisza **<ESC>** następuje automatyczne wejście do edytora wraz ze wskazaniem miejsca występowania błędu w tekście programu. Wyjście z wykonania programu następuje po jego zakończeniu lub po wykryciu błędu (wykonania programu).

Jak uruchomić pierwszy program?

W tym celu należy wprowadzić z klawiatury komendę TURBO i wcisnąć

klawisz RETURN. Spowoduje to wyświetlenie komunikatu wstępnego:

— komunikat wstępny —

zawierającego pytanie o sposób sygnalizowania błędów kompilacji. Na pytanie to należy dać odpowiedź wciskając klawisz **Y** lub **N**. Wciśnięcie klawisza **N** spowoduje, że błędy kompilacji będą sygnalizowane wyłącznie numerami. Wciśnięcie klawisza **Y** sprawi, że sygnalizacja błędów będzie zawierała również słowne ich opisy. Po wprowadzaniu odpowiedzi (zalecamy odpowiedź **Y**) zostaje wyświetlone tzw. menu główne, zawierające zbiór dostępnych komend systemu TURBO PASCAL:

— menu główne —

Po wejściu do systemu TURBO PASCAL należy wywołać edytor (E). Używając komend edytora wprowadzić program źródłowy do zbioru roboczego, skompilować go (C) i uruchomić (R) lub — jeśli wystąpi błąd — przejść ponownie do edycji (wciśnięcie klawisza **<ESC>**. Ciąg czynności prowadzących do uruchomienia programu można przedstawić w postaci następującego algorytmu:
a) wcisnąć klawisz **W** (żądanie wybrania zbioru roboczego),
b) podać nazwę zbioru roboczego edytora i wcisnąć klawisz **RETURN,**
c) wcisnąć klawisz **E** (wywołanie edytora),
d) wprowadzić z klawiatury wiersz tekstu programu źródłowego (zostanie on wyświetlony na ekranie monitora),
e) wcisnąć klawisz **RETURN,**
f) czynności d) i e) wykonać dla kolejnych wierszy tekstu programu źródłowego,
g) po wprowadzeniu ostatniego wiersza tekstu programu źródłowego wcisnąć jednocześnie klawisze **<CTRL>** i **K,** a następnie wcisnąć klawisz **D** (wyjście z edytora),
h) wcisnąć klawisz **C** (wykonanie kompilacji),
i) jeśli nie wystąpi błąd kompilacji wcisnąć klawisz **R** (uruchomienie programu), a w przypadku przeciwnym wcisnąć klawisz **ESC** (ponowne wywołanie edytora w celu poprawienia programu. Po jego poprawieniu przejść do punktu **h**).

Po wciśnięciu klawisza **E** nastąpi wyświetlenie komunikatu:

Work file name:

po którym należy podać nazwę zbioru (zawierającego tekst programu) w postaci:

nazwa lub *nazwa. typ*

gdzie:
nazwa — ciąg od jednego do ośmiu znaków określających nazwę zbioru,
typ — ciąg od jednego do trzech znaków określających typ zbioru (pominięcie typu jest równoważne przyjęciu typu domyślnego PAS).
W obu przypadkach żaden ze znaków nie powinien być znakiem:

= ? * . , < > : ; [ ] { }

Po podaniu nazwy zbioru (np. FILENAME.PAS) i wciśnięciu klawisza RETURN w pierwszym wierszu, u góry ekranu, zostanie wyświetlony opis stanu edytora w postaci

Line 1 Col i Insert Indent A:FILENAME.PAS

Od tego momentu edytor reaguje tylko na własne komendy, a nie na komendy systemu TURBO PASCAL. W następnych wierszach będą wyświetlane kolejne wersy wprowadzanego tekstu programu źródłowego. Gdy cały ekran zostanie zapełniony, wprowadzenie nowego wiersza następuje po przesunięciu zawartości ekranu (bez opisu stanu edytora) o jeden wiersz w górę. Nie należy się jednak obawiać utraty „znikających" wierszy. Można je bowiem wyświetlić wykorzystując odpowiednie komendy edytora.

Wciśnięcie klawisza C (po wyjściu z edytora, po wprowadzeniu tekstu programu) spowoduje wykonanie kompilacji programu.

Jeżeli zakończy się ona poprawnie, program można uruchomić przez wciśnięcie klawisza R. W razie błędnej kompilacji należy ponownie wywołać edytor (tym razem przez wciśnięcie klawisza **<ESC>** i wykorzystując niżej opisane komendy edytora poprawić program źródłowy. Czynności poprawianie—kompilacja wykonujemy aż do uzyskania bezbłędnej kompilacji.

WYBRANE KOMENDY EDYTORA

Pierwsze próby uruchomienia programu z reguły kończą się niepowodzeniem. Przyczyny są następujące:
— w programie źródłowym wystąpił zapis niezgodny z regułami języka PASCAL (sygnalizowany jest błąd kompilacji),
— podczas wykonania programu wystąpił błąd, np. próba wczytania danej o postaci niedopuszczalnej dla typu zmiennej, której ma być przypisana (sygnalizowany jest błąd wykonania),
— w programie wystąpiły błędy logiczne (program został zrealizowany lecz wyniki są niezgodne z oczekiwanymi).

W każdym z tych przypadków należy wywołać edytor przez wciśnięcie klawisza:
— **<ESC>** — po sygnalizacji błędu kompilacji lub wykonania: na ekranie monitora zostanie wyświetlony program źródłowy, a kursor wskaże miejsce, w którym wystąpił błąd,
— E — gdy stwierdziliśmy błędy logiczne; na ekranie monitora zostanie wyświetlony program źródłowy, a kursor będzie wskazywał pierwszy znak w pierwszej linii programu.

Wykorzystując komendy edytora program można poprawić i uruchomić ponownie. Edytor dostarcza 45 komend, z których opiszemy kilka, naszym zdaniem niezbędnych.

Zapisujemy je symbolicznie następująco

<CTRL> — ZN

Oznacza on, że po przy wciśniętym klawiszu **CONTROL** należy nacisnąć klawisz ze znakiem **„ZN"**.

Poprawienie programu może polegać m.in. na:
— wstawieniu nowych znaków w wierszu,
— skasowaniu określonych znaków w wierszu,
— wstawieniu nowego wiersza do programu,
— skasowaniu określonego wiersza programu.

Edytor ma dwa tryby wstawiania znaków w wierszu:
INSERT oraz OVERWRITE. W trybie INSERT nowy znak zostaje wprowadzony na pozycję kursora, a pozostała część wiersza (od kursora w prawo do końca wiersza) jest przesuwana w prawo. W trybie OVERWRITE nowy znak zostaje wprowadzony na miejsce znaku wskazywanego przez kursor. Zmianę trybu wstawiania znaków powoduje komenda **<CTRL> — V.** Do skasowania znaku znajdującego się przed kursorem wykorzystywany jest klawisz <DEL>. Do przesunięcia kursora można wykorzystać jedną z komend:
<CTRL>—D — przesunięcie kursora w prawo o jeden znak,
<CTRL>—S — przesunięcie kursora w lewo o jeden znak,
<CTRL>—E — przesunięcie kursora o jeden wiersz w górę,
<CTRL>—X — przesunięcie kursora o jeden wiersz w dół,
<CTRL>—W — tzw. przewinięcie w górę (przesunięcie tekstu wyświetlonego na ekranie o jeden wiersz w dół),
<CTRL>—Z — tzw. przewinięcie w dół (przesunięcie tekstu wyświetlonego na ekranie o jeden wiersz w górę).

Wstawienie nowego wiesza umożliwia komenda **<CTRL>—N.** Powoduje ona wstawienie nowego wiersza (za wierszem, w którym znajduje się

kursor). Znaki, od pozycji kursora w prawo do końca wiersza, zostają przeniesione do nowego wiersza. Pozycja kursora się nie zmienia. Do kasowania wiersza, w którym znajduje się kursor, służy komenda <CTRL>—**Y.** Po skasowaniu wiersza tekst znajdujący się poniżej kasowanego wiersza jest przesuwany o jeden wiersz w górę. Kursor przesuwa się na pozycję pierwszego znaku. Przy stosowaniu tej komendy należy zachować ostrożność, gdyż nie ma możliwości odzyskania skasowanego wiersza.

PRZECHOWYWANIE ZBIORÓW ROBOCZYCH

Żądanie utworzenia nowego zbioru roboczego, gdy w pamięci istnieje już jakiś zbiór (np. poddawany edycji), powoduje wyprowadzenie pytania dotyczącego składowania istniejącego zbioru. Odpowiedź twierdząca **(Y)** powoduje w tym przypadku przechowanie zbioru w pamięci dyskowej, przecząca zaś — jego zniszczenie poprzez otworzenie nowego zbioru roboczego. Widzimy zatem, że składowanie zbioru zapobiega jego zniszczeniu. Składowanie programu można uzyskać również przez wykonanie komendy S. Jeżeli istnieje poprzednia wersja zbioru o tej samej nazwie, co składowany zbiór roboczy, to typ zbioru w tej wersji zostanie przemianowany na BAK. Wprowadzając nowy program o dużej ilości wierszy źródłowych warto dokonywać jego składowania po wprowadzeniu kolejnej porcji wierszy. Zapobiegnie to utracie całego wprowadzonego programu w przypadku jakiejkolwiek awarii. Wczytanie przechowanego programu umożliwia znana już komenda W.

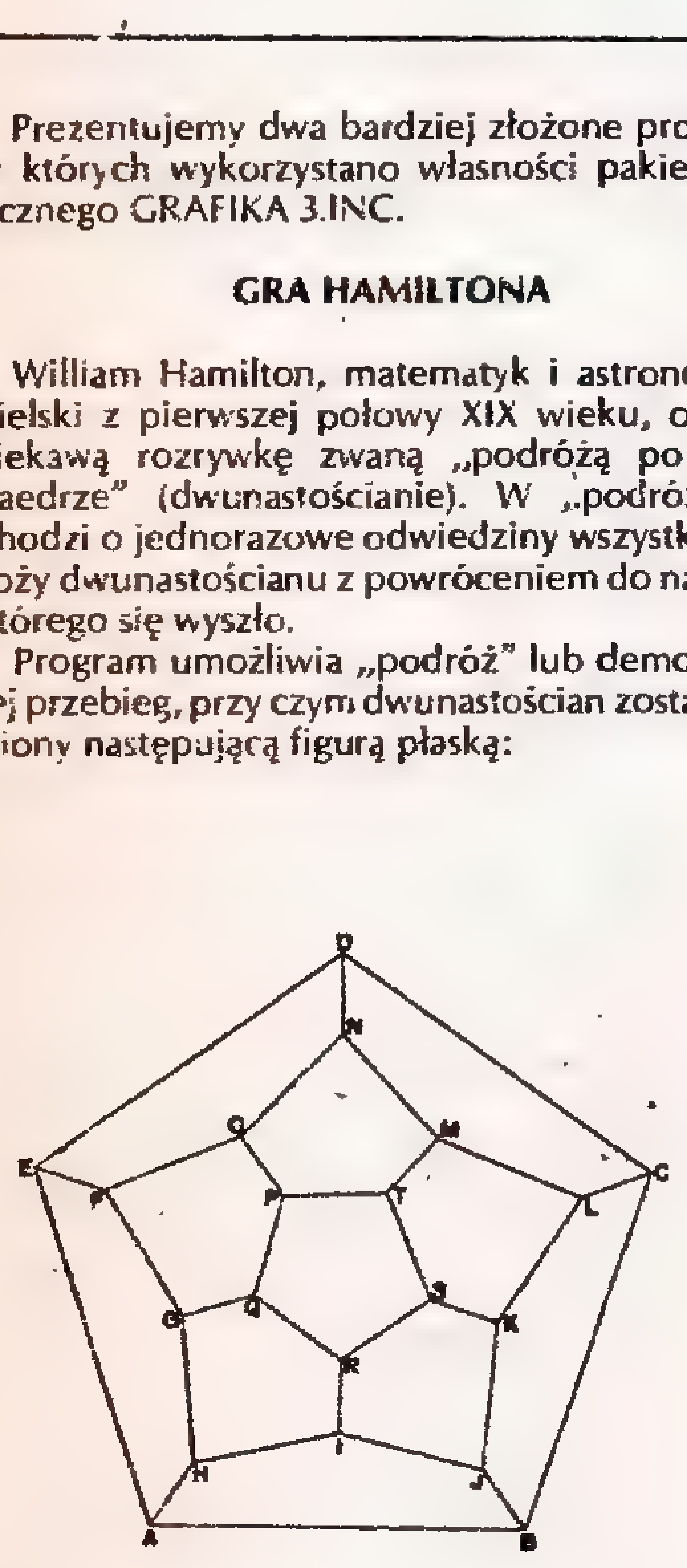
W „Dodatku" omówiono jedynie te właściwości systemu TURBO PASCAL, które są niezbędne do uruchomienia przykładowych programów. Bardziej zaawansowani programiści powinni zapoznać się ze wszystkimi właściwościami systemu, omówionymi w dokumentacji producenta.

Prezentujemy dwa bardziej złożone programy, w których wykorzystano własności pakietu graficznego GRAFIKA 3.INC.

GRA HAMILTONA

William Hamilton, matematyk i astronom angielski z pierwszej połowy XIX wieku, obmyślił ciekawą rozrywkę zwaną „podróżą po dodekaedrze" (dwunastościanie). W „podróży" tej chodzi o jednorazowe odwiedziny wszystkich naroży dwunastościanu z powróceniem do naroża, z którego się wyszło.

Program umożliwia „podróż" lub demonstruje jej przebieg, przy czym dwunastościan został zastąpiony następującą figurą płaską:



Klawisze funkcyjne:

- 0 — wyjście z programu,
- 1 — powrót do „menu",
- 2 — demonstracja gry,
- 3 — gra,
- A÷T — wybór odpowiedniego punktu figury,
- TAB — cofnięcie ruchu,

UWAGA: program kończy się, gdy następuje powrót do punktu początkowego; wygrywa ten, kto przejdzie przez wszystkie punkty zgodnie z podanymi zasadami.

```
PROGRAM grahamiltona;
 {$i grafik3.inc}
 TYPE
  nazwypunktow=(a,b,c,d,e,f,g,h,i,j,k,
                l,m,n,o,p,q,r,s,t);
  opispunktu=RECORD
               x,y:integer;
               wskp:boolean;
               sasiedzi:ARRAY[1..3] OF
                 RECORD
                  poprz,
                  lewy,
                  prawy:nazwypunktow
                 END
             END;
  dodekaedr=ARRAY[nazwypunktow] OF
                          opispunktu;
  lzn=string[40];
 VAR
  fig:dodekaedr;
  ppocz:nazwypunktow;
  ch:char;
  pozlit:ARRAY['A'..'T'] OF
          RECORD
           x,
           y:integer
          END;
  wykgr:boolean;


{*****************************************}
{*   prcedury pomocnicze do manipulacji  *}
{*   zobrazowaniem graficznym            *}
{*                                       *}
 PROCEDURE punkt(nazp:nazwypunktow);
  BEGIN
   plot(fig[nazp].x,fig[nazp].y)
  END;
 PROCEDURE linia(nazp:nazwypunktow);
  BEGIN
   draw(fig[nazp].x,fig[nazp].y)
  END;
 PROCEDURE litera(lit:char);
  BEGIN
   tag;
   plot(pozlit[lit].x,pozlit[lit].y);
```

```
   grafout(lit);
   tagoff
  END;
  PROCEDURE napis(l:lzn;nrlin,
                  nrkol:integer);
   VAR
    i,j,k:integer;
   BEGIN
    i:=nrkol*16-16;
    j:=400-nrlin*16;
    tag;
    FOR k:=1 TO length(l) DO
     BEGIN
      plot(i,j);
      grafout(l[k]);
      i:=i+16
     END;
    tagoff
   END;
{****************************************}
{****************************************}
{*       ustawienie wspolrzednych       *}
{*                                      *}
 PROCEDURE ustawwsp;
  PROCEDURE wspp(nazp:nazwypunktow;
                 xp,yp:integer);
   BEGIN
    WITH fig[nazp] DO
     BEGIN x:=xp; y:=yp; wskp:=false END
   END;
  BEGIN
   wspp(a,200,50);  wspp(b,400,50);
   wspp(c,464,240); wspp(d,300,356);
   wspp(e,136,240); wspp(f,176,226);
   wspp(g,216,162); wspp(h,222,84);
   wspp(i,300,100); wspp(j,378,84);
   wspp(k,384,162); wspp(l,424,226);
   wspp(m,354,258); wspp(n,300,314);
   wspp(o,246,258); wspp(p,270,228);
   wspp(q,254,174); wspp(r,300,140);
   wspp(s,346,174); wspp(t,330,228)
  END;
{****************************************}
{****************************************}
{*     opisy wierzcholkow sasiednich    *}
{*                                      *}
 PROCEDURE ustawsas;
  PROCEDURE sd(nazp,p1,p2,
             p3:nazwypunktow);
   BEGIN
    WITH fig[nazp] DO
     BEGIN
      WITH sasiedzi[1] DO
       BEGIN
        poprz:=p1; lewy:=p2; prawy:=p3
       END;
      WITH sasiedzi[2] DO
       BEGIN
        poprz:=p2; lewy:=p3; prawy:=p1
       END;
      WITH sasiedzi[3] DO
       BEGIN
        poprz:=p3; lewy:=p1; prawy:=p2
       END
     END
   END;
  BEGIN
   sd(a,e,h,b); sd(b,a,j,c); sd(c,b,l,d);
   sd(d,c,n,e); sd(e,d,f,a); sd(f,e,o,g);
   sd(g,f,q,h); sd(h,g,i,a); sd(i,h,r,j);
   sd(j,i,k,b); sd(k,j,s,l); sd(l,k,m,c);
   sd(m,l,t,n); sd(n,m,o,d); sd(o,n,p,f);
   sd(p,o,t,q); sd(q,p,r,g); sd(r,q,s,i);
   sd(s,r,t,k); sd(t,s,p,m)
  END;
{****************************************}
{****************************************}
{*     wypelnienie tablicy opisujacej   *}
{*     polczenie nazw wierzcholkow      *}
{*                                      *}
PROCEDURE nazwyw;
 PROCEDURE wp(l:char;x,y:integer);
  BEGIN
   pozlit[l].x:=x;
   pozlit[l].y:=y
  END;
 BEGIN
  wp('A',195,40);  wp('B',395,40);
  wp('C',475,245); wp('D',295,376);
  wp('E',115,245); wp('F',174,246);
  wp('G',195,160); wp('H',225,78);
  wp('I',295,90);  wp('J',363,78);
  wp('K',392,160); wp('L',420,246);
  wp('M',361,276); wp('N',278,329);
  wp('O',231,276); wp('P',250,230);
  wp('T',317,247); wp('Q',244,164);
  wp('S',350,188); wp('R',307,139)
 END;
{****************************************}

{****************************************}
{*         wyswietlenie figury          *}
{*                                      *}
PPOCEDURE wyswfig;
 VAR
  i,j:nazwypunktow;
  l:char;
 BEGIN
  border(0,0); grafpaper(3);
  clg(3); grafpen(0);
  punkt(a);
  FOR i:=a TO d DO
   linia(succ(i));
  linia(a);
  punkt(f);
  FOR i:=g TO o DO
   linia(i);
  linia(f);
  linia(e);
  j:=h;
  FOR i:=a TO d DO
   BEGIN
    punkt(i);
    linia(j);
    j:=succ(succ(j))
   END;
  punkt(p);
  FOR i:=q TO t DO
   linia(i);
  linia(p);
  linia(o);
  j:=g;
  FOR i:=q TO t DO
   BEGIN
    punkt(i);
    linia(j);
    j:=succ(succ(j))
   END;
  nazwyw;
  grafpen(2);
  FOR l:='A' TO 'T' DO
   litera(l)
 END;

{****************************************}

{****************************************}
{*  pokaz wyznaczania drogi hamiltona   *}
{*                                      *}
```

```
PROCEDURE pokaz;
 VAR
  akt,pop,pom:nazwypunktow;
  i1,i,l1,l2:integer;
 FUNCTION nre:integer;
  VAR zn:boolean;
      i:integer;
  BEGIN
   zn:=false;
   i:=1;
   REPEAT
    WITH fig[akt] DO
      IF sasiedzi[i].poprz=pop THEN
       BEGIN nre:=i; zn:=true END;
    i:=i+1
   UNTIL zn
  END;
 PROCEDURE wlewo;
  BEGIN
   pom:=akt;
   akt:=fig[akt].sasiedzi[nre].lewy;
   pop:=pom;
   linia(akt)
  END;
 PROCEDURE wprawo;
  BEGIN
   pom:=akt;
   akt:=fig[akt].sasiedzi[nre].prawy;
   pop:=pom;
   linia(akt)
  END;
 PROCEDURE opoznij;
  VAR i:integer;
  BEGIN
   i:=1;
   REPEAT
    i:=i+1
   UNTIL i>20000
  END;
 BEGIN
  ustawwsp;
  ustawsas;
  wyswfig;
  opoznij;
  i1:=random(19);
  akt:=a;
  IF i1>0 THEN
   FOR i:=0 TO i1-1 DO
    akt:=succ(akt);
  pop:=fig[akt].sasiedzi[random(2)+1]
       .poprz;
  grafpen(1);
  litera(chr(ord('A')+ord(akt)));
  opoznij;
  punkt(akt);
  FOR l1:=1 TO 2 DO
   BEGIN
    FOR l2:=1 TO 3 DO
     BEGIN wlewo; opoznij END;
    FOR l2:=1 TO 3 DO
     BEGIN wprawo; opoznij END;
    wlewo; opoznij;
    wprawo; opoznij;
    wlewo; opoznij;
    wprawo; opoznij;
   END;
  napis('KONIEC POKAZU',24,1);
  wykgr:=false
 END;
{*****************************************}

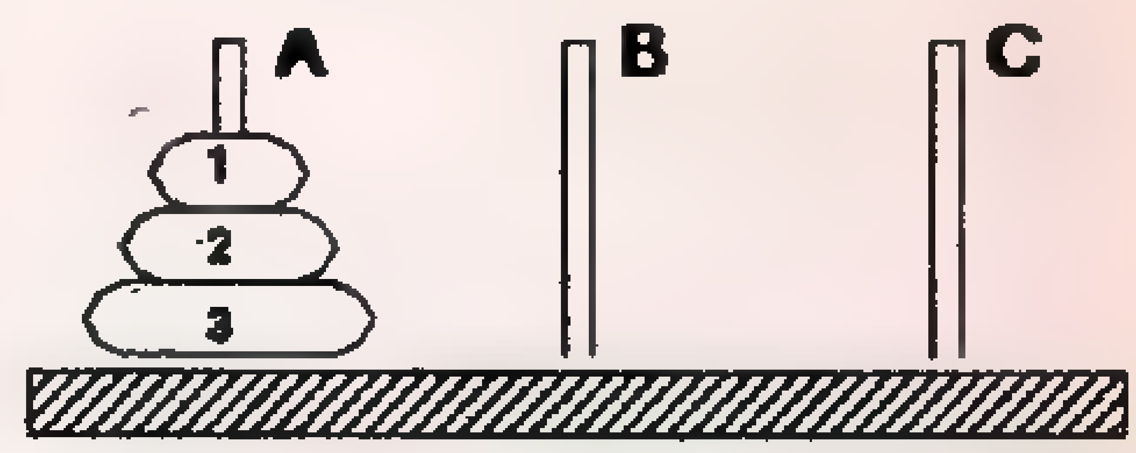
{*****************************************}
{*                 menu                  *}
{*                                       *}
 PROCEDURE menu;
  BEGIN
   border(0,0); grafpaper(3);
   clg(3); grafpen(2);
   napis('GRA HAMILTONA',2,5);
   napis('MENU:',4,5);
   napis('0 - WYJSCIE Z PROGRAMU',6,10);
   napis('1 - POWROT DO MENU',8,10);
   napis('2 - POKAZ',10,10);
   napis('3 - GRA',12,10);
   wykgr:=false
  END;
{*****************************************}

{*****************************************}
{*                  gra                  *}
{*                                       *}
PROCEDURE gra;
 VAR
  koniec:boolean;
  ruchy:ARRAY[1..20] OF nazwypunktow;
  ppocz,pnast:nazwypunktow;
  nrakt:integer;
 FUNCTION nazp(ch:char):nazwypunktow;
  VAR
   i,j:integer;
   p:nazwypunktow;
  BEGIN
   IF (ch>='A') AND (ch<='T') THEN
     i:=ord(ch)-ord('A')
   ELSE
     i:=ord(ch)-ord('a');
   p:=a;
   FOR j:=1 TO i DO
    p:=succ(p);
   nazp:=p
  END;
 PROCEDURE wygaskom;
  VAR
   lpom:lzn;
   i:integer;
  BEGIN
   grafpen(3);
   FOR i:=1 TO 40 DO
    lpom[i]:=' ';
   napis(lpom,24,1);
   grafpen(1)
  END;
 FUNCTION rpoor(pnast:nazwypunktow):
          boolean;
  BEGIN
   WITH fig[pnast] DO
    IF (NOT wsk) AND
     ((sasiedzi[1].poprz=ruchy[nrakt]) OR
      (sasiedzi[2].poprz=ruchy[nrakt]) OR
      (sasiedzi[3].poprz=ruchy[nrakt]))
     THEN rpoor:=true
     ELSE rpoor:=false
  END;
 BEGIN
  ustawwsp;
  ustawsas;
  wyswfig;
  wykgr:=true;
  koniec:=false;
  grafpen(1);
  napis('PODAJ PUNKT POCZATKOWY',24,1);
  REPEAT
   REPEAT UNTIL keypressed;
   READ(kbd,ch);
  UNTIL ((ch>='A') AND (ch<='T')) OR
        ((ch>='a') AND (ch<='t'));
  ppocz:=nazp(ch);
  IF (ch>='a') AND (ch<='t') THEN
             ch:=chr(ord(ch)-32);
  litera(ch);
  nrakt:=1;
  ruchy[1]:=ppocz;
  wygaskom;
  napis('PODAJ NASTEPNY PUNKT',24,1);
  punkt(ppocz);
```

```
WHILE NOT ((ch=#48) OR (ch=#49) OR
  (ch=#50) OR (ch=#51) OR koniec) DO
 BEGIN
  REPEAT UNTIL keypressed;
  READ(kbd,ch);
  IF ch=#8 THEN
   BEGIN
    IF nrakt>1 THEN
    BEGIN
     grafpen(0);
     linia(ruchy[nrakt-1]);
     fig[ruchy[nrakt]].wskp:=false;
     nrakt:=nrakt-1;
     wygaskom;
     napis('PODAJ NASTEPNY PUNKT',24,1);
     punkt(ruchy[nrakt]);
    END
   END
  ELSE
   IF ((ch>='A') AND (ch<='T')) OR
      ((ch>='a') AND (ch<='t')) THEN
      BEGIN
       pnast:=nazp(ch);
       IF rpopr(pnast) THEN
        BEGIN
         linia(pnast);
         IF pnast=ppocz THEN koniec:=true
         ELSE
          BEGIN
           nrakt:=nrakt+1;
           ruchy[nrakt]:=pnast;
           fig[pnast].wskp:=true;
           wygaskom;
           napis('PODAJ NASTEPNY PUNKT',
                  24,1);
           punkt(ruchy[nrakt]);
          END
        END
       ELSE
        BEGIN
         wygaskom;
         napis(
          'ZLY RUCH-PODAJ NASTEPNY PUNKT',
          24,1);
         punkt(ruchy[nrakt])
        END
      END;
  END;
   IF koniec THEN
    BEGIN
     wygaskom;
     IF nrakt=20 THEN
      napis(
       'KONIEC GRY. WYGRALES-GRATULACJE',
       24,1)
     ELSE
      napis(
       'KONIEC GRY. NIESTETY PRZEGRALES',
       24,1);
     wykgr:=false
    END
 END;
{*****************************************}
BEGIN
 crtinit;
 initgrafik;
 mode(1);
 ink(0,26,26);
 ink(1,15,15);
 ink(2,9,9);
 ink(3,0,0);
 menu;
 REPEAT
  IF NOT wykgr THEN
   BEGIN
    REPEAT UNTIL keypressed;
    READ(kbd,ch)
   END;
  CASE ch OF
   #49: menu;
   #50: pokaz;
   #51: gra
  END;
 UNTIL ch=#48;
 mode(2);
 ink(0,1,1);
 ink(1,26,26);
 border(1,1);
 leavegrafik;
END.
```

WIEŻA HANOI

Mamy trzy pręty i n krążków o różnych średnicach. Krążki mogą być nakładane na pręty, tworząc w ten sposób wieżę. Załóżmy, że *n* krążków zostało umieszczonych początkowo na pręcie A w kolejności malejących średnic (na rysunku zilustrowano tę sytuację dla $n = 3$).



Naszym zadaniem jest przesunięcie *n* krążków z prętu A na pręt C z zachowaniem pierwotnego porządku. Zasady przesuwania są następujące:

1) w każdym kroku przesuwa się dokładnie jeden krążek z jednego pręta na inny,
2) krążek nie może być nigdy nałożony na krążek o mniejszej średnicy,
3) pręt B może być użyty jako „magazyn pomocniczy".

Po wprowadzeniu do programu ilości krążków (1—5) program realizuje ich przenoszenie według podanych zasad.

```
PROGRAM wiezahanoi;
 {$a-}
 {$i grafik3.inc}
  CONST
    maxilkr=5; {maksymalna ilosc krazkow}
    py=40;
    podstawa=50;
    sr1=119;
    sr2=299;
    sr3=479;
  TYPE
    znakp='a'..'c'; {nazwa preta}
    wyswiez=1..maxilkr; {maksymalna wysokosc wiezy
                         mierzona iloscia krazkow}
    opiswiez=ARRAY [znakp] OF  {opis wiez}
        RECORD
          w:wyswiez; {wskaznik wierzcholka wiezy}
          wieza:ARRAY [wyswiez] OF wyswiez
        END;
  VAR
    ilkr:integer; {zadana ilosc krazkow}
    start,  {pret, z ktorego przesuwamy krazki}
```

```
    cel,    {pret, na ktory przesuwamy krazki }
    posr:znakp; {pret - magazyn pomocniczy    }
    opw:opiswiez;
ch1,ch2:char;nr:integer;
  PROCEDURE ustawstpocz;
    {procedura ustawiajaca poczatkowy stan wiez}
    VAR
      s:znakp; {zmienne robocze}
      i:wyswiez;
    PROCEDURE pret(x:integer);
      {procedura rysowania preta, na ktory beda}
      {nakladane krazki                         }
     BEGIN
      plot(x-3,podstawa);
      draw(x-3,270);
      plot(x+3,podstawa);
      draw(x+3,270);
     END;
  PROCEDURE krazek(i:byte);
    {procedura rysowania krazka o numerze i}
   VAR
    lin,xl,yl,xp,yp,px:integer;
   BEGIN
    grafpen(i+1);
    yl:=podstawa+(maxilkr-i)*py;
    yp:=yl+py-1;
    px:=(i-1)*10+20;
    xl:=sr1-px;
    xp:=sr1+px;
    FOR lin:=yl TO yp DO
     BEGIN
      plot(xl,lin);
      draw(xp,lin);
     END;
   END;

   BEGIN
    ink(0,0,0);
    ink(1,26,26);
    ink(2,15,15);
    ink(3,17,17);
    ink(4,6,6);
    ink(5,1,1);
    ink(6,21,21);
    grafpaper(0);
    clg(0);
    grafpen(1);
    plot(50,podstawa-1);
    draw(550,podstawa-1);
    plot(50,43);
    draw(550,43);
    plot(sr1-10,20);
    grafout('A');
    pret(sr1);
    plot(sr2-10,20);
    grafout('B');
    pret(sr2);
    plot(sr3-10,20);
    grafout('C');
    pret(sr3);
    WITH opw['a'] DO
     BEGIN
      w:=maxilkr-ilkr+1;
      FOR i:=w TO maxilkr DO
       wieza[i]:=i;
     END;
    opw['b'].w:=0;
    opw['c'].w:=0;
   FOR i:=maxilkr DOWNTO opw['a'].w DO
    krazek(i);
   FOR i:=1 TO 15000 DO
  END;
    PROCEDURE wymkr(nazw:znakp;nrkr:byte;
                    VAR sr,px:integer);
      {procedura ustalajaca wymiary i polozenie}
      {krazka na wiezy                         }
     BEGIN
      CASE nazw OF
       'a':sr:=sr1;
       'b':sr:=sr2;
       'c':sr:=sr3;
      END;
      px:=(nrkr-1)*10+20;
     END;
    FUNCTION przes(start,cel:znakp):integer;
     {funkcja wyznaczajaca wielkosc przesuniecia}
     {krazka w lewo lub w prawo                 }
     BEGIN
      CASE start OF
       'a':IF cel='b' THEN
            przes:=180
           ELSE
            przes:=360;
       'b':przes:=180;
       'c':IF cel='b' THEN
            przes:=180
           ELSE
            przes:=360;
      END;
     END;
    PROCEDURE pgora(start:znakp;nrkr,wskw:byte);
      {procedura przesuwajaca wybrany krazek}
      {z zadanej wiezy w gore              }
     VAR
      sr,px,pocz,kon,
      podst,i:integer;
     BEGIN
      wymkr(start,nrkr,sr,px);
      pocz:=sr-px;
      kon:=sr+px;
      podst:=podstawa+(maxilkr-wskw)*py;
      FOR i:=podst TO 301 DO
       BEGIN
        grafpen(0);
        plot(pocz,i);
        draw(kon,i);
        IF i<=270 THEN
         BEGIN
          grafpen(1);
          plot(sr-3,i);
          draw(sr+3,i);
         END;
        grafpen(nrkr+1);
        plot(pocz,i+py);
        draw(kon,i+py);
       END;
     END;
    PROCEDURE pdol(cel:znakp;nrkr,wskw:byte);
     {procedura przesuwajaca wybrany krazek}
     {w dol na zadana wieze               }
    VAR
     sr,px,pocz,kon,
     podst,i:integer;
    BEGIN
     wymkr(cel,nrkr,sr,px);
     pocz:=sr-px;
     kon:=sr+px;
     podst:=podstawa+(maxilkr-wskw)*py;
     FOR i:=341 DOWNTO podst+py DO
      BEGIN
       grafpen(0);
       plot(pocz,i);
       draw(kon,i);
       IF i<=270 THEN
        BEGIN
         grafpen(1);
         plot(sr-3,i);
         draw(sr+3,i);
        END;
       grafpen(nrkr+1);
       plot(pocz,i-py);
       draw(kon,i-py);
      END;
    END;
   PROCEDURE pprawo(start,cel:znakp;nrkr:byte);
```

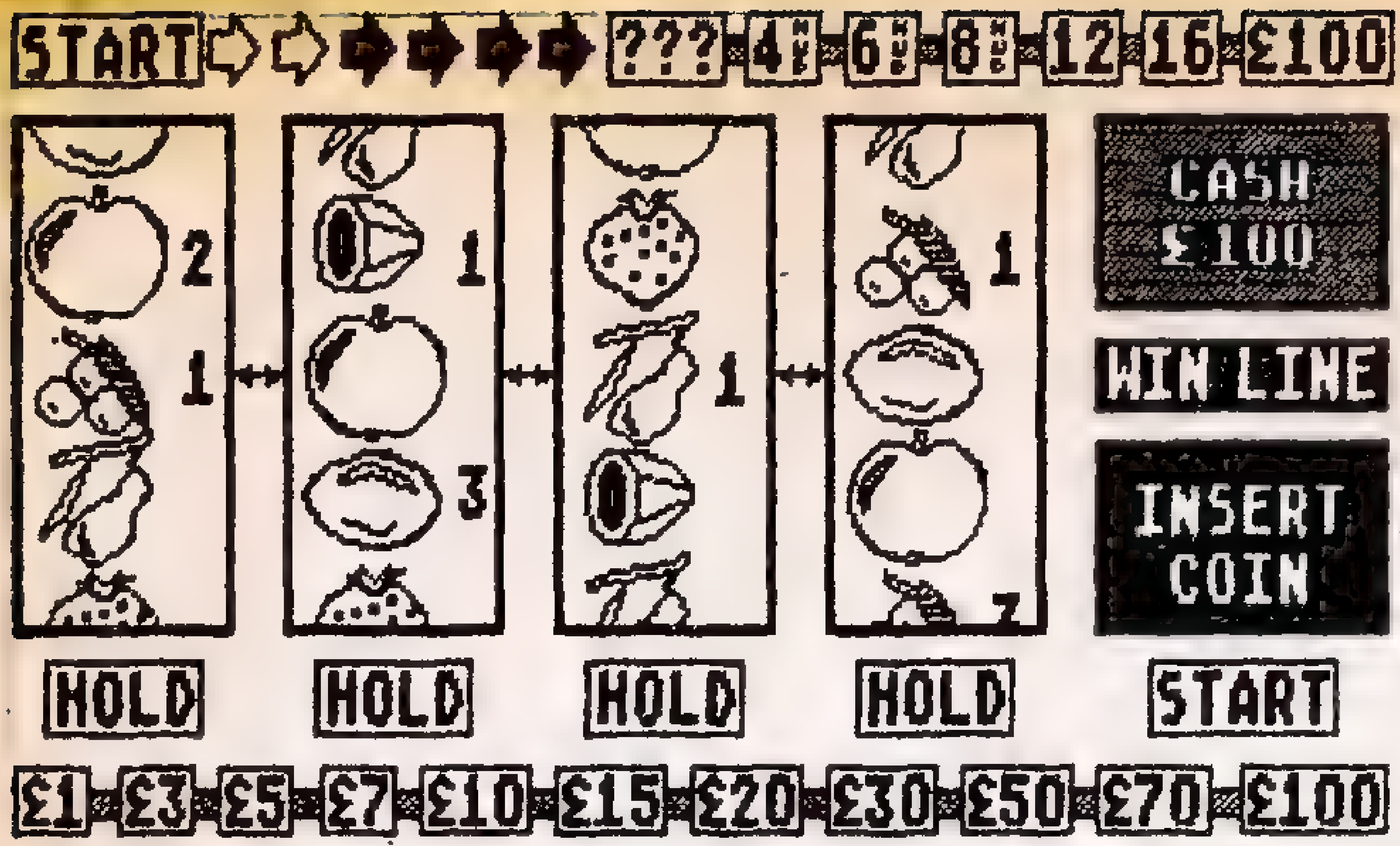
```
    {procedura przesuwajaca wybrany krazek  }
    {w prawo od wiezy 'start' do wiezy 'cel'}
   VAR
    sr,px,pocz,kon,deltax,i:integer;
   BEGIN
    wymkr(start,nrkr,sr,px);
    deltax:=przes(start,cel);
    pocz:=sr-px;
    kon:=sr+px-1;
    FOR i:=0 TO deltax-1 DO
     BEGIN
      grafpen(0);
      plot(pocz+i,302);
      draw(pocz+i,341);
      grafpen(nrkr+1);
      plot(kon+i,302);
      draw(kon+i,341);
     END;
 END;
 PROCEDURE plewo(start,cel:znakp;nrkr:byte);
  {procedura przesuwajaca wybrany krazek  }
  {w prawo od wiezy 'start' do wiezy 'cel'}
 VAR
  sr,px,pocz,kon,deltax,i:integer;
 BEGIN
  wymkr(start,nrkr,sr,px);
  deltax:=przes(start,cel);
  pocz:=sr-px;
  kon:=sr+px-1;
  FOR i:=0 TO deltax-1 DO
   BEGIN
    grafpen(0);
    plot(kon-i+1,302);
    draw(kon-i+1,341);
    grafpen(nrkr+1);
    plot(pocz-i+1,302);
    draw(pocz-i+1,341);
   END;
 END;
PROCEDURE przen(start,cel:znakp);
   {procedura przenoszaca pojedynczy krazek}
   { z preta "start" na pret "cel"         }
  VAR
   nrkr,wskw:byte;
  BEGIN
   wskw:=opw[start].w;
   nrkr:=opw[start].wieza[wskw];
   pgora(start,nrkr,wskw);
   WITH opw[cel] DO
    BEGIN
     IF w=0 THEN  w:=maxilkr
     ELSE  w:=w-1;
     wieza[w]:=nrkr;
    END;
   WITH opw[start] DO
    IF w=maxilkr THEN  w:=0
    ELSE  w:=w+1;
   CASE cel OF
    'a':plewo(start,cel,nrkr);
    'b':IF start='a' THEN
         pprawo(start,cel,nrkr)
        ELSE
         plewo(start,cel,nrkr);
    'c':pprawo(start,cel,nrkr);
   END;
   wskw:=opw[cel].w;
   nrkr:=opw[cel].wieza[wskw];
   pdol(cel,nrkr,wskw);
  END;
 PROCEDURE przel(n:wyswiez; VAR start,posr,cel:znakp);
   {procedura przesuwajaca "n" krazkow z preta
    "start" na pret "cel" z wykorzystaniem preta
    "posr" jako magazynu}
  BEGIN
   IF n=1 THEN przen(start,cel)
   ELSE
    BEGIN
     przel(n-1,start,cel,posr);
     przen(start,cel);
     przel(n-1,posr,start,cel)
    END
  END;
BEGIN
  REPEAT
   WRITELN('podaj ilosc krazkow <= ',maxilkr);
   READLN(ilkr);
  UNTIL (ilkr>0) AND (ilkr<=maxilkr);
  initgrafik;
  mode(0);
  ustawstpocz;
  grafwindow(0,0,639,399);
  start:='a';
  cel:='c';
  posr:='b';
  przel(ilkr,start,posr,cel);
 REPEAT UNTIL keypressed;
 ink(1,26,26);
 grafpen(1);
 ink(0,1,1);
 grafpaper(0);
 clg(0);
 mode(2);
 leavegrafik;
END.
```



	2	3	4
WIŚNIE	3	5	10
CYTRYNA	3	5	10
ŚLIWKA	3	7	20
GRUSZKA	3	7	20
JABŁKO	3	10	30
TRUSKAWKA	3	10	50
ORZECH	3	15	70
CHIP	3	15	100

Hazard jest jednym z tych nałogów, który może bezpośrednio nie odbija się na zdrowiu, lecz pośrednio oddziaływuje bardzo skutecznie. Iluż to ludzi popełniło samobójstwo z powodu pozostawienia swojego całego majątku w kasynie gry. Tobie, oczywiście, to nie grozi z prostej przyczyny, posiadasz przecież komputer. Tym razem daje ci on możność podejmowania ryzykownych decyzji bez negatywnych następstw w razie pomyłki. Wbrew utartym schematom trzeba w tej grze również myśleć. Jako cel postawiono przed grającym zadanie zdobycia żetonów o wartości 250 punktów. Komputer przydziela ci na początku gry 100 punktów. Każdy obrót bębnów kosztuje ciebie żeton o wartości 1 punkta. Po " włożeniu " żetona klawiszem **SELECT**, naciskamy **START**. Wszystkie bębny wystartują, a po chwili kolejno od lewej strony zaczną się zatrzymaywać. Po ich zatrzymaniu się będziesz miał do czynienia z następującymi sytuacjami:

1. Na bębnach na wysokości strzałek są różne owoce. Punktów jest poniżej 7, a komputer reaguje jedynie lampką **INSERT COIN**, polecająca wrzucić żeton. Zrób, to naciskając **SELECT**. Czynność tę możesz powtórzyć kilkakrotnie -zwiększysz w ten sposób swój kredyt w komputerze.



Klawiszem **START** uruchomisz ponownie bębny .

2. Na bębnach są różne owoce . Punktów jest mniej niż 7 . Pod bębnami migają lampki z napisem **HOLD** . Oznacza to , że jeżeli zechcesz , to przed uruchomieniem bębnów możesz niektóre z nich zatrzymać . Dokonujesz tego klawiszami od 1 do 4 . Gdybyś w trakcie wybierania doszedł do wniosku , że zatrzymujesz niewłaściwe bębny , naciśnij **ESC** . Następnie skoryguj swój wybór . Pozostałe czynności jak w p. 1 .

3. Na wysokości strzałek na bębnach patrząc od strony lewej masz dwa , trzy lub cztery takie same owoce . Na dole w zależności od rodzaju i ilości owoców świeci się lampka z ilością punktów . Możesz je zabrać, naciskając **START**. Gdy zechcesz zdobyte punkty postawić i ewentualnie zdobyć ich więcej, naciśnij **OPTION** . Błyskające lampki po bokach , wskazujące wygraną , pokazują , ile możesz zdobyć punktów , jeśli zaryzykujesz (ta po prawej) lub , ile dostaniesz ich w postaci nagrody pocieszenia , gdy ci się nie uda (po lewej). Gdy przegrasz , komputer automatycznie dodaje punkty do twojej kasy , a następnie oczekuje na kontynuację gry . Ilość punktów , jaką możesz zdobyć w zależności od ilości i rodzaju owoców , przedstawiono w tabeli poniżej. Jak z niej widać okruch (CHIP) -w przeciwieństwie do rzeczywistości -jest wart najwięcej .

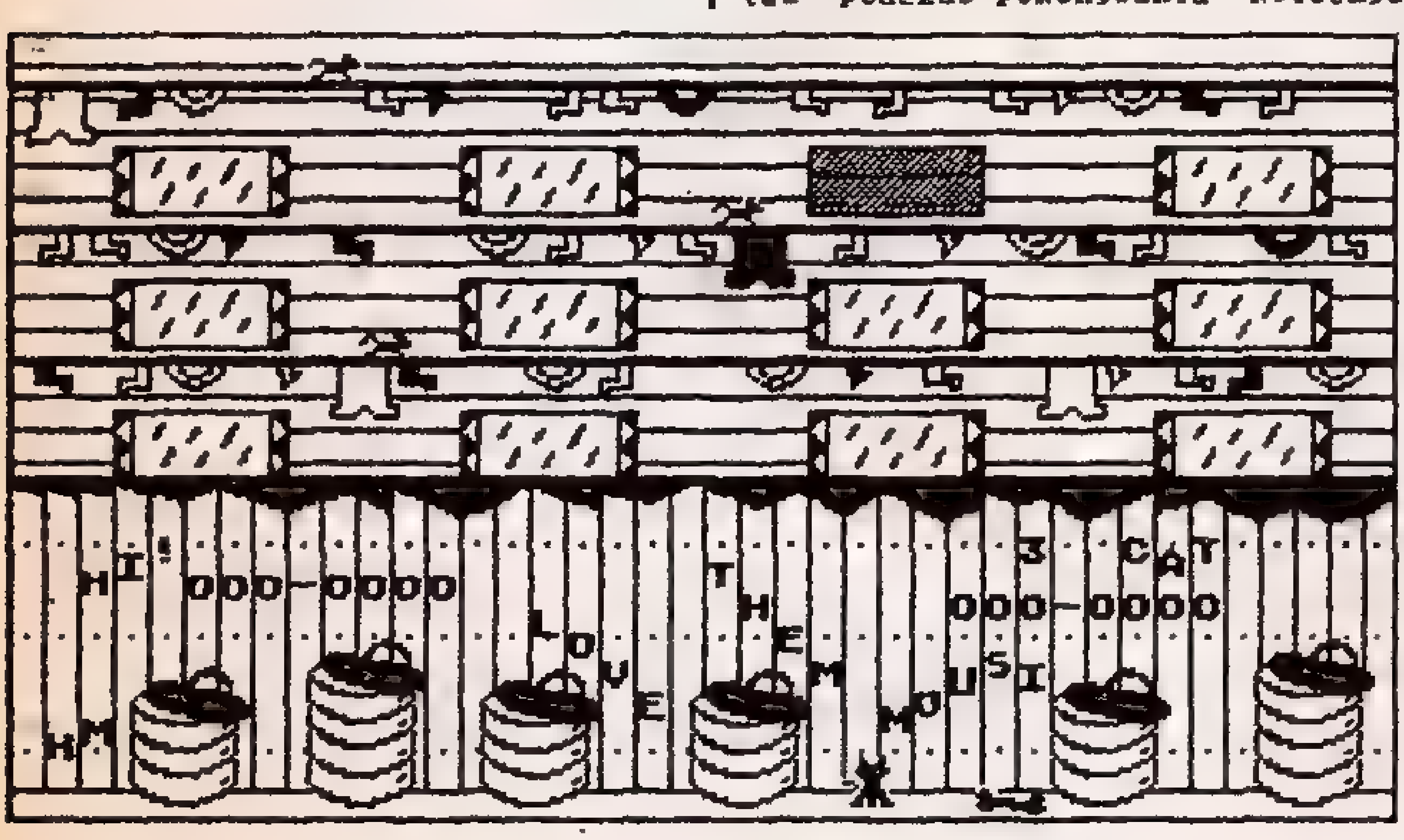
4. Gdy uzyskano na bębnach 7 punktów , komputer automatycznie wykonuje ponowny obrót bębnów . Otrzymasz ślepą wygraną , która ustawi bębny na owoce dające żetony o wartości 3 lub 5 punktów . Suma punktów zliczanych z bębnów jest pokazywana w postaci świecących się strzałek w górnej części ekranu . Punkty te nie są wartością wygranej , choć w istotny sposób wpływają na jej wielkość .

5. W przypadku , gdy na bębnach będzie więcej niż 7 punktów, otrzymujesz premię w postaci nudży . Ilość ich jest zależna od sumy punktów na bębnach , a odczytuje się je z niebiesko świecących wskaźników, znajdujących się w górnej części ekranu . Aby zebrać te nudże w postaci punktów naciśnij **START** , a komputer uruchomi bębny , ustawiając je w możliwie atrakcyjnym położeniu . Jeżeli decydujesz się postawić do dalszej gry zdobyte nudże, przyciśnij **OPTION**. W przypadku gdybyś przegrał , zostaniesz uhonorowany nagrodą pocieszenia . Gdy wygrasz, możesz stawiać dalej aż do uzyskania maksimum , tj. 100 punktów .

Gdy uda ci się zdobyć 250 punktów lub wszystko przegrasz , będziesz mógł grać od nowa po wciśnięciu klawisza **SPACE BAR** lub **HELP**.



Kot jest jednym z tych zwierząt, o którym słyszy się od najmłodszych lat . W przedszkolu z pewnością uczono cię wiersza " Wlazł kotek ..." , a w szkole kazano przeczytać bajkę " Kot w butach " . Nie przejmuj się , tym razem nikt nic nie będzie ci kazał . Sam wcielisz się w kota Feliksa , który będzie musiał wykazać się zwinnością i sprytem podczas pokonywania kolejnych przeszkód . Przed rozpoczęciem gry zdecyduj się , który z czterech stopni trudności wybierasz . Proponujemy pierwszy - zatytułowany KITTEN. Wyboru należy dokonywać klawiszami OPTION i SELECT . Grę uruchamia się klawiszem START lub przez naciśnięcie przycisku joystick'a . Pamiętaj o jednym ; że masz tylko do wykorzystania trzy koty . Te wstrętne myszy będą robiły wszystko aby wyeliminować ciebie z gry , ale nie tylko one. Podczas próby dostania się na płot ze śmietników będzie cię starał zepchnąć biały kot. Jeżeli to mu się uda a w tym czasie przed płotem przebiegnie pies , przepadłeś . Nad płotem są rozwieszone sznurki z bielizną . Możesz po nich skakać i łapać biegające myszy . Za złapaną mysz na najwyższym sznurku otrzymujesz 90 , na środkowym 70 , a na najniższym 50 punktów . Myszy jest dużo, w związku z tym i dużo roboty . Zwróć uwagę na otwierające się co pewien czas okna ; warto w nie wskakiwać . Można zainkasować poważną ilość



punktów , oczywiście nie za darmo . Bo na przykład : w pokoju z serem musisz wyłapać wszystkie myszy , w bibliotece należy wejść na regał i zabrać trzy wazony z kwiatami . W pokoju z akwarium należy wskoczyć do akwarium i zjeść wszystkie ryby. Uwaga na pływające węże elektryczne - kontakt z nimi powoduje śmierć kota . Jeżeli trafisz do pokoju z klatką , zepchnij ją ze stolika ,a następnie złap uwolnionego w ten sposób kanarka . Możesz się również znaleźć w pokoju z miseczkami napełnionymi mlekiem . Masz teraz okazję napić się do syta , lecz pod jednym warunkiem : że będziesz uważał, by inne koty, znajdujące się w tym samym pomieszczeniu, nie zaatakowały ciebie . Zrobią to , gdy zbyt natarczywie będziesz wypijał mleko z ich miseczek .Pamiętaj także , że jako prawdziwy kot, szukasz również Felicji - wybranki twojego serca . Pokazuje się ona czasem w którymś z okien . Wykorzystaj to i wskocz do tego okna - naprawdę warto . Uważaj na braci broniących dostępu do twej miłości oraz strzelające z łuku amorki . Gdy miłość zwycięży otrzymasz pokaźny zastrzyk z punktów oraz dodatkowego kota . Z pewnością to nie wszystkie atrakcje tego programu , chociażby dlatego , że w prawie każdym pokoju będziesz musiał uporać się z pracującą non stop miotłą , jadowitymi pająkami i czasami przypominającym o sobie psie . Grać w tą grę możesz długo , a gdy się zmęczysz , naciśnij SPACE BAR . program zostanie zatrzymany . Aby go kontynuować , naciśnij ponownie ten sam klawisz . Zdobywane przez ciebie punkty wypisywane są na płocie po prawej stronie . Nad nimi podana jest liczba posiadanych jeszcze kotów . Największą ilość zdobytych w grze punktów komputer podaje na płocie po lewej stronie .

INFORMATYKA
KOMPUTERY
SYSTEMY

zeszyty programów komputerowych

nr 2/88

Dodatek „Żołnierza Wolności" Cena 200 zł

NOWOCZESNY KOMPUTER NA!

JĘZYK „C" CZĘŚĆ 1

GRY

DEEP BLUE C

SPIS TREŚCI

1. Wstęp str.3
2. Kompilator DEEP BLUE C, str.3
2.1. Wymagane akcesoria sprzętowo programowe str.3
2.2. Różnice w stosunku do " C " standardowego str.3
2.3. Dyskietka zawierająca kompilator DEEP BLUE C str.4
2.4. Proces kompilacji i konsolidacji str.4
2.5. Rozszerzenie nazw zbiorów str.5
2.6. Tworzenie zbiorów z tekstem źródłowym programu ... str.5
2.7. Kompilacja zbiorów źródłowych.................... str.5
2.8. Łączenie zbiorów półskompilowanych str.5
2.9. Uruchomienie programu wynikowego str.6
2.10. Programy demonstr..... str.6
2.11. Uwagi techniczne str.6
3. Zaczynamy programować str.7
3.1. Podstawowe pojęcia funkje komentarze, instrukcje str.7
3.2. Stałe i zmienne str.8
3.3. Decyzje programowe ... str.9
3.3.1. Instrukcja If str.9
3.3.2. Instrukcja If-else .. str.11
3.3.3. Konstrukcja do-while. str.11
3.3.4. Instrukcja while str.12
4. Operacja na liczbach. str.13
4.1. Piszemy własną funkje str.13
4.2. Program kalkulator .. str.14
4.3. Instrukcja switch ... str.15
4.4. Instrukcja continue . str.15
4.5. Instrukcja break str.16
4.6. Instrukcja for str.16
5. Tablice i wskaźniki . str.17
5.1. Tablice znakowe str.17
5.2. Klasyczne podejście do przetwarzania tablic str.18
5.3. Wskaźniki str.18

PODRĘCZNIK PROGRAMOWANIA

I

OPIS KOMPILATORA JĘZYKA C

dla mikrokomputerów ATARI XL/XE

Język C staje się coraz powszechniej stosowanym narzędziem programowania. Posiada on wiele zalet, takich jak: szczególna przydatność do tworzenia oprogramowania systemowego, możliwość definiowania i przetwarzania nowoczesnych struktur danych oraz prostota i zwartość budowanych programów. Inną jego zaletą, być może najistotniejszą, jest to, że nie jest on związany z konkretnym typem sprzętu i napisane w nim programy mogą być uruchamiane na dowolnym komputerze, dla którego opracowano kompilator tego języka. Intensywny wzrost liczby mikrokomputerów, użytkowanych zarówno przez amatorów jak i poważnych użytkowników, znacznie przybliża możliwość programowania w tym języku. Warto zatem zainteresować się nim bliżej.

Kierując się tym, prezentujemy publikowane po raz pierwszy na łamach popularnych czasopism informatycznych, kompleksowe opracowanie poświęcone językowi C. Składa się ono z dwóch części. Niniejsza, pierwsza część przedstawia pakiet kompilacyjny DEEP BLUE C języka C opracowany dla mikrokomputerów ATARI serii XL/XE. Omówiono w niej zasady posługiwania się programami użytkowymi tego pakietu oraz scharakteryzowano biblioteki funkcji standardowych. Część ta zawiera również obszerny podręcznik programowania, przedstawiający podstawowe elementy i konstrukcje języka ilustrowane licznymi programami przykładowymi. Część druga, która zostanie opublikowana w następnym zeszycie specjalnym IKS-a poświęcona będzie omówieniu obszernej biblioteki funkcji, realizujących arytmetykę zmiennoprzecinkową w kompilatorze DEEP BLUE C oraz zasad ich wykorzystania w tworzonych programach.

Z. Dyjak

H. Krasuski

DODATEK A

A1. Wstęp str.21
A2. Funkcje we/wy zdefiniowane w bibliotece AIO.C. . str.21
A3. Funkcje graficzne zdefiniowane w bibliotece GRAPHICS.C str.24
A4. Funkcje grafiki PMG zdefiniowane w bibliotece PMG.C. str.25
A5. Funkcje zdefiniowane w pliku PRINTF.C. str.25

OPISY GIER KOMPUTEROWYCH

Bilard str.27
Movie Musical Madness str.28
Hacker str.29
The Last Starfighter str.31



1. WSTĘP

Język C - skąd taka dziwna nazwa na określenie języka programowania? Skąd język ten się wywodzi i czym różni się od innych języków programowania? Co za jego pomocą można zrobić lepiej niż przy zastosowaniu innych języków?

Tego typu pytania nurtują każdego, kto po raz pierwszy zetknie się z językiem C. Pytania te są jeszcze bardziej intrygujące, gdy uwzględni się fakt, że prawie każdy użytkownik mikrokomputera pisał użyteczne programy w najbardziej popularnym języku, jakim jest BASIC, a być może również w innych językach wysokopoziomowych jak PASCAL.

Jednak język C uderza każdego swą odmiennością od tych języków.

C jest językiem programowania ogólnego zastosowania, zaprojektowanym w celu wypełnienia luki pomiędzy językiem BASIC a językiem asemblera. Stworzony przez programistów systemowych stanowi wysokopoziomową alternatywę dla języków asemblerowych. Język C jest szybszy i oferuje programiście więcej możliwości niż BASIC, a jednocześnie jest mniej błędotwórczy i bardziej wydajny niż język asemblera.

Kilka słów o historii powstania języka C. Jego twórcami są Dennis M. Ritchie i Brian W. Kernighan. Powstał on w latach siedemdziesiątych jako kolejna modyfikacja języka BCPL, który opracowano na Uniwersytecie w Cambridge. BCPL jest do dzisiaj używany na tej uczelni do nauczania programowania. Język C jest bardzo silnie związany z systemem UNIX.

Pierwsza wersja systemu operacyjnego UNIX została napisana w języku asemblera. W efekcie UNIX mógł być eksploatowany tylko na komputerach określonego typu. Aby uciec od ścisłego przywiązania systemu do określonego typu komputera opracowano specyficzny język, nazwany "B", przeznaczony do przenoszenia systemu UNIX z jednego typu komputera na inny.

Z języka B poddanego pewnym modyfikacjom, dokonanym przez Dennisa M. Ritchie, powstał język C. Ritchie dokonał również przeprogramowania systemu UNIX, stosując nowoopracowany język. W efekcie UNIX może być eksploatowany na dowolnym typie komputera. Również programy napisane w języku C mogą być uruchamiane na dowolnym komputerze, dla którego opracowano kompilator tego języka.

Jednym z kompilatorów C opracowanych dla mikrokomputerów ATARI serii XL/XE jest kompilator DEEP BLUE C. Umożliwia on tworzenie programów o różnym przeznaczeniu (w tym programów graficznych, muzycznych itp.) i różnej wielkości, w tym programów dużych, na które składa się więcej niż sto linii kodu źródłowego.

Programy napisane w języku C dla tych mikrokomputerów, mimo iż działają nieco wolniej niż programy w kodzie maszynowym, to są znacznie łatwiejsze w tworzeniu i testowaniu. Takie elementy języka jak: wskaźniki, funkcje rekurencyjne i wysokopoziomowe struktury sterujące czynią nawet potencjalnie skomplikowany program łatwym do zakodowania i przetestowania. Co więcej, C jest faktycznie językiem oprogramowania systemowego nowej generacji komputerów; programy w nim napisane, w przeciwieństwie do języka asemblera, będą mogły zostać przeniesione na inne komputery (również te, które nie są oparte na mikroprocesorze 6502) kosztem nieznacznych modyfikacji.



2. KOMPILATOR DEEP BLUE C

2.1. Wymagane akcesoria sprzętowo - programowe.

Do poprawnej pracy kompilatora DEEP BLUE C potrzebne są:

- 48 kB pamięci RAM;
- stacja dysków ATARI;
- edytor tekstowy.

Ponadto dla pełnego wykorzystania możliwości oferowanych przez kompilator DEEP BLUE C /DBC/, szczególnie do włączania do programu tworzonego w języku C, procedur w kodzie maszynowym, może okazać się przydatnym ATARI Macro Assembler (MAC65) lub nawet ATARI Assembler Editor.

2.2. Różnice w stosunku do "C" standardowego.

Należy stwierdzić, że język C nie posiada formalnego protokółu, aczkolwiek prace nad jego powstaniem prowadzone są od dłuższego czasu przez ANSI. Jednak we wszystkich publikacjach na jego temat za wersję standardową uznaje się wersję języka zdefiniowaną przez Kernighana i Ritchie'a /K&R/ w pracy "The C programming language" (wydane zostało polskie tłumaczenie tej książki). Niektórzy autorzy za standardowe uważają również pewne rozszerzenia języka C, powstałe przy tworzeniu systemu UNIX.

Z oczywistych powodów kompilator DBC nie posiada zaimplementowanej pełnej wersji języka C, zdefiniowanej przez K&R. W szczególności DBC nie zapewnia realizacji:

1/ struktur i unii;

2/ tablic wielowymiarowych;

3/ liczb zmiennoprzecinkowych;

4/ funkcji zwracających inne wartości niż typu int;

5/ operatora unarnego "sizeof";

6/ operatora binarnego konwersji typów "typecasting".

Ponadto DBC posiada następujące niestandardowe cechy:

1/ ostatni punkt instrukcji "switch", "case" lub "default" musi być zakończony instrukcją "break", "continue" lub "return";

2/ usunięto starą konstrukcję =(op). Zamiast niej należy używać (op)=;

3/ znaki (zmienne typu char) traktowane są jako liczby

bez znaku. Przyjmują one wartości od 0 do 255;

4/ łańcuchy znaków nie mogą być wydłużone do następnej linii logicznej;

5/ długość jednej linii kodu źródłowego nie może przekroczyć 79 znaków;

6/ funkcje mogą mieć maksymalnie 126 argumentów.

DBC zapewnia realizację następujących elementów standardowego "C":

1/ typy danych: char, int, pointer;

2/ tablice jednowymiarowe;

3/ operatory unarne (jednoargumentowe): *, -, &, ++, --, !(wykrzyknik), ~(tylda);

4/ operatory binarne (dwuargumentowe): +, -, *, /, %, |(pionowa kreska), ^, &, ==, !=, <, >, <=, >=, <<, >>, <op>=, &&, ||, przecinek;

5/ trójargumentowy operator warunkowy ?:

6/ instrukcje (słowa kluczowe) języka C: if, else, while, break, continue, return, for, do, switch, case, default;

7/ dyrektywy kompilatora: #define i #include;

8/ relokacyjny linker;

9/ stałe: szesnastkowe, ósemkowe, i typu "backslash".

Standardowy język C wykorzystuje kilka znaków ASCII niedostępnych w komputerach ATARI; i tak: nawiasy klamrowe "{,}" zostały zastąpione kombinacjami znaków $(i $), a znak tyldy (~) zastąpiono parą znaków "$-". Ponieważ znak dolara "$" nie jest wykorzystywany w standardowym C, to można przy pomocy funkcji edytora "znajdź i podmień" zamienić programy napisane w standardowym C na format akceptowany przez DBC.

2.3. Dyskietka zawierająca kompilator DEEP BLUE C.

Przed rozpoczęciem poznawania zasad programowania w języku C warto jest zapoznać się z zawartością dyskietki, na której zapisany jest kompilator DBC, poznać mechanizm kompilacji oraz nauczyć się posługiwania programami realizującymi ten proces.

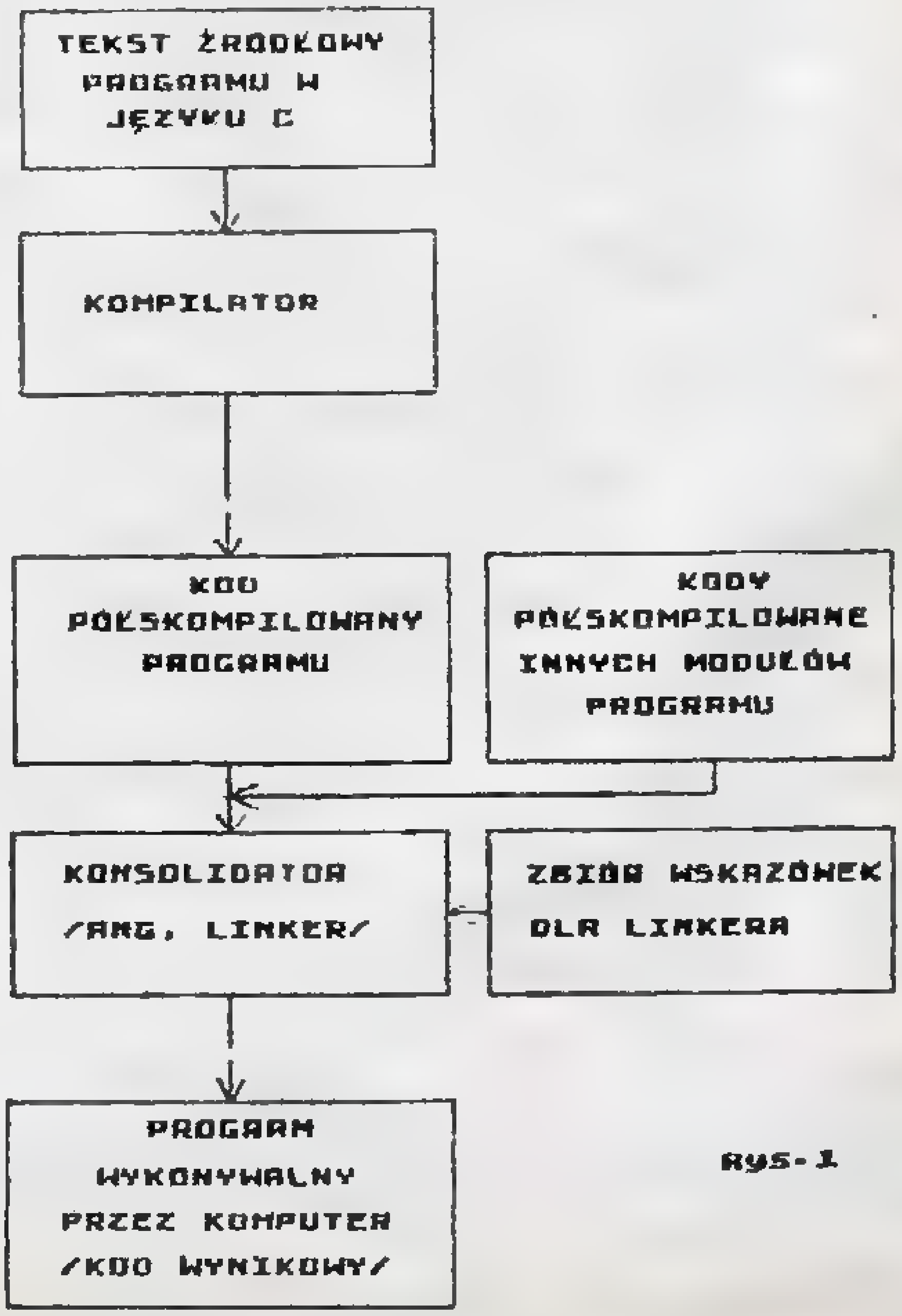
Dyskietka zawierająca kompilator zapisana jest dwustronnie. Na stronie pierwszej zawarte są niżej wymienione zbiory:

1. DOS.SYS - standardowy FMS DOS'a;
2. DUP.SYS - standardowy zbiór zawierający menu DOS'a;
3. CC.COM - kompilator DBC;
4. CLINK.COM - linker DBC;
5. DBC.OBJ - moduł uruchomieniowy DBC;
6. AIO.C - tekst źródłowy funkcji biblioteki we/wy;
7. AIO.CCC - postać półskompilowana funkcji biblioteki AIO;
8. GRAPHICS.C - tekst źródłowy funkcji graficznych i we/wy dla gier;
9. GRAPHICS.CCC - postać półskompilowana biblioteki GRAPHICS.C;
10. PMG.C - tekst źródłowy funkcji realizujących grafikę PMG;
11. PMG.CCC - postać półskompilowana biblioteki PMG;
12. PRINTF.C - tekst źródłowy funkcji wyprowadzania sformatowanego;
13. PRINTF.CCC - postać półskompilowana biblioteki PRINTF;
14. X.C - tekst źródłowy programu demonstracyjnego (patrz opis w dalszej części);
15. X.CCC - postać półskompilowana programu X;
16. X.LNK - zbiór wskazówek dla linkera, wykorzystywany w procesie konsolidacji (linkowania) programu X;
17. X.COM - wykonywalna wersja programu X;
18. BOUNCE.C - tekst źródłowy, postać półskompilowana, itd. programu demonstrującego wykorzystanie funkcji graficznych.

Druga strona dyskietki zawiera bibliotekę funkcji realizujących arytmetykę zmiennoprzecinkową oraz programy demonstrujące sposób ich wykorzystania. Przed rozpoczęciem pracy z kompilatorem należy wykonać roboczą kopię tej dyskietki.

2.4. Proces kompilacji i konsolidacji.

Pisanie programu jest przekazywaniem komputerowi sposobu (algorytmu) postępowania, niezbędnego do rozwiązania określonego problemu, przy czym forma przekazu jest ściśle określona przez język programowania. Język C, podobnie jak każdy inny język programowania posiada ściśle określone reguły syntaktyczne (gramatyczne) i semantyczne (znaczeniowe) - zbliżone do reguł rządzących naturalnym językiem, którym porozumiewają się ludzie. Program napisany w języku C nie może być "zrozumiały" przez komputer w formie takiej, w jakiej został napisany - postać ta nazywana będzie dalej kodem źródłowym programu. Musi on być przetłumaczony przez odpowiedni program do postaci, którą komputer "zrozumie" i wykona. Taki program tłumaczący nazywany jest kompilatorem. Kompilator DBC tłumaczy kod źródłowy programu na postać przejściową (jeszcze nie wykonywalną przez komputer, a potrzebną do przeprowadzenia pełnego procesu kompilacji), zwaną kodem lub programem półskompilowanym. Aby program półskompilowany był wykonywalny przez komputer, musi być najpierw połączony z kodami półskompilowanymi procedur użytkowych, stanowiących część kompilatora, zwaną w języku C - biblioteką kompilatora. Dopiero w wyniku połączenia powstaje kod programu, który może być wykonany przez komputer. Ten



RYS-1

wykonywalny kod nazywany jest kodem wynikowym. Proces łączenia realizowany jest przez odpowiedni program, będący również integralną częścią kompilatora, zwany linkerem. Odpowiednikami nazw "linker" i "linkowanie" są w języku polskim nazwy: "konsolidator" i "proces konsolidacji". Proces konsolidacji, dla programisty przyzwyczajonego do takich języków programowania jak PASCAL czy BASIC, może początkowo wydać się niezrozumiałym i niepotrzebnym. I chociaż jest on dodatkowym przebiegiem w procesie kompilacji programu w języku C, to właśnie dzięki niemu język ten ma wiele zalet w porównaniu z innymi językami programowania. Praktyką stosowaną powszechnie przez programistów piszących programy w "C" jest tworzenie programu "po kawałku", oddzielne kompilowanie każdego fragmentu programu i na końcu łączenie takich "kawałków" w jeden program. Każdy napisany, skompilowany i przetestowany fragment jakiegoś programu może być w postaci półskompilowanej dołączony do bibliotek kompilatora i wykorzystany w innym programie, bez konieczności tworzenia go od początku. Dzięki takiej technice proces tworzenia programów w języku C jest bardzo wygodny i znacznie szybszy niż w innych językach wysokopoziomowych. Proces kompilacji programu w języku C jest przedstawiony na rys. 1.

Reasumując: aby przekształcić kod źródłowy programu w języku C, napisany na papierze, na wykonywalny przez komputer kod wynikowy, należy wykonać niżej przedstawione czynności:

1. Przy pomocy edytora tekstowego przygotować program jako jeden lub kilka zbiorów z tekstem źródłowym;

2. Przekształcić zbiory źródłowe w zbiory z kodem półskompilowanym poprzez skompilowanie tych pierwszych przy pomocy programu CC.COM kompilatora DBC;

3. Stworzyć zbiór wskazówek dla linkera. Zbiór ten powinien zawierać nazwy wszystkich zbiorów z kodem półskompilowanym, które są częściami programu i jest wykorzystywany przez linker (w kroku 4) do połączenia razem wszystkich części programu;

4. Przy pomocy linkera CLINK.COM połączyć wszystkie zbiory, zawierające kod półskompilowany w kompletny program wynikowy.

2.5. Rozszerzenia nazw zbiorów.

Każda nazwa zbioru, wykorzystywanego w wyżej opisanym przebiegu musi mieć swoje własne rozszerzenie, wymagane przez kompilator DBC. Rozszerzenia te są następujące:

.C – zbiór zawierający tekst źródłowy programu;

.CCC – zbiór zawierający postać półskompilowaną programu;

.LNK – zbiór zawierający informacje dla linkera, wskazujące nazwy zbiorów, które mają być łączone;

.COM – zbiór zawierający wykonywalny kod wynikowy programu.

2.6. Tworzenie zbiorów z tekstem źródłowym programu.

Zbiory z tekstem źródłowym programu (zbiory źródłowe dla kompilatora) najlepiej przygotowuje się przy pomocy edytora tekstowego. Do edytowania programów w języku C doskonale nadają się: ATARI Program Text Editor (dołączony do Macro Assemblera), edytor kompilatora KYAN PASCAL lub każdy inny edytor tekstów, jak ATARI WRITER czy Speedscript. Każdy inny, nie wymieniony wyżej edytor tekstowy można z powodzeniem wykorzystać do edycji programów w języku C. Ważne jest, aby stosowany edytor nie wstawiał kolejnego numeru linii na początku każdego wiersza. Nie jest więc możliwe wykorzystanie do edycji edytora BASIC-a ani edytora z pakietu ATARI Assembler Editor, jeżeli nie napiszemy specjalnego programu użytkowego, usuwającego z tekstu źródłowego programu w języku C numery wierszy. Warto jeszcze raz podkreślić, że wszystkie zbiory źródłowe powinny mieć rozszerzenie ".C". Rozszerzenie to przyjęto tradycyjnie i jest ono wymagane przez kompilator DBC, jak również przez inne kompilatory języka C opracowane dla różnorodnych mikrokomputerów, również profesjonalnych.

2.7. Kompilacja zbiorów źródłowych.

Jak zaznaczono, program źródłowy w języku C, aby mógł być uruchomiony na komputerze musi zostać "przetłumaczony" na kod wynikowy. Proces "tłumaczenia" składa się z dwóch etapów: kompilacji i łączenia. Kompilacja wykonywana jest przez program o nazwie CC.COM kompilatora DBC. Czyta on instrukcje programu źródłowego ze zbioru źródłowego (rozszerzenie .C), tłumaczy je na postać półskompilowaną i zapisuje je do tworzonego zbioru z kodem półskompilowanym (rozszerzenie .CCC). Przedstawimy teraz dokładny sposób posługiwania się tym programem. Na przykład: aby skompilować program źródłowy zapisany w zbiorze o nazwie X.C należy wykonać następujące czynności:

1. Włączyć stację dysków i włożyć do niej dyskietkę z kompilatorem DBC;

2. Włączyć komputer przytrzymując wciśnięty klawisz OPTION;

3. Gdy wyświetli się menu DOS-a wybrać opcję L (ładowanie zbioru), napisać CC.COM i wcisnąć RETURN;

4. Gdy kompilator CC.COM załaduje się do pamięci komputera, wyczyści ekran i wyświetli komunikat nagłówkowy:

```
Deep Blue C Compiler version 1.2
(C) 1982 John Howard Palevich
File to compile (or RETURN to exit)
```

(zbiór do kompilacji (lub RETURN – wyjście z kompilatora));

5. Należy włożyć do stacji dysków dyskietkę zawierającą zbiór z programem źródłowym, napisać nazwę tego zbioru i nacisnąć RETURN. O ile pełna nazwa zbioru ma postać, na przykład D:X.C to wystarczy podać tylko główną część tej nazwy, w tym przypadku X, ponieważ kompilator zakłada, że pozostała część nazwy składa się z "D:" i nazwa ma rozszerzenie ".C". Wystarczy zatem napisać tylko X i nacisnąć RETURN. Jeżeli dyskietka z programem źródłowym znajduje się w stacji o numerze innym niż 1, to obowiązkowo należy podać w nazwie, poprzedzony literą D numer stacji dysków, zawierającej dyskietkę z programem do kompilacji, np. "D2:X".

6. Kompilator wypisze komunikat:

```
D:X.C-->D:X.CCC
```

który oznacza, że rozpoczęła się kompilacja programu. Kompilacja może trwać kilka minut, w zależności od długości i stopnia skomplikowania programu źródłowego. Aby poinformować programistę o tym, co się aktualnie dzieje, kompilator wyświetla nazwę funkcji z programu źródłowego, którą w danej chwili analizuje. Jeżeli w programie źródłowym znalezione zostaną błędy syntaktyczne, kompilator wyświetla wiersz kodu źródłowego, w którym wykrył błąd a wyświetlona pod nim strzałka pionowa skierowana ku górze wskazuje dokładne miejsce jego wystąpienia, np:

```
main(){printf("Hello,world/n");
                                ↑
Missing final end.
```

(brak nawiasu zamykającego "}" funkcję main())

Jeżeli błędy syntaktyczne nie występują, kompilator wyświetli komunikat no errors (nie ma błędów) i znów pojawi się komunikat nagłówkowy:

File to compile (or RETURN to exit)

Jeżeli za jednym razem chcemy skompilować więcej niż jeden program, możemy podać nową nazwę zbioru z programem źródłowym i ponownie uruchomić kompilator, naciskając RETURN.

7. Gdy chcemy zakończyć kompilowanie zbiorów źródłowych naciskamy klawisz RETURN (bez podawania nazwy zbioru) aby powrócić do menu Dos-a.

2.8. Łączenie zbiorów półskompilowanych.

Gdy poszczególne zbiory źródłowe zostaną już skompilowane bez błędów, to należy je połączyć i stworzyć kod wynikowy programu. W tym celu należy najpierw utworzyć zbiór informacji dla linkera, wskazujący zbiory z kodem półskompilowanym, które mają być połączone. Zbiór wskazówek dla linkera musi mieć rozszerzenie ".LNK". W zbiorze tym, oprócz nazw zbiorów zawierających kod półskompilowany powstały po kompilacji zbiorów źródłowych, muszą być podane również nazwy bibliotek kompilatora DBC, których funkcje wykorzystywane są przez program źródłowy. Typowy, mały program, wyprowadzający na ekran monitora dowolny tekst przy pomocy funkcji wyprowadzania sformatowanego "printf()" wymagałby zbioru wskazówek dla linkera o niżej przedstawionej postaci:

WYSWIETL

PRINTF

DBC.OBJ

gdzie: "WYSWIETL" jest nazwą zbioru zawierającego kod półskompilowany tego programu a "DBC.OBJ" jest nazwą programu uruchomieniowego kompilatora DBC, który musi być dołączony do kodu wynikowego każdego programu. Jeżeli dyskietka zawierająca wyżej wymienione zbiory znajduje się w stacji o innym numerze niż jeden, to podane nazwy muszą posiadać prefiks "Dn:", gdzie n jest numerem stacji dysków. Jeżeli rozszerzenia nie są podane, to linker przyjmuje rozszerzenia ".CCC". Zbiór wskazówek dla linkera nie może zawierać linii pustych, nawet na końcu zbioru.

Po utworzeniu zbioru wskazówek dla linkera można przystąpić do konsolidacji wszystkich zbiorów półskompilowanych w kod wynikowy programu. W tym celu należy załadować do pamięci komputera DOS i wybrać opcję L z jego menu. Następnie należy załadować linker, pisząc CLINK.COM i naciskając RETURN. Linker DBC po załadowaniu wyświetli komunikat:

Deep Blue C linker version 1.2

(C) 1982 John Howard Palevich

Link program, Duplicate file or Quit

(łączenie, kopiowanie zbioru lub wyjście z linkera)

Linker uruchamia się podając pierwszą literę komendy (L,D lub Q). Funkcja L tworzy działający program w języku C. Funkcja D pozwala na kopiowanie małych zbiorów (o pojemności do ok 6kB) z jednej dyskietki na drugą, bez potrzeby wykorzystania w tym celu opcji O z DOS-a. Na dużych zbiorach funkcja ta nie działa. Funkcja Q powoduje powrót do DOS-a. Zatem, gdy chcemy wykonać proces konsolidacji piszemy:

L X i naciskamy RETURN;

gdzie X jest nazwą zbioru zawierającego wskazówki dla linkera (linker przyjmuje w tym przypadku, że nazwa tego zbioru ma rozszerzenie ".LNK"). Podobnie uruchamia się funkcję kopiowania zbiorów dostępną w linkerze.

Linker nie wykona łączenia jeżeli określone zbiory nie istnieją na dyskietce. Dodatkowo, jeśli w trakcie łączenia linker napotka funkcję lub zmienną zewnętrzną, która nigdzie nie została zdefiniowana, to wyświetli komunikat: undefined label (niezdefiniowaną etykietą) i listę niezdefiniowanych nazw funkcji i zmiennych. Jeżeli np.: źle napisano nazwę zmiennej ("alfa"), która była zdefiniowana z nazwą "alpha", to również zostanie to zasygnalizowane. Jeśli nie ma błędów, linker wyświetli komunikat no errors i ponawia zaproszenie do dalszej pracy "Link program, Duplicate file or Quit". Po zakończeniu łączenia programu można wyjść z linkera pisząc Q i naciskając RETURN.

2.9. Uruchomienie programu wynikowego.

Skompilowany i skonsolidowany program w języku C może być traktowany jak każdy program w kodzie maszynowym. Można go nawet opatrzyć nazwą AUTORUN.SYS aby po włączeniu komputera ładował i uruchamiał się automatycznie. Podobnie jak inne zbiory z programami można go również ładować za pomocą opcji L z DOS-a. Podczas działania takiego programu mogą wystąpić cztery rodzaje błędów, spowodowanych złą konstrukcją programu. Błędy te nazywane są błędami uruchomieniowymi. Gdy jeden z tych błędów wystąpi, program zatrzymuje się i wyświetlany jest komunikat:

dbc i run-time error <litera>

type a key to return to DOS

<litera> może być następująca:

A - stos przekroczył zakres zmiennej RAMTOP. Zdarza się to, gdy w uruchomionym programie występuje nieskończona rekurencja lub gdy zdefiniowano zbyt wiele zmiennych.

B - nielegalny kod operacji. Błąd ten ma miejsce wówczas, gdy program wychodzi z obszaru kodu i próbuje wykonać jako rozkaz, zawartość komórek zawierających "śmieci".

C - błąd wersji. Ma miejsce wówczas, gdy numery wersji programów CC.COM, CLINK.COM i DBC.OBJ są różne.

D - próba dzielenia przez zero. W uruchomionym programie wykryta została operacja dzielenia przez zero.

2.10. Programy demonstracyjne.

Na dyskietce zawierającej pakiet kompilacyjny DBC znajdują się trzy programy demonstracyjne, które pomagają zapoznać się z programami pakietu i z zasadami ich wykorzystania. Dla każdego programu demonstracyjnego podany jest jego kod źródłowy, postać półskompilowana, zbiór wskazówek dla linkera wykorzystywany w procesie konsolidacji tego programu oraz kod wynikowy.

Pierwszy program o nazwie X demonstruje niektóre możliwości wprowadzania i wyprowadzania informacji w DBC na/z urządzenia zewnętrzne jak: stacja dysków, monitor i klawiatura. Aby uruchomić ten program, należy załadować DOS-a a następnie opcją L załadować program X.COM. Program ten uruchomi się automatycznie i wyświetli komunikat:

File reader/writer

Command: r, w, q

Przy pomocy tego programu można odczytać zbiór ze stacji dysków (przez naciśnięcie r), zapisać go (komenda w) lub zakończyć pracę (komenda q). Dobrym ćwiczeniem jest przeanalizowanie postaci źródłowej tego programu, zawartej w zbiorze X.C, ilustrującej sposób użycia funkcji z biblioteki kompilatora DBC, takich jak: open(), close(), gets(), getchar() i printf().

Drugi program demonstruje wykorzystanie grafiki PMG. Zawiera przykłady użycia takich funkcji bibliotecznych jak: color(), plot(), drawto(), sound(), pmcinit(), pmmove() i pmcolor(). Program ten zapisany jest w zbiorze o nazwie BOUNCE.COM a uruchamia się go tak, jak program X.COM.

Przejrzenie jego postaci źródłowej jest również dobrym ćwiczeniem.

Ostatni program demonstruje wykorzystanie biblioteki matematycznej kompilatora DBC, której funkcje służą do realizacji obliczeń w arytmetyce zmiennoprzecinkowej.

Opis poszczególnych bibliotek kompilatora DBC, funkcji w nich zawartych i zasad ich użycia w programie przedstawiony jest w dodatkach, umieszczonych na końcu zeszytu.

2.11. Uwagi techniczne.

Analizując uważnie postać źródłową funkcji wejścia/wyjścia zawartych w bibliotece AIO.C można zauważyć, że niektóre funkcje takie jak: ciov() zdefiniowane są w szczególny sposób, z wykorzystaniem instrukcji asm. Na przykład:

```
ciov(iocb,com,bad,blen,ax1,ax2)
int iocb,com,blen,ax1,ix2;
char *bad;
asm 12291
```

W powyższy sposób definiuje się podprogramy napisane w kodzie maszynowym. Gdy wywołana zostanie funkcja "asm" to jej argumenty przekazywane są na stos, podobnie jak w funkcji USR() z języka BASIC i następuje skok pod adres następujący po słowie "asm". Podobnie, jeżeli chcemy wykonać skok do własnego podprogramu w kodzie maszynowym, który zaczyna się od adresu $600 wystarczy w programie w języku C umieścić poniższą sekwencję instrukcji:

```
foo()
asm 600
```

Jest rzeczą oczywistą, że nasz podprogram musi zostać najpierw dołączony do programu wynikowego w języku C. W tym celu musimy wytworzyć kod maszynowy tego podprogramu za pomocą dowolnego assemblera i zapisać go, jako zbiór binarny na dyskietce. Nazwa tego zbioru powinna mieć rozszerzenie ".OBJ". Następnie nazwę tę wprowadzamy do zbioru wskazówek dla linkera i wykonujemy przebieg konsolidacji. Do dołączonego w ten sposób podprogramu maszynowego można przekazać z programu w języku C argumenty, wykorzystując jeden bajt na szczycie stosu do przekazania informacji o liczbie argumentów.

Moduł uruchomieniowy DBC wykorzystuje pamięć RAM od adresu $3000 do $3FFF, a program użytkownika zaczyna się od adresu $4000 i ciągnie się aż do końca pamięci. Na procedury w kodzie maszynowym można przeznaczyć szóstą stronę pamięci i wolną pamięć pomiędzy końcem obszaru zajmowanego przez DOS a adresem $3000. Chociaż kompilator wymaga 48kB pamięci do pracy, to większość programów w DBC wymaga dużo mniej miejsca. Można na pewno napisać wiele programów, które zajmują tak niewiele pamięci jak 24kB RAM.



3. ZACZYNAMY PROGRAMOWAĆ

3.1. Podstawowe pojęcia: funkcje, komentarze, instrukcje.

Najlepszą metodą nauczenia się programowania w określonym języku jest ... pisanie w nim programów. Zatem napiszmy pierwszy, prosty program w języku C, który wyświetli na monitorze komunikat takiej treści: "Programowanie w języku C jest ciekawym zajęciem". Oto tekst źródłowy takiego programu:

Program 1

```
main()
{          /* początek funkcji main() */
     char c;
     printf("Programowanie w języku C jest ciekawym zajęciem");
     /* i tu właściwie kończy się ten program. Instrukcja  */
     /* char c; i poniższa instrukcja c=getchar() nie są    */
     /* potrzebne do wyprowadzenia komunikatu - patrz opis */
     /* w tekście.                                          */
     c=getchar();
}          /* koniec funkcji main() */
```

Kilka słów o sposobie kompilacji i konsolidacji tego programu. Tekst programu, dokładnie w takiej postaci, w jakiej jest on wydrukowany (oczywiście bez polskich znaków), wprowadzamy przy pomocy edytora tekstowego, np. ATARI WRITER do zbioru dyskowego o nazwie "PROG1.C". Przy okazji tworzymy zbiór wskazówek dla linkera, który będzie potrzebny w procesie konsolidacji naszego programu. Zbiór ten powinien mieć nazwę "PROG1.LNK" i zawierać dokładnie cztery niżej przedstawione wiersze:

```
PROG1
AIO
PRINTF
DBC.OBJ
```

ładujemy do pamięci komputera kompilator CC.COM, piszemy nazwę zbioru z programem źródłowym, tj. PROG1.C i naciskamy RETURN. Gdy kompilacja zakończy się komunikatem "no errors", ładujemy linker CLINK.COM. Gdy kompilacja zakończy się błędami, to weryfikujemy wprowadzony przez nas tekst programu z tekstem powyższym, poprawiamy błędy edytorem tekstowym i ponownie uruchamiamy kompilator. Linker uruchamiamy poleceniem L PROG1 i wciskamy RETURN. Linker utworzy nam wykonywalny kod programu, który zostanie zapisany do zbioru o nazwie PROG1.COM. Gdy załadujemy ten program do pamięci, zostanie on automatyczne wystartowany i wypisze na ekranie poniższy tekst:

Programowanie w języku C jest ciekawym zajęciem

Naciśnięcie dowolnego klawisza i RETURN spowoduje ponowne załadowanie DOS-a.

W postaci źródłowej programu w języku C można wyodrębnić trzy zasadnicze elementy:

1. nagłówek zawierający definicję stałych programu;
2. deklaracje zmiennych programu;
3. funkcje.

Program 1 nie zawiera nagłówka, natomiast zawiera deklarację jednej zmiennej o nazwie "c". Program ten składa się tylko z jednej funkcji o nazwie main(). Każdy program w języku C składa się przynajmniej z jednej funkcji, lecz może mieć ich również więcej. Jeżeli program zawiera tylko jedną funkcję, to musi ona mieć nazwę main(). Gdy program zawiera wiele funkcji, to jedna z nich musi mieć nazwę main(). Funkcja jest częścią programu wykonującą pewną operację, na przykład: wyprowadzenie informacji na ekran, obliczenie wartości wyrażenia, itp. Funkcja wykonuje operacje na przekazanych do niej wartościach zmiennych. Wartości te nazywane są w języku C argumentami funkcji. Listę argumentów umieszcza się w nawiasach okrągłych (..), tuż za nazwą funkcji. Jeżeli funkcja do wykonania operacji nie potrzebuje żadnych argumentów (funkcja bezargumentowa), to za jej nazwą umieszcza się tylko nawiasy, na przykład funkcja main() jest funkcją

bezargumentową. Nawiasy za nazwą funkcji muszą zawsze występować, gdyż umożliwiają one kompilatorowi DBC identyfikację, czy dana nazwa jest nazwą zmiennej (bez nawiasów), czy nazwą funkcji. Funkcja może przekazać z powrotem do funkcji, z której została wywołana, wynik wykonanych obliczeń czy operacji. Ten wynik nazywa się "wartością zwracaną przez funkcję". W języku C funkcja może mieć wiele argumentów, jednak zwracać może tylko jedną wartość. Funkcje zbudowane są z instrukcji języka C. Instrukcją w języku C jest wyrażenie zakończone średnikiem. Natomiast wyrażeniem jest każda konstrukcja zbudowana ze stałych, nazw zmiennych i stałych oraz operatorów. Zatem wyrażeniami w C są: 2+5, a+b, a>b, c, itp. Każda funkcja w języku C ma swoją unikalną nazwę, nadaną przez programistę. Jedynie nazwy funkcji zawartych w bibliotece kompilatora posiadają już nadane nazwy i programiście nie wolno ich zmieniać. Nazwa każdej funkcji musi spełniać poniższe warunki:

1. musi zaczynać się od litery (znak podkreślenia "_" jest traktowany w C jak litera);

2. musi zawierać tylko litery lub liczby (inne znaki nie są dopuszczalne w nazwie funkcji);

Poprawnymi nazwami funkcji są: wynik(), WYNIK(), a1(), a_1(). Nazwami niedopuszczalnymi są: 2ala(), liczba-5(). Należy zwrócić uwagę, że nazwy wynik() i WYNIK() traktowane są przez kompilator jako nazwy różne.

Po nazwie funkcji występuje nawias otwierający "$(", a na jej końcu nawias zamykający "$)". Wszystko,co jest pomiędzy tymi nawiasami, jest traktowane przez kompilator jako jedna funkcja.

Program źródłowy w języku C może mieć dowolny układ, najbardziej odpowiadający indywidualnym upodobaniom każdego programisty. Przy jego redagowaniu należy jednak przestrzegać pewnych zasad:

- wszystkie słowa występujące w jednej linii programu muszą być oddzielone od siebie jedną lub kilkoma znakami spacji;
- instrukcje muszą być zakończone średnikiem;
- znak "#" (hash) stojący przed dyrektywą kompilatora (#define lub #include) musi być umieszczony zawsze na początku linii (musi być pierwszym znakiem w linii).

Tekst zawarty pomiędzy znakami /*...*/ jest w języku C komentarzem i jest ignorowany przez kompilator. Komentarz może być dowolnie długi. Powyższe zasady umożliwiają takie redagowanie programu źródłowego w języku C, że jego tekst może z powodzeniem stanowić dokumentację programu. Ma to istotne znaczenie w przypadku poprawiania napisanego programu po upływie pewnego czasu.

Instrukcje:

```
char c;
c=getchar();
```

nie są konieczne do poprawnego działania programu 1. Jednakże program skompilowany kompilatorem DBC ma tę cechę, że w momencie, gdy wykonana zostanie ostatnia instrukcja programu następuje automatyczne ładowanie DOS-a. Zatem powyższe instrukcje wprowadzono do programu wyłącznie po to, aby móc zaobserwować efekt działania programu na ekranie monitora. Więcej o tych instrukcjach powiemy w dalszej części tego rozdziału.

Rozważmy kolejny program:

```
Program 2
    main()
    $(     /* początek funkcji main() */
    char c;
    printf("Nazywam się Jan Kowalski.");
    printf(A Ty jak się nazywasz?");
    c=getchar();
    $)     /* koniec funkcji main() */
```

Po jego uruchomieniu zobaczymy na ekranie poniższy komunikat:

Nazywam się Jan Kowalski.A Ty jak się nazywasz?

Załóżmy, że chcemy, aby drugie zdanie komunikatu zaczynało się od nowej linii. W tym celu zamieniamy linię z programu 2

```
printf("Nazywam się Jan Kowalski.");
```

na linię:

```
printf("Nazywam się Jan Kowalski.\n");
```

Po takiej modyfikacji uzyskamy na ekranie tekst wyświetlony w dwóch wierszach:

```
Nazywam się Jan Kowalski.
A Ty jak się nazywasz?
```

Wstawiony na końcu linii symbol składający się z dwóch znaków "\n" (backslash i mała litera n) jest w języku C znakiem nowej linii. Pomiędzy znakami "\" i "n" nie może być spacji. Znak nowej linii może być użyty w dowolnym miejscu wyprowadzanego komunikatu i spowoduje, że tekst występujący po nim zostanie wyświetlony na ekranie od początku nowej linii. A więc identyczny efekt uzyskalibyśmy, zamieniając zamiast pierwszej - drugą linię printf(...), nadając jej postać:

```
printf("\nA Ty jak się nazywasz?");
```

Oprócz znaku nowej linii kompilator DBC dopuszcza stosowanie innych znaków mających specjalne znaczenie:

\f - czyść ekran;

\g - włącza sygnał dźwiękowy;

\h - kasuje znak z lewej strony aktualnej pozycji kursora i przesuwa kursor o jedną pozycję w lewo;

\r - kasuje całą linię;

\\ - wyprowadza znak backslash;

\' - wyprowadza apostrof;

\" - wyprowadza cudzysłów;

\t - wyprowadza znak tabulacji;

\### - wyprowadza znak, którego kod ATASCII jest trzycyfrową liczbą oktalną o postaci ###. Np.: litera "A" ma kod ATASCII równy 65 a odpowiada mu liczba oktalna 101. Zatem użycie w komunikacie sekwencji \101 spowoduje wyprowadzenie litery A.

W łańcuchach znaków wyprowadzanych przy pomocy funkcji wyjścia sformatowanego printf() można umieszczać dowolne znaki (również w inwersie video) poza jednym wyjątkiem. Znak "CTRL-," (serduszko) wykorzystywany jest przez kompilator do oznaczenia końca łańcucha znaków i z tego powodu nie powinien być w nim używany. Znak "\" (backslash) również ma specjalne znaczenie, wyżej przedstawione, o którym nie należy zapominać.

3.2. Stałe i zmienne. Typy i deklaracje.

Chcemy dodać do siebie dwie liczby i ich sumę wyprowadzić na monitor. Oto program realizujący to zadanie:

```
Program 3
    main()
    $(          /* początek funkcji main() */
    char c;
    int suma;   /* deklaracja zmiennej suma */
    suma=15+34;
    printf("suma jest równa: %d\n",suma);
    c=getchar();
    $)          /* koniec funkcji main()    */
```

W programie 3 "suma"jest nazwą zmiennej. Zmienne w języku C, podobnie jak w innych językach programowania są obiektami, na których program wykonuje operacje. Każdą zmienną kompilator DBC identyfikuje poprzez nazwę. Nazwy zmiennych w języku C podlegają identycznym ograniczeniom jak nazwy funkcji (patrz 3.1.)

Kompilator DBC musi być "poinformowany", że programista

będzie używał w programie zmiennych. Do przekazywania mu tych informacji służą deklaracje zmiennych. Innymi słowy: każda zmienna w programie w języku C musi być zadeklarowana przed jej użyciem. Instrukcja "int suma;" jest deklaracją. Wszystkie deklaracje zmiennych muszą wystąpić na początku funkcji, tuż za nawiasem otwierającym "#(" (są wyjątki od tej zasady). W jednej instrukcji będącej deklaracją można wymienić kilka nazw zmiennych tego samego typu np. instrukcja:

int suma,suma1,a,b,c;

jest poprawną deklaracją pięciu zmiennych typu całkowitoliczbowego o nazwach : suma, suma1, a, b, c.

DBC dopuszcza deklarację trzech typów zmiennych:

char - oznacza zmienną typu znakowego;

int - oznacza zmienną numeryczną całkowitoliczbową;

zmienne wskaźnikowe (typ pointer) o których powiemy nieco później.

Zmienna suma została zadeklarowana jako zmienna typu całkowitoliczbowego, co oznacza, że można w niej przechowywać liczby całkowite, dodatnie lub ujemne należące do przedziału [-32768, +32767]. Zmienna typu int w implementacji DBC zajmuje dwa bajty pamięci, przy czym liczba zapisana jest na piętnastu bitach, a jeden bit wykorzystywany jest jako bit znaku. Zmienna c została zadeklarowana w programie 3 jako zmienna typu znakowego. Ten typ zmiennych w implementacji DBC zajmuje jeden bajt pamięci i jest traktowany jako liczba dodatnia bez znaku z zakresu [0, 255].

Wszystkim zadeklarowanym zmiennym kompilator DBC przypisuje początkową wartość równą zero. Do zmiany wartości określonej zmiennej służy instrukcja przypisania, zbudowana w oparciu o operator przypisania "=". W instrukcji tej zmienna, której chcemy nadać wartość, występuje po lewej stronie znaku równości, natomiast po prawej stronie znaku umieszcza się wyrażenie, którego wartość zostanie przypisana zmiennej. Zatem instrukcję: suma=15+24; w języku C należy czytać: "przypisz zmiennej suma wartość wyrażenia 15+25". Operatorem przypisania jest znak równości. W języku C istnieje możliwość tworzenia również innych operatorów przypisania. Umożliwia to konstrukcja "op=", gdzie "op" jest jednym z następujących operatorów dwuargumentowych: +, -, #, /, %, <<, >>, &, ^, |. Jak kompilator DBC interpretuje utworzone w ten sposób operatory i w jakich sytuacjach wolno ich stosować? Wyjaśnimy to na przykładzie: jeśli mamy instrukcję przypisania typu a=a+b; to możemy ją zapisać w taki sposób: a+=b. Odnosi się to również do instrukcji a=a-b; a=a/b; a=a#b; itd. Wszystkie tego typu instrukcje mogą być zastąpione odpowiednimi instrukcjami: a-=b; a/=b; a#=b itd, przy czym b nie musi być tylko zmienną, lecz może być również wyrażeniem np.: instrukcja a=a+(c#2) jest równoważna instrukcji a+=c#2.

Program 3 można zapisać również tak:

```
Program 4
     #define      LICZBA1       15
     #define      LICZBA2       24
     main()#(     /# początek main()    #/
      int suma;
      char c;
      suma=LICZBA1+LICZBA2;
      printf("Suma jest równa: %d\n",suma);
      c=getchar();
     #)           /# koniec main()  #/
```

Efekt działania tego programu będzie identyczny jak programu 3. Wprowadzone zmiany dotyczą sposobu definiowania stałych w programie. W programie 3 stałe zostały zdefiniowane wprost, przez umieszczenie ich wartości w instrukcji programu. Jednak często dobrze jest używać stałych w programie poprzez nazwę, jak to ma miejsce w programie 4. #define jest dyrektywą kompilatora DBC, która informuje go, że gdy analizując program źródłowy, spotka nazwę wymienioną w tej dyrektywie ,to ma ją zastąpić wartością przypisaną tej nazwie. Nie dotyczy to oczywiście nazw występujących w łańcuchach znakowych. Definiowanie stałych za pomocą dyrektywy #define gwarantuje, że w przypadku gdy zajdzie potrzeba zmiany wartości stałych, nie trzeba będzie zmieniać instrukcji programu (czasami wielu), a jedynie dyrektywę #define.

Dyrektywą #define można definiować następujące typy stałych:

1. dziesiętne np.: 173, -45;

2. oktalne np.: 017, 045;

3. szesnastkowe np.: 0xd400, 0xff, 0xC000;

4. znakowe np.: 'a', 'b', '1';

5. łańcuchowe np.: "okno", "ala";

A oto poprawne przykłady definiowania różnych typów stałych przy wykorzystaniu dyrektywy #define:

```
#define     11     25              #define liczba  -1348
#define     12     0101            #define adres 0xe465
#define     znak   'T'             #define kom "cześć"
```

Przyjęło się, że nazwy stałych pisze się w języku C dużymi literami, jednak nie jest to konieczne. Zasada ta ma tę zaletę, że łatwo jest odróżnić w programie źródłowym stałe od zmiennych.

Zdefiniowane dyrektywą #define nazwy nie mogą być używane w programie jako zmienne. Nie można wykonywać na nich operacji przypisania i innych. Instrukcja LICZBA1=30; gdyby została umieszczona w programie 4 byłaby traktowana przez kompilator DBC jako błędna. Dyrektywy #define nie wolno kończyć średnikiem.

Oprócz dyrektywy #define kompilator DBC dopuszcza stosowanie jeszcze jednej dyrektywy, a mianowicie #include. Informuje ona kompilator, że ma on odczytać z dyskietki zbiór o nazwie podanej w dyrektywie i włączyć jego zawartość do kompilowanego programu. Włączony zbiór zawiera z reguły tekst źródłowy jednej lub kilku funkcji programu. Można jej również użyć do wczytania zbioru zawierającego definicje wszystkich stałych programu (zbiór taki zawiera tylko dyrektywy #define).

3.3. Decyzje programowe.

Bardzo często program musi podjąć określone działanie w zależności od zaistniałych warunków. Sytuację taką ilustruje poniższy problem: należy wyświetlić na monitorze pytanie, na które użytkownik może odpowiedzieć "tak" lub "nie". Jeżeli odpowiedź jest "tak", należy wyprowadzić określony komunikat; gdy odpowiedź jest "nie" - inny komunikat. Program działający w taki właśnie sposób wygląda następująco:

```
Program 5
     main()#(                 /# początek main #/
     char c,odp;
     printf("Chcesz programować w języku C? (T/N) \n");
     odp=getchar();           /# wczytaj odpowiedź #/
     c=getchar();             /# wczytaj return    #/
       if(odp=='T')#(         /# początek "if"     #/
       printf("Czytaj dalej ten rozdział\n");
     #)                       /# koniec "if"       #/
       if(odp=='N')#(         /# początek "if"  #/
       printf("To nie męcz się dłużej\n");
     #)                       /# koniec "if"    #/
     c=getchar();
     #)                       /# koniec main()  #/
```

Program 5 zbudowany jest z jednej funkcji main(), z której wywoływane są dwie funkcje z biblioteki DBC, poznana wcześniej funkcja printf() i funkcja getchar() z biblioteki AIO. Funkcja getchar() działa podobnie jak instrukcja INPUT z BASIC-a, tzn. praca komputera zostaje zawieszona aż do momentu, gdy użytkownik naciśnie klawisz RETURN. Kod klawisza naciśniętego przed naciśnięciem RETURN zwracany jest do funkcji wywołującej funkcję getchar(). Stąd instrukcja przypisania:

odp=getchar();

powoduje, że zmiennej "odp" przypisana zostaje wartość zwrócona przez funkcję getchar(). Program 5 po uruchomieniu wyświetla pytanie: "Chcesz programować w języku C? (T/N)" i czeka aż użytkownik naciśnie jeden z klawiszy T lub N. Gdy użytkownik naciśnie T program wyświetli tekst "Czytaj dalej ten rozdział" — gdy N — tekst "To nie męcz się dłużej". Takie rozstrzygnięcie przez program, jaka akcja ma być podjęta, nazywa się decyzją programową.

3.3.1. Instrukcja if.

Decyzja programowa podejmowana jest w oparciu o wynik badania warunku lub inaczej, zdarzenia programowego. Do badania warunków (zdarzeń programowych) służy w języku C, między innymi instrukcja if, będąca jednym ze słów kluczowych tego języka. W tym miejscu wypada uczynić również zastrzeżenie, że nazwy zmiennych, stałych i funkcji muszą być różne od słów kluczowych języka C. Wykaz wszystkich słów kluczowych DBC wymieniony został we wstępie.

Składnia instrukcji if jest następująca:

```
if(warunek)
$(        /* nawias otwierający          */
   instrukcja1;
   instrukcja2;
   .......
$)        /* nawias zamykający           */
instrukcja;   /* ta instrukcja wykonywana jest wówczas
               * gdy warunek instrukcji if(..) nie
               * jest spełniony                     */
```

Po słowie kluczowym if musi być zdefiniowany warunek ujęty w nawiasy (...). Warunek definiowany jest przy pomocy wyrażenia warunkowego, które może przyjmować dwie wartości: TRUE i FALSE. Wyrażeniem warunkowym jest:

odp=='T'

a odczytuje się je: "czy wartość zmiennej odp jest równa 'T'". Gdy wyrażenie warunkowe przyjmuje wartość TRUE, to wykonywana jest sekwencja instrukcji ujętych w nawiasy $(..$). Gdy przyjmie wartość FALSE, wykonywana jest pierwsza instrukcja występująca po nawiasie zamykającym. Nawiasy $(i $) mogą być pominięte jedynie wówczas, gdy przy spełnionym warunku ma być wykonana tylko jedna instrukcja.

W języku C wartości logicznej TRUE odpowiada wartość różna od zera, natomiast wartości logicznej FALSE odpowiada zero. Dzięki tej własności wyrażenia warunkowe w języku C mogą być budowane z instrukcji przypisania, wyrażeń arytmetycznych czy wartości zwracanych przez funkcje. Oczywiście można je również konstruować w klasyczny sposób, jak w języku BASIC, wykorzystując do tego operatory relacji, którymi są:

== oznacza "jest równe";

!= oznacza "jest różne";

> oznacza "jest większe";

< oznacza "jest mniejsze";

>= oznacza "jest większe lub równe";

<= oznacza "jest mniejsze lub równe".

A oto kilka przykładów wyrażeń warunkowych zbudowanych przy wykorzystaniu operatorów relacji:

10==7+3 — "czy 10 jest równe 7+3";

ala1>=ala2 — "czy wartość zmiennej ala1 jest większa lub równa wartości zmiennej ala2;

Powyższe wyrażenia warunkowe nazywane są prostymi wyrażeniami. W języku C można konstruować również złożone wyrażenia warunkowe przy pomocy wyrażeń prostych i operatorów logicznych, którymi są:

!! — oznacza logiczną alternatywę (OR);

&& — oznacza logiczną koniunkcję (AND);

! (wykrzyknik) — oznacza logiczną negację (NOT).

Przy konstruowaniu złożonych wyrażeń warunkowych można używać nawiasów (..) do wskazywania kolejności obliczania wartości poszczególnych wyrażeń prostych. Przykładami złożonych wyrażeń warunkowych są:

a>='0' && a<='9'   "czy a jest cyfrą"

a!='T' && a!='t'   "czy a nie jest literą t"

Również poprawnymi w języku C są konstrukcje warunkowe:

if(5);   if(c=a);   if(czydalej());

pomimo że wyrażenia warunkowe w nich zawarte nie są zbudowane z operatorów relacji. Pierwsze z nich przyjmuje zawsze wartość logiczną TRUE (ponieważ 5 jest różne od zera), drugie przyjmie wartość logiczną FALSE jedynie wówczas, gdy zmiennej c zostanie przypisana wartość zmiennej a równa zero, a trzecie przyjmie wartość logiczną FALSE jedynie wówczas, gdy funkcja czydalej() zwróci do miejsca wywołania zero.

Do budowy wyrażeń służą również w języku C bitowe operatory logiczne, pozwalające na manipulację pojedynczymi bitami ich argumentów. Do operatorów tych należą:

& — bitowa koniunkcja;

! — bitowa alternatywa;

^ — bitowa różnica symetryczna;

<< — przesunięcie w lewo;

>> — przesunięcie w prawo;

$~ — uzupełnienie jedynkowe.

Działanie tych operatorów ilustrują poniższe tabelki, przy założeniu, że zmienne x, y, z są typu char oraz x='A' i y='C'.

bit	7	6	5	4	3	2	1	0
arg. x	0	1	0	0	0	0	0	1
arg. y	0	1	0	0	0	0	1	1
z=x&y	0	1	0	0	0	0	0	1

bit	7	6	5	4	3	2	1	0
arg. x	0	1	0	0	0	0	0	1
arg. y	0	1	0	0	0	0	1	1
z=x!y	0	1	0	0	0	0	1	1

bit	7	6	5	4	3	2	1	0
arg. x	0	1	0	0	0	0	0	1
arg. y	0	1	0	0	0	0	1	1
z=x^y	0	0	0	0	0	0	1	0

bit	7	6	5	4	3	2	1	0
arg. x	0	1	0	0	0	0	0	1
z=$~x	1	0	1	1	1	1	1	0

Operatory << i >> służą do przesuwania bitów argumentu stojącego po lewej stronie operatora o liczbę pozycji stojącą po prawej stronie. Na przykład: x<<2 przesuwa zawartość zmiennej x o dwa bity w lewo. Zwolnione w wyniku przesunięcia pozycje bitowe zmiennej x wypełniane są zerami.

Tworzone za pomocą wyżej przedstawionych operatorów wyrażenia warunkowe jak: if(a&b), if(a!b), itp są również akceptowane przez DBC.

Gdy dokładniej przeanalizujemy działanie programu 5, to stwierdzimy, że posiada on pewne wady. Gdy użytkownik wciśnie małe "t" lub małe "n", program nie podejmie żadnego działania. Znając zasady konstruowania wyrażeń warunkowych, możemy go zatem poprawić, aby wyeliminować tę wadę.

Program 6

```
     #define TD     'T'
     #define TM     't'
     #define ND     'N'
     #define NM     'n'
     main()$(
          char c,odp;
          printf("Chcesz programować w języku C? (T/N)\n");
          odp=getchar();
          c=getchar();
          if(odp==TD !! odp==TM)
               printf("Czytaj dalej ten rozdział\n");
          if(odp==ND !! odp==NM)
               printf("to nie męcz się dłużej\n");
          c=getchar();
     $)        /* koniec main()    */
```

Zasadnicze zmiany wprowadzone w tym programie w stosunku do programu 5 są następujące:

— stałe znakowe 'T' i 'N' zostały zdefiniowane w dyrektywie #define;

— zdefiniowane zostały dwie dodatkowe stałe znakowe 't' i 'n'

po to, aby program działał bez względu na to, czy w odpowiedzi naciśnięto dużą czy małą literę;

- usunięto nawiasy $(i $) ograniczające zakres instrukcji if. Było to możliwe dlatego, że zarówno przy spełnieniu pierwszego jak i drugiego warunku wykonywana jest tylko jedna instrukcja;

- wprowadzono złożone wyrażenia warunkowe w miejsce prostych.

Przedstawimy teraz dwie pożyteczne funkcje z biblioteki kompilatora DBC, dzięki którym złożone wyrażenia warunkowe z programu 6 będziemy mogli ponownie zastąpić wyrażeniami prostymi. Funkcjami tymi są: tolower() i toupper(). Funkcja tolower() zwraca kod ATASCII małej litery, gdy jej argumentem jest kod dużej litery. Gdy jej argumentem jest kod małej litery, to jest on zwracany bez zmian. Natomiast funkcja toupper() zwraca kod ATASCII dużej litery, gdy jej argumentem jest kod małej litery. Kod dużej litery jest zwracany przez tę funkcję bez zmian. Wykorzystując jedną z tych dwóch funkcji program 6, możemy zmodyfikować nadając mu postać:

Program 7
```
     #define TD      'T'
     #define ND      'N'
     main()$(
          char c,odp;
          printf("Chcesz programować w języku C? (T/N)\n");
          odp=toupper(getchar());
          c=getchar();/* przeskocz RETURN */
          if(odp==TD)
               printf("Czytaj dalej ten rozdział\n");
          if(odp==ND)
               prinft("to nie męcz się dłużej\n");
          c=getchar();
     $)        /* koniec main()   */
```

W programie tym nie ma sprawdzania, czy odpowiedź na wyświetlone pytanie udzielona była przy użyciu dużej czy małej litery. Stało się to możliwe dzięki instrukcji przypisania:

```
          odp=toupper(getchar());
```

po wykonaniu której zmienna "odp" zawsze zawierać będzie kod ATASCII odpowiadający dużej literze alfabetu, o ile naciśnięty został klawisz odpowiadający literze (A - Z). W sytuacji, gdy naciśnięty zostanie klawisz nie będący literą, np. cyfra, SHIFT, RETURN itp. zmienna "odp" zawierać będzie niezmieniony kod tego klawisza.

Pozostańmy nadal przy programie 7. Jego dokładna analiza prowadzi do wniosku, że nie działa on poprawnie w każdej sytuacji. Co ten program "zrobi", gdy na wyświetlone pytanie użytkownik naciśnie, świadomie lub pomyłkowo, klawisz inny niż "t" lub "n". Program nie wyświetli w takiej sytuacji żadnego komunikatu i zakończy się. Poprawmy go zatem tak, aby uwzględniał on również ten przypadek. Oto jego nowa wersja:

Program 8
```
     #define TD      'T'
     #define ND      'N'
     main()$(
          char c,odp;
          printf("Chcesz programować w języku C? (T/N)\n");
          odp=toupper(getchar());
          c=getchar();/* przeskocz RETURN */
          if(odp==TD)
               printf("Czytaj dalej ten rozdział\n");
          else if(odp==ND)
               prinft("to nie męcz się dłużej\n");
          else
          $(
          printf("\nUdzieliłeś niewłaściwej odpowiedzi");
          printf("\nmiałeś nacisnąć T lub N");
          printf("\na nacisnąłeś '%c'",odp);
          $)
          c=getchar();
     $)        /* koniec main()  */
```

Jak widać, aby poprawić nasz program, musieliśmy użyć nowej instrukcji else.

3.3.2. Konstrukcja if-else

Konstrukcje programowe typu if-else mają zastosowanie w sytuacji, gdy program musi wykonać pewne działanie przy spełnionym określonym warunku a inne działanie, gdy warunek ten jest niespełniony. Działanie takich konstrukcji programowych można zilustrować poniższym schematem:

```
        if(wyrażenie)         /* gdy to wyrażenie przyjmuje
             $(                * wartość logiczną TRUE
             działanie1;       * wykonywane jest to działanie
             $)
        else                   * w przeciwnym wypadku
             $(
             działanie2;       * program podejmuje tę akcję
             $)                */
```

Ze schematu tego wynika, że konstrukcje if-else służą do wyboru jednej z dwóch możliwych ścieżek przetwarzania. Jednak w praktyce programowania często spotyka się bardziej skomplikowane sytuacje, gdy program musi wybrać jedną z kilku możliwych akcji. W języku C istnieje konstrukcja programowa umożliwiająca programiście wygodne kodowanie takich sytuacji. Konstrukcja ta zbudowana jest również ze słów kluczowych if i else, jednak dla odróżnienia jej od wyżej przedstawionej, określa się ją niekiedy nazwą else-if. Konstrukcja else-if ma postać poniższą:

```
     if(wyrażenie1)         /* gdy to wyrażenie ma wartość
          $(                 * logiczną TRUE
          akcja1;            * program wykonuje tę akcję
          $)                 *
     else if(wyrażenie2)     * gdy to wyrażenie ma wartość
          $(                 * logiczną TRUE
          akcja2;            * program wykonuje tę akcję
          $)                 *
     ..........              * itd
     else                    * gdy wszystkie w/w wyrażenia
          $(                 * mają wartość logiczną FALSE
          akcja;             * program wykonuje tę akcję
          $)                 */
```

Nawiasy ograniczające $(..$) mogą być pominięte, gdy określona akcja składa się tylko z jednej instrukcji.

Omawiając konstrukcje sterujące programem (do takich należą if-else i else-if), należy dodać, że implementacja DBC nie posiada rozkazu skoku (goto), chociaż standardowy język C rozkaz taki zawiera. Ma to swoje dobre i złe strony. Rozkazy skoku są nagminnie nadużywane we wszystkich językach programowania, w których są dostępne. DBC jest wyposażony w podstawowe konstrukcje sterujące, umożliwiające wybór właściwej drogi przetwarzania. W praktyce instrukcja skoku nigdy nie jest konieczna i zawsze można się bez niej obejść. Zatem jej brak w implementacji DBC zmusza programistę do stosowania dobrego stylu programowania - programowania strukturalnego. Wadą spowodowaną jej brakiem jest konieczność powtarzania sprawdzeń warunków i większa liczba zmiennych programowych.

Wracając jednak do naszego programu musimy stwierdzić, że jego działanie nie dostarcza nam jeszcze pełnej satysfakcji. Program 8 poinformuje użytkownika o naciśnięciu niewłaściwego klawisza i co dalej? Nic, zakończy działanie. Jednak dobrze byłoby, gdyby nasz program ponowił w takiej sytuacji pytanie.

3.3.3 Konstrukcja do-while.

Sytuacja, w której program musi powtórzyć przyjęcie określonej informacji od użytkownika, bardzo często występuje w praktyce programowania. Na przykład DOS: wyświetlając swoje menu oczekuje na polecenie wybierające jedną z możliwych opcji. Gdy użytkownik naciśnie literę nie odpowiadającą żadnej opcji, zostanie poinformowany o niewłaściwym wyborze i DOS ponownie będzie oczekiwał na komendę. Ta procedura będzie wykonywana tak długo, aż użytkownik wybierze poprawną (możliwą do wykonania) opcję.

Język C dostarcza programiście narzędzi umożliwiających powtarzanie przez program określonej akcji dowolną ilość razy. Narzędziem tym jest konstrukcja do - while. Konstrukcja ta zbudowana jest ze słów kluczowych do i while i jej postać jest następująca:

```
     do
     $(
     działanie;
     $)
     while(wyrażenie);
```

Po słowie kluczowym do następuje ciąg instrukcji ujętych w

nawiasy @(..@), który jest wykonywany tak długo, jak długo wyrażenie ujęte w nawiasy (..), występujące po słowie kluczowym while przyjmuje wartość różną od zera. W momencie, gdy wyrażenie to przyjmie wartość zero, sterowanie w programie przenoszone jest do pierwszej instrukcji występującej po słowie while i związanym z tym słowem wyrażeniem. Zatem aby program 8 ponawiał pytanie w przypadku naciśnięcia klawisz innego niż "t" lub "n", powinien mieć poniższą postać:

```
 Program 9
      #define TAK     'T'
      #define NIE     'N'
      main()@(
           char c,odp;
           do
      @(/* początek do  */
      printf("Chcesz programować w języku C? (T/N)\n");
      odp=toupper(getchar());
      c=getchar();/* przeskocz RETURN */
      if(odp!=TAK && odp!=NIE)@(
       printf("\nUdzieliłeś niewłaściwej odpowiedzi");
       printf("\nmiałeś nacisnąć T lub N");
       printf("\na nacisnąłeś '%c'",odp);
      @)/* koniec if*/
      @)/* koniec do  */
      while(odp!=TAK && odp!=NIE);
      /* od tego miejsca rozpocznij wykonywanie programu,
       * gdy zostanie naciśnięty jeden z klawiszy "t"
       * lub "n" */
      if(odp==TAK)
           printf("Czytaj dalej ten rozdział\n");
      else
           printf("to nie męcz się dłużej\n");
      c=getchar();
@)         /* koniec main()   */
```

Program 9 po wyświetleniu pytania czeka na odpowiedź. Gdy naciśnięty zostanie klawisz inny niż "t" lub "n", program wyświetli komunikat o udzieleniu niewłaściwej odpowiedzi i ponownie wyświetli pytanie. Akcję tę program będzie kontynuował tak długo, aż zostanie naciśnięty albo klawisz "t" albo "n". Wówczas nastąpi wyjście z pętli do - while i wyprowadzony zostanie jeden z dwóch, przewidzianych w takiej sytuacji komunikatów.

3.3.4 Instrukcja while.

Przedstawiona w poprzednim rozdziale pętla programowa może być również skonstruowana inaczej - w sposób bardziej elegancki. Szczegółowa analiza pętli do - while z programu 9 prowadzi do następujących wniosków:

- wyświetlane jest pytanie: "Czy chcesz...?.";
- wczytywany jest znak z klawiatury;
- badane jest (instrukcja if), czy wczytany znak nie jest ani literą T ani N;
- jeżeli nie jest, to wyświetlany jest komunikat o udzieleniu niewłaściwej odpwiedzi;
- ponownie badane jest, czy wczytany znak jest inny niż T i N (tym razem badanie tego warunku realizowane jest w instrukcji while);
- jeżeli jest inny, to pętla wykonywana jest od początku.

Nieracjonalnym w tej pętli jest dwukrotne badanie tego samego warunku. Jest to cena, którą trzeba płacić za brak instrukcji skoku. Ale czy naprawdę trzeba? Nie zawsze; program 10 zawiera pętlę realizującą to samo zadanie co program 9, jednak wyeliminowano z niego dwukrotne badanie tego samego warunku. Do zbudowania tej pętli zastosowana została instrukcja while, będąca również słowem kluczowym języka C. Postać tej instrukcji jest następująca:

```
     while (wyrażenie logiczne)
     @(
          instrukcja1
          instrukcja2
          ...
     @)
```

Słowo kluczowe while otwiera pętlę. Tuż za nim musi wystąpić wyrażenie logiczne ujęte w nawiasy (..). Bezpośrednio za wyrażeniem logicznym występuje ciąg instrukcji ujętych w nawiasy @(..@). Jeśli wystąpi tylko jedna instrukcja nawiasy można pominąć. Instrukcja while działa w ten sposób, że na początku obliczana jest wartość wyrażenia logicznego i jeśli

Program 10

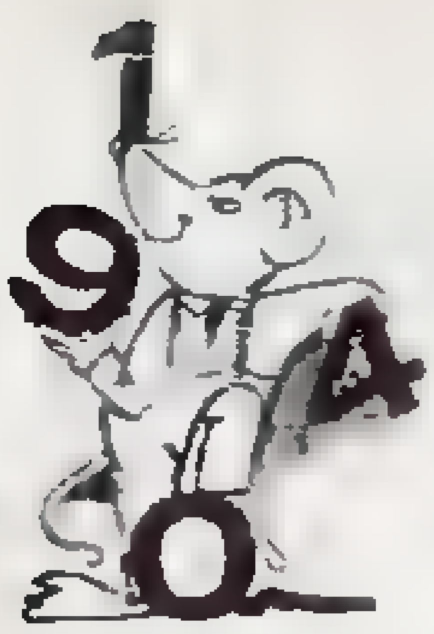
```
   #define   TAK   'T'
   #define   NIE   'N'
   main()
   @(
    char odp,ret;
    printf("Czy chcesz programowac w Jezyku C?\n);
    printf("odpowiedz T lub N\n");
    printf("--> ");
    odp=toupper(getchar());
    ret=getchar();  /* pomin RETURN */
    while(odp!=TAK && odp!=NIE)
        @(   /* poczatek petli while */
        printf("\nUdzieliles zlej odpowiedzi %c",odp);
        printf("\nnacisnij T lub N");
        printf("--> ");
        odp=toupper(getchar());
        ret=getchar();
        @)   /* koniec petli while */
    if(odp==TAK)
      printf("\nCzytaj dalej ten rozdzial");
    else
      printf("\nTo nie mecz sie dluzej");
    odp=getchar();  /* zawies program przed */
                    /* zaladowaniem DOS-a */
  @)                /* koniec main() */
```

jest ona różna od zera (TRUE), wykonywane są instrukcje zawarte w pętli (pomiędzy nawiasami @(..@)). Proces ten powtarzany jest tak długo, aż wyrażenie logiczne nie przyjmie wartości zerowej (FALSE). Wówczas żadna z instrukcji zawartych w pętli nie zostanie wykonana, a sterowanie zostanie przeniesione do instrukcji występującej po nawiasie zamykającym pętlę.

Warto zapamiętać podstawową różnicę w realizacji pętli do - while i while. W pierwszym przypadku instrukcje składające się na pętlę wykonywane są przed sprawdzeniem warunku. Zatem zawsze pętla do - while zostanie wykonana przynajmniej jeden raz. Natomiast w drugim przypadku, najpierw następuje sprawdzenie warunku i w zależności od jego wyniku pętla może zostać wykonana lub nie. Zatem w szczególnej sytuacji pętla while może być w ogóle nie wykonana.

Uważnego czytelnika intryguje zapewne średnik stojący tuż za nawiasem zamykającym wyrażenie warunkowe pętli while; zadaje on sobie pytanie: dlaczego wyrażenie warunkowe w instrukcji if nie jest zakończone średnikiem, a w instrukcji while jest? Skąd się bierze taka niekonsekwencja? Otóż, nie jest to brak konsekwencji, lecz wprost przeciwnie, średnik jest dowodem wnikliwego przemyślenia przez twórców języka C wszystkich stosowanych w nim konstrukcji programowych. Instrukcja while występująca w pętli do - while i w pętli while jest tą samą konstrukcją języka C. Zatem kompilator, interpretując kod źródłowy programu oczekuje wystąpienia po słowie kluczowym while ciągu instrukcji do wykonania. Gdyby ich nie znalazł sygnalizowałby błąd. Aby go "oszukać", umieszcza się po słowie kluczowym while w pętli do - while instrukcję nieistotną dla całego programu, tzw. instrukcję pustą. Taką właśnie instrukcją jest sam średnik.

Przedstawione instrukcje if, else, do, while nie wyczerpują wszystkich możliwości sterowania programem w języku C. O ile instrukcja if jest wygodna przy wyborze jednej z dwóch możliwych ścieżek przetwarzania, to jej zastosowanie przy wyborze jednej z wielu dróg staje się już mniej wygodne, a ponadto ogranicza czytelność programu źródłowego. Język C posiada również dużo wydajniejsze instrukcje umożliwiające sterowanie programem i organizowanie pętli programowych - instrukcje switch , for, które zostaną przedstawione w następnych rozdziałach.



4. OPERACJE NA LICZBACH

Przykładowe programy przedstawione w poprzednim rozdziale zawierały proste operacje arytmetyczne. Język C umożliwia realizowanie również skomplikowanych operacji arytmetycznych, podobnie jak inne języki wysokopoziomowe (Pascal, Basic). Wystarczy zadeklarować nazwy stałych i zmiennych występujących w wyrażeniu arytmetycznym i skonstruować to wyrażenie, wykorzystując niżej przedstawione operatory arytmetyczne:

+ – dodawanie;

(-) – odejmowanie;

* – mnożenie;

/ – dzielenie.

Wartość wyrażenia arytmetycznego może być przypisana do pewnej zmiennej programowej lub wyprowadzona bezpośrednio, np. na ekran. Ilustruje to poniższy program:

```
main()
{
 int a,b,c;
 a=20;
 b=25;
 c=a+b;
 printf("a+b = %d ",c);
}
```

Po uruchomieniu tego programu na ekranie zostanie wyświetlona linia

```
a+b = 45
```

Można oczywiście obliczać wartości bardziej skomplikowanych wyrażeń, np.:

```
b=20;
c=25;
d=-10;
a=b+c+d;
```

W przykładzie tym zmiennej d została przypisana wartość minus 10 (-10). Przypisanie to wykorzystuje jednoargumentowy operator (-), tzw. minus unarny. Operator ten działa według zasady: "pobierz wartość argumentu i zmień jej znak na przeciwny". W przykładzie tym widać również, że w jednej linii (instrukcji) może być użytych wiele operatorów arytmetycznych. Następny przykład ilustruje nowy problem:

```
b=20;
c=25;
d=3;
a=b+c*d;
```

Jak w języku C zostanie wykonana taka operacja? Czy wartość b zostanie dodana do c i następnie suma pomnożona przez d? A może inaczej: najpierw c zostanie pomnożone przez d i iloczyn zostanie dodany do wartości b.

W języku C istnieją ściśle określone reguły, według których obliczana jest wartość wyrażeń. Reguły te definiują ważność każdego operatora, a co za tym idzie, kolejność wykonywania operacji.

Największą wagę mają operatory mnożenia i dzielenia i te działania wykonywane są zawsze jako pierwsze (licząc od lewej do prawej strony wyrażenia). Następne w kolejności są operatory dodawania i odejmowania. Ponieważ C umożliwia stosowanie operatorów arytmetycznych i logicznych w jednym wyrażeniu, stąd następnymi w kolejności są operatory logiczne porównania (==, >, < itd.), koniunkcji (&&), alternatywy (||). Na samym końcu w tej hierarchii ważności znajduje się operator przypisania (=).

Wracając do powyższego przykładu, możemy teraz stwierdzić, że najpierw wartość c zostanie pomnożona przez d i iloczyn zostanie dodany do wartości b. Obliczony w ten sposób wynik (=95) zostanie przypisany zmiennej a. Rozważmy wyrażenie:

```
if (a==5+8 && b>20)
```

Opierając się na wyżej przedstawionych zasadach przeanalizujemy, jak obliczona zostanie jego wartość:

– w pierwszej kolejności wykonana zostanie operacja dodawania 5+8 i wynik 13 zostanie gdziekolwiek zapamiętany;

– następnie zostaną sprawdzone warunki: czy wartość zmiennej a jest równa 13, a wartość zmiennej b jest większa od 20. Załóżmy, że a=13 a b=28. Ponieważ przy tym założeniu obydwa wyrażenia logiczne przyjmą wartość logiczną TRUE, zatem i całe wyrażenie przyjmie taką wartość.

Należy zapamiętać, że w języku C kolejność wykonywania operacji, zarówno arytmetycznych jak i logicznych jest następująca:

```
działanie               wykonywane
* i /                     przed
+ i -                     przed
==,!=,>,<,>=,<=           przed
&& i ||                   przed
= (przypisanie)
```

Kolejność ta może być zmieniona przez zastosowanie nawiasów. Wracając do przedstawionego przykładu, jeśli wyrażenie arytmetyczne napiszemy stosując nawiasy:

```
a=(b+c)*d
```

to zawartość zmiennej b zostanie dodana do c i suma pomnożona przez wartość zmiennej d, a zatem inaczej niż bez użycia nawiasów. Nawiasy mogą być stosowane w dowolny sposób; ważne, aby ich liczba była parzysta.

Przedstawimy teraz program realizujący funkcję prostego kalkulatora. Działał on będzie w następujący sposób:

–zapytaj użytkownika o rodzaj operacji, która ma być wykonana;

–zapytaj go o dwa argumenty, na których należy wykonać wybraną operację;

–wykonaj operację;

–wyświetl wynik operacji;

–powtórz wszystko od początku, gdy użytkownik wyrazi takie życzenie. Na początek zajmiemy się ostatnim elementem powyższego algorytmu.

4.1. Piszemy własną funkcję.

Każdy dobrze zaprojektowany program powinien przed zakończeniem swojego działania zadać użytkownikowi pytanie, czy chce on ponownie wykonać przebieg programu czy też nie i w zależności od uzyskanej odpowiedzi podjąć odpowiednią akcję. W języku C istnieje szczególnie wygodny mechanizm do realizacji takiego celu. Są nim funkcje. Dobrze zaprojektowana i przetestowana funkcja może zostać w postaci półskompilowanej włączona do biblioteki kompilatora i używana w każdym nowotworzonym programie.

Przedstawimy teraz funkcję, której zadaniem będzie ustalenie, czy użytkownik programu ma zamiar kontynuować pracę, czy też ją zakończyć. Będzie to funkcja bezargumentowa, która będzie zwracać do miejsca wywołania wartość 1, gdy użytkownik będzie chciał kontynuować pracę, a zero w przeciwnym wypadku. Tekst źródłowy takiej funkcji może wyglądać następująco:

Funkcja 1

```
/****************************
 *   funkcja  CZYDALEJ      *
 ****************************/
czydalej()$(
int stat,odp,T,t,N,n;
char *u,od,zer,i;
i='1';
od='4';
zer='0';
u="K:";
T=84;  /* kod litery T */
t=116; /* kod litery t */
N=78;  /* kod litery N */
n=110; /* kod litery n */
stat=open(i,od,zer,u);   /* otwarcie "K:" z iocb i   */
if(stat!=1)$(
   printf("\n\tBlad otwarcia K: ");
   return;
$)    /* koniec if  */
do$(  /* poczatek do */
printf("\nCzy chcesz kontynuowac?");
odp=cgetc(i);
if(odp!=T && odp!=t && odp!=N && odp!=n)$(
  printf("\n\tOdpowiedz albo T albo N");
  continue;
 $)  /* koniec if */
$)   /* koniec do */
while(odp!=T && odp!=t && odp!=N && odp!=n);
close(i);
return((odp==T || odp==t) ? 1 : 0);
$) /* koniec czydalej() */
```

Funkcja czydalej() działa w następujący sposób:

-wyświetla pytanie: "Czy chcesz kontynuować?" i czeka na naciśnięcie jednego z dwóch klawiszy T lub N;

-gdy zostanie naciśnięty jakikolwiek klawisz, sprawdzane jest czy jest to klawisz T lub N, jeżeli nie jest, to funkcja wyświetla komunikat "Odpowiedz albo T albo N", ponawia pytanie i ponownie czeka na naciśnięcie klawisza;

-gdy naciśnięty zostanie klawisz T (lub małe t), funkcja czydalej() kończy działanie i zwraca do miejsca wywołania wartość 1;

-gdy naciśnięty zostanie klawisz N (lub małe n), funkcja również kończy działanie, zwracając w takiej sytuacji zero do miejsca wywołania.

Zakończenie działania funkcji realizowane jest przy pomocy instrukcji return, która umożliwia przekazanie sterowania w programie z funkcji wywołanej do funkcji wywołującej. Postać tej instrukcji jest następująca:

return(wyrażenie);

Jeśli instrukcja return nie zawiera żadnego wyrażenia, w takiej sytuacji do funkcji wywołującej nie zostanie zwrócona żadna wartość (nastąpi tylko przekazanie sterowania). Gdy instrukcja return zawiera wyrażenie, to wraz z przekazaniem sterowania, do funkcji wywołującej przekazana zostanie wartość, będąca rezultatem funkcji wywołanej. Instrukcja

return((odp==T || odp==t) ? 1: 0);

powoduje wyjście z funkcji czydalej() i jednoczesne przekazanie rezultatu tej funkcji do funkcji wywołującej (main()). Gdy wyrażenie logiczne (odp==T ||odp==t) przyjmie wartość TRUE, to funkcja czydalej() zwróci do miejsca wywołania wartość 1, gdy wyrażenie to przyjmie wartość FALSE - zostanie zwrócone zero.

Uzyskanie tak zwartego zapisu instrukcji return jest możliwe dzięki trójargumentowemu operatorowi warunkowemu "?". Wyrażenia logiczne budowane przy pomocy tego operatora mają ogólną postać:

wyrażenie1 ? wyrażenie2 : wyrażenie3

a ich wartość obliczana jest według następującego algorytmu:

-najpierw obliczana jest wartość wyrażenia1;

-jeśli obliczona wartość jest różna od zera (TRUE), to obliczana jest wartość wyrażenia2 i ona jest wartością wynikową całego wyrażenia logicznego;

-w przeciwnym wypadku wartością wynikową całego wyrażenia jest wartość wyrażenia3.

Operator ? zastępuje konstrukcje logiczne typu:

```
if (y>x)
   z=y;
else
   z=x;
```

dzięki niemu można je zapisać w zwartej postaci:

```
z=(y>x) ? y : x
```

Należy zauważyć, że x,y,z mogą być dowolnymi wartościami zmiennych. Zatem wyżej przedstawionej instrukcji można użyć np. do wybrania większej z dwóch liczb.

Kilka słów na temat kompilacji funkcji czydalej(). Przedstawiony wyżej kod źródłowy funkcji należy wprowadzić za pomocą edytora tekstowego do zbioru o nazwie CZYDALEJ.C. Kompilator DEEP BLUE C jej kod półskompilowany zapisze do zbioru o takiej samej nazwie, z rozszerzeniem .CCC. Chcąc dołączyć tę funkcję do każdego nowotworzonego programu, należy przy konsolidacji tego programu umieścić dodatkowy wiersz CZYDALEJ.CCC w zbiorze wskazówek dla linkera.

4.8. Program kalkulator.

Możemy teraz przystąpić do napisania wspomnianego wyżej programu kalkulatora. Oczywiście wykorzystamy w nim funkcję czydalej(). (Nie zapomnij o umieszczeniu wiersza CZYDALEJ.CCC w potrzebnym do konsolidacji tego programu zbiorze wskazówek dla linkera). Oto postać źródłowa tego programu

program 11

```
/****************************/
/* program11 KALKULATOR    */
/****************************/
#define  TAK   1
#define  NIE   0
main()
$(       /*poczatek funkcji main()  */
 int l1,l2,wynik;
 char oper,zlyoper,znak;
 char arg1[120],arg2[120];
 do     /* poczatek do  */
 $(
  zlyoper=NIE;
  printf("\nWybierz rodzaj dzialania\n");
  printf("+,-,*,/ ");
  oper=getchar();
  znak=getchar(); /* przeskocz RETURN*/

  printf("\nPodaj pierwszy argument: ");
  gets(arg1);
  l1=val(arg1);
  printf("\nPodaj drugi argument: ");
  gets(arg2);
  l2=val(arg2);
  switch(oper)
   $(    /* poczatek switch */
    case '+' : wynik=l1+l2;
               break;
    case '-' : wynik=l1-l2;
               break;
    case '*' : wynik=l1*l2;
               break;
    case '/' : wynik=l1/l2;
               break;
    default  : printf("\nWybrales zle dzialanie");
               zlyoper=TAK;
               break;
   $)   /* koniec switch */
  if(zlyoper==NIE)
   printf("\nWynik dzialania: %d\n",wynik);
 $)     /* koniec do   */
 while(czydalej());
$)    /* koniec main()  */
```

Jest to pierwszy większy program i zasługuje na specjalne

wyjaśnienie. Trzy linie:

```
printf("\nWybierz rodzaj dzialania\n");
printf("+,-,*,/");
oper = getchar();
```

informują o działaniach, jakie program może wykonać i wczytują z klawiatury znak, odpowiadający wybranej operacji do zmiennej oper. Linie:

```
printf("\nPodaj pierwszy argument: ");
gets(arg1);
l1=val(arg1);
printf("\nPodaj drugi argument: ");
gets(arg2);
l2=val(arg2);
```

wprowadzają z klawiatury wartości argumentów, na których ma być wykonana wybrana operacja. Wprowadzanie argumentów realizowane jest dwuetapowo. Najpierw wczytywany jest łańcuch znaków stanowiących argument (funkcja gets()), a następnie łańcuch ten zamieniany jest na liczbę za pomocą funkcji val(). Funkcja ta działa podobnie, jak instrukcja VAL z BASIC-a. W tym miejscu można zadać pytanie: czy nie można wprowadzić z klawiatury ciągu cyfr, dokonując automatycznie konwersji tego ciągu na odpowiadającą mu liczbę. Niestety, implementacja DBCC nie zawiera takiej możliwości, aczkolwiek w standardowym języku C istnieje funkcja realizująca takie zadanie (scanf()). Następnie za pomocą instrukcji switch wybierany jest rodzaj operacji, która ma być wykonana i następuje jej realizacja.

4.3. Instrukcja switch.

W programie "kalkulator" rodzaj akcji, którą program ma podjąć zależy od rodzaju działania, który wybiera użytkownik. W oparciu o informacje dotyczące instrukcji if i else fragment programu realizujący ten wybór można byłoby zapisać w poniższy sposób:

```
if (oper=='+')
  wynik = l1+l2;
else if (oper=='-')
  wynik = l1-l2;
else if (oper=='*');
  wynik = l1*l2;
else if (oper=='/')
  wynik = l1/l2;
else $(
printf("wybrales zle dzialanie\n");
zlyoper = TAK $)
```

Instrukcja switch pozwala uprościć zapis i uczynić go bardziej czytelnym.

Konstrukcję zbudowaną z instrukcji switch rozpoczyna słowo kluczowe switch, po którym występuje wyrażenie (lub zmienna) ujęte w nawiasy (..). Następnie w nawiasach $(..$) umieszczona jest lista wszystkich możliwych przypadków (wyborów), od których zależy rodzaj podejmowanego przez program działania. Każdy przypadek definiowany jest za pomocą instrukcji case, za którą, po dwukropku, definiowane jest działanie, które musi być podjęte przez program, gdy zaistnieje zdefiniowany przypadek. Ciąg instrukcji definiujących działanie programu w zaistniałym przypadku musi być zakończony instrukcją break lub nawiasem zamykającym "$)" definiowanie wszystkich przypadków. Na przykład:

```
case '+' :   wynik = l1+l2
             break;
```

Instrukcja break powoduje, że gdy wykonana zostanie akcja związana z zaistnieniem określonego przypadku, nie będą sprawdzane przypadki zdefiniowane za nim i dalsze wykonywanie programu rozpocznie się od instrukcji umieszczonej po nawiasie zamykającym "$)" całą konstrukcję switch.

Słowo kluczowe default służy do definiowania działania, które program podejmuje w sytuacji, gdy nie zostanie spełniony żaden z przypadków zdefiniowanych instrukcjami case. Zatem z programie "kalkulator", gdy użytkownik wprowadzi niedopuszczalny rodzaj działania, wykonane zostaną instrukcje:

```
printf("wybrales zle dzialanie\n");
zlyoper = TAK;
```

i nie zostanie wyświetlony wynik działania. Wyświetlenie wyniku działania uzależnione jest od wartości wyrażenia (zlyoper==NIE). Jeśli wyrażenie to przyjmie wartość różną od zera (nie zostanie wybrany niedozwolony rodzaj działania), wynik zostanie wyświetlony, w przeciwnym wypadku wyświetlanie wyniku jest pominięte.

4.4. Instrukcja continue.

Wnikliwy programista, analizując program "kalkulator", dostrzeże zapewne nieeleganckie rozwiązanie w nim zawarte. Polega ono na tym, że w sytuacji, gdy użytkownik wybiera niedozwolony rodzaj operacji (inny niż +,-,*,/) program wykona zupełnie niepotrzebnie całą pętlę. Znacznie lepiej byłoby, gdyby w takiej sytuacji program zażądał dokonania ponownego wyboru, natychmiast po stwierdzeniu wybrania niedozwolonej operacji. Poprawmy zatem ten program, aby wyeliminować tę wadę. W tym celu użyjemy instrukcji continue. Oto poprawiona wersja programu:

Program 12

```
/***************************/
/* program12 KALKULATOR    */
/* wersja z inst. continue */
/***************************/
#define   TAK   1
#define   NIE   0
main()
$(        /*poczatek funkcji main()  */
 int l1,l2,wynik;
 char oper,zlyoper,znak;
 char arg1[120],arg2[120];
 do     /* poczatek do  */
 $(
  printf("\nWybierz rodzaj dzialania\n");
  printf("+,-,*,/ ");
  oper=getchar();
  znak=getchar(); /* przeskocz RETURN*/

  /* Teraz sprawdz, czy wprowadzono dozwolony rodzaj dzialania */
  /* i powtorz ponownie wybor, gdy wybrano dzialanie niedozw.  */
  if(oper!='+' && oper!='-' && oper!='*' && oper!='/')
    $(
     printf("Wybrales zle dzialanie\n");
     continue;
    $)
  /* od tego miejsca rozpocznij wykonywanie programu   */
  /* gdy wybrano dopuszczalne dzialanie                */
  printf("\nPodaj pierwszy argument: ");
  gets(arg1);
  l1=val(arg1);
  printf("\nPodaj drugi argument: ");
  gets(arg2);
  l2=val(arg2);
  if(l2==0) break;
  /* zatrzymaj program gdy drugi argument jest zerem   */
  switch(oper)
   $(     /* poczatek switch */
    case '+' : wynik=l1+l2;
               break;
    case '-' : wynik=l1-l2;
               break;
    case '*' : wynik=l1*l2;
               break;
    case '/' : wynik=l1/l2;
               break;
   $)   /* koniec switch */
  printf("\nWynik dzialania: %d\n",wynik);
 $)    /* koniec do   */
while(czydalej());
$)    /* koniec main()  */
```

Dzięki użyciu instrukcji continue program ten po stwierdzeniu, że użytkownik wybrał niedozwolony rodzaj operacji, wyświetli komunikat "Wybrałeś złe działanie" i nakaże dokonać

ponownie wyboru.

poniższy schemat ilustruje działanie instrukcji continue:

```
do
$(       /$ początek pętli $/
   if (wyrażenie logiczne1)
     $(
     .....
 ┌───continue;
 │   $)
 │
 │     /$ pozostała część pętli $/
 │     $)
 └>while (wyrażenie logiczne2);
```

Wykonywany jest początek pętli do i obliczana jest wartość wyrażenia logicznego1. Gdy jest ona różna od zera, wykonywane są instrukcje umieszczone po nim, na których końcu znajduje się instrukcja continue. Jej wykonanie powoduje przeniesienie sterowania do instrukcji while, a stąd, gdy spełniony jest warunek zdefiniowany wyrażeniem logicznym2, następuje skok na początek pętli. W tej sytuacji pozostała część pętli (w programie "kalkulator" zawiera ona instrukcję switch) zostanie wykonana jedynie wówczas, gdy wyrażenie logiczne1 przyjmie wartość zero (FALSE). Należy zauważyć, że w takiej sytuacji nie ma potrzeby używania instrukcji default.

4.5. Instrukcja break.

W programie 11 zastosowano instrukcję break do "przeskoczenia" pozostałych do zbadania przypadków zdefiniowanych w instrukcji wyboru switch. Natomiast w programie 12 instrukcja ta została użyta w celu opuszczenia pętli programowej:

```
if(12==0) break;
```

Jeżeli założymy, że użytkownik wprowadził drugi argument działania równy zero, to powyższa instrukcja spowoduje opuszczenie pętli i wykonanie instrukcji występującej po while, które zamyka całą pętlę. Ogólnie, sposób działania instrukcji break można zilustrować schematem:

```
do
$(    /$ początek pętli  $/
 ....
┌──if (warunek) break;
│       /$ pozostała część pętli $/
│  $)
│  while(warunek);
└──────> /$ koniec pętli $/
```

Takie zastosowanie instrukcji break bywa czasami bardzo przydatne. Na przykład: w programie 11 brak było badania wartości drugiego argumentu wykonywanej operacji; zakładając, że wartość ta była zerem i wykonywana była operacja dzielenia, wystąpiłby w takiej sytuacji błąd wykonania programu (próba dzielenia przez zero). W programie 12 przewidziano taką ewentualność i za pomocą instrukcji break pominięto wykonywanie działania. Oczywiście, że zamiast zakończenia w takiej sytuacji działania programu należałoby napisać sekwencję rozkazów wczytujących ponownie wartość drugiego argumentu.

4.6. Instrukcja for.

Instrukcja for jest prawdopodobnie najczęściej używaną instrukcją języka C, przeznaczoną do konstruowania pętli programowych. Najważniejsza różnica pomiędzy pętlą for a przedstawionymi wcześniej pętlami while i do..while polega na tym, że te ostatnie stosuje się, gdy dokładnie nie wiadomo, ile razy mają być wykonane sekwencje instrukcji w nich zawarte. Natomiast instrukcja for jest bardzo pożyteczna do tworzenia pętli, o których z góry wiadomo, że mają być wykonane ściśle określoną liczbę razy. Działanie tej instrukcji zilustrujemy programem obliczającym kwadraty kolejnych liczb naturalnych, np.: kwadraty liczb od 1 do 100. Oto postać takiego programu:

Program 13

```
main()
$(
int liczba;
char c;
printf("\Kwadraty liczb naturalnych\n\n");
for(liczba=1;liczba<=100;liczba=liczba+1)
  $(
  printf("\nliczba: %d kwadrat %d",liczba,liczba$liczba);
  $)
c=getchar();
$)
```

Instrukcja for zawiera ujęte w nawiasy (...) trzy części, oddzielone od siebie średnikami. Część pierwsza (liczba=1) jest częścią inicjującą pętli, wykonywana jest ona raz na początku pętli i ustawia wartość początkową licznika pętli. Część druga, zwana częścią sterującą pętli, (liczba<=100) jest wyrażeniem logicznym, od wartości którego zależy czy pętla zostanie wykonana ponownie, czy nastąpi jej opuszczenie. Pętla jest wykonywana tak długo, jak długo wyrażenie to przyjmuje wartość logiczną TRUE. Trzecią część pętli (liczba=liczba+1) definiuje krok, z jakim zmieniana jest wartość licznika pętli. Część ta nazywana jest również częścią modyfikującą pętli. Po instrukcji for umieszczone są, ujęte w nawiasy `$(...$)` instrukcje stanowiące pętlę. Jeżeli pętla składa się tylko z jednej instrukcji, to nawiasy `$(` i `$)` można pominąć.

Nietrudno dostrzec, że program 13 można skonstruować również używając pętli while. Będzie on miał wówczas taką postać:

Program 14

```
main()
$(
int liczba;
char c;
printf("\Kwadraty liczb naturalnych\n\n");
liczba=1;     /$ część inicjująca pętli $/
while(liczba<=100)    /$ część sterująca $/
  $(
  printf("\nliczba: %d kwadrat %d",liczba,liczba$liczba);
  liczba=liczba+1;    /$ część modyfikująca $/
  $)
c=getchar();
$)
```

Efekty działania programów 13 i 14 są identyczne. Którą zatem konstrukcję należy stosować w tworzonych programach? Nie ma jednoznacznej odpowiedzi na takie pytanie. Obie konstrukcje są równoważne. Pętla for jest bardziej przejrzysta, gdyż skupia wszystkie elementy sterujące jej wykonaniem w jednym miejscu programu.

Dowolną z trzech części instrukcji for można pominąć, jednak oddzielający je średnik musi pozostać. Pominięcie części inicjującej i modyfikującej jest równoznaczne z usunięciem odpowiednich instrukcji z programu 14. (Można przeanalizować, jak ten program będzie działał w takiej sytuacji). Pominięcie części sterującej jest równoznaczne założeniu, że definiujące ją wyrażenie logiczne zawsze przyjmuje logiczną wartość TRUE. Korzystając z tej cechy, można za pomocą instrukcji for skonstruować tzw. wieczną pętlę: for(;;).

Z reguły wyjście z takiej pętli następuje przy pomocy instrukcji continue, break lub return. Jednak taka konstrukcja bywa czasami bardzo przydatna.

Omawiając instrukcję for, warto przy okazji przedstawić dwa jednoargumentowe operatory dodawania i odejmowania, bardzo wygodne w użyciu i stosowane najczęściej do modyfikowania wartości licznika pętli. Na przykład: instrukcję przypisania

```
liczba=liczba+1;
```

można w języku C zastąpić prostym wyrażeniem: liczba++. Zatem instrukcji for z programu 13 można nadać taką postać:

```
for(liczba=1;liczba<=100;liczba++)
```

Operator "++" realizuje działanie: "zwiększ o jeden wartość zmiennej". Operator ten może być również użyty w instrukcji przypisania:

```
j=i++;
```

Instrukcja powyższa zostanie wykonana w następujący sposób:

-wartość zmiennej i przypisana jest do zmiennej j i jest zwiększana o jeden. Zatem instrukcja j=i++ jest równoważna dwom instrukcjom:

```
j=i;
i=i+1;
```

Można budować również takie instrukcje:

```
j=++i;
```

są one wykonywane nieco inaczej: najpierw wartość zmiennej i jest zwiększana o jeden, a potem zmiennej j przypisywana jest wartość zmiennej i. Zatem odpowiada to instrukcjom:

```
i=i+1;
j=i;
```

Analogicznie działa jednoargumentowy operator odejmowania "--". Instrukcja j=i--; jest równoważna instrukcjom:

```
j=i;
i=i-1;
```

natomiast instrukcja j=--i; zastępuje:

```
i=i-1;
j=i;
```

Operatory "++" i "--" mogą być użyte w złożonych wyrażeniach arytmetycznych, jak na przykład:

```
j=i++-k+3;
```

które zastępuje poniższe instrukcje:

```
j=i-k+3;
i=i+1;
```

Uważny czytelnik, który wprowadził i skompilował program "kalkulator", mógł zauważyć, że gdy dodane (lub pomnożone) zostaną dwie duże liczby np.: 10000 i 30000, to program wyprowadza niepoprawne wyniki. Również błędne wyniki będzie generował program 13, gdy podnoszone do kwadratu liczby będą większe od 182. Co jest tego przyczyną? Dzieje się tak, ponieważ w pierwszym jak i w drugim programie mnożone, dodawane, dzielone i odejmowane liczby zostały zadeklarowane jako liczby całkowite (typu int). Na wstępie zaznaczono, że w implementacji DBC liczba typu int może przyjmować wartości z przedziału [-32768,32767]. Jeżeli w wyniku wykonania operacji arytmetycznej powstaje rezultat nie mieszczący się w tym przedziale, następuje przepełnienie (tzn. wynik przekracza zakres liczby, którą można zapisać na 15 bitach, bit 16 jest bitem znaku), które powoduje, że wyprowadzane wyniki są błędne. Cóż zatem robić w takiej sytuacji? Otóż, standardowy język C dopuszcza deklaracje zmiennych typu float i double, których zakres jest nieporównywalnie większy od zmiennych typu int. Aczkolwiek DBC nie posiada zaimplementowanych tego typu danych, to zawiera bogatą bibliotekę funkcji przeznaczonych do realizacji arytmetyki zmiennoprzecinkowej. Ze względu na liczbę funkcji zawartych w tej bibliotece, ich opis oraz sposób wykorzystania, ilustrowany odpowiednimi programami, zostanie przedstawiony w następnym wydaniu specjalnym IKS-a.



5. TABLICE I WSKAŹNIKI

W języku C tablicami są zmienne przeznaczone do przechowywania wielu wartości tego samego typu, do których można odwoływać się poprzez jedną, wspólną dla wszystkich tych wartości nazwę. Elementy tablic numerowane są od zera. Kompilator DBC dopuszcza stosowanie w programach dwóch typów tablic: tablic znakowych (ich elementami są zmienne typu char) i tablic, których elementami są zmienne typu int. Zarówno pierwsze jak i drugie tablice mogą być jednowymiarowe. Tablice, podobnie jak zmienne, muszą być zadeklarowane przed ich użyciem. Deklaracja tablicy składa się z określenia typu tablicy (int,char), jej nazwy i rozmiaru podanego w nawiasach kwadratowych [..]. Na przykład:

```
char tab[9];
```

jest deklaracją tablicy znakowej o nazwie "tab", zawierającej 9 elementów [0,1,...8], a deklaracja:

```
int liczby[9];
```

jest deklaracją tablicy 9 elementowej o nazwie liczby, której elementami są zmienne typu int (liczby całkowite). Należy pamiętać, że obszar pamięci zajmowany przez tablicę typu int jest dwa razy większy od liczby jej elementów (każda zmienna typu int zajmuje dwa bajty pamięci). Odwołanie do określonego elementu tablicy realizowane jest poprzez nazwę tablicy i numer (indeks) tego elementu ujęty w nawiasy [..]. Ponieważ elementy tablic numerowane są od zera, to indeks o wartości zero wskazuje pierwszy element, indeks 1 - drugi element, indeks 2 - trzeci, itd. Zatem instrukcja:

```
liczby[2]=2;
```

przypisuje trzeciemu elementowi tablicy o nazwie "liczby" wartość 2. Natomiast instrukcja: liczby[9]=14; jest błędną w języku C, ponieważ element o indeksie 9 fizycznie nie istnieje. Tego typu błędy w programie są bardzo trudne do wykrycia, tym bardziej, że żaden kompilator języka C, DBC również, błędów takich nie sygnalizuje podczas kompilacji programu. Tylko nieliczne programy (np. program Lint systemu UNIX), napisane specjalnie w celu kontrolowania poprawności składni kodu źródłowego programu w C, są w stanie błędy takie wykryć.Kontrola "wzrokowa" tekstu źródłowego programu, też z reguły takich błędów nie wykryma - wszystko wygląda poprawnie: tablica jest 9 elementowa, operacja jest wykonana na elemencie o indeksie 9 a skompilowany program nie działa poprawnie.

5.1. Tablice znakowe.

Ponieważ DBC dopuszcza jedynie stosowanie tablic jednowymiarowych, to tablice znakowe są traktowane jako łańcuchy znaków lub krótko - łańcuchy. Kompilator DBC oznacza, w wewnętrznej reprezentacji, koniec łańcucha znakiem, którego kod ATASCII jest równy zeru (zerduszko). Pisząc własne funkcje operujące na łańcuchach, należy ten znak (reprezentacja oktalna '\0') traktować jako wskaźnik końca łańcucha.

Każdy, kto pisał programy w języku BASIC wie, że przypisanie wartości zmiennej tekstowej może być wykonane za

pomocą instrukcji:

ALA$="WITAJ"

Natomiast w języku C instrukcja przypisania:

tab="okno";

jest niedopuszczalna (tab – tablica znakowa). W DBC wprowadzenie wartości do elementów tablicy znakowej wymaga albo zdefiniowania wskaźnika (będzie o nich mowa trochę później) do tej tablicy albo wykonania szeregu instrukcji przypisania, z których każda przypisze określoną wartość określonemu elementowi tablicy. W ogóle język C nie posiada instrukcji operujących na łańcuchach jako całości. Na pierwszy rzut oka wydaje się to być niewygodne. Jednak operując wskaźnikami, problem ten można doskonale rozwiązać.

5.2. Klasyczne podejście do przetwarzania tablic.

Tablice w języku C można przetwarzać, podobnie jak w innych językach wysokopoziomowych, odwołując się do ich elementów poprzez indeksy. Taką metodę przetwarzania ilustrują przedstawione niżej dwa programy.

Program 15

```
/*********************************
 * program15 PRZESTAWIANKA       *
 *********************************
 */
main()$(
char znak,wyraz[128];
int i,j,dl;
 do$(
 printf("\nNapisz dowolny wyraz:\n");
 dl=gets(wyraz);
 if(dl<=0) continue;
 printf("\nNapisales:\n");
   for(i=0;i<=dl-1;i++)
        printf(" %c",wyraz[i]);
/* teraz przestaw znaki we wprowadzonym wyrazie */
   for(i=0,j=dl-1;i<j;i++,j--)$(
     znak=wyraz[i];
     wyraz[i]=wyraz[j];
     wyraz[j]=znak;
    $)   /* koniec petli for  */
/* wyswietl przestawiony wyraz   */
 printf("\nA tak to wyglada po przestawieniu\n");
   for(i=0;i<=dl-1;i++)
      printf(" %c",wyraz[i]);
  $)   /* koniec do  */
 while(czydalej());
$)  /* koniec main  */
```

Program 15 wykonuje przestawienie znaków w łańcuchu, w taki sposób, że pierwszy znak staje się ostatnim, i drugi przedostatnim, itd. Łańcuch znaków po wczytaniu z klawiatury jest wyświetlany w normalnej postaci i w postaci przestawionej. Tekst źródłowy tego programu jest na tyle prosty, że nie wymaga szczegółowego komentarza. Na uwagę zasługuje jedynie instrukcja for organizująca pętlę, w której dokonywane jest przestawienie znaków w łańcuchu:

```
for(i=0,j=dl-1;i<j;i++,j--)
```

Instrukcja ta steruje jednocześnie dwoma indeksami do tablicy wyraz[]. Indeks i "przesuwa się" po tablicy od jej początku do końca, natomiast indeks j [illegible] od końca do początku. Taka konstrukcja w języku C jest możliwa, dzięki operatorowi przecinkowemu. Znak "," jest operatorem jedynie wówczas, gdy występuje pomiędzy wyrażeniami, np.:

```
i=0,j=0,k=0,l=0;
```

jest instrukcją ustawiającą wartości czterech zmiennych: i, j, k, l na zero. Operatora przecinkowego można używać również do tworzenia takich instrukcji:

```
i=5,i+2;
```

po jej wykonaniu zmienna i ma wartość 7. Przecinek nie pełni funkcji operatora, gdy oddziela zmienne w deklaracjach lub argumenty w wywołaniu funkcji.

Program 16

```
/**********************************
 * program16 Sortowanie tablicy   *
 *       liczb calkowitych        *
 *   wedlug algorytmu Shella      *
 **********************************
 */
#define   dl  10
main()$(
int tab[dl],i,j,k,krok;
char liczba[6];
do$(
 /* wczytaj wartosci do tab  */
 for(i=0;i<=dl-1;i++)$(
  printf("\npodaj %d element",i);
  gets(liczba);
  tab[i]=val(liczba);
 $)   /* for */
 /* teraz wyswietl wszystkie elementy
  * w takiej kolejnosci w jakiej zostaly
  * wczytane
  */
 printf("\nwprowadziles nastepujace liczby:\n");
 for(i=0;i<=dl-1;i++)
   printf(" %d",tab[i]);
 /* teraz uporzadkuj rosnaco tablice */
 for(krok=dl/2;krok>0;krok/=2)
   for(i=krok;i<dl;i++)
     for(j=i-krok;j>=0 && tab[j]>tab[j+krok];j-=krok)$(
       k=tab[j];
       tab[j]=tab[j+krok];
       tab[j+krok]=k;
     $)  /* for */
 /* teraz wyswietl posortowana tablice */
 printf("\n\ttablica posortowana\n");
  for(i=0;i<=dl-1;i++)
   printf(" %d",tab[i]);
 $) /* do */
while(czydalej());
$) /* main */
```

Program 16 porządkuje 10 elementową tablicę liczb. Najpierw wprowadza z klawiatury wartości do elementów tej tablicy, wyświetla je, porządkuje metodą Shell'a elementy od najmniejszego do największego i wyświetla elementy uporządkowane.

Oba programy wykorzystują przedstawioną wcześniej funkcję czydalej(). Przyjrzyjmy się jej jeszcze raz, ponieważ zastosowano w niej nowy typ zmiennej – zmienną wskaźnikową.

5.3. Wskaźniki.

Zmienne wskaźnikowe, lub krótko wskaźniki, podobnie jak inne zmienne w języku C muszą być zadeklarowane przed ich użyciem. Każdy wskaźnik posiada typ związany z obiektem, który wskazuje. Deklaracja:

char *u

nakazuje kompilatorowi DBC utworzenie wskaźnika typu znakowego, który wskazywał będzie obiekty tego samego typu, tzn. zmienne i tablice znakowe typu char. Natomiast deklaracja:

int *i

nakazuje kompilatorowi utworzenie wskaźnika typu integer (wskazywał on będzie zmienne i tablice tego samego typu). DBC nie dopuszcza deklaracji wskaźników innych typów.

W języku C wskaźnik jest zmienną zawierającą adres obiektu (zmiennej lub elementu tablicy). Na przykład: jeśli w programie zadeklarowaliśmy tablicę typu integer:

int tab[10];

oraz wskaźnik:

int *wsk

to sam fakt deklaracji nie wiąże jeszcze wskaźnika ze wskazywanym przez niego obiektem (np. zadeklarowaną wyżej tablicą). Jeśli chcemy, aby wskaźnik został związany z jakąkolwiek zmienną musimy przypisać adres tej zmiennej do tego wskaźnika. Przypisanie adresu dokonywane jest przez operator adresu "&". Na przykład instrukcja:

wsk=&tab[0];

powoduje, że wskaźnikowi przypisany zostanie adres pierwszego elementu tablicy tab. Jeżeli teraz takiego wskaźnika użyjemy w instrukcji:

```
element=*wsk;
```

to zmienna element zawierać będzie pierwszy element tablicy tab. Operator "*" powoduje, że jego argument jest traktowany jako adres zmiennej, której zawartość ma być użyta w instrukcji, w której operator ten występuje. Łatwo zauważyć, że instrukcje:

```
wsk=&tab[1];
element=*wsk;
```

są równoważne jednej:

```
element=tab[1];
```

gdzie zmienna element jest zadeklarowaną wcześniej zmienną typu integer. Po cóż zatem wskaźniki w języku C, skoro można bez ich pomocy zrealizować potrzebne operacje?

Najistotniejszym powodem, dla którego konstruktorzy języka C wyposażyli go w mechanizm wskaźników, jest sposób realizacji funkcji i przekazywania do nich argumentów.

Argumenty w języku C przekazywane są do wywoływanych funkcji poprzez wartość. Chcąc zatem przekazać do funkcji, jako argument np. 100 elementową tablicę, należałoby z każdego elemenetu tej tablicy uczynić osobny argument. Byłoby to bardzo niewygodne. Posługując się wskaźnikami, można przekazać funkcji, jako argument, adres początku tej tablicy, zawarty we wskaźniku. Działanie tego mechanizmu pokażemy na przykładzie programu 15. Napiszemy ten program w innej postaci, tworząc funkcję odwroc(), której zadaniem będzie przestawienie znaków w łańcuchu. Argumentem tej funkcji będzie adres łańcucha i jego długość. Oto postać tej funkcji.

Funkcja 2

```
/**********************
 * funkcja ODWROC()   *
 ***********************/
odwroc(wyraz,dlugosc)
char *wyraz;
int dlugosc;
$( /* poczatek funkcji */
char znak;
char *koniec;
dlugosc-=1;
koniec=wyraz+dlugosc;
for(;wyraz<koniec;wyraz++,koniec--)$(
 znak=*wyraz;
 *wyraz=*koniec;
 *koniec=znak;
 $) /* for */
return;
$) /* koniec odwroc() */
```

Tekst źródłowy tej funkcji należy wprowadzić do zbioru o nazwie ODWROC.C. Kompilator DBC po jej skompilowaniu, zapisze kod półskompilowany do zbioru ODWROC.CCC. Chcąc użyć tej funkcji w programie 15, należy go zmodyfikować, zastępując pętlę, w której realizowane było przestawianie znaków łańcucha wywołaniem funkcji odwroc(). Wersja zmodyfikowana tego programu ma postać:

Program 17

```
/********************************
 * program17 PRZESTAWIANKA      *
 * wykorzystuje funkcje odwroc() *
 * do przestawiania znakow w    *
 * lancuchu                     *
 ********************************
 */
main()$(
char znak,wyraz[120];
int i,j,dl;
 do$(
 printf("\nNapisz dowolny wyraz:\n");
 dl=gets(wyraz);
 if(dl<=0) continue;
 printf("\nNapisales:\n");
   for(i=0;i<=dl-1;i++)
        printf(" %c",wyraz[i]);
/* teraz przestaw znaki we wprowadzonym wyrazie */
 odwroc(wyraz,dl);
/* wyswietl przestawiony wyraz   */
 printf("\nA tak to wyglada po przestawieniu\n");
   for(i=0;i<=dl-1;i++)
      printf(" %c",wyraz[i]);
  $)   /* koniec do  */
 while(czydalej());
$)  /* koniec main  */
```

Przed konsolidacją programu 17 należy wprowadzić do zbioru wskazówek dla linkera dodatkowy wiersz (ODWROC.CCC).

Efekty działania programów 15 i 17 są identyczne.

Przedstawimy teraz dokładnie, jak funkcjonują wskaźniki w funkcji odwroc() i w jaki sposób przekazywane są do niej argumenty. Instrukcja odwroc(wyraz,dl); w programie 17 stanowi wywołanie funkcji odwroc(). Funkcja ta otrzymuje dwa argumenty:

- nazwę tablicy "wyraz" z funkcji main();
- liczbę elementów tablicy (zmienna dl)

Wewnątrz funkcji odwroc() deklaracja char *wyraz; tworzy wskaźnik do tej tablicy. Zbieżność nazw tablicy i zmiennej wskaźnikowej nie jest przypadkowa. Gdybyśmy użyli wskaźnika o innej nazwie, np. char *poczatek; to jego wartość po takiej deklaracji byłaby nieokreślona.

Wewnątrz funkcji odwroc() mamy zadeklarowane dwie zmienne: char znak; i char *koniec;. Zmienna "znak" jest zwykła zmienną roboczą, wykorzystywaną do chwilowego przechowania przestawianych elementów tablicy. Natomiast wskaźnik "koniec" jest "prywatnym" wskaźnikiem funkcji odwroc() i wskazuje ostatni element przestawianej tablicy. Instrukcja:

```
koniec=wyraz+dlugosc;
```

inicjuje wartość wskaźnika "koniec", przypisując mu adres ostatniego elementu tablicy "wyraz" (do adresu początkowego tablicy zawartego we wskaźniku "wyraz" dodawana jest liczba elementów tablicy). Należy w tym miejscu zwrócić uwagę na jeden aspekt takiego inicjowania wartości wskaźników. Powyższe inicjowanie jest poprawne, gdyż tablica "wyraz" jest tablicą znakową (każdy jej element zajmuje jeden bajt pamięci). Natomiast gdyby tablica "wyraz" była typu int (każdy element zajmuje dwa bajty pamięci), to poprzednia inicjacja wskaźnika "koniec" byłaby błędna. Należałoby wówczas użyć takiej:

```
koniec=wyraz+(2*dlugosc);
```

Przestawianie elementów tablicy realizowane jest w pętli for(..). Pętla ta nie posiada części inicjującej, ponieważ nie ma potrzeby definiowania takiej części. Część sterującą pętli stanowi wyrażenie:

```
wyraz<koniec;
```

będące porównaniem wartości dwóch wskaźników (adresów). Pętla będzie wykonywana tak długo, dopóki wartość wskaźnika wskazującego na początku pętli pierwszy element tablicy "wyraz" będzie mniejsza od wartości wskaźnika wskazującego ostatni element.

Część modyfikującą pętli stanowią wyrażenia: wyraz++, koniec--. Oznacza to, że po każdorazowym wykonaniu pętli wskaźnik początku tablicy zwiększany jest w taki sposób, aby wskazywał on następny element, a wskaźnik końca - zmniejszany w taki sposób, aby wskazywał poprzedzający element. Jak widać, na wskaźnikach można wykonywać operacje arytmetyczne. Wskaźniki są

adresami, zatem wykonywanie na nich operacji dodawania lub odejmowania jest zwiększaniem lub zmniejszaniem adresu pamięci. Można również dodawać do siebie (lub odejmować) wartości wskaźników. Jednak zawsze należy pamiętać, aby poddawane takim operacjom wskaźniki były tego samego typu. Dodanie wartości wskaźnika typu znakowego do wartości wskaźnika typu int i użycie tak wyznaczonego adresu do jakiejkolwiek operacji może spowodować nieobliczalne skutki w programie. Z czego to wynika? Deklarując wskaźnik, np. char *znak; informujemy kompilator, że wskaźnik ten wskazywał będzie zmienne znakowe. Jeżeli na takim wskaźniku wykonamy operację znak++, to jego wartość zostanie zwiększona o jeden i tak jest poprawnie. Natomiast, gdyby wskaźnik ten został zadeklarowany jako typu int, to operacja znak++ zwiększyłaby jego wartość o dwa (nie o jeden), gdyż po niej ma on wskazywać następny obiekt typu int. Jest to logiczne, gdyż zmienne typu int zajmują dwa bajty pamięci.

Należy zwrócić również uwagę na deklarację argumentów funkcji odwroc() wewnątrz funkcji. Jeżeli funkcja otrzymuje jakieś argumenty, to muszą być one zadeklarowane tuż po nazwie funkcji (poza nawiasami $(..$)). Wszystkie zmienne deklarowane wewnątrz nawiasów $(..$) są prywatnymi zmiennymi funkcji, w której zostały zadeklarowane i żadna inna funkcja, w tym również main() nie ma do nich dostępu.

Pozostańmy jednak nadal przy wskaźnikach. Jak zaznaczono wcześniej, sam fakt zadeklarowania wskaźnika nie powoduje, że wskazuje on obiekt programu. Aby wskaźnik można było sensownie wykorzystać, musi on zostać prawidłowo zainicjowany, tzn. musi mu zostać przypisana wartość, będąca adresem jednej ze zmiennych programu. Używanie wskaźników bez nadania im wartości początkowej (inicjacji) jest częstym błędem popełnianym przez początkujących programistów w języku C. Przed opisaniem metod inicjowania wartości wskaźników należy zwrócić uwagę na jeszcze jeden aspekt ich stosowania. W języku C stanowią one alternatywną metodę przetwarzania dużych obszarów danych, zapisanych w pamięci komputera. Alternatywną, gdyż jak wynika z przedstawionych wcześniej przykładów, prawie wszystko to, co można wykonać w programie przy ich zastosowaniu, można również zrealizować stosując metodę indeksowania elementów tych obszarów. Często stosowanie indeksów bywa wygodniejsze, a często jest odwrotnie.

Kompilator DSC realizując deklarację zmiennej wskaźnikowej nie sprawdza, czy został zadeklarowany obiekt programu, z którym ten wskaźnik ma być skojarzony. Jest to kłopot programisty. Stąd tak istotne znaczenie ma inicjowanie wartości wskaźników. Sposoby inicjowania wskaźników pokażemy na niżej przedstawionych przykładach:

Niech deklaracja

```
char tablica[100];
```

będzie deklaracją tablicy znakowej, a

```
char *wsk;
```

deklaracją wskaźnika tego samego typu. Instrukcja przypisania:

```
wsk=&tablica[0];
```

inicjuje wskaźnik wsk na pierwszy (początkowy) element tablicy. Poprawne jest również zainicjowanie wskaźnika w taki sposób:

```
wsk=&tablica[i];
```

wskazywał on wówczas będzie i-ty element tablicy. Wskaźnik zainicjować można również w poniższy sposób:

```
wsk="to jest tekst";
```

Tak zainicjowany wskaźnik zawiera adres obszaru pamięci, zawierającego łańcuch "to jest tekst". Jeżeli tak zainicjowanego wskaźnika użyjemy w instrukcji przypisania:

```
znak=*(wsk+2);
```

gdzie znak jest zmienną znakową, to zawierać ona będzie kod trzeciego znaku z tego łańcucha, tj kod spacji (#20).

Wskaźnik może być zainicjowany na początkowy element tablicy poprzez instrukcję:

```
wsk=tablica;
```

Z ostatniego sposobu inicjowania wynika, że nazwa tablicy bez nawiasów [..] traktowana jest jak wskaźnik. Tak jest w istocie. W języku C każde odwołanie do elementu tablicy realizowane jest poprzez wskaźnik do jej początku. Czy można zatem traktować jako równoważne deklaracje wskaźników i funkcji, np.:?

```
char tablica[100];

char *tablica;
```

Niestety, nie. Deklaracja tablicy musi zawsze wystąpić wszędzie tam, gdzie konieczne jest zarezerwowanie pewnego obszaru pamięci na dane programu. Deklaracja wskaźnika problemu tego nie rozwiązuje. Natomiast jeżeli chodzi o metodę dostępu do elementów tych danych, to oba podejścia, tj. klasyczne, oparte na indeksowaniu elementów, jak i na wskaźnikach - są równorzędne.

Istnieje tylko jedna sytuacja, w której powyższe deklaracje są równoważne. Ma ona miejsce wówczas, gdy dotyczą one parametrów formalnych funkcji. Można to zilustrować na przykładzie prezentowanej już funkcji odwróć(), której jednym z argumentów jest tablica wyraz[]. Wewnątrz funkcji jest ona zadeklarowana jako wskaźnik:

```
odwroc(wyraz,dlugosc)
char *wyraz;
```

i jej elementy przetwarzane są przy zastosowaniu wskaźników. Tablicę wyraz[] można zadeklarować wewnątrz funkcji również tak:

```
odwroc(wyraz,dlugosc);
char wyraz[];
```

i wówczas jej elementy powinny być przetwarzane przy zastosowaniu metody klasycznej, tj opartej na indeksach. Funkcja odwroc() powinna mieć w tym przypadku taką postać:

```
/****************************
 * funkcja ODWROC()         *
 * wersja "indeksowa"       *
 ****************************/
odwroc(wyraz,dlugosc)
int dlugosc;
char wyraz[dlugosc];
$( /* poczatek funkcji */
char znak;
int i,j;
dlugosc-=1;
for(i=0,j=dlugosc;i<j;i++,j--)$(
 znak=wyraz[i];
 wyraz[i]=wyraz[j];
 wyraz[j]=znak;
 $) /* for */
return;
$) /* koniec odwroc() */
```

DODATEK A

"BIBLIOTEKA DEEP BLUE C"

A1. Wstęp.

W przeciwieństwie do innych języków C nie posiada na stałe wbudowanych instrukcji we/wy. Zamiast BASIC'owej instrukcji PRINT czy PASCAL'owej WRITE język C wykorzystuje funkcje we/wy. Jest to cecha bardzo wygodna, ale oznacza że każda implementacja posiada swoją własną wersję podstawowych instrukcji we/wy. DEEP BLUE C nie stanowi tu wyjątku, ale jeżeli zaproponowany zbiór funkcji nie będzie Ci odpowiadał możesz zdefiniować swoje własne, bardziej odpowiadające Twoim potrzebom.

Funkcje zdefiniowane w plikach AIO.C, GRAPHICS.C, PMG.C i PRINTF.C zapewniają dostęp do możliwości sprzętowych komputera na takim poziomie jak BASIC. Funkcje biblioteczne o tak znanych nazwach jak *plot()*, *drawto()* czy *poke()* działają podobnie jak ich odpowiedniki z języka BASIC. Pomijając oczywisty fakt, że najbardziej dokładną definicją każdej funkcji jest jej kod źródłowy w języku C podajemy jednak krótki opis każdej z nich.



A2. FUNKCJE WE/WY ZDEFINIOWANE W BIBLIOTECE AIO.C

```
clear( s, len)
char *s;
int len;
```

Funkcja clear ładuje do tablicy *s[0 .len-1]* bajty zerowe, co jest wygodne przy inicjowaniu dużych tablic. Dla tablic złożonych z liczb całkowitych argument długości *len* powinien być pomnożony przez dwa, gdyż podaje się go w bajtach a nie w słowach.

```
copen( fn, mode)
char *fn, mode;
```

Funkcja *copen* otwiera plik, o nazwie zawartej w łańcuchu *fn*, dla odczytu, zapisu lub dołączenia, w zależności od wartości znaku *mode*:

r - odczyt pliku (jak OPEN #n, 4, fn$)
w - zapis pliku (jak OPEN #n, 8, fn$)
a - dołączanie do pliku (jak OPEN #n, 12, 0, fn$)

Jeżeli plik zostanie otwarty poprawnie, to funkcja zwraca numer IOCB (0-7). Należy go zapamiętać w jakiejś zmiennej, gdyż będzie potrzebny do komunikowania się z danym plikiem. Jeżeli plik nie został otwarty funkcja zwraca liczbę ujemną, będącą negacją kodu błędu CIO. Na przykład wciśnięcie klucza BREAK w czasie próby otwarcia pliku spowoduje, że funkcja *copen()* zwróci wartość -128.

```
open( iocb, ax1, ax2, fname)
char iocb, ax1, ax2, *fname;
```

Funkcja ta jest podobna do instrukcji OPEN z języka BASIC. Funkcja *open()* zwraca wartość 1 jeżeli nie ma problemów z otwarciem pliku. W przypadku przeciwnym zwraca ujemny kod błędu CIO.

```
close( i)
char i;
```

Ta funkcja jest podobna do instrukcji CLOSE z BASIC'a. Zamyka IOCB i zwraca wartość 1 lub w przypadku wystąpienia błędu ujemny kod błędu CIO.

```
cclose( i)
int i;
```

Gdy chcesz zamknąć konkretny plik powinieneś wywołać funkcję *cclose()* z liczbą zwracaną przez *copen()*. Funkcja *cclose()* zwraca wartość 1 gdy wszystko zakończyło się poprawnie lub negację błędu CIO, gdy nie udało się zamknąć pliku.

```
cgetc( iocb)
int iocb;
```

Funkcja *cgetc()* jest bardzo podobna do instrukcji GET z BASIC'a. Po przekazaniu numeru IOCB funkcja zwraca kolejny znak z pliku (o kodzie od 0 do 255) lub liczbę ujemną, która jest kodem błędu CIO.

```
getchar()
```

Funkcja *getchar()* wprowadza jeden znak z ekranu (iocb 0) i zwraca wartość 1 lub ujemny kod błędu CIO.

```
putchar( c)
char c;
```

Funkcja *putchar()* wyświetla dany znak na ekranie (iocb 0) i zwraca wartość 1 lub ujemny kod błędu CIO.

```
gets( string)
char *string;
```

Funkcja *gets()* jest podobna do instrukcji INPUT z BASIC'a. Wprowadza cały logiczny wiersz tekstu i umieszcza go w tablicy znakowej podanej jako argument. Tablica ta powinna mieć przynajmniej 120 bajtów długości, gdyż w przeciwnym przypadku operator może ją przepełnić poprzez wprowadzenie zbyt długiego wiersza. Jeżeli nie wystąpił błąd funkcja *gets()* zwraca liczbę znaków wprowadzonego wiersza (od 0 do 120). W przypadku wystąpienia błędu zwracany jest ujemny kod błędu CIO.

```
cprints( string)
char *string;
```

Funkcja *cprints()* jest podobna do instrukcji PRINT z BASIC'a. Wyświetla na ekranie łańcuch będący jej argumentem. Nie wyprowadza znaku RETURN ale do tego celu można wykorzystać instrukcję *putchar(155)*, która wymusi przejście kursora na początek następnego wiersza.

```
cputs( strig, iocb)
char *string;
int iocb;
```

Funkcja *cputs()* jest podobna do instrukcji PRINT # z BASIC'a. Wyprowadza przekazany łańcuch do konkretnego pliku. Należy jako drugiego argumentu funkcji użyć numeru IOCB zwróconego przez instrukcję *copen()*. Jeżeli brak błędu funkcja *cputs()* zwraca wartość 1, a w przypadku przeciwnym ujemny kod błędu CIO.

```
ciov( iocb, com, bad, blen, ax1, ax2)
int iocb, com, blen, ax1, ax2;
char *bad;
```

Funkcja *ciov()* jest podobna do wywołania XIO z BASIC'a. Możesz ustawić *iocb* i wywoływać przy pomocy tej funkcji CIO. Argument *iocb* powinien być z przedziału 0-7 i musi określać blok we/wy, którego używasz. *com* jest kodem rozkazu ICCOM. *bad* jest adresem bufora ICBAD, a *blen* długością bufora ICBLEN. *ax1* jest pomocniczym bajtem ICAX1, a *ax2* pomocniczym bajtem ICAX2. Jeżeli nie chcesz zmieniać bieżącej wartości któregoś z czterech ostatnich argumentów (*bad*, *blen*, *ax1*, *ax2*) użyj zastępczo wartości -1. Tak więc chcąc wczytać kolejny wiersz do bufora powinieneś napisać:

```
ciov(1, 5, -1, -1, -1, -1);
```

Większość podanych funkcji we/wy zostało zaimplementowana z wykorzystaniem *ciov()*. Dwa wyjątki *cgetc()* i *cputc()* zostały zaimplementowane w asemblerze, aby zwiększyć szybkość ich działania. Jeżeli CIO zwraca wynik mniejszy niż 128, to funkcja *ciov()* przekazuje go w niezmienionej postaci, ale jeżeli kod wynikowy jest większy lub równy 128 (oznacza to wystąpienie błędu), to *ciov()* zwraca ujemną wartość tego kodu. Takie podejście jest zgodne ze standardem języka C, w którym przyjęto, że kody błędów są podawane jako liczby ujemne.

```
normalize( fname, fext)
char *fname, *fext;
```

normalize() jest funkcją użytkową służącą do przekształcenia dowolnej formy nazwy pliku w standardową postać wymaganą przez CIO i FMS. Najpierw nazwa pliku zamieniana jest na nazwę pisaną dużymi literami, a następnie, jeżeli nie występuje przedrostek nazwy urządzenia, na początek nazwy dodawane są znaki "D1:". Jeżeli brak rozszerzenia, to do nazwy pliku dodawana jest

kropka i łańcuch *fext*. Oto typowy przykład wykorzystania tej funkcji:

```
char fname[20];
gets( fname);
normalize( fname, "BAS");
```

Wykorzystanie tej funkcji gwarantuje otrzymanie nazwy pliku akceptowalnej przez system CIO. Na przykład jeżeli użytkownik wprowadzi nazwę "prog", to po

```
normalize( fname, "BAS");
```

łańcuch wynikowy zawiera "D:PROG.BAS"

```
toupper( c)
char c;
```

Jeżeli c jest małą litera, to funkcja *toupper()* zwraca kod odpowiedniej dużej litery.

```
tolower( c)
char c;
```

Jeżeli c jest duża litera, to *tolower()* zwraca kod odpowiedniej małej litery.

```
strcpy( a, b)
char *a, *b;
```

Funkcja *strcpy()* kopiuje łańcuch znaków z tablicy *b* do tablicy *a*. Funkcja zwraca długość kopiowanego łańcucha z pominięciem bajtów zerowych.

```
move( a, b, len)
char *a, *b;
int len;
```

Funkcja *move()* przepisuje len bajtów z *a* do *b* zaczynając od elementu *a[0]*, a kończąc na *a[len-1]*. Dzieją się dziwne rzeczy jeżeli *a* <= *b* <= *a* + len.

```
usr( adr,...)
int adr;
```

Funkcja *usr()* jest podobna do funkcji USR() z BASIC'a. Pierwszy argument jest adresem podprogramu napisanego w języku maszynowym, natomiast reszta stanowi parametry jakie są do niego przekazywane. Wynik obliczeń zwracany jest w rejestrach A (część mniej znacząca) i X (część bardziej znacząca). Po wywołaniu podprogramu użytkownika stos wygląda nastepująco:

<oprócz adresu liczba argumentów (0-120)>
<bardziej znaczący bajt pierwszego argumentu>
<mniej znaczący bajt pierwszego argumentu>
<bardziej znaczący bajt drugiego argumentu>
itd.
<adres powrotu (dwa bajty)>

Procedura użytkownika może wykorzystywać bajty #f6-#ff strony zerowej.

```
find( addr, len, ch)
char *addr, ch;
int len;
```

Funkcja *find()* przeszukuje obszar pamięci od adresu *addr* do adresu *addr* + *len* - *1*, aż do momentu wystąpienia znaku *ch*. Jeżeli funkcja nie znajdzie znaku *ch* zwraca wartość -1, a w przeciwnym przypadku podaje odległość (w bajtach) znaku *ch* od adresu *addr*.

```
peek( i)
char *i, d;
```

Funkcja *peek()* zwraca bajt pamięci o adresie i.

```
poke( i, d)
char *i, d;
```

Funkcja *poke()* wpisuje bajt d pod adres i i zwraca poprzednią zawartość tej komórki.

```
dpeek( i)
char *i;
```

Funkcja *dpeek()* zwraca zawartość słowa (dwa bajty) o adresie i.

```
dpoke( i, w)
char *i;
int w;
```

Funkcja *dpoke()* wstawia słowo w pod adres *i*, *i+1* i zwraca poprzednią zawartość tego słowa.

```
val( s)
char *s;
```

Funkcja *val()*, podobnie jak instrukcja VAL z BASIC'a, zwraca wartość liczbową łańcucha *s*.

```
hval( s)
char *s;
```

Funkcja *hval()* zwraca wartość liczbową łańcucha *s* w kodzie szesnastkowym.

```
60 pp65 of6 lp54
POSTAC ZRODLOWA FUNKCJI ZAWARTYCH
        W ZBIORZE AIO.C

#define EOL 155
#define TAB 127
#define SPACE 32
/******************************************
 * FUNKCJA val(s) - zwraca do miejsca
 *  wywolania liczbe odpowiadajaca lancuchowi
 * (ciagowi cyfr) zawartemu w s
 ******************************************
val(s)
 char *s;
$(
 char c;
 int v,sign;
 while(c= *s)$(
  if(c==SPACE | c==EOL | c==TAB)++s;
  else break;
 $)
 sign=1;
 if(c=='+' | c=='-')$(
  if(c=='-')sign= -1;
  c= *++s;
 $)
 v=0;
 while(c>='0' & c<='9')$(
  v=v*10+c-'0';
  c=*++s;
 $)
 return v*sign;
$)
/******************************************
 * FUNKCJA hval(s)
 * zamienia liczbe szesnastkowa na
 * liczbe typu integer
 ******************************************
hval(s)
 char *s;
 $(
 int v;
 char c;
 v=0;
 while(c= *s)$(
  if(c==TAB | c==SPACE | c==EOL)++s;
  else break;
 $)
 while(1)$(
  if(c>='0' & c<='9')c=c-'0';
  else if(c>='a' & c<='f')c=c-87;
  else if(c>='A' & c<='F')c=c-55;
  else break;
  v=(v<<4)+c;


  c=*++s;
 $)
 return v;
$)
/******************************************
 * FUNKCJA copen()
 * otwiera zbior o nazwie zawartej
 * w lancuchu fn
 ******************************************
copen(fn,mode)
 char *fn,mode;
$(
 int k,io,r,dir;
 io=0;
 while(io<8)$(
  if(peek(832+(io<<4))==255)break;
  ++io;
 $)
 if(io==8)return -1; /* -1 otwarcie niemozliwe
 fn[k=strlen(fn)]=EOL;
 if(mode=='r')dir=4;
 else if(mode=='w')dir=8;
 else if(mode=='a')dir=9;
 else return -132;
 r=ciov(io,3,fn,k+1,dir,0);
 fn[k]=0;
 if(r<0)$(
  cclose(io);
```

```
  return r;     /* ujemny kod bledu */
 $)
 return io;     /* numer iocb        */
$)
/*********************************************
 * FUNKCJA open()
 * otwiera blok IOCB do operacji we/wy
 *********************************************/
open(iocb,ax1,ax2,fname)
 char iocb,ax1,ax2,*fname;
$(
 return ciov(iocb,3,fname,
  strlen(fname),ax1,ax2);
$)
/*********************************************
 * FUNKCJA close(i)
 * zamyka blok IOCB
 *********************************************/
close(i)
 char i;
$(
 return cclose(i);
$)
cclose(i)
 int i;
$(
 return ciov(i,12,-1,-1,-1,-1);
$)
/*********************************************
 * FUNKCJA cputc() - przesyla znak
 * do urzadzenia skojarzonego
 * z otwartym blokiem IOCB
 *********************************************/
cputc(c,unit)
 char c;
 int unit;
asm 12297;
/*********************************************
 * FUNKCJA cgetc()
 * odczytuje kolejny znak z urzadzenia
 * skojarzonego z otwartym blokiem IOCB
 *********************************************/
cgetc(unit)
 int unit;
 asm 12294;
/*********************************************
 * FUNKCJA getchar()
 * wczytuje jeden znak z ekranu (IOCB=0)
 *********************************************/
getchar()
$(
 return cgetc(0);
$)
/*********************************************
 * FUNKCJA putchar()
 * przesyla jeden znak na ekran (IOCB=0)
 *********************************************/
putchar(c)
 char c;
$(
 return cputc(c,0);
$)
/*********************************************
 * FUNKCJA gets(str)
 * wprowadza logiczny wiersz tekstu
 * z ekranu (IOCB=0) do lancucha str
 *********************************************/
gets(str)
 char *str;   /* wskaznik do lancucha str */
$(
 int r;
 if((r=ciov(0,5,str,120,-1,-1))<0)
  return r;
 str[r=dpeek(840)-1]=0;
 return r;
$)
/*********************************************
 * FUNKCJA cprints()
 * wyswietla na ekranie lancuch znakow
 * zawarty w lancuchu str

 *********************************************/
cprints(str)
 char *str;
$(
 return cputs(str,0);
$)
/*********************************************
 * FUNKCJA cputs() - wysyla na urzadzenie
 * skojarzone z otwartym blokiem IOCB
 * lancuch znakow zawarty w str
```

```
 *********************************************/
cputs(str,i)
 char *str;
 int i;
$(
 int k;
 if(k=strlen(str))
  return ciov(i,11,str,k,-1,-1);
 else return i;
$)
/*********************************************
 * FUNKCJA ciov() - gdy argumenty bad, blen, ax1
 * lub ax2 sa przy wywolaniu ustawione na -1,
 * to ich biezaca wartosc nie ulega zmianie
 *********************************************/
ciov(iocb,com,bad,blen,ax1,ax2)
 int iocb,com,blen,ax1,ax2;
 char *bad;
asm 12291; /* $3003 */
/*********************************************
 * FUNKCJA normalize()
 * zamienia litery z nazwy zbioru na duze litery
 * dodaje przedrostek D:, gdy go brak
 * dodaje rozszerzenie zdefiniowane przez fext
 *********************************************/
normalize(fname,fext)
 char *fname,*fext; /* wskazniki do nazwy i rozsz.
$(
 int i;
 char c,temp[20];
 /* zamiana nazwy zbioru na duze litery */
 i=0;
 while(c=fname[i])
   fname[i++]=toupper(c);
 if(--i < 0)return 0; /* nazwa pusta (brak) */
 /* fname[i] jest ostatnim znakiem nazwy */
 /* przedrostek D: */
 if(i==0 | (i==1 & fname[1]!=':')
  |(fname[1]!=':'&fname[2]!=':'))$(
  strcpy(temp,"D:");
  strcpy(temp+2,fname);
  strcpy(fname,temp);
  i=i+2;
 $)


 /* gdy brak '.', dodaj '.'fext */
 if(find(fname,i+1,'.')<0)$(
  fname[i+1]='.';
  strcpy(fname+i+2,fext);
  $)
$)
/*********************************************
 * FUNKCJA toupper() - zwraca kod duzej
 * (A .. Z), gdy argumentem
 * jest kod malej litery (a .. z)
 *********************************************/
toupper(c)
 char c;
$(
 return (c < 'a') || (c > 'z')
  ? c : c-32;
$)
/*********************************************
 * FUNKCJA tolower() - zwraca kod malej
 * (a .. z), gdy argumentem
 * jest kod duzej litery (A .. Z)
 *********************************************/
tolower(c)
 char c;
$(
 return (c < 'A') || (c > 'Z')
  ? c : c+32;
$)
/*********************************************
 * FUNKCJA strcpy(a,b)
 * kopiuje lancuch "b" do "a"
 *********************************************/
strcpy(a,b)
 char *a,*b; /* wskazniki do kop. lancuchow */
asm 12300;
/*********************************************
 * FUNKCJA move(a,b,len)
 * przepisuje len znakow z "a" do "b"
 *********************************************/
move(a,b,len)
 char *a,*b;
 int len;
asm 12303;
/*********************************************
 * FUNKCJA clear(a,len) -- zeruje len elementow
 * tablicy "a" poczawszy od a[0] */
```

```
clear(a,len)
 char *a;
 int len;
$(
 *a=0;
 move(a,a+1,len-1);
$)
/**********************************************

 * FUNKCJA usr() - wywolanie podpr. maszynowego
 * rezydujacego od adresu addr
 **********************************************
usr(addr)
 char *addr;
asm 12306;
/**********************************************
 * FUNKCJA find() - szuka pierwszego wystap.
 * znaku ch w obszarze pamieci od adresu addr
 * do adresu addr+len-1
 **********************************************
find(addr,len,ch)
 char *addr,ch;
 int len;
asm 12309;
/**********************************************
 * FUNKCJA strlen(str) - zwraca liczbe
 * znakow zawartych w lancuchu str
 **********************************************
strlen(str)
 char *str;
$(
 return find(str,30000,0);
$)
/**********************************************
 * FUNKCJA peek(i) - zwraca zawartosc
 * bajtu o adresie i
 **********************************************
peek(i)
 char *i;
asm 12312;
/**********************************************
 * FUNKCJA poke(i,d) - wpisuje wartosc
 * d do bajtu o adresie i i zwraca
 * stara zawartosc tego bajtu */
poke(i,d)
 char *i,d;
asm 12315;
/**********************************************
 * FUNKCJA dpeek(i) - zwraca zawartosc slowa
 * (dwoch bajtow) o adresie i
 **********************************************
dpeek(i)
 char *i;
asm 12318;
/**********************************************
 * FUNKCJA dpoke(i,w)
 **********************************************
dpoke(i,w)
 char *i;
 int w;
asm 12321;
```



A3. FUNKCJE GRAFICZNE ZDEFINIOWANE W BIBLIOTECE GRAPHICS.C

graphics(n)
char n;

Funkcja *graphics()* zmienia tryb graficzny ekranu podobnie jak instrukcja GRAPHICS w języku BASIC. Zwraca takie same wartości jak funkcja *open()*.

color(c)
char c;

Funkcja *color()* ustawia kolor pisaka dla funkcji *plot()* lub *drawto()* podobnie jak instrukcja COLOR w BASIC'u. Zwraca wartości przypadkowe.

drawto(x, y)
int x, y;

Funkcja *drawto()*, podobnie jak instrukcja DRAWTO w BASIC'u, kreśli linię od ostatnio oznaczonego punktu (*plot()*) do punktu o współrzędnych (x, y). Zwraca wartość 1 gdy nie ma błędów lub ujemny kod błędu CIO.

locate(x, y)
int x, y;

Funkcja *locate()* umieszcza kursor graficzny w pozycji (x,y) i zwraca wartość tego piksla lub ujemny kod błędu CIO. Stanowi ona odpowiednik instrukcji LOCATE w języku BASIC.

plot(x, y)
int x, y;

Funkcja *plot()* zapala punkt (x,Y) tak jak instrukcja PLOT języka BASIC. Zwraca wartość 1 lub kod błędu CIO.

position(x, y)
int x, y;

Funkcja *position()* umieszcza kursor graficzny w nowym miejscu o współrzędnych (x,y). Kursor zmienia swoje położenie dopiero przy kolejnej instrukcji wyprowadzenia.

setcolor(reg, hue, lum)
char reg, hue, lum;

Funkcja *setcolor()* ustawia kolor rejestru reg kombinacją wartości hue i lum. Działa podobnie jak instrukcja SETCOLOR BASIC'a.

fill(x, y, c)
int x, y;
char c;

Funkcja *fill()* stanowi implementację rozkazu FILL BASIC'a dla urządzenia S (ekran). Kreśli linię od ostatnio narysowanego punktu do punktu o współrzędnych (x,y) zmieniając kolor tła po prawej stronie kreślonej linii. Jest bardzo przydatna przy wypełnianiu kolorem dużych trapezoidalnych obszarów ekranu.

paddle(n)
char n;

Funkcja *paddle()* zwraca wartość wiosełka określonego przez n. Działa tak jak instrukcja PADDLE języka BASIC.

ptrig(n)
char n;

Funkcja ptrig() zwraca wartość przycisku określonego wiosełka. Działa podobnie jak instrukcja PTRIG BASIC'a.

stick(n)
char n;

Funkcja *stick()* zwraca wartość określonego dżojstika. Działa podobnie jak instrukcja STICK w języku BASIC.

strig(n)
char n;

Funkcja *strig()* zwraca wartość przycisku określonego dżojstika. Działa podobnie jak funkcja STRIG BASIC'a.

vstick(n)
char n;

Funkcja *vstick()* zwraca pionową składową położenia dżojstika. Gdy jest wychylony w górę zwraca wartość 1, gdy w dół -1, natomiast gdy znajduje się w położeniu neutralnym wartość 0.

hstick(n)
char n;

Funkcja *hstick()* zwraca poziomą składową położenia dżojstika. Gdy jest wychylony w lewo zwraca wartość -1, gdy w prawo 1, a gdy jest w położeniu neutralnym 0.

```
sound( voice, pitch, dist, volume)
char voice, pitch, dist, volume;
```
Funkcja *sound()* wywołuje efekty dźwiękowe podobnie jak instrukcja SOUND BASIC'a.

```
rnd( n)
int n;
```
Funkcja *rnd()* zwraca liczbę losową z zakresu od 0 do n-1 włącznie. Jeżeli chcemy uzyskać liczbę losową z zakresu od 1 do 10 należy użyć wyrażenia 1+rnd(10). Jeżeli n jest mniejsze niż 2, to funkcja *rnd()* zwraca wartość 0.



A4. FUNKCJE GRAFIKI PMG ZDEFINIOWANE W BIBLIOTECE PMG.C

```
pmcinit()
```
Funkcja *pmcinit()* inicjuje grafikę "graczy i pocisków" oraz zbiór znaków. Funkcja *pmcinit()* powinna być użyta tylko raz, przed wywołaniem funkcji *graphics()*.

```
pmcflush()
```
Funkcja *pmcflush()* kasuje bufory znaków graficznych i grafiki PMG zwalniając 4K bajtów RAM. Funkcja *pmcflush()* powinna być użyta tylko raz, tuż przed powrotem do systemu operacyjnego.

```
pmgraphics( i)
int i;
```
Funkcja *pmgraphics()* powinna być wywoływana po każdym odwołaniu do *graphics()*, aby ustawić rozdzielczość grafiki PMG. Funkcja *pmgraphics(1)* ustawia rozdzielczość jednoliniową, *pmgraphics(2)* ustawia rozdzielczość dwuliniową, a *pmgraphics(0)* wstrzymuje grafikę PMG.

```
hitclear()
```
Funkcja *hitclear()* czyści rejestry kolizji.

```
hitp2pf( from, to)
char from, to;
```
Funkcja *hitp2pf()* zwraca wartość 1 gdy gracz *#from* uderzy w pole *#to*.

```
hitp2pl( from, to)
char from, to;
```
Funkcja *hitp2pl()* zwraca wartość 1, gdy gracz *#from* trafi w gracza *#to*, a 0 gdy nie trafi. Gdy *#from* jest równe *#to*, to funkcja zwraca wartość 1.

```
pmclear( n)
char n;
```
Funkcja *pmclear()* zmazuje gracza n.

```
pmcolor( n, c, i)
char n, c, i;
```
Funkcja *pmcolor()* ustawia kolor gracza/pocisku na c, a jasność na i.

```
pmwidth( n, w)
char n, w;
```
Funkcja *pmwidth()* ustawia szerokość gracza n na wartość w:

w==0 normalny wymiar;
w==1 podwójny wymiar;
w==2 poczwórny wymiar;

```
pladdr( n)
char n;
```
Funkcja *pladdr()* zwraca adres bufora zawierającego gracza n.

```
plmove( n, x, y, shape)
char n, x, y, *shape;
```
Funkcja *plmove()* przesuwa gracza n na pozycję (x,y) (we współrzędnych aktualnego trybu graficznego) i pobiera jego kształt z tablicy znakowej *shape*. *Shape[0]* to wielkość gracza, a *Shape[1..size]* to bajty tworzące kształt gracza. Należy zostawić kilka bajtów zerowych przed i po, aby mieć pewność, że poprzednie kształty zostaną prawidłowo wymazane.

```
chget( c, s)
char c, *s;
```
Funkcja *chget()* wypełnia tablicę *s[0..7]* zawierającą kształt wewnętrznego znaku Atari, którego kod znajduje się w c.

```
choget( c, s)
char c, *s;
```
Funkcja *choget()* wypełnia tablicę *s[0..7]* znakiem oryginalnym z ROM.

```
chput( c, s)
char c, *s;
```
Funkcja *chput()* wypełnia zawartość tablicy *s[0..7]* dla znaku c. Aby umieścić np. kropkę na środku pustego pola znaku należy napisać:

```
chput(0,"\0\0\0\60\0\0\0");
```

UWAGA: Przed wykorzystaniem jakiejkolwiek funkcji z biblioteki PMG.C należy wywołać najpierw funkcję *pmcinit()*.



A5. FUNKCJE ZDEFINIOWANE W PLIKU PRINTF.C

Oprócz funkcji znajdujących się w pliku AIO.C istnieją w pliku PRINTF.C jeszcze dwie funkcje biblioteczne.

```
printf(s,...)
char *s;
```
Funkcja *printf()* jest standardową funkcją sformatowanego wyprowadzania zaimplementowaną w języku C. Akceptuje ona zmienną liczbę argumentów. Pierwszym z nich jest łańcuch formatujący, zawierający tekst jaki ma być wyprowadzony i znaki określające gdzie wstawić resztę argumentów. Znak % występujący w łańcuchu formatującym ma znaczenie specjalne. Znaki występujące po % określają sposób drukowania kolejnego argumentu. Pierwszy znak % dotyczy pierwszego argumentu, drugi znak % drugiego argumentu, itd. Jeżeli podano zbyt mało argumentów (lub zbyt mało znaków %), to zostaną wyprowadzone przypadkowe wartości.

Po znaku % można napisać jedną z następujących liter:

d – aby wydrukować liczbę dziesiętną
x – aby wydrukować liczbę szesnastkową
c – aby wydrukować znak
s – aby wydrukować łańcuch znaków
% – aby wydrukować znak %

Jeżeli chcesz aby argument miał określoną liczbę znaków podaj ją między znakiem % a znakiem formatu.

Standardowo następuje równanie drukowanej wartości do lewego krańca pola. Jeżeli chcesz aby równanie następowało do prawego skraju wstaw przed liczbą znak minus (-). Oto kilka przykładów:

```
printf("abcd"); -> abcd
printf("=%s=","abcd"); -> =abcd=
printf("=%5d=", 99); -> =99   =
printf("=%-5d=",99); -> =   99=
printf(%c %d %x", 65, 65, 65); -> A 65 41
```

```
fprintf(iocb, s, ...)
int iocb;
```

```
char $s;
```

Funkcja *fprintf()* jest podobna do *printf()*, ale ma dodatkowy argument iocb i wyprowadza dane do pliku o tym iocb. Funkcja *printf()* to właściwie *fprintf(0,...)*.

POSTAC ZRODLOWA FUNKCJI ZAWARTYCH W ZBIORZE PRINTF.C

```
/* funkcje wyprowadzania sformatowanego */

/**********************************************
 * FUNKCJA printf()
 **********************************************
printf()
$(
 int sp;
 sp=dpeek(210);
 sp=sp-(6+peek(dpeek(sp-8)));
 return fprint(sp,0);
$)
/**********************************************
 * FUNKCJA fprintf()
 **********************************************
fprintf()
$(
 int sp;
 sp=dpeek(210);
 sp=sp-(6+peek(dpeek(sp-8)));
 return fprint(sp+2,dpeek(sp));
$)
fprint(s,io)
 char *s;
 int io;
$(
 char *fstring,*arg,c;
 int cleft,i,size,sign;
 cleft=strlen(fstring=dpeek(s));
 while(cleft)$(
  if((i=find(fstring,cleft,'%'))<0)$(
   cputs(fstring,io);
   return; /* nie ma wiecej zn. formatujacych */
  $)
  if(i>0)$( /* znak przed % */
   ciov(io,11,fstring,i,-1,-1);
   fstring=fstring+i;
   cleft=cleft-i;
  $)
  /* pobierz argument numeryczny*/
  size=0;
  sign=1;
  while(cleft)$(
   ++fstring;
   --cleft;
   c= *fstring;
   if(c=='-')sign= -sign;
   else if(c>='0' & c<='9')
    size=size*10+(c-'0');
   else break;
  $)
  size=size*sign;

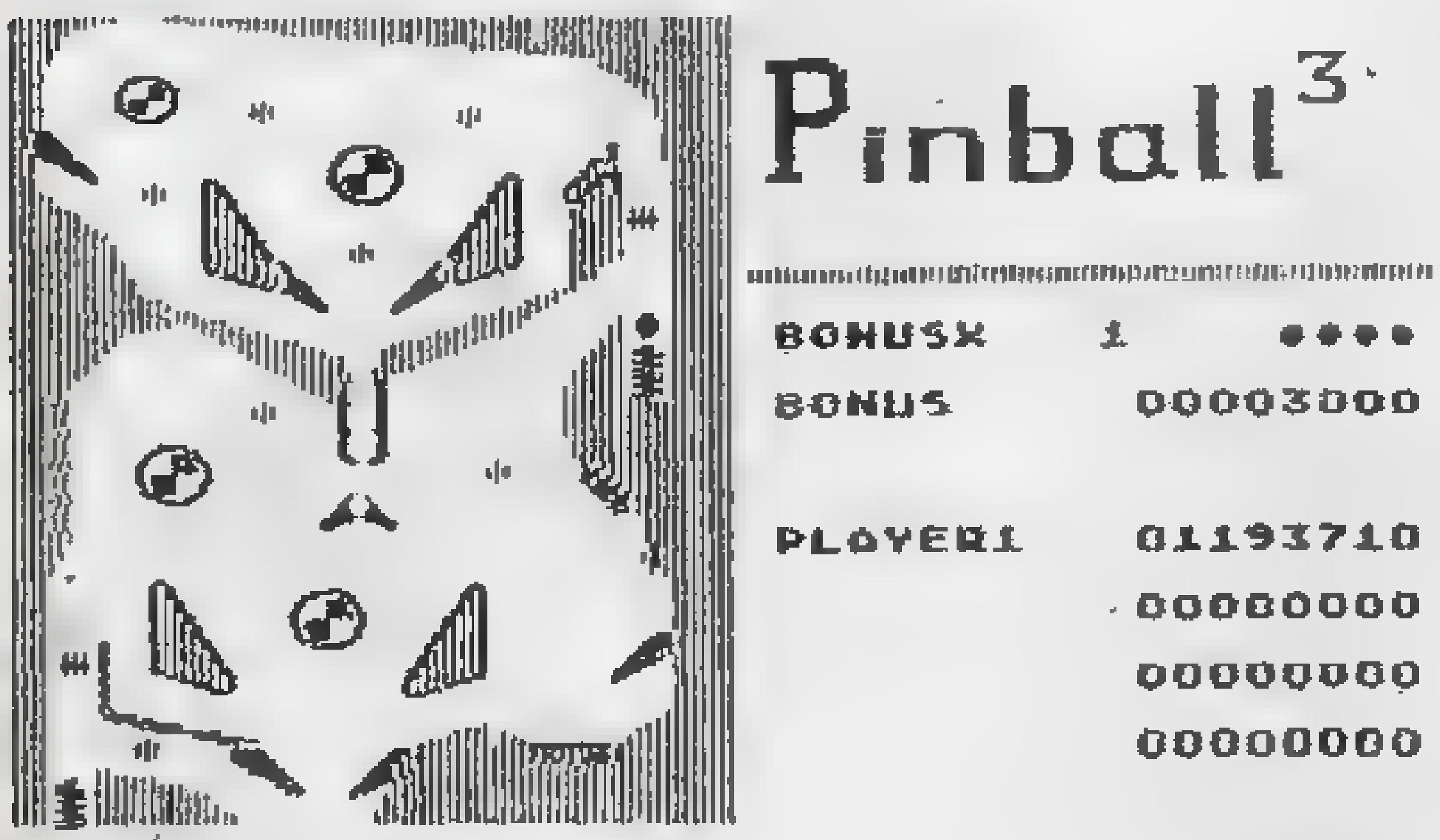
  arg=dpeek(s=s+2);
  /* sprawdzenie znaku formatujacego */
  /* tj. znaku wystepujacego po %    */
  if(c=='d')fprintd(arg,size,io);
    /* gdy znak formatujacy = d,      */
    /* to funkcja fprintd             */
  else if(c=='s')
    /* gdy znak formatujacy = s,      */
    /* to funkcja fprints             */
   fprints(arg,size,io);
  else if(c=='c')
    /* gdy znak formatujacy = c,      */
    /* to funkcja fprintc             */
   fprintc(arg,size,io);
  else if(c=='x')
    /* gdy znak formatujacy = x,      */
    /* to funkcja fprintx             */
   fprintx(arg,size,io);
  else $( /*  %%, etc. */
    /* gdy znak formatujacy = %,      */
    /* to wyprowadz ten znak          */
   cputc(c,io);
   s=s-2; /* i w tej sytuacji brak    */
          /* argumentu do wyprow.     */
  $)
  ++fstring;
  --cleft;
 $)
$)
/**********************************************
 * Funkcja fprintd() - wyprowadza liczbe dzies.
 **********************************************/
fprintd(arg,size,io)
 int arg,size,io;
$(
 /* bufor na lancuch odpowiadajacy    */
 /* wyprowadzanej liczbie             */
 char buf[7],flag,c;
 int i,k;
 i=flag=0;
 k=10000;
 if(arg<0)$(
  buf[i++]='-';
  arg= -arg;
 $)
 while(k)$(
  if(flag | (c=arg/k)>0)$(
   buf[i++]=c+'0';
   flag=1;
  $)
  arg=arg%k;
  k=k/10;
 $)
 if(flag==0) /* zero */

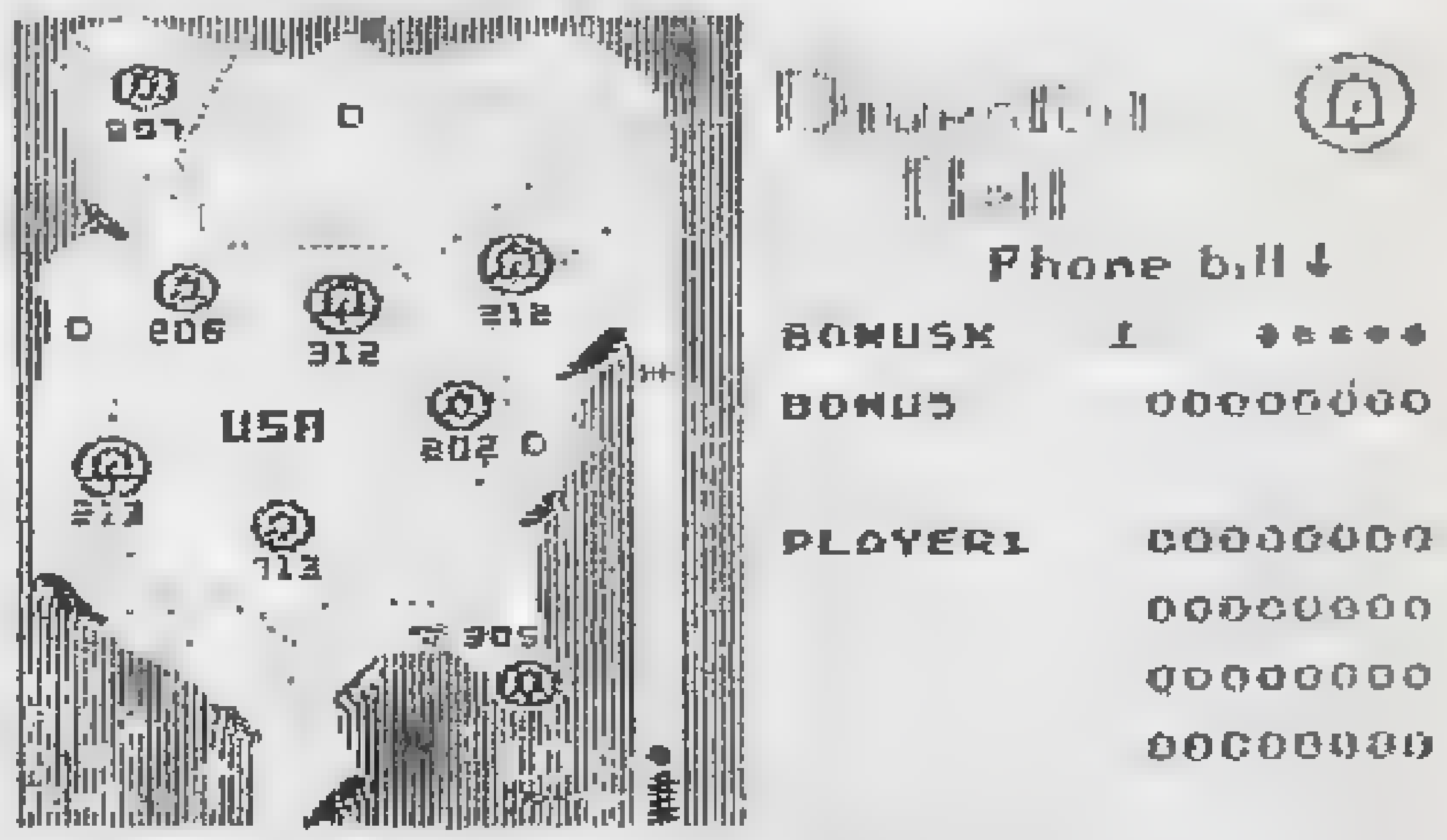
   buf[i++]='0';
  buf[i]=0;
  if(size > 0)
   fprintws(io,size-i);
  cputs(buf,io);
  if(size < 0)
   fprintws(io,(-size)-i);
$)
/**********************************************
 * FUNKCJA fprintx() - wyprowadza liczbe hex.
 **********************************************
fprintx(arg,size,io)
 int arg,size,io;
$(
 char buf[7],flag,c;
 int i,field;
 i=flag=0;
 field=12;
 while(field>=0)$(
  if(flag|
   (c=(arg >> field)&15))0)$(
   if(c<=9)buf[i++]=c+'0';
   else buf[i++]=c+'a'-10;
   flag=1;
  $)
  field=field-4;
 $)
 if(flag==0) /* zero */
  buf[i++]='0';
 buf[i]=0;
 if(size>0)fprintws(io,size-i);
 cputs(buf,io);
 if(size<0)fprintws(io,(-size)-i);
$)
/**********************************************
 * FUNKCJA fprintc() - wyprowadza znak
 **********************************************
fprintc(arg,size,io)
 char arg;
 int size,io;
$(
 if(size>0)fprintws(io,size-1);
 cputc(arg,io);
 if(size<0)fprintws(io,(-size)-1);
$)
/**********************************************
 * FUNKCJA fprints() - wyprowadza lancuch znakow
 **********************************************
fprints(arg,size,io)
 char *arg;
 int size,io;
$(
 int i;
 i=strlen(arg);

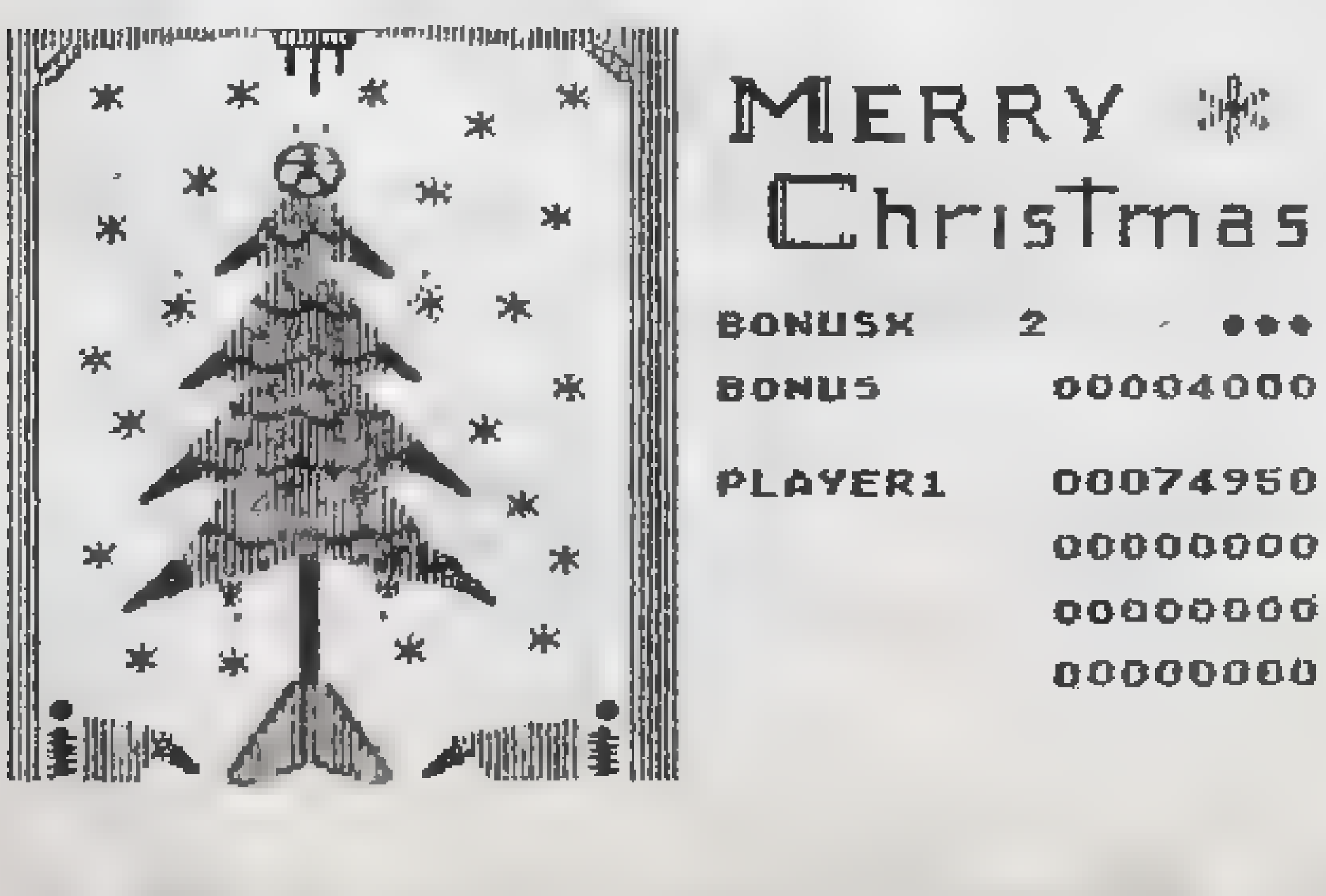
 if(size>0)fprintws(io,size-i);
 cputs(arg,io);
 if(size<0)fprintws(io,(-size)-i);
$)
/**********************************************
 * FUNKCJA fprintws() - wyprowadza n odstepow
 **********************************************
fprintws(io,n)
 int io,n;
$(
 while(n-- > 0)cputc(' ',io);
$)
```



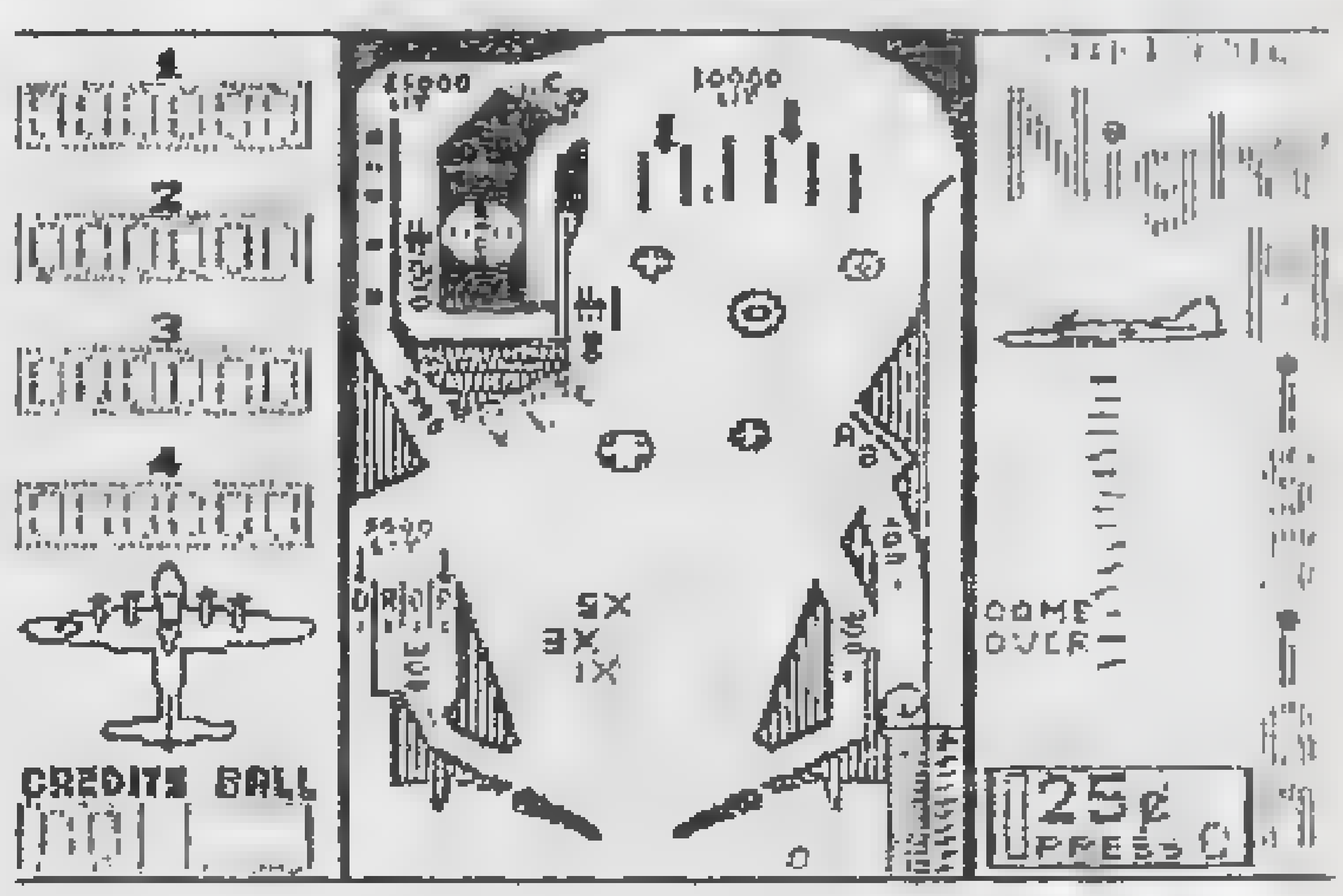
Jednymi z pierwszych automatów do gry, instalowanych w kraju, były bilardy. Zasada gry między poszczególnymi jego rodzajami mimo różnego stopnia skomplikowania czy kształtu automatu była ta sama. Po wrzuceniu monety wybijało się kulkę, a następnie dwoma przyciskami sterowało zestawem dźwigni tak, aby kulka jak najdłużej odbijała się od tych dźwigni oraz innych elementów automatu. Każde takie odbicie wiązało się z uzyskaniem kolejnych punktów, premii w postaci punktów lub dodatkowej kulki. Zdenerwowany gracz niejednokrotnie, chcąc przechylić szalę zwycięstwa na swoją korzyść, przyłożył pięścią w obudowę lub zaczął szarpać całym automatem. Wszystkie chwyty były dozwolone, byle tylko jak najdłużej utrzymać się przy grze. Powyższe metody zdobywania punktów stają się nieaktualne w przypadku przedstawionych poniżej bilardów komputerowych. Gier tego typu stworzono wiele. Różnica między nimi polega na wymyślnej grafice, bogactwie dźwięku, itd. W pierwszych trzech przykładach, tzn. PINPBALL 3, MERRY CHRISTMAS i DIVESTED BELL obsługa jest identyczna. Punktacja może być prowadzona jednocześnie dla czterech graczy. W tym celu naciskamy klawisz **OPTION** tyle razy, ile osób bierze udział w grze. Aby kulka została wystrzelona, należy uprzednio spowodować naciągnięcie sprężyny klawiszem **START** lub przyciskiem (FIRE) manipulatora podłączonego do portu pierwszego. We wszystkich tych grach dysponujesz pięciowa seriami rzutów. W każdej z tych gier przypada inna ilość kulek na serię. DIVESTED BELL - jedna, MERRY CHRISTMAS - dwie, i PINBALL 3 od trzech wzwyż; górna granica zależna jest od ilości zdobytych punktów, za które możesz otrzymać premię w postaci dodatkowych kulek. W grze możesz stosować jeden lub dwa manipulatory. Jeśli masz jeden manipulator, używaj portu 1. Użycie przycisku manipulatora powoduje wystrzelenie kulki. Ruch lewych dźwigni w bilardzie uzyskuje się przez naciskanie przycisku lub zwrot drążka manipulatora w lewo, natomiast ruch prawych dźwigni przez zwrot drążka w prawo. Mając dwa manipulatory, używa się tylko przycisków manipulatorów. Czasami zachodzi konieczność zwiększenia lub zmniejszenia naciągu sprężyny wystrzeliwującej kulkę. Efekt ten otrzymuje się, kierując drążek manipulatora do siebie lub od siebie.







Czwarty program NIGHT MISSION stanowi znacznie bogatszą ofertę niż poprzednie. Zarówno po stronie wizualnej jak i dźwiękowej wykorzystano w pełni możliwości Atari. Osobie stojącej w pobliżu mogłoby się wydawać, że to nie bilard a gra symulująca walkę powietrzną. Aby jednak mogło to mieć miejsce, należy wrzucić kilka monet 25 C, by otworzyć sobie kredyt. Robi się to klawiszem Q. Ilość gier, za które zapłaciliśmy, możemy odczytać pod napisem CREDITS. Następnie klawiszem S zapełniamy cztery tablice wyników po lewej stronie ekranu. Jedno naciśnięcie tego klawisza oznacza



że w grze uczestniczyć będzie jeden gracz. Dalsze naciskanie umożliwi grę kolejnym chętnym (maks. czterem). W tym czasie następuje również automatyczne naciągnięcie sprężyny wyrzucającej kulki. Naciąg tej sprężyny możesz regulować podobnie jak w poprzednich grach : manipulatorem podłączonym do portu 1 (do siebie naprężanie , od siebie zmniejszenie naciągu sprężyny) . Ruchem w lewo lub prawo tego manipulatora ustawia się kolor ramki . Wystrzelenie kulki następuje po naciśnięciu przycisku jednego z manipulatorów . Manipulatorem z portu 2 można wykonywać jeszcze dwie czynności : zmiana koloru planszy (drążek w lewo lub prawo) , zmiana nasycenia koloru (drążek w górę lub dół). Stwarza to możliwość wprowadzenia utrudnień . W celu poprawienia ci możliwości orientacji jak długo będziesz jeszcze grał , pod napisem BALL podawana jest liczba wystrzelonych kulek .

MOVIE  MUSICAL

MADNESS

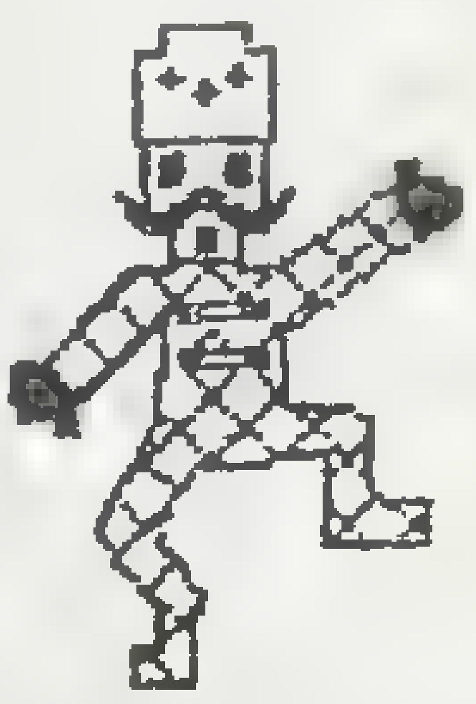
Jeżeli chcesz pobawić się w tworzeniu scenografii filmowej , a także komponowania do niej akcji , załaduj ten program . Gdy pojawią się na ekranie jego wstępne możliwości : COLOR (kolor) czy BLACK & WHITE (czarno - białe) ; ustaw manipulatorem gwiazdkę nad tym wariantem , który wybierasz , a następnie naciśnij przycisk . Pojawią się trzy postacie . Lewa z literką S na piersiach, prawa z literką C. Wybranie którejś z nich następuje przez przesunięcie manipulatorem w lewo lub prawo znajdującej się nad nimi gwiazdki a następnie po naciśnięciu przycisku. Postacie znikają . Pozostają tylko dwa rzędy po 20 prostokątów (góra i dół ekranu) . W tym momencie program jest gotów do tworzenia scenografii . Dolny rząd prostokątów to 20 różnych melodii , które możesz wykorzystać przy realizacji swojego dzieła. Górny rząd to twój magazyn rekwizytów , a pierwszy i ostatni prostokąt są "garderobami" statystów, którzy mogą uatrakcyjnić twoje dzieło . Niestety, w scenie może brać udział tylko jeden aktor i jeden statysta . Przy pomocy pierwszego prostokąta możesz otrzymać następujących statystów

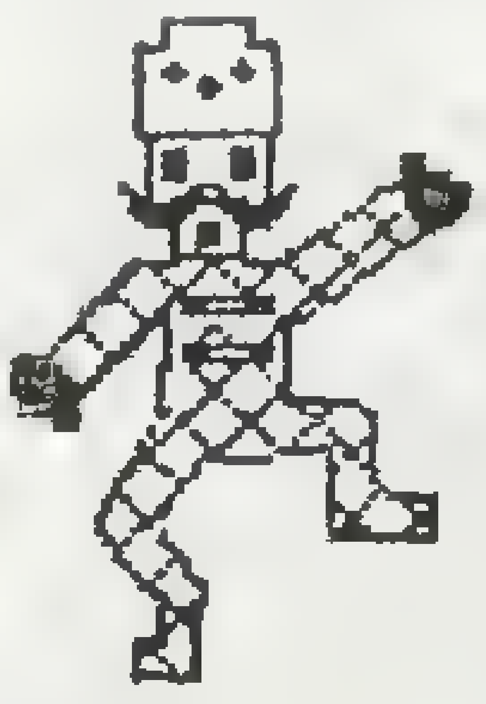
- czarodziej ;
- zbójnik ;
- statek kosmiczny ;
- piesek .
- reklama (EAT)

Dwudziesty prostokąt daje Ci

- statek kosmiczny ;
- babę Jagę ;
- pociąg ;
- kosmoludka ;
- duszka

Aby otrzymać np. zbójnika, należy manipulatorem nasunąć migającą gwiazdkę na pierwszy prostokąt , nacisnąć jego przycisk , ukazującego się czrodzieja skasować klawiszem SPACE i ponownie nacisnąć przycisk manipulatora, a gdy ukaże się zbójnik , potwierdzić jego wybór przez naciśnięcie przycisku .





Innych statystów uzyskuje się identycznie w kolejności, w jakiej zostały wyszczególnione powyżej . Teraz możesz przystąpić do tworzenia scenografii . Poszczególne jej elementy wybieramy identycznie jak statystów z tym , że jeden prostokąt odpowiada jednemu rekwizytowi. Jest ich w sumie osiemnaście .

Możesz je ustawić dowolnie na planie, lecz pamiętaj , że dopóki nie potwierdzisz przez naciśnięcie przycisku manipulatora , możesz go dowolnie przemieszczać na planie lub skasować klawiszem spacji . A oto wykaz wszystkich rekwizytów , jakie możesz otrzymać w kolejności od strony lewej

1. baszta ,
2. minaret ;
3. dom ;
4. fragment muru ;
5. palma ;
6. drzewo ,
7. płot ;
8. pianino ;
9. planeta ,
10. tulipan ;
11. drogowskaz ,
12. dom (hotel) ,
13. wieżowiec ,
14. słoń
15. reklama (SHOW) ;
16. księżyc ,
17. latarnia uliczna ,
18. stara lampa .

Sposób ich doboru oraz ilość jest zupełnie dowolna . Kiedy uznasz że scenografia jest już dostatecznie rozbudowana , przychodzi kolej na dobranie do niej odpowiedniego podkładu muzycznego . Przejście z górnego rzędu prostokątów następuje przez skierowanie drążka manipulatora do dołu i następnie wybraniu jednej z dwudziestu melodii . Przy jej wyborze postępuje się identycznie jak przy rekwizytach . Gdy dojdziesz do wniosku , że melodia odpowiada twojej koncepcji , naciśnij jeszcze raz przycisk manipulatora . Pojawi się polecenie PRESS START TO BEGIN ACTING . Naciśnij wtedy klawisz



START . Rozpocznie się realizacja sceny filmu . Przemieszczaj postać zgodnie z Twoją koncepcją . Tworzysz w ten sposób akcję danej sceny . Czas trwania sceny jest uzależniony od długości melodii . Jeżeli uprzednio był wprowadzony aktor np. czarodziej , pies itp. , pojawi się on i będzie wykonywał charakterystyczną dla siebie czyn-



mość. Koniec melodii oznacza koniec sceny. Na ekranie pojawiają się ponownie trzy postacie. Możesz rozpocząć tworzenie scenografii następnej sceny jak również jej pozostałych elementów np. zmiana statysty, melodii, itp. Możesz również poprzez naciśnięcie klawisza SELECT przejść do nowej opcji, tzn. PUSHES (odtwarzanie całego utworu), RETAKES (zlikwidowanie ostatnio utworzonej sceny), THE END (koniec filmu). Jak zwykle, wyboru dokonuje się przy pomocy manipulatora. Nie należy przerażać się przypadkowym wybraniem opcji THE END ponieważ komputer prezentuje Ci cały film, a następnie proponuje: PRESS OPTION TO VIEW IT AGAIN (ponowne odtworzenie filmu po naciśnięciu klawisza OPTION) i PRESS SELECT TO MAKE A NEW FILM (skasowanie stworzonego filmu, a następnie przejście do tworzenia nowego przez naciśnięcie klawisza SELECT). Przedstawiony powyżej program jak również jego możliwości z pewnością nie usatysfakcjonują wszystkich. Brak jest tu działek laserowych, strzelających kosmitów czy zdradliwych tuneli. Jednak szansa sprawdzenia się w bądź co bądź prestiżowym zawodzie powinna skusić wielu młodych ludzi. Nikt nie rodzi się reżyserem czy scenografem, tego trzeba się nauczyć.



H A C K E R

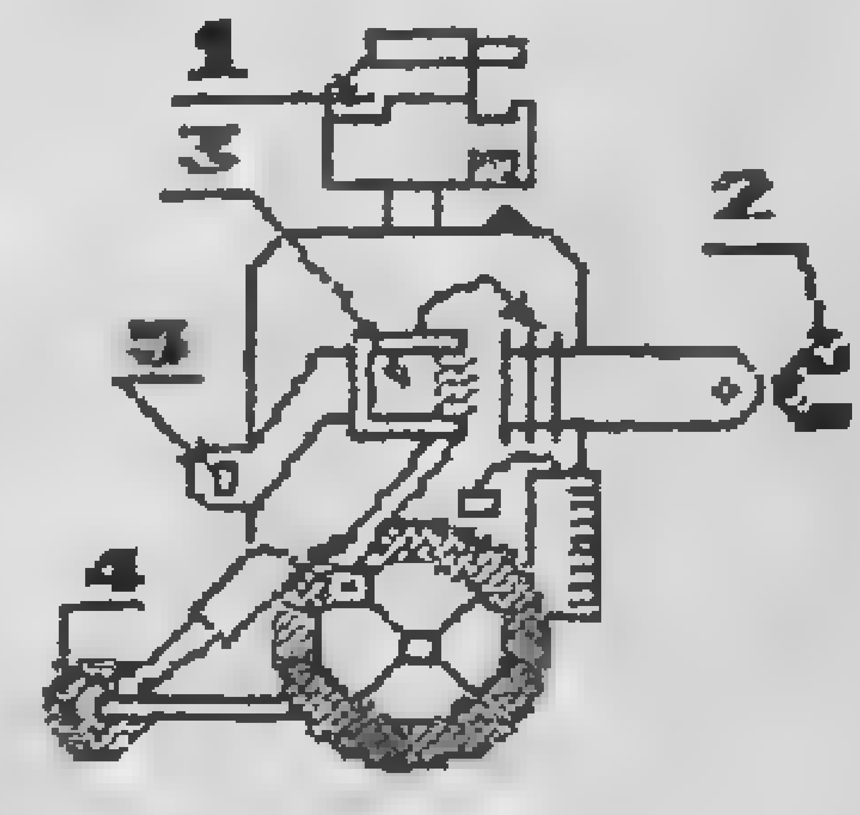
Właściwie nie wiadomo w jakim celu i kiedy została wybudowana sieć tuneli łącząca prawie wszystkie stolice i największe miasta świata. Do naszych czasów dotrwały w stanie nienaruszonym tylko ich fragmenty. Dzięki przezorności Narodów Zjednoczonych zostały wyremontowane, a następnie zaadoptowane do super szybkiego przemieszczania ludzi i sprzętu. I nadszedł taki czas, kiedy trzeba było praktycznie to wykorzystać. Wydzielona grupa przedstawicieli Narodów Zjednoczonych opracowała plan ochrony Ziemi przed intruzami z zewnątrz. Nazwano go planem MAGMA. Aby nie dopuścić do rozszyfrowania go w razie wykradzenia, został on podzielony na dziewięć części. Niestety, jak można było przypuszczać, wrogowie dowiedzieli się o istnieniu planu. Polecili swoim szpiegom wykraść go. Szpiedzy dokonali tego. W obecnej chwili poszczególne części planu znajdują się w różnych miastach rozrzuconych po całej kuli ziemskiej. Jeżeli w odpowiednim czasie nie odzyskasz skradzionych części planu, wrogowie przejmą kontrolę nad Ziemią i jej mieszkańcami. A więc do dzieła.

Uruchomienie programu i gra.

Po załadowaniu się programu komputer sprawdza, czy jesteś fachowcem, czy jeszcze człowiekiem którego należy podszkolić. Zada Ci pytanie - LOGON PLEASE ? Wprowadź teraz kod wywoławczy. Dla ułatwienia podajemy, że jest nim nazwa jednego z kontynentów. Gdy będziesz znał to hasło, ominie cię test robota i przejdziesz od razu do drugiego etapu. Załóżmy jednak, że nie znasz hasła, a więc na wymienione wyżej pytanie napisz H i **RETURN**. Komputer powiadamia was wtedy o zmianie hasła i zadaje jeszcze raz pytanie jak wyżej. Należy naciskać wtedy tylko **RETURN**. Gdy to wykonasz, pojawi się rysunek robota. Twoim zadaniem będzie odgadnięcie tych w robocie miejsc, których nazwy będą pojawiały się na dole ekranu. Aby przejść do następnego etapu, należy bezbłędnie odnaleźć wszystkie części. Kursor sterowany jest przy pomocy manipulatora. Po ustaleniu miejsca występowania części należy naciskać przycisk (FIRE). Jeżeli odpowiedź będzie właściwa, komputer po pytaniu napisze ci OK. i zada następne. Jeżeli natomiast pomylisz się, pytanie będzie ponawiane dotąd, aż odpowiesz prawidłowo a ponadto test zostanie jeszcze raz powtórzony. Dla ułatwienia, pokazany został poniżej robot z umiejscowieniem poszczególnych części.

TEST ROBOTA SRU.

1. INFRARED VIDEO IMAGE SENSOR - ośrodek wizyjny
2. ASYNCHRONOUS DATA COMPACTOR - manipulator
3. HYDRAULIC MOTIVATOR - napęd
4. PHLAMSON JOINT - mechanizm kierunku
5. THELMAN PORT - gniazdo połączeń zewnętrznych



Drugi etap rozpoczyna się od pytania o twe imię. Gdy je wprowadzisz, zostaniesz poproszony o ustawienie strefy czasowej na Ziemi. Poruszając manipulatorem w lewo lub prawo ustawiasz tak porę dnia, by ci odpowiadała. Dla utrwalenia sytuacji naciskasz przycisk. Komputer informuje wtedy, że nastąpiła weryfikacja czasu. Robot SRU znajduje się w tej chwili na południowym Atlantyku. Naciśnij **RETURN**. Gdy wykonasz to polecenie, rozpoczniesz właściwą grę. Zobaczysz tablicę wskaźników (rysunek poniżej), na której będziesz otrzymywał dane

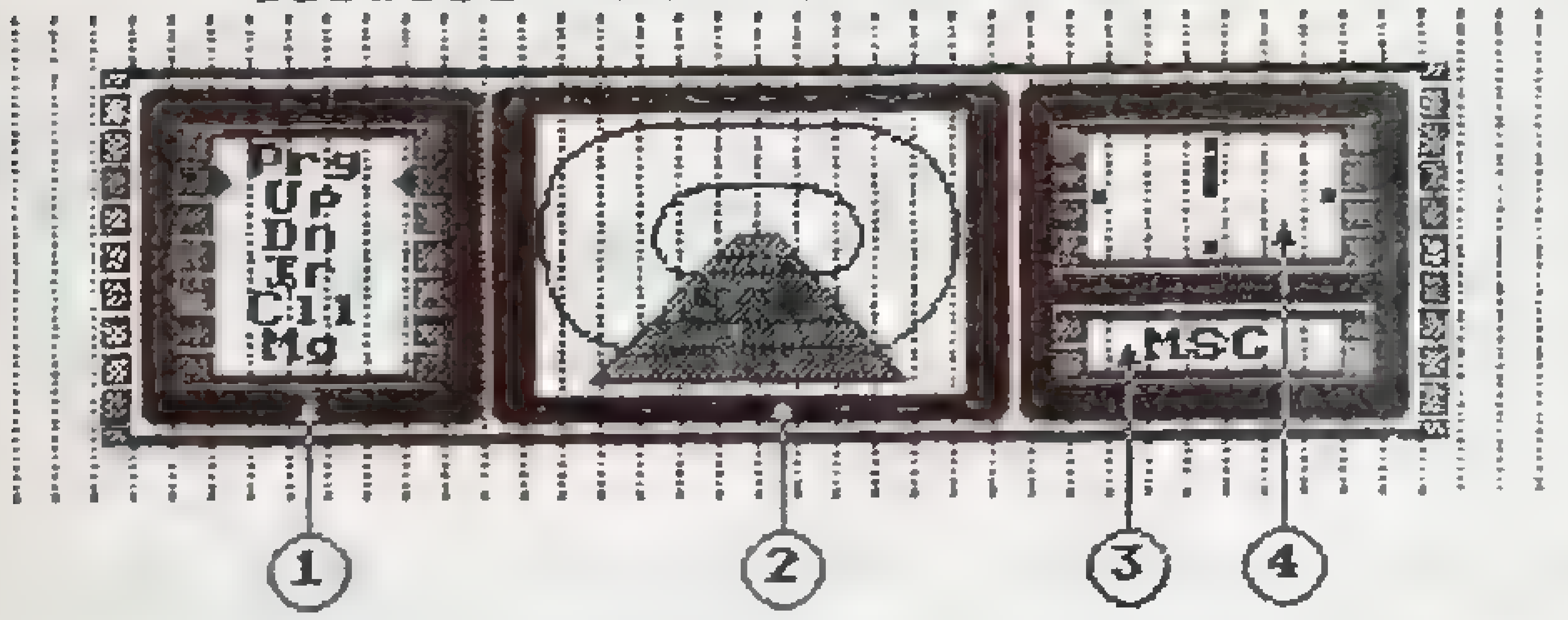
1. Jaki jest aktualnie wykonywany rozkaz,
2. Na monitorze obraz, jaki widzi

przed sobą robot;
3. Sygnalizacje zgłaszania się kierownictwa akcji (migający skrót MSG);
4. Na wskaźniku określenie kierunku w jakim zwrócony jest robot.

START - powoduje powrót na początek gry

Kierowanie robotem w tunelu odbywa się za pomocą manipulatora. Przy skierowaniu drążka w kierunku, którym chcemy się poruszać, robot obraca się i jeżeli dalej przytrzymamy, tym samym powodujemy zaprogramowanie fragmentu drogi. Gdy naciśniemy przycisk, robot przebędzie ten fragment tunelu. W zależności od tego, w którą stronę został zwrócony robot, będziesz widział dalszy ciąg tunelu lub ścianę. Schemat tunelu przedstawiono na rysunku poniżej.

Tablica wskaźników robota



Po rozpoczęciu gry z pewnością zgłosi się kierownictwo akcji; wciśnij wtedy klawisz M. Otrzymasz szereg instrukcji. Spróbuj je przetłumaczyć. Możesz w ten sposób wzbogacisz swój słownik o nowe określenia. Jeżeli jednak nie poradzisz sobie, nie przejmuj się tym. W czasie gry będziesz miał do czynienia z dwoma grupami rozkazów. Pierwsze to te, które będziesz wydawał pod Ziemią, a mianowicie klawisz

U - oznacza wyprowadzenie robota na powierzchnię, gdy znajduje się on pod Ziemią;

I - włącza noktowizor gdy okazuje się, że na powierzchni jest noc.

C - przywołuje się szpiega, by dokonać transakcji;

D - powrót robota z powrotem do tunelu;

M - nawiązanie (na polecenie MSG) kontaktu z kierownictwem w celu otrzymania instrukcji.

Druga grupa jest związana tylko z operacjami wykonywanymi na powierzchni Ziemi (po przywołaniu szpiega) i tak naciśnięcie klawisza:

D - jak poprzednio oznacza powrót robota do tunelu;

I - zmiana proponowanego szpiegowi przedmiotu na inny, gdy ten nie jest nim zainteresowany,

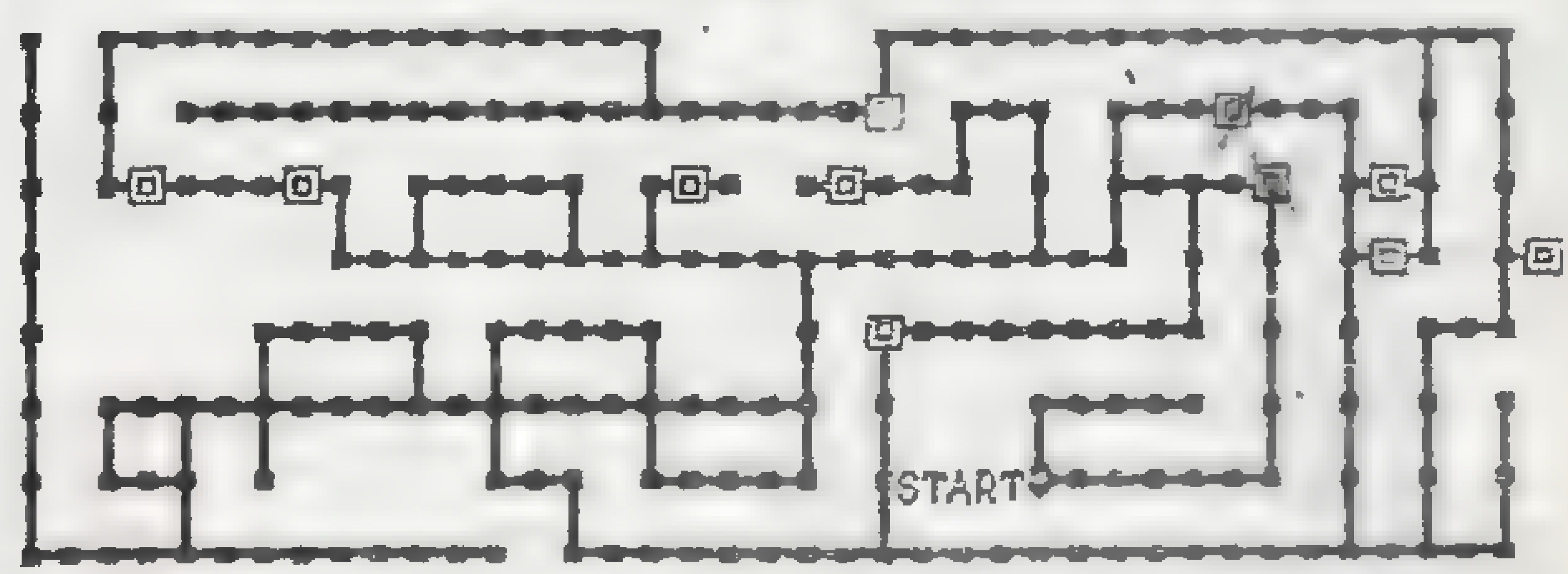
Y - wyrażenie zgody na kupienie od szpiega przedmiotu,

N - odmowa kupna;

E - daje możliwość obejrzenia dotychczas wykupionych części planu;

U - jak w pierwszym rodzaju rozkazów;

Plan podziemnych korytarzy.



Załóżmy teraz, że znajdujesz się pod miastem. W takim przypadku należy nacisnąć klawisz U. Robot zostanie wyniesiony na powierzchnię. W zależności od tego, w jakiej strefie czasowej znajdujesz się, może być na powierzchni dzień (nic nie naciskasz) lub noc (wtedy musisz włączyć noktowizor klawiszem I).

Manipulatorem możesz uzyskać widok tego, co znajduje się w zasięgu widzialności robota. Po naciśnięciu klawisza C zostaje przywołany do robota szpieg. Po przyjściu proponuje ci dokonanie wymiany. Nie zawsze pytania szpiega będą zrozumiałe, gdyż z reguły będą posługiwać się swoimi narodowymi językami. Klawiszem I będziesz mógł pokazywać swoje oferty. Gdy szpieg zaakceptuje którąś z nich, automatycznie da ci za nią część planu (zobaczysz ją na ekranie), a gdy naciśniesz RETURN, zacznie przedstawiać swoje propozycje do sprzedania. Przedmioty te często będą związane z krajem, w którym będziesz aktualnie się znajdował. Dla przykładu w Egipcie, szpieg zaproponuje ci szmaragdowego skarabeusza i złotą statuetkę TUT, w Grecji imitację starożytnego przedmiotu i urnę grecką, itd. Możesz je kupować, wtedy naciskasz klawisz Y lub nie (klawisz N). Następnie przyciskając D, ponownie opuszczasz robota do tunelu i kierujesz go do następnego miasta w celu zdobycia kolejnych części planu. Przy odwiedzaniu miast powinna być zachowana pewna kolejność, gdyż szpiedzy najczęściej za plan będą żądali przedmiotów, które kupiłeś wcześniej. Odkrywając rąbka tajemnicy, można powiedzieć, że np. szpieg w Anglii będzie chciał za plan chronograf, w Grecji szmaragdowego skarabeusza czy w USA (San Francisko) album z autografami Beatlesów, itd. Ze względu na ograniczony czas gry musisz sprawnie pokonywać poszczególne etapy. Jeżeli zbyt wolno będziesz zdobywał poszczególne części planu, przeciwnik może cię uprzedzić i zakończysz swą wędrówkę. Z tego samego powodu przeciwnik może wysłać satelity szpiegowskie do unieszkodliwienia robota. Gdy zostaje on wykryty, przybywa pytanie o kod rozpoznawczy. Jeśli odpowiesz źle, robot zostanie zlikwidowany. Znając ten kod zmylisz satelity podając robota za ich sprzymierzeńca. A oto kody w kolejności, w jakiej będą żądane przez satelity

1. MAGMA. LTD.
2. AX-0310479

3. HYDRAULIC
4. AUSTRALIA
Krążąc po tunelu, uważaj na rejon zastrzeżony. Znajduje się on w rejonie Australii. Gdybyś przypadkiem wkroczył w jego obszar, gra skończy się ze względu na zniszczenie robota. Jak widać, zabawa nie jest taka prosta, ale czyż nie warto dowiedzieć się co kryje plan MAGMA ?



W historii istot myślących mieliśmy okresy walk między rodzajem ludzkim, następnie między ludźmi a mieszkańcami innych układów słonecznych, aby w czwartym tysiącleciu naszej ery doprowadzić do zawarcia pokoju w całym dotychczas zbadanym obszarze galaktyki. Doprowadziło to do eksplozji dobrobytu. Ludzie z poprzednich epok nie byliby w stanie zrozumieć tego pojęcia ani tego, że przez blisko tysiąc lat nie było konfliktów, których rozstrzygnięcie odbywałoby się przez walkę zbrojną.

Oczywiście taka sielanka nie mogła trwać wiecznie. Znalazła się w końcu grupa ludzi, która wydostała się spod kontroli unii. Utworzyli oni w układzie słonecznym AUR niezależną bazę. W tym momencie stało się jasne, że kończy się okres spokoju. Mieszkańcy układu słonecznego PROCYON, którego Ty jesteś przedstawicielem, widząc zagrożenie swojego bytu, przystąpili do budowy super statku mogącego stawić czoło ewentualnym najeźdźcom. To właśnie tobie przypadł zaszczyt go pilotować i walczyć. Po załadowaniu programu wybierz klawiszem **SELECT** stopień trudności (DIFFICULTY) od 1 do 3. Natomiast wciśnięcie **START** lub przycisku manipulatora uruchamia silniki statku wyrzucające go w przestrzeń międzyplanetarną. Od tego momentu rozpoczyna się Twoja przygoda. Jak zapewne już zauważyłeś, siedzisz przed pulpitem sterowniczym. Na dużym ekranie obserwujesz sytuację wokół najbliższego otoczenia statku. Powyżej ekranu po lewej stronie masz aktualny wynik zdobytych punktów, po prawej wartość rekordową. Środek to różnego rodzaju komentarze, z których najczęściej występujące to

- ENEMY IN RANGE (wróg w zasięgu),
- SYSTEM MALFUNCTION (wadliwie działa system);
- ALLIED CITY DESTROYED (zniszczone zostało bronione miasto);
- BARRIER UNDER ATTACK (zaatakowana bariera układu);
- HULL TEMP CRITICAL (krytyczna temperatura kadłuba statku);
- WARP ENGINES ENGAGED (niebezpieczeństwo uszkodzenia silników);
- TRAJECTORY PLOTTED (plan układu słonecznego);
- ENERGY LEVEL CRITICAL (krytyczny poziom energii).

Poniżej ekranu znajduje się cały szereg wskaźników, a mianowicie

1. Sygnalizator pracy działek laserowych ;
2. Sygnalizator pracy działek termicznych ;
3. Sygnalizator pracy działek neutronowych ;
4. Wskaźnik ilości amunicji termicznej ;
5. Wskaźniki ostrzegawcze ;
6. Wskaźnik poziomu energii statku ;
7. Wskaźnik temperatury działek laserowych
8. Urządzenie czasowe
9. Ekran radaru .
10. Połączenie foniczne z centrum unii ;
11. Wskaźniki informujące o aktualnej sytuacji na zewnątrz statku



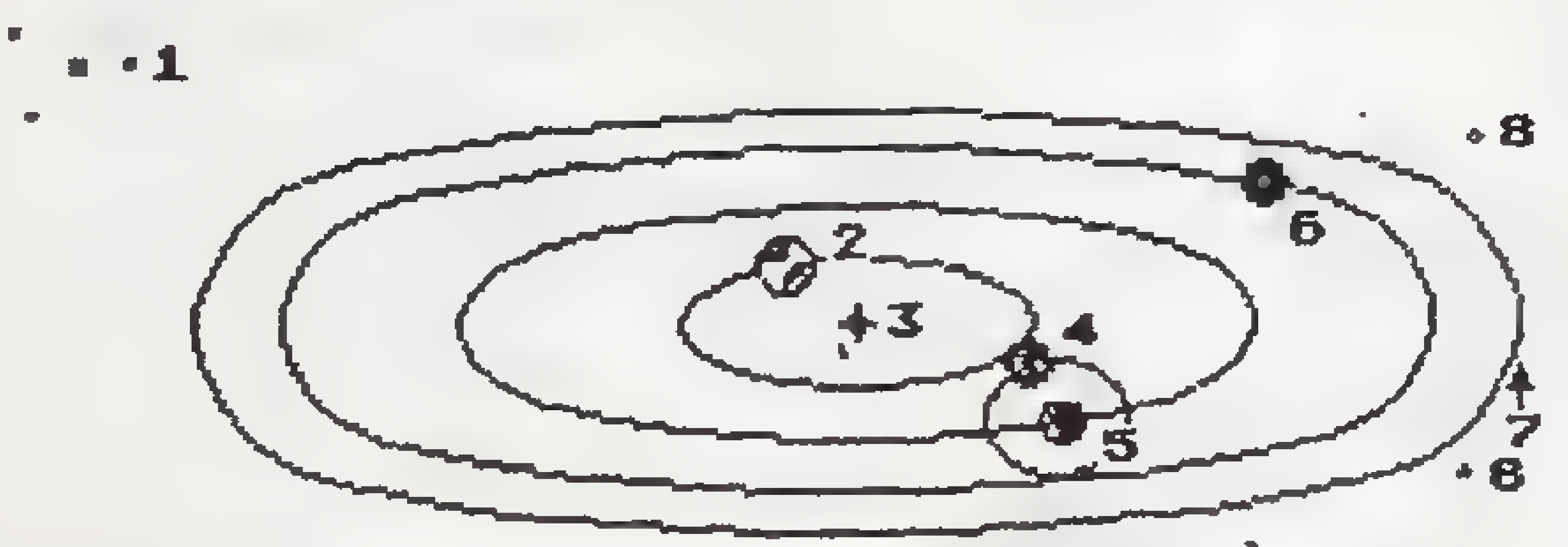
Najprawdopodobniej za chwilę nadlecą pojazdy ze szwadronu KODAN-a. Poruszają się po trzy atakując trzema falami. Ich pociski nie są groźne, lecz potrafią być dokuczliwe. Twój jeden celny strzał likwiduje pojazd przeciwnika, dając 100 punktów premii. Jeżeli uda ci się zlikwidować wszystkich atakujących, czeka Cię przeprawa z następną partią znacznie trudniejszych do zniszczenia trzech statków. Każdy musi być trafiony trzykrotnie pociskiem termicznym. Za zniszczenie statku otrzymuje się 500 punktów. Zmiana rodzaju pocisków wykonuje się automatycznie. Po zniszczeniu armady należy odnaleźć następną i zniszczyć, itd. W celu przeszukania układu słonecznego wciśnij **SPACE**. Otrzymasz na ekranie obraz tego układu z zaznaczonymi aktualnymi pozycjami przeciwnika. Chcąc przenieść się w inne miejsce układu, należy manipulatorem przeciągnąć linię do punktu, w którym chcemy się znaleźć, a następnie nacisnąć przycisk. Prawidłowe ustawienie punktu docelowego zostanie potwierdzone przez powstanie otoczki wokół tego miej-

sca . Najszybciej armadę można zniszczyć , gdy przekracza ona barierę ochronną układu , wystarczy wtedy przenieść się w to miejsce i oddać jeden strzał , a armada przestanie istnieć , za co otrzymasz 500 punktów . Na mapie układu słonecznego statki KO-DAN-a zazna-nowymi , które można wystrzelić , gdy przełączysz rodzaj prowadzonego ognia , np. klawiszem **OPTION** .

W ferworze walki nie zapomnij o wyczerpującej się Ci energii . Gdy stwierdzisz, że posiadasz jej zbyt mało do prowadzenia dalszej walki, przenieś statek w pobliże gwiazdy

- nazwę , np. KO-DAN
- kod identyfikacyjny , np.

ID 606.4615.4961
- masę planety , np. M=5.3E27 g
- promień planety, np. R=5.6E08 cm
- otoczenie (ENVIRONMENT) , np. umiarkowane (TEMPERATE)



Skład układu planetarnego Celos IV .
1. Układ planetarny XUR
2. SERIDUS
3. Słońce CELOS IV
4. GALEN
5. RYLOS
6. ARCANUS
7. Bariera ochronna układu
8. Eskadry statków z KO-DAN

Układ planetarny gwiazdy Celos IV

czone są w postaci trzech punktów. Gdy dojdziesz do wniosku , że pozbyłeś się wojsk przeciwnika w swoim układzie słonecznym , powinieneś zaatakować układ słoneczny XUR . Zasady walki w układzie przeciwnika są prawie identyczne jak wtedy , gdy broniłeś swojego terenu . Różnica polega jedynie na tym , że atakować Cię będą znacznie liczniejsze grupy statków. Ponadto będziesz musiał niszczyć bazy przeciwnika rozmieszczone na planetach układu . Za zniszczenie bazy otrzymujesz 1000 punktów. Baze niszczy się pociskami neutro- układu , w którym się znajdujesz . Nie możesz jednak zbyt długo tam przebywać . Po usłyszeniu sygnału ostrzegawczego lub wcześniej postaraj się , jak najszybciej przenieść w inne miejsce układu . Gdy tego nie uczynisz , Twój statek spali się i zabawę będziesz musiał rozpocząć od nowa poprzez jednoczesne przyciśnięcie klawiszy **START** i **SELECT** . W czasie przenoszenia się otrzymujesz szereg mniej lub bardziej potrzebnych informacji o miejscu , w które chcesz się udać . Jeżeli jest to planeta , uzyskujesz jej -

- ilość miast (CITIES) lub baz (BASES)
- liczbę statków przeciwnika(DESTROYERS) nad planetą
- liczbę pojazdów bojowych (FIGHTERS) przeciwnika nad planetą .

Aby zwyciężyć przeciwników z KO-DAN , powinieneś nie tylko celnie strzelać , ale wykazać się przebiegłością i pewną dozą strategicznego myślenia . Nie zawsze prawidłowe posunięcia muszą w sposób zadowalający wpływać na następne decyzje . A więc walczcie i zwyciężajcie .



Skład układu planetarnego XUR .
1. Słońce XUR
2. GRON
3. KO-DAN
4. MORKOTH
5. Eskadry z KO-DAN
6. Układ planetarny Celos IV

Układ planetarny gwiazdy XUR

" IKS " - dodatek " Żołnierza Wolności ". Redaguje Wiesław Catera (kierownik zespołu). Rada programowa: Krzysztof Chwarra, Romuald Gład, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasucki , Ireneusz Miernik , Ludwik Piela . Jacek Stanisławski . Adres redakcji 00-950 Warszawa ul . Grzybowska 77 , telefon centrali 20-12-61 w. 485. Telex 813664 . Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów . Nakładem Wydawnictwa " Czasopisma Wojskowe ", Warszawa ul. Grzybowska 77. Druk offsetowy - Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego.

Nr zam 1507 Nr ind 39628 U-33

INFORMATYKA
KOMPUTERY
SYSTEMY

zeszyty programów komputerowych

nr 3/88

Dodatek „Żołnierza Wolności" Cena 200 zł

GRAFIKA

ATARI

GRY



SPIS TREŚCI

Kreślenie odcinków prostych — 3
Kolory — 4
Przekształcanie rysunków — 7
Zmiana miejsca położenia — 8
Zmiana rozmiarów (skalowanie) — 9
Zmiana orientacji (obrót) — 10
Przekształcenie kombinowane — 11
Ruchome rysunki — 13
Przekształcenia (program — wersja CPC 6128) — 24
Alfabet Morse'a (program) — 25

Opis gry „Oils Well" — 32

„IKS" — dodatek „Żołnierza Wolności". Redagują: Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77, Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego.

U-3

Nr zam. 1690. Nr. ind. 396281.

ATARI BASIC

w przykładach

Kontynuując nasz cykl, rozpoczęty w „Zeszytach programów komputerowych" nr 4 z ubiegłego roku, prezentujemy proste programy umożliwiające wykorzystanie własności graficznych komputera Atari oraz jego interpretera.

Podczas działania komputera, w jednym z trybów graficznych — w odróżnieniu od trybów znakowych — mamy bezpośredni dostęp do pojedynczych punktów ekranu (elementów rastra), których wielkość zależy od używanego trybu. Przejście do takiego trybu uzyskujemy po wykonaniu instrukcji GRAPHICS z odpowiednią liczbą. Wykreślenie czegokolwiek na ekranie polega na „zapaleniu" odpowiednich punktów. Dlatego podstawową instrukcją graficzną jest „zapalenie" jednego elementu rastra, położonego w miejscu określonym przez współrzędne, tj. parę liczb całkowitych (X,Y). Pierwsza z nich określa odległość do zadanego punktu w poziomie, a druga — w pionie. W komputerach Atari współrzędne zmieniają się od strony lewej ku prawej i z góry na dół. Oznacza to, że ekran związany jest z układem współrzędnych, a środek tego układu znajduje się w lewym górnym rogu ekranu. Współrzędna X — pozioma — przyjmuje wartości rosnące od strony lewej do prawej, od 0 do XM. Współrzędna Y — pionowa — wzrasta od góry do dołu, przyjmując wartości 0, 1, 2, ..., YM. Punkt położony w lewym górnym rogu ekranu ma współrzędne (0, 0), punkt w prawym dolnym rogu — (XM, YM).

Wielkości charakteryzujące szerokość i wysokość ekranu (w jednostkach elementów rastra), tj. XM i YM, zmieniają się od 39 do 319 (X) oraz od 19 do 191 (Y). Im większe są wartości XM i YM, tym wyższa jakość wyświetlanych na ekranie linii, ponieważ każdy punkt zajmuje mniejszy obszar ekranu. Komputery Atari mają kilka trybów działania o różnej rozdzielczości, które prezentuje **program nr 34**.

W celu „zapalenia" punktu na ekranie należy podać odpowiednią instrukcję graficzną. Przedtem jednak należy ustawić odpowiedni tryb pracy (wybrać wymaganą rozdzielczość) za pomocą instrukcji GRAPHICS. Dopiero teraz można „zapalić" punkt instrukcją PLOT X,Y, gdzie X i Y określają współrzędne punktu na ekranie. W zasadzie wartości te powinny mieć postać liczb całkowitych. Jeżeli są to liczby rzeczywiste, zostaną zaokrąglone do najbliższych liczb całkowitych. Należy pamiętać, że w miejsce X i Y mogą być podstawione dowolne wyrażenia arytmetyczne.

Jeżeli wartości X i Y przekraczają granice ekranu, komputer przekazuje komunikat błędu. Aby mieć pewność, że rysunek tworzony jest w granicach ekranu, należy włączyć do programu sprawdzenie następujących warunków:

0 <= X <= XM
0 <= Y <= YM

Często, po „zapaleniu" wielu punktów, przydatna okazuje się możliwość oczyszczenia powierzchni ekranu. Służy do tego instrukcja PRINT #6; CHR$(125). Zastosowanie opisanej instrukcji pokazane jest w **programie nr 1**.

```
FK 1 REM ***************
FJ 2 REM *             *
XA 3 REM * Program nr 1 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
BH 100 ? "PODAJ TRYB GRAFICZNY ",
   INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
UI 120 XM=0+40*(GR=3)+80*(GR=4 OR
   GR=5)+160*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+320*(GR=8)
UN 125 XM=XM+80*(GR=9 OR GR=10 OR
   GR=11)
JD 130 YM=0+24*(GR=3)+48*(GR=4 OR
   GR=5)+96*(GR=6 OR GR=7)+192*(
   GR>7)
LS 140 GRAPHICS GR+16:POKE 705,5
   POKE 764,255 SETCOLOR 2,0,0
RA 145 COLOR 1+5*(GR=9)
YT 150 FOR K=1 TO 100
WZ 160 X=INT(RND(0)*XM)
XY 170 Y=INT(RND(0)*YM)
TI 180 PLOT X,Y
HD 190 NEXT K
YK 200 IF PEEK(764)=255 THEN 200
ON 210 POKE 764,255 GOTO 100
```

Uwaga: Litery z lewej strony numerów linii nie są częścią programu, lecz sumą kontrolną ułatwiającą wprowadzenie programu za pomocą Edytora BASIC u IKS nr 4 z 1987 roku). Znaki, które są podkreślone, należy wprowadzić w negatywie. Znaki wypisane drukiem rozszerzonym należy wprowadzić z wciśniętym klawiszem CONTROL.

Zarówno w tym, jak i w następnych programach, staramy się uwolnić czytelników od pamiętania rozmiarów ekranu w różnych trybach graficznych. Wystarczy podać numer trybu, a program przypomni nam wartości XM i YM oraz dopilnuje, aby w instrukcjach graficznych X i Y nie przekroczyły wartości maksymalnych. Jednocześnie kontrolowany jest numer trybu, dzięki czemu przyjmowane są tylko wartości trybów graficznych.



KREŚLENIE ODCINKÓW PROSTYCH

Za pomocą instrukcji PLOT można wykreślić odcinek poprzez wyświetlenie ciągu punktów wzdłuż prostej, przechodzącej przez oba jego końce. Wszystkie punkty prostej są niezależne, dlatego na przykład, w celu przedstawienia odcinka prostej pionowej, za pomocą instrukcji PLOT, należy zwiększać Y z krokiem 1 przy stałej wartości X. W **programach nr 2** i **nr 3** pokazano jak można wykreślać linie pionowe i poziome. Programy te tworzą linie ciągłe, ponieważ każdy zapalany element rastra jest małym prostokątem, dodawanym do już wyświetlonego elementu.

Kreślenie linii ukośnych jest nieco trudniejsze. W celu wykreślenia linii pochyłej należy obliczyć współrzędne wszystkich elementów rastra położonych na tej linii. Można to zrobić za pomocą wzoru:

$$Y = M \cdot X + B$$

określającego zależność między Y i X dla linii prostej. Współczynnik M określa nachylenie linii i może przyjmować wartości dodatnie, ujemne lub zerowe. Jeżeli M = 0, otrzymujemy linię pionową. Współczynnik B równy jest wartości, którą będzie miała współrzędna Y, jeśli przedłużymy linię do przecięcia z osią Y (gdzie X = 0). Jeśli B = 0, otrzymujemy odcinek leżący na prostej, przechodzącej przez początek układu współrzędnych.

Opierając się na instrukcji PLOT i powyższym równaniu, można zaprogramować ogólny algorytm kreślenia odcinków prostych. **Program nr 3** ilustruje wykreślanie odcinków

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Program nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
BH 100 ? "PODAJ TRYB GRAFICZNY "
   INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
II 120 XM=0+39*(GR=3)+79*(GR=4 OR
   GR=5)+159*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+319*(GR=8)
HG 125 XM=XM+79*(GR=9 OR GR=10 OR
   GR=11)
EE 130 YM=0+23*(GR=3)+47*(GR=4 OR
   GR=5)+95*(GR=6 OR GR=7)+191*(
   GR>7)
UK 140 ? "XMAX=";XM;" YMAX=";YM
CC 150 ? :? "Podaj wsp. x kreslon
   ej linii "; INPUT X
JS 160 X=0+X*(X>0 AND X<=XM)
SI 170 ? "Podaj poczatek i koniec
   kreslonej      linii (y1,y2)"
   INPUT Y1,Y2
ZH 180 Y1=0+Y1*(Y1>0 AND Y1<=YM)+
   YM*(Y1>YM)
AK 190 Y2=0+Y2*(Y2>0 AND Y2<=YM)+
   YM*(Y2>YM)
LL 200 GRAPHICS GR+16 POKE 705,5
   POKE 764,255 SETCOLOR 2,0,0
OG 210 COLOR 1+5*(GR=9)
XS 220 FOR Y=Y1 TO Y2
SZ 230 PLOT X,Y
ME 240 NEXT Y
EE 250 IF PEEK(764)=255 THEN 250
OX 260 POKE 764,255 GOTO 100
```

```
FK 1 REM ****************
FJ 2 REM *              *
YO 3 REM * Program nr 3 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
BH 100 ? "PODAJ TRYB GRAFICZNY ";
   INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
II 120 XM=0+39*(GR=3)+79*(GR=4 OR
   GR=5)+159*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+319*(GR=8)
HG 125 XM=XM+79*(GR=9 OR GR=10 OR
   GR=11)
EE 130 YM=0+23*(GR=3)+47*(GR=4 OR
   GR=5)+95*(GR=6 OR GR=7)+191*(
   GR>7)
UK 140 ? "XMAX=";XM;" YMAX=";YM
EV 150 ? :? "Podaj wsp. y kreslon
   ej linii ";:INPUT Y
TJ 160 Y=0+Y*(Y>0 AND Y<=YM)
IU 170 ? "Podaj poczatek i koniec
   kreslonej      linii (x1,x2)";
   :INPUT X1,X2
QG 180 X1=0+X1*(X1>0 AND X1<=XM)+
   XM*(X1>XM)
TW 190 X2=0+X2*(X2>0 AND X2<=XM)+
   XM*(X2>XM)
LL 200 GRAPHICS GR+16:POKE 705,5
   POKE 764,255:SETCOLOR 2,0,0
QG 210 COLOR 1+5*(GR=9)
WH 220 FOR X=X1 TO X2
SZ 230 PLOT X,Y
LU 240 NEXT X
EE 250 IF PEEK(764)=255 THEN 250
OX 260 POKE 764,255:GOTO 100
```

po wprowadzeniu wartości współczynników M i B. Określa on przede wszystkim, czy zadana linia może być wykreślona na ekranie. Jeśli część tej linii wykracza poza granice ekranu, program nie przyjmuje parametrów i sygnalizuje to dźwiękiem.

Jeśli chcemy poprowadzić odcinek między dwoma punktami o zadanych współrzędnych, to należy określić współczynniki M i B dla obliczenia współrzędnych punktów wzdłuż pro-

```
FK 1 REM ****************
FJ 2 REM *              *
ZI 3 REM * Program nr 4 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
BH 100 ? "PODAJ TRYB GRAFICZNY ";
   :INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
II 120 XM=0+39*(GR=3)+79*(GR=4 OR
   GR=5)+159*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+319*(GR=8)
HG 125 XM=XM+79*(GR=9 OR GR=10 OR
   GR=11)
EE 130 YM=0+23*(GR=3)+47*(GR=4 OR
   GR=5)+95*(GR=6 OR GR=7)+191*(
   GR>7)
UK 140 ? "XMAX=";XM;" YMAX=";YM
TP 150 ? :? "Podaj nachylenie i p
   rzesuniecie"
YA 160 ? "na osi y (M,B) ";:INPUT
   M,B
TV 170 REM Jesli M i B sa ujemne
HA 180 REM to linia jest poza ekr
   anem
MO 190 IF M<0 AND B<0 THEN GOSUB
   1000:GOTO 150
CG 200 REM Jesli M jest dodatnie
   ale B
YI 210 REM jest > YMAX to linia j
   est
BW 220 REM poza ekranem
[illegible] 230 IF M>0 AND B>YM THEN GOSUB
   1000 GOTO 150
HE 240 REM Trzeba teraz znalezc p
   oczatkowy
PI 250 REM punkt linii.
XZ 260 X1=0
GV 270 REM Jesli M jest ujemne a
   linia
ST 280 REM przecina os y poza ekr
   anem
BN 290 REM to poczatek linii jest
   w dole
BN 300 REM ekranu
SD 310 IF M<0 AND B>YM THEN X1=(Y
   M-B)/M
GH 320 IF X1>XM THEN GOSUB 1000:G
   OTO 150
[illegible] 330 REM Jesli M jest dodatnie
DH 335 REM i linia przecina os y
   w
EJ 336 REM punkcie ujemnym to poc
   zatek
OV 337 REM linii jest w gorze ekr
   anu
TS 340 IF M>0 AND B<0 THEN X1=-B/
   M
GN 350 IF X1>XM THEN GOSUB 1000:G
   OTO 150

LN 400 GRAPHICS GR+16:POKE 705,5
   POKE 764,255:SETCOLOR 2,0,0
QI 410 COLOR 1+5*(GR=9)
YB 420 FOR X=X1 TO XM:Y=M*X+B
DM 425 IF Y<0 OR Y>YM THEN 450
TB 430 PLOT X,Y
LW 440 NEXT X
   450 IF PEEK(764)=255 THEN 450
   460 POKE 764,255:GOTO 100
   1000 ? CHR$(253),CHR$(253);CHR
   $(253):RETURN
```

stej. Oznaczając współrzędne punktów końcowych przez (X1, Y1) i (X2, Y2) można wyznaczyć M i B na podstawie następujących równań:

$$M = (Y2 - Y1)/(X2 - X1)$$

$$B = Y1 - M \cdot X1$$

Program kreślenia odcinków według powyższych równań, mógłby wprowadzać współrzędne punktów końcowych, obliczyć M i B, a następnie kreślić linię.

Atari, podobnie jak większość systemów mikrokomputerowych, posiada instrukcję graficzną pozwalającą automatycznie prowadzić linię

między dwoma zadanymi punktami, a dokładniej między punktem bieżącym i zadanym:

DRAWTO X, Y

Instrukcja ta informuje, że należy poprowadzić linię prostą od punktu bieżącego do punktu o współrzędnych X, Y. X i Y mogą być stałymi, zmiennymi lub wyrażeniami arytmetycznymi. Wartości ułamkowe są zaokrąglane.

Program nr 5 demonstruje użycie ww. instrukcji do kreślenia linii prostych między dwoma dowolnymi punktami.

```
FK 1 REM ***************
FJ 2 REM *             *
AC 3 REM * Program nr 5 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
BH 100 ? "PODAJ TRYB GRAFICZNY ";
   INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
II 120 XM=0+39*(GR=3)+79*(GR=4 OR
   GR=5)+159*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+319*(GR=8)
HG 125 XM=XM+79*(GR=9 OR GR=10 OR
   GR=11)
EE 130 YM=0+23*(GR=3)+47*(GR=4 OR
   GR=5)+95*(GR=6 OR GR=7)+131*(
   GR>7)
UK 140 ? "XMAX=";XM;" YMAX=";YM
OO 150 ? "Podaj wsp. 1 punktu (x,
   y) ";:INPUT X1,Y1
NH 160 IF X1<0 OR X1>XM THEN GOSU
   B 1000:GOTO 150
PA 170 IF Y1<0 OR Y1>YM THEN GOSU
   B 1000:GOTO 150
TA 180 ? "Podaj wsp. 2 punktu (x,
   y) ";:INPUT X2,Y2
OG 190 IF X2<0 OR X2>XM OR Y2<0 O
   R Y2>YM THEN GOSUB 1000:GOTO 1
   80
LN 400 GRAPHICS GR+16:POKE 705,5:
   POKE 764,255:SETCOLOR 2,0,0
QI 410 COLOR 1+5*(GR=9)
TH 430 PLOT X1,Y1
ZE 440 DRAWTO X2,Y2
GI 450 IF PEEK(764)=255 THEN 450
XJ 455 POKE 764,255
OZ 460 POKE 764,255:GOTO 100
SW 1000 ? CHR$(253);CHR$(253);CHR
   $(253):RETURN
```

```
FK 1 REM ***************
FJ 2 REM *             *
AW 3 REM * Program nr 6 *
FL 4 REM *             *
FO 5 REM ***************
NL 6 REM
EK 90 DIM X(10),Y(10)
BH 100 ? "PODAJ TRYB GRAFICZNY ";
   :INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
II 120 XM=0+39*(GR=3)+79*(GR=4 OR
   GR=5)+159*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+319*(GR=8)
HG 125 XM=XM+79*(GR=9 OR GR=10 OR
   GR=11)
EE 130 YM=0+23*(GR=3)+47*(GR=4 OR
   GR=5)+95*(GR=6 OR GR=7)+191*(
   GR>7)
UK 140 ? "XMAX=";XM;" YMAX=";YM
NV 150 ? :? "Podaj ilosc rogow wi
   elokata "; INPUT N
DJ 160 FOR I=1 TO N
IJ 170 ? "Punkt ";I;" (x,y)=";:IN
   PUT X,Y
EW 180 IF X<0 OR Y<0 OR X>XM OR Y
   >YM THEN GOSUB 1000:GOTO 170
KL 190 X(I)=X:Y(I)=Y
FS 200 NEXT I
DI 400 GRAPHICS GR+16:REM POKE 70
   5,5:POKE 764,255:SETCOLOR 2,0,
   0
XO 410 COLOR 1
XK 420 PLOT X(1),Y(1)
BW 430 FOR K=1 TO N:DRAWTO X(K),Y
   (K):NEXT K
OS 440 DRAWTO X(1),Y(1)
RT 445 GOSUB 2000
GI 450 IF PEEK(764)=255 THEN 450
LV 460 GOTO 100
SW 1000 ? CHR$(253);CHR$(253);CHR
   $(253):RETURN
AS 2000 IF GR<8 OR GR>13 THEN T=4
   :R=0
QG 2010 IF GR=8 THEN T=2:R=1
EY 2020 IF GR=9 OR GR=11 THEN 220
   0
OQ 2030 IF GR=10 THEN T=-4:R=-3
HO 2100 FOR I=0 TO 255 STEP 2
EP 2110 POKE 708+R,I
IE 2120 FOR J=0 TO 255 STEP 2
GQ 2130 POKE 708+T,J
VX 2140 IF PEEK(764)=255 THEN 214
   0
GJ 2150 POKE 764,255
FV 2160 NEXT J
FN 2170 NEXT I
BA 2180 RETURN
DB 2190 REM ZMIANA KOLOROW DLA TR
   YBOW 9,11
FW 2200 FOR I=0 TO 15
JF 2210 SETCOLOR 4,I,0:IF GR=11 T
   HEN SETCOLOR 4,0,I
GM 2220 FOR J=0 TO 15
JX 2230 COLOR J
MK 2240 PLOT X(1),Y(1)
WU 2250 FOR K=2 TO N:DRAWTO X(K),
   Y(K):NEXT K
JK 2260 DRAWTO X(1),Y(1)
CE 2280 IF PEEK(764)=255 THEN 228
   0
GX 2290 POKE 764,255
FH 2300 NEXT J
EZ 2310 NEXT I
AM 2320 RETURN
```

Rozkazy wyświetlania punktów i kreślenia linii są podstawowymi środkami tworzenia rysunków. Za ich pomocą można kreślić kontury dowolnych przedmiotów. Jakość rysunków można poprawić za pomocą koloru. Rysunki wielobarwne są wygodniejsze w odbiorze. Staną się one bardziej przejrzyste, jeśli poszczególne obiekty lub strefy zabarwimy różnymi kolorami. Pozwala to skupić uwagę na najważniejszych elementach obrazu.



KOLORY

Komputer Atari może wytworzyć 16 jaskrawości 16 kolorów. Pozwala to na wybór jednej z 256 możliwości. Kody używanych kolorów przechowywane są w komórkach (rejestrach kolorów) o adresach 704 do 712. Cztery pierwsze rejestry używane są do określania kolorów graczy i pocisków. Pięć następnych określa kolory ramki, tła i kreślonych linii. Po włączeniu zasilania w rejestrach tych ustawione są następujące wartości domyślne:

Adres	Wartość	Kolor
708	40	pomarańczowy
709	202	jasnozielony
710	148	dark blue
711	70	czerwony
712	0	czarny

Zawartość rejestrów kolorów możemy zmienić instrukcją POKE albo SETCOLOR. Ta ostatnia ma trzy parametry: numer zmienianego rejestru (który zawsze odpowiada jednej z powyższych komórek), wybrany kolor (liczba od zera do piętnastu) i jego jaskrawość (liczba od zera do piętnastu). Zmiana koloru lub jaskrawości tła widoczna jest bezpośrednio po wykonaniu instrukcji SETCOLOR. Zmiana zawartości jednego z rejestrów kolorów pierwszoplanowych będzie widoczna tylko wtedy, gdy zmieniany rejestr używany już był do kreślenia.

Przed użyciem instrukcji PLOT lub DRAWTO musimy określić, który rejestr koloru będzie używany. Służy do tego instrukcja COLOR. Nie należy jej mylić z SETCOLOR. W pewnych trybach graficznych dostępnych jest kilka rejestrów kolorów pierwszoplanowych. Instrukcja COLOR wybiera jeden z dostępnych rejestrów. Na przykład, w trybie graficznym 7 możliwe jest kreślenie rysunku w trzech kolorach pierwszoplanowych. Instrukcja COLOR określa, który z rejestrów ma być zastosowany podczas kreślenia punktów lub linii.

Wyrażenie numeryczne po terminie COLOR wybiera często różne rejestry koloru, w zależności od aktualnego trybu graficznego. Regułą jest, że COLOR 0 zawsze wybiera rejestr koloru tła, a COLOR 1 — rejestr koloru pierwszoplanowego.

Kłopotliwy jest fakt, że liczba w instrukcji COLOR nie odpowiada numerowi rejestru w SETCOLOR. W dodatku ma ona różne wartości dla różnych trybów graficznych.

Musimy także pamiętać o poprzedzeniu instrukcji PLOT lub DRAWTO instrukcją COLOR.

Przykłady używania instrukcji PLOT, DRAWTO i COLOR występują w kolejnych programach. **Program nr 6** kreśli dowolny wielobok. Współrzędne wierzchołków wprowadzane są w kolejności ich łączenia i zapamiętywane w tablicach X i Y. Następnie program tworzy wielokąt, kreśląc linie od punktu X(1), Y(1) do X(2), Y(2), itd. Naciskając dowolny klawisz możemy zmieniać kolor rysunku.

Program nr 7 jest ilustracją sposobu tworzenia kolorowych obszarów. Następuje to poprzez kreślenie linii le-

```
FK 1 REM ****************
FJ 2 REM *              *
BQ 3 REM * Program nr 7 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
EK 90 DIM X(10),Y(10)
BH 100 ? "PODAJ TRYB GRAFICZNY ";
   INPUT GR
YY 110 IF GR<3 OR GR>15 OR GR=12
   OR GR=13 THEN 100
II 120 XM=0+39*(GR=3)+79*(GR=4 OR
   GR=5)+159*(GR=6 OR GR=14 OR G
   R=7 OR GR=15)+319*(GR=8)
HG 125 XM=XM+79*(GR=9 OR GR=10 OR
   GR=11)
EE 130 YM=0+23*(GR=3)+47*(GR=4 OR
   GR=5)+95*(GR=6 OR GR=7)+191*(
   GR>7)
UK 140 ? "XMAX=";XM;" YMAX=";YM
VB 150 ? "  PODAJ WSPOLRZEDNE LEW
   EGO GORNEGO       ROGU PROSTOKA
   TA (X,Y) ";
JR 160 INPUT X1,Y1
VY 170 IF X1<0 OR Y1<0 OR X1>XM O
   R Y1>YM THEN GOSUB 1000:GOTO 1
   50
FV 180 ? "  PODAJ WSPOLRZEDNE PRA
   WEGO DOLNEGO      ROGU PROSTOKA
   TA (X,Y) ";
KY 190 INPUT X2,Y2
FT 200 IF X2<0 OR Y2<0 OR X2>XM O
   R Y2>YM THEN GOSUB 1000:GOTO 1
   80
YD 210 ? "PODAJ KOLOR TLA I RYSUN
   KU (T,R) ";
AF 220 INPUT T,R
UA 230 IF T<0 OR R<0 OR T>15 OR R
   >15 THEN GOSUB 1000:GOTO 210
RY 400 GRAPHICS GR+16
HL 410 IF GR<8 OR GR>13 THEN SETC
   OLOR 4,T,0:SETCOLOR 0,R,10:COL
   OR 1
TP 420 IF GR=8 THEN SETCOLOR 2,T,
   0:SETCOLOR 1,R,10:COLOR 1
MS 430 IF GR=9 THEN SETCOLOR 4,T,
   0:COLOR R
YL 440 IF GR=10 THEN POKE 704,T*1
   6:POKE 705,R*16+10:COLOR 1
LP 450 IF GR=11 THEN SETCOLOR 4,0
   ,T:COLOR R
YC 460 FOR Y=Y1 TO Y2
LT 470 PLOT X1,Y:DRAWTO X2,Y
MO 480 NEXT Y
KY 490 IF PEEK(764)=255 THEN 490
XR 495 POKE 764,255
LK 500 GOTO 100
SW 1000 ? CHR$(253);CHR$(253);CHR
   $(253):RETURN
```

```
FK 1 REM ****************
FJ 2 REM *              *
CK 3 REM * Program nr 8 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
PC 10 GRAPHICS 7
YU 20 COLOR 1
JC 30 PLOT 70,40
AE 40 SETCOLOR 2,0,0
ZM 50 DRAWTO 35,0
ZB 60 DRAWTO 34,0
AI 70 POSITION 0,40
HV 80 POKE 765,1
AY 90 XIO 18,#6,0,0,"S "
HL 100 END
```

żących obok siebie. Prostokątny obszar ekranu określany jest współrzędnymi przeciwległych wierzchołków.

Pewne komputery, w tym i Atari, posiadają specjalną instrukcję do zamalowywania obszarów ekranu. Nie jest to instrukcja języka BASIC, lecz specjalny rozkaz dla systemu operacyjnego, zwany instrukcją XIO. Jej zastosowanie ilustruje **program nr 8**.

Linie 80 i 90 dotyczą użycia instrukcji XIO. POKE w linii 80 używa tej samej liczby co COLOR. Kolor wypełniania odpowiada instrukcji SETCOLOR, jak dla punktu lub linii na ekranie. Instrukcja XIO w linii 90 zawsze będzie miała tę samą postać. Należy stosować ją dokładnie tak, jak pokazano w przykładzie.

XIO Fill przeznaczona jest do wypełniania figur czworobocznych. Po uruchomieniu tego przykładu zobaczymy jednak trójkąt. Powyższa instrukcja będzie działać w oczekiwany sposób, jeśli wykonamy następujące czynności:

1. Użyjemy instrukcji PLOT do „zapalenia” punktu w prawym dolnym rogu figury.
2. Użyjemy DRAWTO do wykreślenia linii do prawego górnego rogu.
3. Wykreślimy linię do lewego górnego rogu.
4. Użyjemy instrukcji POSITION do przesunięcia kursora w prawy dolny róg.
5. Użyjemy instrukcji POKE do umieszczenia liczby (równej tej, która jest używana w instrukcji COLOR do kreślenia) w komórce o adresie 765.
6. Wykonamy XIO 18,#6,0,0,,,S:".

Instrukcje te muszą być wykonywane w określonej kolejności. XIO działa w sposób niemożliwy do przewidzenia, jeśli kolejność pierwszych pięciu czynności zostanie naruszona. XIO ma również inne ograniczenia. Wypełnia obszar tylko od lewej strony do prawej. Napotkanie przez kursor podświetlonego punktu nie wypełnia figury w tej linii. Jeśli zdefiniowana figura rozpoczyna się w prawym dolnym rogu, rozkaz wypełnienia realizowany jest od góry figury; jeśli natomiast rozpoczyna się w prawym górnym rogu, operacja wypełniania rozpocznie się od dołu figury.

Kolorowe obszary można również uzyskać kreśląc linie poziome, pionowe lub ukośne. Dla otrzymania specjalnych efektów można zmieniać położenie wykreślonych linii względem siebie i ich kolor. Można również uzyskać obszar kreskowany, prowadząc na przykład tylko co drugą lub co trzecią linię. Można także prowadzić linie w dwóch kierunkach. Zmieniając krok kreskowania można otrzymać płynne przejście od obrazu ciemnego do jasnego lub odwrotnie. Krok kreskowania może być zmieniany stopniowo, z przyrostem 1 lub 2, albo szybciej, poprzez podwojenie wartości linii.

Przykładem interesującego barwienia obszaru ekranu jest **program**

```
FK 1 REM ****************
FJ 2 REM *              *
DE 3 REM * Program nr 9 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
UM 100 GRAPHICS 10
MX 105 FOR I=1 TO 3:POKE 704+I,I*
   16+(I-1)*2:NEXT I
EW 110 FOR X=0 TO 79
LH 120 C=INT(RND(0)*8)+1
FF 130 COLOR C
DH 140 PLOT 40,96:DRAWTO X,0
LV 150 NEXT X
JV 160 FOR Y=0 TO 191
TJ 170 C=INT(RND(0)*8)+1:COLOR C
TO 180 PLOT 40,96:DRAWTO 79,Y
MN 190 NEXT Y
AK 200 FOR X=79 TO 0 STEP -1
LG 210 C=INT(RND(0)*8)+1
FE 220 COLOR C
IC 230 PLOT 40,96:DRAWTO X,191
LU 240 NEXT X
AK 250 FOR Y=191 TO 0 STEP -1
LO 260 C=INT(RND(0)*8)+1
FO 270 COLOR C
HR 280 PLOT 40,96:DRAWTO 0,Y
MO 290 NEXT Y
ZM 300 IF PEEK(764)=255 THEN 300
WL 310 POKE 764,255
HD 399 REM CYKLICZNA ZMIANA KOLOR
   OW
TZ 400 A=PEEK(705):FOR I=705 TO 7
   12:POKE I,PEEK(I+1):NEXT I:POK
   E 712,A
AQ 410 IF PEEK(764)=255 THEN 400
PP 420 POKE 764,255:GOTO 110
```

nr 9, który umożliwia otrzymanie kolorowego wzoru (po wykreśleniu radzimy nacisnąć dowolny klawisz). Wykorzystano w nim tylko instrukcje kreślenia linii, wybierania koloru i generowania liczb losowych.

Rysunki mogą być również tworzone w trybie dialogowym. Współdziałanie z programem może odbywać się za pomocą klawiatury, manipulatora, pióra świetlnego lub tabliczki graficznej. **Program nr 10** kreśli punkty w kierunku określonym za pomocą klawiatury. Dialogowy sposób tworzenia rysunków za pomocą odcinków linii prostych ilustruje program nr 11.

Otrzymane na ekranie rysunki wymagają często pewnych opisów. Podczas korzystania z programów może okazać się konieczne wprowadzenie lub wyprowadzenie pewnych informacji. Wykorzystanie całego ekranu na rysunek powoduje, w razie wystąpienia instrukcji INPUT, PRINT lub komunikatu błędu, wykasowanie ekranu i przejście do trybu 0. Dla uniknięcia takich nieoczekiwanych sytuacji wskazane jest użycie rozszczepionego ekranu, na którym (w różnych trybach graficznych), dysponujemy czterema liniami u dołu ekranu w trybie graficznym 0. Jest to tak zwane okno tekstowe, które otrzymujemy bezpośrednio po podaniu numeru trybu. W celu wyeliminowania okna tekstowego, dla uzyskania dodatkowych linii na rysunek, dodajemy liczbę 16 po wyrażeniu GRAPHICS. Poprzednie dodanie liczby 32 eliminujemy automatyczne czyszcze-

```
QO 1 REM ****************
IB 2 REM *              *
MQ 3 REM * Program nr 10 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
QJ 70 GRAPHICS 0·POKE 752,1
JN 80 ? :? "     Program ten umoz
   liwia"
KN 90 ? "  kreslenie dowolnego ry
   sunku"
DB 100 ? " przy pomocy klawiszy s
   trzalek."
HO 110 ? :? CHR$(27);CHR$(156),"r
   ysowanie do gory"
FO 120 ? CHR$(27);CHR$(157),"ryso
   wanie do dolu"
MH 130 ? CHR$(27);CHR$(158),"ryso
   wanie w lewo"
UJ 140 ? CHR$(27);CHR$(159),"ryso
   wanie w prawo"
ET 150 ? :? ,"podanie wspolrzedny
   ch"
UU 160 ? "RETURN","punktu, od kto
   rego"
IV 170 ? ,"zaczynamy kreslenie."
DI 180 ? :? ? "NACISNIJ DOWOLNY
   KLAWISZ"
YN 190 IF PEEK(755)=2 THEN POKE 7
   55,0
PF 200 FOR I=0 TO 120·NEXT I
ZE 210 IF PEEK(755)=0 THEN POKE 7
   55,2
PW 215 FOR I=0 TO 120·NEXT I
HF 220 IF PEEK(764)=255 THEN 190
UU 230 GRAPHICS 15·COLOR 1
NT 240 POKE 764,255·POKE 752,1·?
   CHR$(125);"Podaj wspolrzedne X
   ,Y ";:INPUT X,Y
XF 245 GRAPHICS 15+16+32
LF 250 TRAP 400
JJ 260 PLOT X,Y·POKE 764,255
GI 270 A=PEEK(764):IF A=255 THEN
   270
BS 280 X=X+1*(A=7)-1*(A=6)
NP 290 Y=Y+1*(A=15)-1*(A=14)
XH 300 IF A=12 THEN GRAPHICS 15+3
   2·GOTO 240
OI 310 GOTO 260
TL 400 X=X*(X>=0 AND X<160)+0*(X<
   0)+159*(X>159)
PO 410 Y=Y*(Y>=0 AND Y<192)+0*(Y<
   0)+191*(Y>191)
OA 420 GOTO 250
```

```
QO 1 REM ****************
IB 2 REM *              *
NL 3 REM * Program nr 11 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
WF 80 DIM X1(20),Y1(20),X2(20),Y2
   (20),C(20),A$(1)
RA 120 XM=159·YM=191
KD 130 T=1·REM T=ilosc linii
VI 140 X=80·Y=46·S=1
LB 150 GRAPHICS 15·POKE 752,1
QU 160 GOSUB 1000
UV 170 X1(T)=X·Y1(T)=Y
QX 270 GOSUB 1000
VS 280 X2(T)=X·Y2(T)=Y
MG 350 ? CHR$(125);"Podaj kolor l
   inii (1-3)";:POKE 764,255:INPU
   T S
WS 370 S=1+(S-1)*(S<4 AND S>0)
WO 400 C(T)=S
VP 410 GOSUB 570
HW 430 ? CHR$(125);"Wykreslic czy
   skasowac ostatnia linie (W lu
   b S)"
HW 440 INPUT A$:? CHR$(125)
LT 450 IF A$<>"W" AND A$<>"S" THE
   N 430
RT 460 IF A$<>"W" THEN 530
QI 470 REM WYKRESLIC LINIE
VV 480 T=T+1
OP 490 GOTO 160
WC 530 T=T-1
VW 540 GOSUB 570
VQ 550 T=T+1
OK 560 GOTO 160
[illegible]
[illegible] TRAP 40000·POK
   E 752,1
ID 590 FOR K=1 TO T
[illegible]
[illegible]
GU 620 NEXT K
HS 630 S=1
[illegible]
QG 1000 TRAP 2000·COLOR S·PLOT X,
   [illegible]
[illegible]
   1010
[illegible]
[illegible]
UX 1040 ? CHR$(125),"X= ";X1;"
   [illegible]
WG 1050 IF A=12 THEN RETURN
[illegible]
   1 GOTO 1000
[illegible]
[illegible]
   (0)+191*(Y>191)
[illegible]
```

nie ekranu, wykonywane normalnie przez instrukcję GRAPHICS.

W celu połączenia opcji grafiki pełnoekranowej z zaletą zachowania poprzedniej zawartości ekranu, musimy dodać 48 do żądanego trybu graficznego. Interesującym zastosowaniem tej możliwości jest selektywne otwieranie i zamykanie okna tekstowego w określonym punkcie programu. Demonstruje to **program nr 12**.

```
GO 1 REM ****************
IB 2 REM *              *
OG 3 REM * Program nr 12 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
[illegible]
TW 20 COLOR 1 PLOT 0,0
[illegible]
   E PRZERHTNE
[illegible]
[illegible]
[illegible]
[illegible]
   E UKNA TEKSTOWEGO
[illegible]
```

Oprócz wykorzystania okna tekstowego nie mamy możliwości bezpośredniego umieszczenia tekstu na rysunku. Dosyć łatwo można to zrealizować dla trybów dwukolorowych, tzn. 4, 6, 8 i 14.

Odwzorowanie bitowe zestawu znaków rezyduje w obszarze pamięci. Komórka o adresie 756 zawiera adres strony pamięci, od której zaczyna się ten obszar. Każdy znak zdefiniowany jest w ośmiobitowych segmentach. Taka bitowa reprezentacja znaku może być przesyłana, bajt po bajcie, do wcześniej określonego obszaru pamięci ekranu. Technikę tę ilustruje **program nr 13**.

Zmienne X i Y powinny zawierać numery kolumny i wiersza, na przecięciu których rozpocznie się wyświetlanie tekstu (kolumna we współrzędnych tekstowych, a wiersz w graficznych). Zmienna TXT$ może mieć dowolną wielkość.

```
QO 1 REM ****************
IB 2 REM *              *
PB 3 REM * Program nr 13 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
ED 10 DIM TEKST$(64)
QA 20 GRAPHICS 0·POSITION 5,5·? "
   PODAJ TRYB: ";:INPUT TRYB·GRAP
   HICS TRYB
PE 25 ? "Podaj polozenie (X,Y) i
   tekst "
FI 30 INPUT X,Y,TEKST$
MR 50 GOSUB 2000
SF 40 GOTO 30
EY 1995 REM PODPROGRAM KONWERSJI
   TEKSTU
RS 2000 SA=PEEK(89)*256+PEEK(88):
   REM POCZ PAMIECI EKRANU
DU 2010 TRYB=PEEK(87):REM OKRESLE
   NIE TRYBU GRAFICZNEGO
NH 2020 IF TRYB=8 THEN KOL=40·WIE
   R=192
BI 2030 IF TRYB=6 THEN KOL=20·WIE
   R=96
NX 2040 IF TRYB=4 THEN KOL=10·WIE
   R=24
IE 2050 IF Y>WIER OR X>KOL THEN R
   ETURN
LX 2060 START=SA+Y*KOL+X·REM ADRE
   S POCZATKOWY DLA WYSWIETLANIA
OL 2070 FOR E1=1 TO LEN(TEKST$)
YU 2080 GOSUB 2200
VN 2085 REM ODCZYT ADRESU ZESTAWU
   ZNAKOW
KL 2090 ZNAKI=PEEK(756)*256
OV 2100 ZNAKI=ZNAKI+E3*8
WM 2110 FOR E2=7 TO 0 STEP -1
CI 2120 POKE START+E2*KOL,PEEK(ZN
   AKI+E2)
AL 2130 NEXT E2
SL 2140 X=X+1
BK 2145 IF X>=KOL THEN START=STAR
   T+KOL*8·X=0·REM PRZEJSCIE DO N
   ASTEPNEJ LINII
HZ 2150 START=START+1
HT 2160 NEXT E1
HY 2170 RETURN
TP 2195 REM PROCEDURA KONWERSJI A
   TASCII
TR 2200 E3=ASC(TEKST$(E1,E1))
IP 2210 IF (E3<E2) OR (E3>127 AND
   E3<160) THEN E3=E3+64·RETURN
DQ 2220 IF E3>31 AND E3<96 THEN E
   3=E3-32
AN 2230 RETURN
```

Linie 2000—2060 określają używany tryb graficzny, ustalają szerokość ekranu dla tekstu i wyznaczają początkowy adres pamięci dla umieszczenia tekstu. Podprogram w liniach 2200—2230 przekształca kod znaku każdej litery umieszczonej w TXT$ na przesunięcie, to jest liczbę bajtów od początku tablicy zestawu znaków do binarnej definicji bieżącego znaku.

Linie 2100—2130 przesyłają bitową definicję znaku (8 bajtów) do odpowiedniego obszaru ekranu. Linia 2140 przesuwa tekst do następnej linii, jeśli następny znak nie mieści się w tej linii.

Zmienna tekstowa TXT$ przechowuje tekst do wyświetlania, a zmienne X i Y przechowują współrzędne kolumny i wiersza pierwszej wyświetlanej litery. Współrzędne kolumny mogą zmieniać się od 0 do liczby kolumn dostępnych w bieżącym trybie.

W trybie 4 ekran pomieści poziomo 10 znaków; w trybie 6 — 20 znaków, a w trybie 8 — 40 znaków. Numer wiersza może zmieniać się od 0 do maksymalnej liczby wierszy dostępnych w używanym trybie graficznym. Napisy w pionie możemy zatem umieszczać dosyć elastycznie.

Czas realizacji podprogramu jest dosyć długi, ponieważ zastosowano w nim instrukcje PEEK i POKE. Możliwe jest jednak przyspieszenie przesyłania danych z tablicy zestawu znaków do obszaru pamięci ekranu przez napisanie programu, w języku asemblera, do konwersji danych o znaku.

Za pomocą przedstawionego podprogramu można wyświetlać duże i małe litery oraz inne znaki graficzne. Nie przewidziano natomiast znaków w negatywie.

Kolejny, **program nr 14**, jest procedurą w języku maszynowym, przeznaczoną do kopiowania wzorów znaków do obszaru pamięci ekranu. Realizuje on podobne zadania jak poprzedni, lecz nieco szybciej. Dotyczy również trybów dwukolorowych (tj. 4, 6, 8 i 14).

```
QO 1 REM ****************
IB 2 REM *              *
PW 3 REM * Prosram nr 14 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
QI 100 GOSUB 1000
ME 102 TRAP 105
BE 105 GRAPHICS 0:POSITION 2,0
ND 110 ? "Podaj tryb:";:INPUT TRY
   B
BA 120 GRAPHICS TRYB
LQ 125 TRAP 130
EK 130 ? "Podaj wspolrzedne: ";:I
   NPUT K,W
LL 140 ? "Podaj tekst: ";:INPUT Q
   $
UG 150 GOSUB 900
MD 160 GOTO 110
LQ 900 Q=USR(ADR(Z$),K,W,ADR(Q$),
   LEN(Q$))
ZH 910 RETURN
RR 1000 REM PROCEDURA
VW 1001 REM Q=USR(ADR(Z$),K,W,ADR
   (Q$),LEN(Q$))
IC 1002 T=0:DIM Z$(169),Q$(48)
ZL 1004 RESTORE 1120:FOR T=1 TO 1
   68:READ Q:Z$(LEN(Z$)+1)=CHR$(Q
   ):NEXT T
JZ 1120 DATA 104,201,4,240,9,170,
   240,5,104,104,202,208,251,96,1
   04,133,215,104,133,214,104,104
   ,168,104,133
SY 1130 DATA 217,104,133,216,104,
   104,240,236,133,212,24;165,214
   ,101,88,133,214,165,89,101,215
   ,133,215,152,240,15
AA 1140 DATA 165,214,105,64,133,2
   14,165,215,105,1,133,215,136,2
   08,241,132,221,160,0,132,220,1
   77,216,160,0,170
CD 1150 DATA 16,1,136,132,213,138
   ,41,96,208,4,169,64,16,14,201,
   32,208,4,169,0,16,6,201,64,208
   ,2
MU 1160 DATA 169,32,133,218,138,4
   1,31,5,218,133,218,169,0,162,3
   ,6,218,42,202,208,250,109,244,
   2,133,219
CH 1170 DATA 164,221,177,218,69,2
   13,164,220,145,214,200,132,220
   ,196,212,208,182,24,165,214,10
   5,40,133,214,144,2
CX 1180 DATA 230,215,230,221,169,
   8,197,221,208,159,96,207,96
BC 1190 RETURN
```

Jak każda procedura w języku maszynowym, ta również — zanim zostanie użyta — wymaga umieszczenia kodów rozkazów w odpowiednim obszarze pamięci. Jest ona szczególnie interesująca, ponieważ można ją przemieszczać. Może być zatem umieszczona w dowolnym obszarze pamięci, w tym wypadku w zmiennej tekstowej Z$. Dzięki temu nie zajmuje strony 6 pamięci, która może być wykorzystana do innych celów.

Procedura ta wymaga podania następujących parametrów: współrzędnych X i Y pierwszej litery tekstu, nazwy zmiennej zawierającej tekst do wprowadzenia oraz długości tekstu w znakach. Wywołanie jej ma następującą postać:

a = USR(ADR(Z$),X,Y,ADR(Q$),LEN(Q$))

Współrzędne X i Y wyrażone są w kolumnach i wierszach tekstu (jak dla trybów 0, 1 lub 2), zmienna Z$ zawiera procedurę wyprowadzania, a zmienna Q$ — wyprowadzany tekst. Najlepiej działa w trybie graficznym 8. Umożliwia wyświetlanie wszystkich znaków.

Interesujące rozwiązanie zaprezentowaliśmy w **programie nr 15** zaczerpniętym z książki „Atari BASIC". Jest to w rzeczywistości „driver" urządzenia o nazwie T. Umożliwia on wyprowadzenie tekstu na ekran w dowolnym trybie graficznym (z wyjątkiem tekstowych), w podobny sposób jak na urządzenie zewnętrzne, za pomocą PRINT #3.

```
QO 1 REM ****************
IB 2 REM *              *
QR 3 REM * Prosram nr 15 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
TI 100 GOSUB 31000
YN 110 GRAPHICS 0:POSITION 3,5
BB 115 ? :? "Sposob uzycia jest n
   astepujacy "
DT 120 ? :? "PRINT #3;";CHR$(34);
   "TEKST";CHR$(34);","
VT 130 ? :? "Podaj tryb graficzny
   :";:INPUT TRYB
BE 140 GRAPHICS TRYB
BN 150 ? "Teraz sprobuj drukowac!
   "
TT 160 STOP
NP 31000 REM DANE DRIVER'A
CR 31010 RESTORE 31100:TDEVICE=1:
   PRUEF=0
GL 31020 FOR I=1536 TO 1536+255
SL 31030 READ BYTE:PRUEF=BYTE+PRU
   EF
XE 31040 POKE I,BYTE
HC 31045 NEXT I
ZH 31050 BYTE=USR(1536):POKE 765,
   1
FI 31090 OPEN #3,4,0,"T:"
LN 31095 IF PRUEF=28449 THEN RETU
   RN
LP 31100 DATA 104,162,0,189,26,3,
   240,12,201,84,240,23,232,232,2
   32,224
RE 31110 DATA 36,48,240,96,169,84
   ,157,26,3,169,35,157,27 3,169,
   6
XQ 31120 DATA 157,28,3,96,192,6,1
   32,6,47,6 50,6,47 6,4? 6
LG 31130 DATA 160,146,96,133,208,
   41,127,201,72,16,6,24,105,64,7
   6,72
BI 31140 DATA 6,201,96,16,3,56,23
   3,32,162,0,134,204,133,203,134
   ,206
KH 31150 DATA 134,207,160,16,208,
   13,24,165,206,105,8,133,206,16
   5,207,105
LB 31160 DATA 0,133,207,70,207,10
   2,206,102,204,102,203,136,48,4
   ,144,243
DT 31170 DATA 176,228,24,165,204,
   109,244,2,133,204,160,0,177,20
   3,162,7
SR 31180 DATA 10,72,165,208,144,1
   0,201,128,176,13,32,220,6,76,1
   51,6
EX 31190 DATA 201,128,144,3,32,22
   0,6,230,85,208,2,230,86,104,20
   2,16
[illegible]
[illegible]
[illegible]
MD 312[illegible] DATA [illegible] 141 [illegible]
[illegible]
[illegible]
NA 31260 GRAPHICS 0:? "POMYLKA W
   [illegible]
```

Procedura ta posiada wiele zalet i jedną wadę, zajmuje stronę 6 pamięci komputera. Ponadto, podczas jej używania należy pamiętać o umieszczeniu średnika lub przecinka za cudzysłowem kończącym tekst.

Mamy nadzieję, że przedstawione programy ułatwią Wam tworzenie różnorodnych rysunków i opisywanie ich.



PRZEKSZTAŁCANIE RYSUNKÓW

W celu zwiększenia możliwości tworzenia efektywnych programów graficznych należy zapoznać się z zasadami przekształcania elementów rysunków. Umożliwi to:

a) przemieszczanie rysunków z jednego miejsca ekranu w inne;
b) utworzenie łącznego rysunku z mniejszych elementów;
c) dodawanie do istniejącego rysunku nowych elementów;

d) powiększanie rysunku np. w celu przyjrzenia się drobnym elementom;
e) pomniejszanie rysunku w celu wprowadzenia opisów lub dodatkowych rysunków;
f) tworzenie ruchomych obrazów.

Wszelkie zmiany wykonuje się za pomocą trzech podstawowych operacji:

1) przenoszenia wykreślonych obiektów z jednego miejsca ekranu w inne (przemieszczenie);
2) zwiększania lub zmniejszania rozmiarów wyświetlanego rysunku (skalowanie);
3) zmian orientacji rysunku (obrót).



ZMIANA MIEJSCA POŁOŻENIA

Metody przemieszczania elementów zobrazowania pozwalają tworzyć rysunek w dowolnym obszarze ekranu i przenosić go w inną część ekranu. Można przy tym przekomponować rysunek, albo tworzyć go częściami z zestawu gotowych elementów.

Zanim zaczniemy przemieszczać skomplikowane rysunki, zobaczmy na czym polega zmiana położenia pojedynczego punktu. W tym celu należy zamienić aktualne współrzędne punktu na współrzędne odpowiadające nowemu położeniu. Ogólna reguła przemieszczania punktu z pozycji o współrzędnych (X, Y) do pozycji (XT, YT) w terminach odległości H i V zapisywana jest następująco:

$$XT = X + H, \qquad YT = Y + V$$

Dodatnia wartość H oznacza przemieszczenie punktu w poziomie na prawo, a dodatnia wartość V — przemieszczenie punktu pionowo w dół. Natomiast ujemne wartości H i V powodują przemieszczenie punktu na lewo i w górę.

Podczas przemieszczania wyświetlanych punktów powinniśmy używać tylko takich wartości H i V, które pozostawiają punkt w granicach ekranu. Nie należy jednak używać zbyt małych wartości H i V. Jeśli są one mniejsze niż 0.5, punkt będzie wyświetlony w tym samym miejscu. Wielokrotne wyświetlanie punktu może znacznie zwolnić działanie programu.

Przemieszczenie rysunku z jednego obszaru do drugiego równoważne jest przemieszczeniu wszystkich punktów rysunku, a następnie ponownemu wyświetleniu wszystkich linii połączeniowych. Aby uniknąć naruszenia kształtu przedmiotu, wszystkie punkty rysunku należy przemieścić na jednakową odległość. Oznacza to, że przy przeliczaniu współrzędnych za pomocą powyższych wzorów, należy dla wszystkich punktów używać tych samych wartości H i V. Zastosowanie odmiennych wartości H i V dla różnych punktów powoduje zniekształcenie rysunku. Zwykle wymagane jest przemieszczenie rysunku bez zniekształceń, jednak niekiedy mogą być one przydatne w poszukiwaniu najlepszego kształtu przedmiotu.

Współrzędne punktów końcowych wszystkich odcinków rysunku wygodnie jest przechowywać w postaci tablicy. Dla prostych rysunków — w tablicy jednowymiarowej (w takiej kolejności, w jakiej będą one potem połączone). W bardziej złożonych rysunkach przez jeden punkt może przechodzić kilka linii lub mogą istnieć fragmenty rysunku nie połączone liniami z innymi fragmentami. W takich przypadkach należy przechowywać wartości współrzędnych w tablicy dwuwymiarowej. Jeden indeks tablicy oznacza numer fragmentu, a drugi wskazuje kolejny numer punktu we fragmencie. W ten sposób para wartości (X(2,1),Y(2,1)) określa współrzędne pierwszego punktu drugiego fragmentu rysunku. Tablice dwuwymiarowe wykorzystane są w **programie nr 16** do przechowywania współrzędnych punktów. Pozwala on przemieszczać dowolną ilość razy oddzielne fragmenty rysunku w wybrany obszar ekranu.

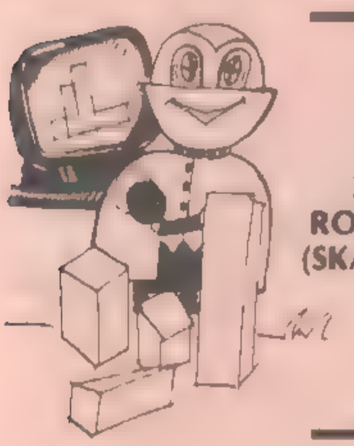
W omawianym programie (16) wyliczone wartości współrzędnych punktów zastępują poprzednie wartości w tablicy X i Y. Jeśli chcemy zachować początkowe wartości współrzędnych, wartości obliczone można umieścić (zapamiętać) w dodatko-

```
QO 1 REM ****************
IB 2 REM *              *
RM 3 REM * Prosram nr 16 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
ES 60 GRAPHICS 8:COLOR 1:POKE 752
   ,1
AR 70 DIM X(5,50),Y(5,50),N(5),A$
   (1)
WN 80 XM=319
DD 90 YM=191:W=0
SR 95 ? "Czy własne rysunki ( T/N
   )? ";:INPUT A$:IF A$="T" THEN
   [illegible]
WN 100 REM ODCZYT RYSUNKOW
DA 110 PN=1
EM 120 E=0
BN 130 READ XD,YD
OF 140 IF XD>=0 THEN 200
QW 150 REM
XD 160 N(PN)=E
WL 170 IF XD=-100 THEN 240
ZJ 180 PN=PN+1
MU 190 GOTO 120
OF 200 E=E+1
VT 210 X(PN,E)=XD
WN 220 Y(PN,E)=YD
MU 230 GOTO 130
QV 240 REM
TB 250 GOSUB 510
LR 280 ? CHR$(125);
MM 290 ? "Podaj numer rysunku do
   przesuniecia "
BI 295 IF W=1 THEN FOR J=1 TO PN:
   ? J;"    ";:NEXT J:? :GOTO 310
AQ 300 ? "1-Postac    2-Pies    3
   -Hydrant"
PE 310 INPUT P
LG 320 ? CHR$(125);
NE 370 ? "Podaj przesuniecie dX i
   dY."
XQ 380 INPUT H,V
KL 400 REM PRZELICZANIE PUNKTOW
KY 410 FOR J=1 TO N(P)
MO 420 X(P,J)=X(P,J)+H
XN 430 Y(P,J)=Y(P,J)+V
GS 470 NEXT J
TJ 480 GOSUB 510
PV 490 GOTO 280
HQ 510 REM PODPROGRAM KRESLENIA R
   YSUNKU
EQ 520 GRAPHICS 8:COLOR 1:POKE 75
   2,1
EF 530 FOR K=1 TO PN
DY 540 FOR J=1 TO N(K)-1
OT 545 TRAP 560
GN 550 PLOT X(K,J),Y(K,J):DRAWTO
   X(K,J+1),Y(K,J+1)
GR 560 NEXT J
HD 570 NEXT K
ZR 580 RETURN
KS 590 REM WSPOLRZEDNE RYSUNKOW
NP 600 DATA 85,70,90,75,105,60,10
   5,80,85,110,95,110,110,85
XX 610 DATA 125,110,135,110,115,8
   0,115,60,130,75,135,70,115,45
ST 620 DATA 115,40,125,30,125,15,
   110,10,95,15,95,30,105,40,105,
   45,85,70
XM 630 DATA -1,-1
QN 640 DATA 50,90,62,110,58,110,5
   0,90,42,110,38,110,50,90
DW 650 DATA 45,100,20,100,10,90,2
   2,110,18,110,10,90,3,110
EK 660 DATA 0,110,10,90,0,80,10,9
   0,50,90,40,80,50,70,55,75,60,7
   8,60,82,50,90
XU 670 DATA -1,-1
SF 680 DATA 290,110,290,90,288,90
   ,288,92,286,92,286,88
RD 690 DATA 286,88,288,88,288,82,
   290,82,290,73,287,73,287,70,29
   0,70,290,65
GD 700 DATA 295,60,305,60,310,65,
   310,70,313,70,313,73,310,73,31
   0,110,290,110
OT 710 DATA -100,-100
VP 1000 REM KRESLENIE WLASNYCH RY
   SUNKOW
TJ 1010 PN=1:X=0:Y=120:W=1
MU 1020 GRAPHICS 8+16:COLOR 1:SET
   COLOR 2,0,0
ZX 1030 E=0:X=X+60:Y=120
RD 1040 POKE 764,255:IF PN>5 THEN
    PN=5:GOTO 250
LF 1050 A=PEEK(764):IF A=255 THEN
    1050
LC 1054 IF A<>6 AND A<>7 AND A<>1
   4 AND A<>15 AND A<>12 AND A<>2
   8 AND A<>44 THEN 1040
RU 1055 IF A=44 THEN N(PN)=E:PN=P
   N+1:GOTO 1020
NT 1056 IF A=28 THEN N(PN)=E:GOTO
    250
XZ 1060 X=X+1*(A=7)-1*(A=6)
NP 1070 Y=Y+1*(A=15)-1*(A=14)
IO 1075 TRAP 1080:PLOT X,Y
NZ 1080 IF A<>12 THEN 1040
JA 1090 E=E+1
JY 1100 X(PN,E)=X:Y(PN,E)=Y:PLOT
   X,Y
OT 1110 GOTO 1040
```

wych tablicach, np. w XT i YT. Przechowanie początkowych wartości współrzędnych zalecane jest wtedy, gdy chcemy sprawdzić, jak będzie wyglądał rysunek lub jeśli zamierzamy przedstawić go na ekranie w obu położeniach.

Pewnych wyjaśnień wymaga opcja tworzenia własnych rysunków w programie nr 16. W celu właściwego ich zapamiętania (umożliwiającego późniejsze przekształcenia), kreślenie rysunku, po uprzednim wstawieniu punktu początkowego, należy rozpocząć naciskając klawisz <RETURN>. Również koniec każdego odcinka należy zasygnalizować naciśnięciem klawisza <RETURN>. Po zakończeniu kreślenia fragmentu rysunku musimy nacisnąć klawisz <TAB>, a po zakończeniu rysunku — <ESC>.

Obiekty symetryczne, a także takie, których granice wyznaczone są równaniami, przemieszczane są w inny sposób. Nie ma potrzeby dodawania przesunięcia do współrzędnych wszystkich punktów. Na przykład, w celu przemieszczenia okręgu lub elipsy wystarczy przemieścić tylko środek figury i wyliczyć nowe współrzędne wszystkich punktów. Dla przemieszczenia prostokąta należy przemieścić jeden z jego wierzchołków i ponownie wykreślić go na podstawie znanych wartości wysokości i szerokości.



ZMIANA ROZMIARÓW (SKALOWANIE)

Po utworzeniu obrazu na ekranie można zmienić jego wymiary w celu poprawienia przejrzystości przedstawionej informacji. Można też wstawić ten obraz w większy rysunek. Rozmiar rysunku można zmienić poprzez pomnożenie wszystkich odległości między punktami przez określony współczynnik skalowania. Na przykład, jeśli chcemy zwiększyć rozmiary prostokąta dwa razy, musimy długość każdej linii pomnożyć przez dwa; jeśli chcemy zmniejszyć dwa razy — przez 0.5.

Podczas skalowania obrazu należy także wskazać jego położenie po zwiększeniu lub zmniejszeniu rozmiarów. Możemy skalować rysunki względem pewnego centralnego punktu rysunku, punktu na granicy rysunku lub dowolnego punktu wewnątrz niego. Można np. zwiększyć rozmiary prostokąta ustalając położenie lewego dolnego rogu prostokąta lub jego środka.

Zacznijmy od czynności prostych, tzn. od skalowania odcinka. Poziomy odcinek między dwoma punktami z odciętymi X(1) i X(2) ma długość:

$$L = X(2) - X(1)$$

Zmiana długości odcinka oznacza, że mnożymy L przez współczynnik skalowania w poziomie, np. HS, co daje nową wartość długości odcinka:

$$LS = L \cdot HS = X(2) \cdot HS - X(1) \cdot HS = XS(2) - XS(1)$$

Tablica XS wykorzystywana jest tu do przechowywania wartości obliczonych współrzędnych, a współczynnik HS powinien mieć wartość dodatnią (HS > O). Jeśli współczynnik skalowania HS > 1, wówczas odcinki są dłuższe. Jeśli HS <, wszystkie odcinki stają się krótsze. Przy HS = 1 długości odcinków nie ulegną zmianie.

Skalowanie odcinków prowadzi do zmiany współrzędnych punktów końcowych, należy zatem wskazać nowe położenie odcinka. Dla odcinka poziomego można ustalić albo jeden z końców odcinka albo jego środek. Można też przemieścić otrzymany odcinek w inne miejsce. Załóżmy, że chcemy zostawić na miejscu lewy koniec odcinka. W tym przypadku nowe współrzędne odcinka będą równe:

$$XS(1) = X(1)$$
$$XS(2) = X(1) + (X(2) - X(1)) \cdot HS$$

Jeśli chcemy zachować położenie prawego końca odcinka, jego nowe współrzędne będą równe:

$$XS(1) = XS(2) - (X(2) - X(1)) \cdot HS$$
$$XS(2) = X(2)$$

Ustalając środek odcinka otrzymamy:

$$XS(1) = XM - ((X(2) - X(1))/2) \cdot HS$$
$$XS(2) = XM + ((X(2) - X(1))/2) \cdot HS$$

gdzie: XM — współrzędne środka odcinka, równa

$$XM = (X(1) + X(2))/2$$

Jeśli trzeba przemieścić zmieniony odcinek w inne miejsce ekranu, to najpierw dokonuje się czynności przeniesienia go w nowe miejsce, a następnie skaluje względem jednego z końców lub środka odcinka. Ogólne podejście polega na wyborze punktu odniesienia, który będzie traktowany jako ustalony. Następnie należy obliczyć współrzędne końców odcinka, mnożąc odległości od nich do punktu XF przez współczynnik skalowania HS. Współrzędne końców odcinka po skalowaniu można obliczyć według wzorów:

$$XS(1) = (X(1) - XF) \cdot HS + XF$$
$$XS(2) = (X(2) - XF) \cdot HS + XF$$

```
QO 1 REM ****************
IB 2 REM *              *
SH 3 REM * Program nr 17 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
PE 80 DIM X(5,100),Y(5,100),N(5)
DZ 90 XM=319:YM=191
DG 140 PN=1
ES 150 E=0
BT 160 READ XD,YD
RN 170 IF XD>=0 THEN 240
XJ 190 N(PN)=E
ZI 200 IF XD=-100 THEN 280
YY 220 PN=PN+1
NQ 230 GOTO 150
ON 240 E=E+1
WB 250 X(PN,E)=XD
WV 260 Y(PN,E)=YD
OJ 270 GOTO 160
RD 280 REM
WZ 290 GOSUB 770
PV 320 GRAPHICS 8+32:POKE 82,0:?
   CHR$(125);
AL 330 ? "1-Przod 2-Srodek 3-Tyl
   4-Calosc 5-koniec"
KF 340 ? "Podaj numer czesci do s
   kalowania";:INPUT P
SK 350 IF P=5 THEN END
TH 360 ? CHR$(125);"Podaj skale n
   a osi x i y";:INPUT XS,YS
PZ 370 ? CHR$(125),"Prosze czekac
   "
DC 500 ON P GOTO 560,590,620,670
NC 510 GOTO 320
GD 560 XF=X(1,2)
GS 570 YF=Y(1,2)
PQ 580 GOTO 640
GT 590 XF=X(2,2)
GP 600 YF=Y(2,2)
RO 605 GOSUB 1100
PD 610 GOTO 640
GQ 620 XF=X(3,2)
HF 630 YF=Y(3,2)
XE 640 GOSUB 870
WV 650 GOSUB 770
NN 660 GOTO 320
VW 670 ? "Podaj punkt skalowania"
   ;:INPUT XF,YF
FU 700 FOR P=1 TO PN
WZ 710 GOSUB 870
IT 720 NEXT P
PT 730 GOTO 650
EC 770 REM *************
FP 780 REM * KRESLENIE *
EG 790 REM *************
UB 800 GRAPHICS 8+16 COLOR 1:SETC
   OLOR 2,0,0
ZH 810 FOR K=1 TO PN:FOR J=1 TO N
   (K)-1
GQ 820 TRAP 830:PLOT X(K,J),Y(K,J
   ):DRAWTO X(K,J+1),Y(K,J+1)
GO 830 NEXT J
HA 840 NEXT K
PQ 845 IF PEEK(764)=255 THEN 845
XS 850 POKE 764,255:RETURN
NP 860 REM **************
OO 870 REM * SKALOWANIE *
NT 880 REM **************
LS 890 FOR J=1 TO N(P)
QE 900 X(P,J)=X(P,J)*XS+XF*(1-XS)
TO 910 Y(P,J)=Y(P,J)*YS+YF*(1-YS)
ND 920 NEXT J RETURN
GK 930 REM ********
ZA 940 REM * DANE *
GO 950 REM ********
FK 960 REM ***** PRZOD *****
EM 970 DATA 220,90,210,90,210,60,
   250,65,260,90,250,30
YD 980 DATA 250,100,240,110,230,1
   10,220 100,220,90
MF 990 DATA 230,80,240,80,250,90,
   -1,-1
KB 1000 REM ***** SRODEK *****
GU 1010 DATA 210,90,160,90,160,60
   ,205,60,13?,35,160,35
SA 1020 DATA 160,30,190,30,210,60
   ,-1,-1
EG 1030 REM ***** TYL *****
RC 1040 DATA 120,90,160,90,160,60
   ,120,60,135,35,160,35
GQ 1050 DATA 160,30,130,30,115,60
   ,80,60,75,90,90,90
```

```
EC 1080 DATA 90,100,100,110,110,1
   10,120,100,120,90,110,80,100,8
   0,90,90,90,100,100,-100
HN 1090 REM PODPROGRAM PRZESUNIEC
   IA CZESCI 1 RYSUNKU PRZY ZMIAN
   IE CZESCI 2
UD 1100 DX=INT(X(1,2)*(XS-1)+XF*(
   1-XS))
WF 1110 FOR J=1 TO N(1):X(1,J)=X(
   1,J)+DX
FL 1130 NEXT J
AN 1140 RETURN
```

Jako punkt odniesienia XF można wybrać dowolny punkt na ekranie lub na zewnątrz niego. Jeśli za XF przyjąć jeden z punktów X(1), X(2) lub XM, otrzymamy wyniki odpowiadające wcześniejszym wzorom.

Skalowania odcinków pionowych dokonuje się analogicznie. Długość pionowego odcinka, poprowadzonego między punktami Y(1) i Y(2) równa jest:

$$L = Y(2) - Y(1)$$

Przekształcenie tego odcinka za pomocą współczynnika skalowania w pionie VS daje odcinek o nowej długości:

$$LS = L \cdot VS = Y(2) \cdot VS - Y(1) \cdot VS = YS(2) - YS(1)$$

gdzie tablica YS wykorzystywana jest do przechowywania nowych współrzędnych końców odcinka. Współczynnik skalowania VS powinien być dodatni, a położenie przekształconego odcinka można wyznaczyć względem dowolnie wybranego punktu, np. Y(1), Y(2) lub środka odcinka. Oznaczając ten punkt przez XF można wyznaczyć współrzędne odcinka:

$$YS(1) = (Y(1) - YF) \cdot VS + YF$$
$$YS(2) = (Y(2) - YF) \cdot VS + YF$$

Długość odcinka poprowadzonego pod dowolnym kątem zależy od jednoczesnego skalowania rzutu poziomego X(2) — X(1) i rzutu pionowego Y(2) — Y(1) względem ustalonego punktu (XF, YF).

Powyższe równania można zapisać w następującej postaci:

$$XS(1) = X(1) \cdot HS + XF \cdot (1 - HS)$$
$$YS(1) = Y(1) \cdot VS + YF \cdot (1 - VS)$$
$$XS(2) = X(2) \cdot HS + XF \cdot (1 - HS)$$
$$YS(2) = Y(2) \cdot VS + YF \cdot (1 - VS)$$

Wyrażenia $XF \cdot (1 - HS)$ i $YF \cdot (1 - VS)$ są stałymi, dlatego obliczane są tylko raz. W celu zwiększenia lub zmniejszenia długości pochyłego odcinka (z zachowaniem kąta pochylenia) musi HS = VS.

Rozmiary skalowanych rysunków zwiększają się lub zmniejszają równomiernie, jeśli HS = VS = S. Powyższe równania używane są do obliczania nowych współrzędnych wszystkich punktów skalowanego zobrazowania. Powiększony lub pomniejszony rysunek otrzymuje się poprzez wykreślenie linii łączących punkty o nowo obliczonych współrzędnych. Dla rysunków zawierających linie krzywe nie musimy stosować przekształceń do każdego ich punktu. Podczas skalowania okręgu o promieniu R wystarczy tylko wyliczyć za pomocą powyższych równań nowe współrzędne środka okręgu i wziąć za promień wielkość R · S.

W pewnych przypadkach trzeba rozciągnąć zobrazowanie w jednym kierunku (np. w pionie) i zachować początkowe rozmiary w innym (np. poziomym) kierunku. Aby przekształcić kwadrat w prostokąt, wystarczy zwiększyć dwa razy jego wysokość. W tym celu podczas przeliczania współrzędnych punktów współczynnik skalowania w pionie VS powinien być równy 2, a współczynnik skalowania w poziomie HS — równy 1. Takie podejście jest efektywne podczas eksperymentowania lub dobierania proporcji fragmentów rysunku. Oprócz tego nierównomierne skalowanie można stosować podczas tworzenia złożonych rysunków ze zbioru figur standardowych, np. kwadratów, które mogą być przekształcone w prostokąty dowolnych rozmiarów.

Program nr 17 zawiera przykład skalowania rysunku. Dane wejściowe programu składają się z opisu rysunku i współczynników skalowania HS i VS. Program skaluje cały rysunek lub dowolny jego fragment.



ZMIANA ORIENTACJI (OBRÓT)

W wielu wypadkach powstaje konieczność zmiany orientacji zobrazowania. Podobnie jak poprzednio, zaczniemy od najprostszej sytuacji.

Obrót punktu polega na przemieszczeniu go z punktu (X, Y) do punktu (XR, YR) po łuku okręgu z środkiem obrotu (X0, Y0). Kąt A określa wielkość obrotu z punktu (X, Y) do punktu (XR, YR). Współrzędne punktu (XR, YR) po przekształceniu oblicza się z wykorzystaniem wielkości A, (X0, Y0) i (X, Y) według wzorów:

$$XR = X0 + ((X - X0) \cdot \cos(A)) + ((Y - Y0) \cdot \sin(A))$$
$$YR = Y0 + ((Y - Y0) \cdot \cos(A)) + ((X - X0) \cdot \sin(A))$$

Centrum obrotu (X0, Y0) może znajdować się w dowolnym miejscu — na ekranie lub poza jego granicami. Ten punkt nie jest wyświetlany i wykorzystywany jest tylko jako punkt odniesienia (dla określenia przemieszczenia po łuku okręgu podczas obrotu).

```
QO 1 REM ****************
IB 2 REM *              *
TC 3 REM * Program nr 18 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
DD 80 DIM X(7,100),Y(7,100),N(7),
   XR(7,100),YR(7,100)
DZ 90 XM=319:YM=191
HL 140 PN=1:DEG
ES 150 E=0
BT 160 READ XD,YD
RN 170 IF XD>=0 THEN 240
XJ 190 N(PN)=E
ZI 200 IF XD=-100 THEN 280
YY 220 PN=PN+1
NQ 230 GOTO 150
ON 240 E=E+1
WB 250 X(PN,E)=XD
WV 260 Y(PN,E)=YD
OJ 270 GOTO 160
RD 280 REM
II 290 FOR P=1 TO PN:FOR E=1 TO N
   (P)
WX 300 XR(P,E)=X(P,E)
XR 310 YR(P,E)=Y(P,E)
HG 320 NEXT E:NEXT P
WU 350 GOSUB 590
OG 360 GRAPHICS 8+32:POKE 82,0
LQ 370 ? CHR$(125);
NK 380 ? "  Podaj kat obrotu wzgl
   edem pozycji          pierwotnej"
   ;
NF 390 ? " (0-360) ";:INPUT A
LD 400 ? CHR$(125);
EU 410 ? "  Podaj wspolrzedne pun
   ktu obrotu"
IL 420 INPUT X0,Y0
KR 430 REM PRZELICZANIE PUNKTOW
FZ 440 FOR P=1 TO PN
WU 450 GOSUB 680
IY 460 NEXT P
WZ 470 GOSUB 590
PH 480 GOTO 360
IG 590 REM PODPROGRAM KRESLENIA R
   YSUNKU
TZ 600 GRAPHICS 8+16:COLOR 1:SETC
   OLOR 2,0,0
FY 610 FOR P=1 TO PN
FO 620 FOR E=1 TO N(P)-1
OC 630 TRAP 650
ZQ 640 PLOT XR(P,E),YR(P,E):DRAWT
   O XR(P,E+1),YR(P,E+1)
ES 650 NEXT E
JA 660 NEXT P
KU 670 IF PEEK(764)=255 THEN 670
YJ 675 POKE 764,255:RETURN
ZR 680 REM PODPROGRAM OBROTU
JX 690 FOR E=1 TO N(P)
QE 700 XR(P,E)=X0+(X(P,E)-X0)*COS
   (A)+(Y(P,E)-Y0)*SIN(A)
SC 710 YR(P,E)=Y0+(Y(P,E)-Y0)*COS
   (A)-(X(P,E)-X0)*SIN(A)
EN 720 NEXT E
ZJ 730 RETURN
XG 740 REM RYSUNEK
WW 750 DATA 160,60,170,50,170,40,
   160,30,150,30,140,40,140,50
OL 760 DATA 150,60,170,60,165,65,
   170,70,160,67,155,70,150,67
LV 770 DATA 140,72,145,65,130,65,
   150,60,100,70,90,70,90,80,100
GI 780 DATA 80,110,90,100,80,120,
   140,120,150,130,160,120,170,12
   5
ZL 790 DATA 175,140,170,140,160,1
   50,150,155,110,170,150,180,160
   ,180
GP 800 DATA 170,200,170,200,185,1
   90,160,195,150,195,140,175,80,
   200
FP 810 DATA 30,190,20,190,10,180,
   10,180,20,170,60
XM 820 DATA -1,-1
LS 830 DATA 147,42,153,42,-1,-1
MG 840 DATA 150,38,150,45,-1,-1
MJ 850 DATA 157,42,163,42,-1,-1
EH 860 DATA 160,38,160,45,-100,-1
   00
```

Kąt obrotu A mierzy się w kierunku przeciwnym do ruchu wskazówek zegara od początkowego położenia punktu (X, Y). Zakłada się przy tym, że początek układu współrzędnych znajduje się w lewym górnym rogu ekranu. Zwykle wartości tego kąta leżą w zakresie 0—2 · PI (6.283185 rad). Inne wartości kąta obrotu także są dopuszczalne, jednak są one równoważne obrotowi w podanym zakresie.

Droga, jaką przechodzi punkt po łuku okręgu podczas obrotu o kąt A, zależy do odległości między punktem (X, Y) i środkiem obrotu (X0, Y0). Im bardziej oddalony jest środek obrotu (X0, Y0) od punktu (X, Y), tym dłuższa jest trajektoria ruchu od punktu (X, Y) do punktu (XR, YR). Przy małych kątach obrotu wielkość drogi D w przybliżeniu równa jest iloczynowi R · A. Jeśli R = 50, punkt przemieszcza się o 1 krok rastra (do sąsiedniego punktu rastra) przy kącie obrotu A
0.02 rad.

Dla zmiany orientacji zobrazowanego przedmiotu przede wszystkim należy wybrać środek obrotu. Następnie oblicza się nowe współrzędne wszystkich punktów przedmiotu według powyższych równań. Na zakończenie, po oczyszczeniu ekranu, wykreśla się linie według nowych współrzędnych.

Program nr 18 obraca rysunek wokół wskazanego punktu (X0, Y0) o zadany kąt A, przy którym — po obrocie — rysunek zostaje na ekranie.

Obrót rysunku może doprowadzić do jego zniekształcenia, jeśli rozdzielczość systemu wzdłuż osi X i Y nie jest jednakowa.



Jako przykład programu realizującego przekształcenia kombinowane proponujemy program nr 19, który umożliwia składanie rysunku z kilku podanych elementów poprzez przemieszczanie, skalowanie i obrót.

Podczas konstruowania rysunków z oddzielnych części powszechnie stosuje się metody dialogowe. W uproszczonej postaci wystąpiło to w programie nr 11, gdzie w trybie dialogo-

```
QO 1 REM ****************
IB 2 REM *              *
TX 3 REM * Program nr 19 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
HI 80 GOSUB 6000
XK 90 CLOSE #1 OPEN #1,4,0,"T:"
RI 100 [illegible]
RC 110 [illegible]
EM 120 GRAPHICS [illegible]
   ,0
JB 130 GOSUB 930
ZB 140 [illegible]
TD 150 REM WCZYTANIE WYBORU
ET 160 POSITION [illegible]
MW 170 ? #1;"Wybierz:";
PR 180 FOR K=1 TO 10 NEXT K
FW 190 POSITION [illegible]
MJ 200 ? #1;"Wybierz ';
RE 210 FOR K=1 TO 10 NEXT K
DC 220 IF PEEK(764)=255 THEN [illegible]
WI 230 Z=PEEK(764):POKE 764,255 I
   F Z<>31 AND Z<>30 AND Z<>26 AN
   D Z<>24 THEN 160
VS 240 B=0:B=1*(Z=31)+2*(Z=30)+3*
   [illegible]
GM 250 [illegible]
SY 260 POSITION 8,174 POKE 765,1
MX 270 ? #1;"Wybierz
TC 280 POSITION 8,RO POKE 765,0
BO 290 ? #1;B;
RD 300 FOR K=1 TO 10 NEXT K
EW 310 POSITION 8,174
MO 320 ? #1;"Wybierz:";
IY 330 POSITION [illegible]
BF 340 ? #1;B;
RN 350 FOR K=1 TO 10:NEXT K
FJ 360 IF PEEK(764)=255 THEN 260
FS 370 [illegible] I
   F Z<>10 AND Z<>62 AND Z<>8 AND
   Z<>5 AND Z<>35 AND Z<>28 THEN
   260
OW 380 IF [illegible] THEN END
PD 390 IF Z<>35 THEN 440
LD 400 RESTORE
YF 410 GOSUB 990
TB 420 [illegible]
NH 430 GOTO 140
XE 440 IF [illegible] THEN 530
XL 450 COLOR 0
LV 460 IF B=1 AND N=1 THEN GOSUB
   1070
EQ 470 IF B=2 AND N=1 THEN GOSUB
   1100
MS 480 IF B=3 AND N=1 THEN GOSUB
   1160
NB 490 IF B=4 AND N=1 THEN GOSUB
   1430
CE 500 RESTORE
YG 510 GOSUB 990
NG 520 GOTO 140
KW 530 POKE 765,0
QF 540 GOSUB [illegible] POKE 765,1
CH 550 [illegible]
NH 560 IF Z<>10 THEN 640
XF 570 GOSUB [illegible]
[illegible] 580 COLOR 0
VB 590 IF B=1 THEN GOSUB 1510
CM 600 IF B=2 THEN GOSUB 1590
DW 610 IF B=3 THEN GOSUB 1690
LJ 620 IF B=4 THEN GOSUB [illegible]
YG 630 REM SKALOWANIE
FK 640 IF [illegible]
RB 645 IF [illegible] THEN GOSUB 960 GOTO
   160
UF 650 GOSUB 2330
[illegible] 660 [illegible]
ZP 670 IF B=1 THEN GOSUB 1920
SW 680 IF B=2 THEN GOSUB 2020
VY 690 IF B=3 THEN GOSUB 2140
UR 700 IF B=4 THEN GOSUB 2220
KN 710 REM OBROT
EE 720 IF Z<>8 THEN 790
[illegible]
   160
[illegible]
[illegible] 740 [illegible]
TV 750 IF B=1 THEN GOSUB 2400
ZB 760 IF B=2 THEN GOSUB 2540
BT 770 IF B=3 THEN GOSUB 2740
BL 780 IF B=4 THEN GOSUB [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] 830 REM WYSWIETLENIE MENU
[illegible] 840 POSITION 8,8 ? #1;"Wybierz
   ".
JC 850 POSITION 8,24 ? #1;"1.";
MH 860 POSITION 8,56:? #1;"2.",
QM 870 POSITION 8,88:? #1;"3.",
[illegible] 880 POSITION 8,120 ? #1;"4.",
[illegible] 890 COLOR [illegible]
[illegible] 900 GOSUB 990
[illegible] 910 GOSUB 1080
PG 920 GOSUB 1100
XH 930 AA=0
UK 940 GOSUB 1160
UJ 950 GOSUB 1430
RC 960 POSITION 8,174 ? #1;"Wybie
   rz
UG 970 POSITION 0,183:? #1;"P-prz
   [illegible] obrot K-kasuj N
   -nowy";
ZC 980 RETURN
ZC 990 REM [illegible] DANYCH
SI 1000 RESTORE READ XP,YP,XQ,YQ
FP 1010 [illegible] XV,YV,XW,YW,XX
   ,YX
FS 1020 READ [illegible] RX,RY
RI 1030 READ XR,YR,XS,YS,XT,YT
AL 1040 RETURN
[illegible] 1050 REM
IP 1060 REM
FJ 1070 REM LINIA
GA 1080 TRAP 1090 PLOT XP,YP DRAW
   TO XQ,YQ
BH 1090 RETURN
KF 1100 REM PROSTOKAT
KV 1110 TRAP 1150 PLOT [illegible] DRAW
   [illegible]
[illegible] 1120 DRAWTO XW,YW
[illegible] 1130 DRAWTO [illegible]
[illegible] 1140 DRAWTO [illegible]
HC 1150 RETURN
HO 1160 REM KOLO
KS 1170 IF [illegible] THEN 1200
NF 1180 R=RY
FI 1190 GOTO 1210
LW 1200 [illegible]
TH 1210 IF AA<>0 THEN 1310
OF 1220 FOR A=0 TO 1.5708 STEP 1/
   R
EI 1230 DX=RX*COS(A)
HE 1240 [illegible] TRAP 1290
WI 1250 PLOT XC+DX,YC+DY
[illegible] 1260 PLOT [illegible]
[illegible] 1270 PLOT XC-DX,YC+DY
[illegible] 1280 PLOT XC-DX,YC-DY
[illegible] 1290 [illegible]
[illegible] 1300 GOTO 1420
AT 1310 [illegible]
WT 1320 [illegible]
[illegible] 1330 FOR A=0 TO 3.14159 STEP 1
   R
MN 1340 [illegible]
[illegible] 1350 [illegible]
KN 1360 [illegible]
[illegible] 1370 [illegible]
[illegible] 1380 [illegible] TRAP 14
   10
[illegible] 1390 PLOT XC+X,YC+Y
TU 1400 PLOT XC-X,YC-Y
BQ 1410 NEXT A
HN 1420 RETURN
[illegible] 1430 REM TROJKAT
ZY 1440 TRAP 1470 PLOT XR,YR DRAW
   TO XS,YS
CG 1450 DRAWTO XT,YT
[illegible] 1460 DRAWTO [illegible]
RI 1470 RETURN
[illegible] 1480 REM
RN 1490 REM PRZESUN
IU 1500 REM PRZESUN LINIE
[illegible] 1510 IF N=0 THEN 1530
HT 1520 GOSUB [illegible]
[illegible] 1530 [illegible]
[illegible] 1540 XQ=XQ+HT YQ=YQ+VT
YU 1550 [illegible]
BF 1560 GOSUB 1070
BE 1570 RETURN
JF 1580 REM PRZESUN PROSTOKAT
CN 1590 IF N=0 THEN 1610
[illegible] 1600 GOSUB 1100
```

```
NX 1610 XU=XU+HT YU=YU+VT
QA 1620 XV=XV+HT YV=YV+VT
SD 1630 XW=XW+HT YW=YW+VT
UG 1640 XX=XX+HT YX=YX+VT
YW 1650 COLOR 1
YO 1660 GOSUB 1110
BG 1670 RETURN
KK 1680 REM PRZESUN KOLO
DI 1690 IF N=0 THEN 1710
AQ 1700 GOSUB 1160
DZ 1710 XC=XC+HT YC=YC+VT
YP 1720 COLOR 1
AZ 1730 GOSUB 1160
AZ 1740 RETURN
HI 1750 REM PRZESUN TROJKAT
IL 1760 IF N=0 THEN 1780
BI 1770 GOSUB 1430
IU 1780 XR=XR+HT YR=YR+VT
KX 1790 XS=XS+HT YS=YS+VT
LY 1800 XT=XT+HT:YT=YT+VT
YO 1810 COLOR 1
AV 1820 GOSUB 1430
AY 1830 RETURN
BH 1840 REM INSTRUKCJA
CT 1850 GRAPHICS 8+32:SETCOLOR 2,
   0,0
NQ 1860 ? "Podaj x i y przesuniec
   ia"
AB 1870 INPUT HT,VT
TH 1880 ? CHR$(125);:GRAPHICS 8+1
   6+32:SETCOLOR 2,0,0
BQ 1890 RETURN
IO 1900 REM
UC 1910 REM SKALOWANIE
IU 1920 REM
HK 1930 IF N=0 THEN 1950
BH 1940 GOSUB 1070
UD 1950 XF=(XP+XQ)/2
VJ 1960 YF=(YP+YQ)/2
HG 1970 XP=XP*HS+XF*(1-HS):YP=YP*
   VS+YF*(1-VS)
IA 1980 XQ=XQ*HS+XF*(1-HS):YP=YP*
   VS+YF*(1-VS)
ZO 1990 COLOR 1
AE 2000 GOSUB 1070
AD 2010 RETURN
KG 2020 REM PROSTOKAT
AW 2030 IF N=0 THEN 2050
XJ 2040 GOSUB 1100
RN 2050 XF=(XU+XV+XW+XX)/4
UC 2060 YF=(YU+YV+YW+YX)/4
UL 2070 XU=XU*HS+XF*(1-HS):YU=YU*
   VS+YF*(1-VS)
XI 2080 XV=XV*HS+XF*(1-HS):YV=YV*
   VS+YF*(1-VS)

AF 2090 XW=XW*HS+XF*(1-HS):YW=YW*
   VS+YF*(1-VS)
CA 2100 XX=XX*HS+XF*(1-HS):YX=YX*
   VS+YF*(1-VS)
YB 2110 COLOR 1
XF 2120 GOSUB 1100
AL 2130 RETURN
AJ 2140 REM KOLO
DL 2150 IF N=0 THEN 2170
AX 2160 GOSUB 1160
HS 2170 RX=RX*HS
OY 2180 RY=RY*VS
YZ 2190 COLOR 1
AH 2200 GOSUB 1160
AH 2210 RETURN
DK 2220 REM TROJKAT
CM 2230 IF N=0 THEN 2250
AQ 2240 GOSUB 1430
QD 2250 XF=(XR+XS+XT)/3
RZ 2260 YF=(YR+YS+YT)/3
MH 2270 XR=XR*HS+XF*(1-HS):YR=YR*
   VS+YF*(1-VS)
PE 2280 XS=XS*HS+XF*(1-HS):YS=YS*
   VS+YF*(1-VS)
SB 2290 XT=XT*HS+XF*(1-HS):YT=YT*
   VS+YF*(1-VS)
YC 2300 COLOR 1
AJ 2310 GOSUB 1430
AM 2320 RETURN
AV 2330 REM INSTRUKCJA
CH 2340 GRAPHICS 8+32:SETCOLOR 2,
   0,0
GL 2350 ? "Podaj skale na osi x i
   y"
YM 2360 INPUT HS,VS
AL 2370 ? CHR$(125);:GRAPHICS 8+1
   6+32:SETCOLOR 2,0,0:RETURN
JB 2380 REM
TD 2390 REM OBROT
EX 2400 REM LINIA
CI 2410 IF N=0 THEN 2430
AS 2420 GOSUB 1070
VZ 2430 XO=(XP+XQ)/2
XF 2440 YO=(YP+YQ)/2
PH 2450 XH=XP
TV 2460 XP=XO+(XP-XO)*COS(AR)+(YP
   -YO)*SIN(AR)
MD 2470 YP=YO+(YP-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
QA 2480 XH=XQ
WD 2490 XQ=XO+(XQ-XO)*COS(AR)+(YQ
   -YO)*SIN(AR)
MF 2500 YQ=YO+(YQ-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
YJ 2510 COLOR 1
AU 2520 GOSUB 1070
AT 2530 RETURN
KW 2540 REM PROSTOKAT
GR 2550 IF N=0 THEN 2570
XZ 2560 GOSUB 1100

UO 2570 XO=(XU+XV+XW+XX)/4
XD 2580 YO=(YU+YV+YW+YX)/4
RT 2590 XH=XU
DC 2600 XU=XO+(XU-XO)*COS(AR)+(YU
   -YO)*SIN(AR)
FQ 2610 YU=YO+(YU-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
RK 2620 XH=XV
FK 2630 XV=XO+(XV-XO)*COS(AR)+(YV
   -YO)*SIN(AR)
QU 2640 YV=YO+(YV-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
SD 2650 XH=XW
HS 2660 XW=XO+(XW-XO)*COS(AR)+(YW
   -YO)*SIN(AR)
RY 2670 YW=YO+(YW-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
SW 2680 XH=XX
KA 2690 XX=XO+(XX-XO)*COS(AR)+(YX
   -YO)*SIN(AR)
SA 2700 YX=YO+(YX-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
YN 2710 COLOR 1
XR 2720 GOSUB 1100
AX 2730 RETURN
AV 2740 REM KOLO
JB 2750 IF N=0 THEN 2780
BQ 2760 HA=AS
BM 2770 GOSUB 1160
NE 2780 AA=AR+HS
ZL 2790 COLOR 1
AT 2800 GOSUB 1160
ZB 2810 AS=AA
AW 2820 RETURN
DZ 2830 REM TROJKAT
IF 2840 IF N=0 THEN 2860
BF 2850 GOSUB 1430
TD 2860 XO=(XR+XS+XT)/3
UZ 2870 YO=(YR+YS+YT)/3
QS 2880 XH=XR
YK 2890 XR=XO+(XR-XO)*COS(AR)+(YR
   -YO)*SIN(AR)
NI 2900 YR=YO+(YR-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
QJ 2910 XH=XS
ZQ 2920 XS=XO+(XS-XO)*COS(AR)+(YS
   -YO)*SIN(AR)
OM 2930 YS=YO+(YS-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
RC 2940 XH=XT
BY 2950 XT=XO+(XT-XO)*COS(AR)+(YT
   -YO)*SIN(AR)
PQ 2960 YT=YO+(YT-YO)*COS(AR)-(XH
   -XO)*SIN(AR)
ZJ 2970 COLOR 1
BQ 2980 GOSUB 1430
BT 2990 RETURN
AH 3000 REM INSTRUKCJA
BT 3010 GRAPHICS 8+32:SETCOLOR 2,
   0,0
OW 3020 ? "Podaj kat obrotu ";
WA 3030 INPUT AR:AR=AR*3.14159/18
   0
SH 3040 ? CHR$(125);:GRAPHICS 8+1
   6+32:SETCOLOR 2,0,0
AQ 3050 RETURN
UA 3060 REM *****************
NT 3070 DATA 30,27,80,27
HH 3080 DATA 80,45,80,70,30,70,30
   ,45
IW 3090 DATA 55,90,12,12
AM 3100 DATA 30,130,80,130,55,110
TG 6000 REM TEKST W GRAFICE
PT 6005 IF PEEK(1536+255)=96 THEN
   6070
OT 6010 RESTORE 7000:S=0
AJ 6020 FOR I=1536 TO 1536+255
AS 6030 READ A:POKE I,A
WI 6040 S=S+A
FJ 6050 NEXT I
XA 6060 IF S<>28449 THEN ? "BLAD"
   :STOP
XH 6070 A=USR(1536):POKE 765,1
BC 6080 RETURN
PR 7000 DATA 104,162,0,189,26,3,2
   40,12,201,84,240,23,232,232,23
   2,224
CE 7010 DATA 36,48,240,96,169,84,
   157,26,3,169 [illegible],157,[illegible],3,169,[illegible]
WO 7020 DATA 157,28,3,96,192,6,19
   2,6,47,6,50,6,47,6,4[illegible],6
SA 7030 DATA 160,146,96,133,208,4
   1,127,201,32,16,6,24,105,64,76
   ,72
IW 7040 DATA 6,201,96,16,3,56,233
   ,32,162,0,134,204,133,203,134,
   206
HW 7050 DATA 134,207,160,16,208,1
   3,24,165,206,105,8,133,206,165
   ,207,105
LI 7060 DATA 0,133,207,70,207,102
   ,206,102,204,102,203,136,48,4,
   144,243
EX 7070 DATA 176,228,24,165,204,1
   09,244,2,133,204,160,0,177,203
   ,162,7
ZV 7080 DATA 10,72,165,208,144,10
   ,201,128,176,13,32,220,6,76,15
   1,6
MJ 7090 DATA 201,128,144,3,32,220
   ,6,230,85,208,2,230,86,104,202
   ,16
RG 7100 DATA 223,200,192,8,16,20,
   72,56,165,85,233,8,133,85,165,
   86
RR 7110 DATA 233,0,133,86,104,230
   ,84,76,124,6,56,165,84,233,7,1
   33

HC 7120 DATA 84,160,1,96,173,253,
   2,141,251,[illegible],169,[illegible],141,[illegible],3,1
   [illegible]
TY 7130 DATA 0,141,168,3,141,169,
   3,162,96,76,86,228,138,72,152,
   72
EZ 7140 DATA 165,85,133,91,165,86
   ,133,9[illegible],165,[illegible] [illegible]
MM 7150 DATA 76,245,6,198,90,32,1
   [illegible]
```

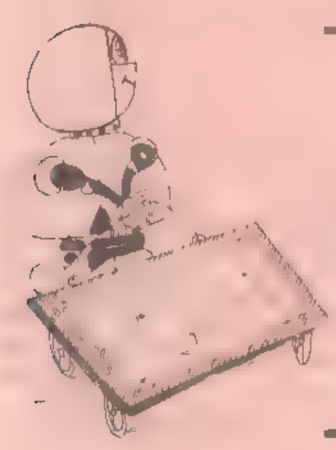
wym kolejno kreślona była linia po linii. Ten sposób tworzenia rysunku może być rozszerzony poprzez włączenie innych figur i sposobów ich przekształcania. W taki sposób można wybrać zestaw podstawowych elementów i składać z nich rysunek. W trybie dialogowym możemy sterować przemieszczaniem figur w zadane miejsce rysunku z realizacją wybranego przekształcenia. Program nr 19 tworzy rysunek z oddzielnych linii, trójkątów, prostokątów i okręgów. Z lewej strony ekranu program wyświetla menu, gdzie wymienione są wszystkie dopuszczalne elementy rysunku. Wykaz możliwych przekształceń wyświetlany jest u dołu ekranu. Poprzez migotanie słowa SELECT program wskazuje, który z dwóch zestawów jest aktualnie używany.

Program nr 19 składa się z kilku modułów realizujących całkowicie jedną operację logiczną. Do wskazy-

wania wybranego z menu wariantu używana jest klawiatura. Istnieje możliwość przystosowania go do współpracy z manipulatorem. Można by wskazywać wielkość przemieszczenia i wartości współczynników skalowania oraz kątów obrotu za pomocą trzeciego menu. Może być ono przedstawione w postaci jednej lub kilku skal liniowych, oznaczonych dla wybrania niezbędnych wartości.

Tworzenie rysunków etapami, zastosowane w powyższym programie, jest ogólnym sposobem rozwiązywania wielu zadań. Metoda ta z powodzeniem może być stosowana w zadaniach projektowania, tworzenia wykresów oraz podczas eksperymentowania z kształtem obiektów.

RUCHOME RYSUNKI



Dla „ożywienia" rysunków możemy posługiwać się podstawowymi operacjami ich przekształcania. Najpierw wyświetlamy rysunek, następnie kasujemy go, wykonujemy przekształcenia i wyświetlamy zmieniony rysunek. Jeśli tę procedurę powtórzymy kilka razy, otrzymamy ruchomy obraz. W podobny sposób tworzy się filmy animowane.

Zaczniemy, jak zwykle, od najprostszych przykładów. Programy realizujące ruch punktu po ekranie są modyfikacjami poprzednich, które wykonywały operacje przemieszczania rysunków. Zamiast pojedynczego przemieszczenia punktu, wykonuje się jego wielokrotne przemieszczenia. Można przy tym poruszać punkt zarówno wzdłuż linii prostej jak i krzywej.

Jako prosty przykład ruchu prostoliniowego rozważymy poziome przemieszczanie punktu po ekranie. Jeśli chcemy, aby punkt poruszał się z lewa na prawo, rozpoczniemy ruch od współrzędnych (X1, Y), a zakończymy w (X2, Y), przy czym X2 > X1. Ruch możemy przedstawić w postaci ciągu przemieszczeń punktu od X1 do X2, a każde przemieszczenie równe jest krokowi rastra. W każdym kroku gasimy wcześniej zapalony punkt (X, Y) i wyświetlamy nowy (X+1, Y). Ruch w lewo uzyskuje się

```
QO 1 REM *****************
IB 2 REM *               *
NK 3 REM * Program nr 20 *
ID 4 REM *               *
QS 5 REM *****************
NL 6 REM
ER 15 GRAPHICS 0 POSITION 3.10
DZ 30 ? "Podaj x dla lewej i praw
   ej sciany "
UW 50 XM=79:YM=47
EM 60 GRAPHICS 0:POSITION 3,10
ED 70 ? "Podaj x dla lewej i praw
   ej sciany "
LJ 80 ? ,:INPUT XL,XP
PN 90 IF XL =XP OR XL<0 OR XP>XM
    THEN 70
UH 100 DX=1 X=XL+INT((XP-XL)/2)
WS 110 Y=INT(YM 2)
HJ 120 GRAPHICS 5+16
TF 140 COLOR 1 PLOT XL,0 DRAWTO X
   L,YM PLOT XP,0 DRAWTO XP YM
VE 150 COLOR 1 PLOT X,Y
WM 155 FOR k=1 TO 5 NEXT k
ZV 160 IF X<=XL+1 OR X>=XP-1 THEN
    DX=-DX
UZ 180 COLOR 0 PLOT X,Y
MB 190 X=X+DX
NK 200 GOTO 150
```

poprzez zmniejszanie współrzędnej X o 1 w każdym kroku. Łącząc je razem możemy otrzymać ruch punktu w prawo i w lewo między X1 i X2. Ilustruje to program nr 20. Początkowo punkt porusza się w prawo z dodatnim przyrostem współrzędnej X (DX = 1). Kiedy osiągnie współrzędne (X2, Y), znak przyrostu zmienia się w (DX = — DX) i punkt porusza się w lewo. W punkcie (X1, Y) znów zmienia się znak przyrostu. Program podtrzymuje ten ruch przez czas nieograniczony.

Powyższy program można rozszerzyć dodając granice pionowe we współrzędnych X1 i X2, aby ruchomy punkt odskakiwał od nich.

Należy jednak zmienić punkt odbicia, aby nie ścierać punktu, który należy do ścianki. Dlatego punkt ruchomy powinien zmienić kierunek o jeden krok przed uderzeniem o ściankę.

Ruch punktu w górę i w dół oznacza, że ustalamy granice poziome o współrzędnych Y1 i Y2 i przemieszczamy punkt przy ustalonej wartości współrzędnej X. Aby wykonać ruch w dowolnym kierunku, należy wprowadzić jednocześnie przyrosty dla obu współrzędnych. Proces ten jest podobny do wykreślania linii prostej, z wyjątkiem tego, że ścieramy poprzedni punkt zanim wyświetlimy nowy. Trajektoria ruchu może być określona różnymi sposobami: można wskazać graficzne punkty drogi lub posłużyć się równaniem prostej (tj. określić współczynniki M i B) lub wybrać odpowiednie przyrosty współrzędnych DX i DY.

Przykład ruchu po przekątnej demonstruje **program nr 21**, który symuluje odbicie punktu od wewnętrznych ścian pudełka. Wprowadzamy rozmiary pudełka i punkt rozpoczyna ruch w prawo w dół z jednakowym przyrostem obu współrzędnych. Kie-

```
QO 1 REM *****************
IB 2 REM *               *
OF 3 REM * Program nr 21 *
ID 4 REM *               *
QS 5 REM *****************
NL 6 REM
UW 50 XM=79 YM=47
NQ 60 GRAPHICS 0:POSITION 3,8
RL 70 ? :? "Podaj x dla lewej i p
   rawej sciany "
LJ 80 ? ,:INPUT XL,XP
PN 90 IF XL>=XP OR XL<0 OR XP>XM
   THEN 70
AH 100 ? :? "Podaj y dla gornej i
    dolnej sciany"
UA 110 ? ,:INPUT YG,YD
UP 120 IF YG>=YD OR YG<0 OR YD>YM
    THEN 100
DE 130 DX=1
DM 140 DY=1
UT 150 X=XL+INT((XP-XL)/2)
JZ 160 Y=YG+INT((YD-YG)/2)
HT 170 GRAPHICS 5+16
XZ 180 COLOR 1
WZ 190 REM *** KRESLENIE PROSTOKA
   TA ***
LZ 200 PLOT XL,YG:DRAWTO XP,YG
YY 210 DRAWTO XP,YD
XA 220 DRAWTO XL,YD
YY 230 DRAWTO XL,YG
OL 240 REM *** RUCH PUNKTU ***
VF 250 COLOR 1:PLOT X,Y
WN 255 FOR K=1 TO 5:NEXT K
ZW 260 IF X<=XL+1 OR X>=XP-1 THEN
    DX=-DX
SA 270 IF Y<=YG+1 OR Y>=YD-1 THEN
    DY=-DY
VA 280 COLOR 0:PLOT X,Y
MC 290 X=X+DX
MH 300 Y=Y+DY
NX 310 GOTO 250
```

dy punkt dotknie ściany pudełka, zmienia kierunek. Jeśli odbija się od ścianki pionowej, zmienia się znak przyrostu współrzędnej X (DX= — DX). Jeśli punkt odbija się od ściany poziomej, zmieni się znak przyrostu współrzędnej Y (DY= — DX). Gdy punkt wpadnie w kąt pudełka, zmienia się znak obu przyrostów.

Ruch punktu można przyspieszyć zwiększając wartość przyrostów DX i DY lub zwolnić (w zadanym kierunku), jeśli wybierze się krok mniejszy od 1 (np. 0.5).

Program ten można również zmodyfikować tak, aby działał prawidłowo przy dowolnych wartościach przyrostów DX i DY, poprzez zmianę warunków odbicia. Kierunek ruchu punktu powinien być zmieniony, jeśli kolejna zmiana choćby jednej współrzędnej prowadzi do przekroczenia granic pudełka. W tym celu w programie nr 21 należy zamienić linie 260 i 270 następującym fragmentem (**program nr 22**):

```
260  IF DX>0 THEN 268
264  IF X+DX<=XL THEN DX=—DX
266  GOTO 270
268  IF X+DX>=XP THEN DX=—DX
270  IF DY>0 THEN 278
274  IF Y+DY<=YG THEN DY=—DY
276  GOTO 280
278  IF Y+DY>=YD THEN DY=—DY
```

W powyższym fragmencie programu punkt zmienia kierunek ruchu jeszcze przed dotknięciem ściany. Dla bardziej prawdopodobnego odwzorowania odbicia można wyświetlić

```
QO 1 REM ****************
IB 2 REM * [illegible] *
PA 3 REM * Program nr 22 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
UW 50 XM=79 YM=47
NV 60 GRAPHICS 0 POSITION 3 8
RL 70 ? :? "Podaj x dla lewej i p
   rawej sciany "
LJ 80 ? ,:INPUT XL,XP
FN 90 IF XL>=XP OR XL<0 OR XP>XM
   THEN 70
AH 100 ? :? "Podaj y dla gornej i
    dolnej sciany"
UA 110 ? ,:INPUT YG,YD
UP 120 IF YG>=YD OR YG<0 OR YD>YM
    THEN 100
DE 130 DX=1
DM 140 DY=1
UT 150 X=XL+INT((XP-XL)/2)
JZ 160 Y=YG+INT((YD-YG)/2)
HT 170 GRAPHICS 5+16
XZ 180 COLOR 1
WZ 190 REM *** KRESLENIE PROSTOKA
   TA ***
LZ 200 PLOT XL,YG DRAWTO XP,YG
YY 210 DRAWTO XP,YD
XA 220 DRAWTO XL,YD
YY 230 DRAWTO XL,YG
OL 240 REM *** RUCH PUNKTU ***
VF 250 COLOR 1:PLOT X,Y
SB 255 FOR K=1 TO 10:NEXT K
EI 260 IF DX>0 THEN 268
DR 264 IF X+DX<=XL THEN DX=-DX
PU 266 GOTO 270
HJ 268 IF X+DX>=XP THEN DX=-DX
FM 270 IF DY>0 THEN 278
EB 274 IF Y+DY<=YG THEN DY=-DY
QH 276 GOTO 280
DS 278 IF Y+DY>=YD THEN DY=-DY
VA 280 COLOR 0 PLOT X Y
MC 290 X=X+DX
MH 300 Y=Y+DY
NX 310 GOTO 250
```

punkt w pobliżu ściany przed zmianą kierunku ruchu. W tym celu tak dobieramy rozmiary pudełka, aby były wielokrotnościami przyrostów w odpowiednich kierunkach. Ogólny sposób rozwiązania jest następujący: trajektoria ruchu punktu rzutowana jest na ściany i określane są współrzędne punktu przecięcia. W tym wypadku można dokładniej pokazać odbicie punktu od ściany przy dowolnych wartościach przyrostu współrzędnych.

Na podstawie bieżących wartości współrzędnych można obliczyć współrzędne punktu przecięcia (XI, YI), według równania prostoliniowego:

$$XI = M \cdot XI + (Y - M \cdot X)$$

gdzie XI = XP prawej granicy, XI = XL lewej granicy obszaru. Współczynnik nachylenia trajektorii ruchu obliczamy następująco:

$$M = DY/DX$$

Powyższe równania można wykorzystać do wyznaczenia punktu przecięcia przy dowolnych granicach i kierunkach ruchu. Przy wyznaczaniu punktów przecięcia z górą lub dnem pudełka wartość XI można obliczyć za pomocą powyższego równania w następujący sposób: $XI = X + (YI - Y)/M$, gdzie YI = YG lub YI = YD. W tych wszystkich wypadkach wyświetlamy punkt odbicia w odległości kroku rastra od ścianki, a następnie zmieniamy kierunek ruchu.

Ruch okręgu jest podstawą przy rozwiązywaniu różnorodnych zadań tworzenia ruchomych obrazów. Rozpatrzone powyżej metody tworzenia efektu ruchu punktów, w pełni stosuje się do symulowania odbicia piłki od ścian, jeśli przyjąć, że piłka jest większym punktem: przemieszczamy środek okręgu i obserwujemy moment dotknięcia okręgu do ścianki pudełka. Imitacja odbić piłki od ścian pudełka realizowana jest przez program nr 23 z wykorzystaniem rozpatrzonych powyżej metod. Dla modelowania odbicia znajdujemy punkt przecięcia trajektorii piłki ze ścianką i zmieniamy kierunek ruchu piłki przy odległości 1 kroku rastra między piłką a ścianką.

```
QO 1 REM ****************
IB 2 REM *              *
PV 3 REM * Program nr 23 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
DV 50 XM=319:YM=191
SK 55 TRAP 70
NQ 60 GRAPHICS 0:POSITION 3,8
RL 70 ? :? "Podaj x dla lewej i p
   rawej sciany "
LJ 80 ?/,:INPUT XL,XP
PN 90 IF XL>=XP OR XL<0 OR XP>XM
   THEN 70
KU 95 TRAP 100
AH 100 ? :? "Podaj y dla gornej i
    dolnej sciany"
UA 110 ? ,:INPUT YG,YD
UP 120 IF YG>=YD OR YG<0 OR YD>YM
    THEN 100
EK 130 DX=5
BR 140 DY=5:R=3
FA 150 X=XL+INT((XP-XL)/2):XN=X
WH 160 Y=YG+INT((YD-YG)/2) YN=Y
TJ 170 GRAPHICS 8+16
XZ 180 COLOR 1
WZ 190 REM *** KRESLENIE PROSTOKA
   TA ***
LZ 200 PLOT XL,YG:DRAWTO XP,YG
YY 210 DRAWTO XP,YD
XA 220 DRAWTO XL,YD
YY 230 DRAWTO XL,YG
RO 240 REM *** RUCH PILKI ***
XR 290 COLOR 0
SQ 300 A=X:B=Y:GOSUB 1000
XN 310 COLOR 1
NQ 320 A=XN B=YN:GOSUB 1000
RB 330 X=XN
RP 340 Y=YN
CS 350 BX=0
DA 360 BY=0
OX 370 M=DY/DX
AS 410 IF DX>0 THEN 480
XI 420 REM *** RUCH W LEWO ***
OF 430 IF X+DX-R>XL THEN 530
CZ 440 BX=1
IW 450 XN=XL+R+1
OQ 460 GOTO 530
RD 470 REM
AV 480 REM *** RUCH W PRAWO ***
PD 490 IF X+DX+R<XP THEN 530
CS 500 BX=1
LR 510 XN=XP-R-1
QU 520 REM
QW 530 REM
XP 540 IF DY>0 THEN 610
OT 550 REM *** RUCH W GORE ***
RH 560 IF Y+DY-R>YG THEN 670
DM 570 BY=1
HX 580 YN=YG+R+1

QZ 590 GOTO 670
QR 600 REM
RZ 610 REM *** RUCH W DOL ***
NE 620 IF Y+DY+R<YD THEN 670
DF 630 BY=1
IH 640 YN=YD-R-1
RB 650 REM
RD 660 REM
NA 670 REM *** ROZPOZNANIE ODBICI
   A ***
TO 680 IF BX=0 AND BY=0 THEN 730
YS 690 IF BX=0 AND BY=1 THEN 770
SH 700 IF BX=1 AND BY=0 THEN 810
XL 710 IF BX=1 AND BY=1 THEN 850
QW 720 REM
JV 730 REM *** BEZ ODBICIA ***
RH 740 XN=X+DX
SH 750 YN=Y+DY
QD 760 GOTO 290
CM 770 REM *** ODBICIE OD SCIANY
   Y ***
LS 780 XN=(YN-Y)/M+X
IQ 790 DY=-DY
PS 800 GOTO 290
AW 810 REM *** ODBICIE OD SCIANY
   X ***
IQ 820 YN=(XN-X)*M+Y
HP 830 DX=-DX
QA 840 GOTO 290
AO 850 REM *** ODBICIE W ROGU ***
RF 860 REM
CA 870 IF ABS(XN-X)<ABS(YN-Y) THE
   N 810
HX 880 IF ABS(YN-Y)<ABS(XN-X) THE
   N 770
RL 890 REM
HK 900 DX=-DX
IC 910 DY=-DY
PX 920 GOTO 290
MF 1000 REM PODPROGRAM KRESLENIA
   PILKI
OS 1010 FI=0:Y1=0:X1=R
SY 1020 FIY=FI+Y1+Y1+1
HY 1030 FIXY=FIY-X1-X1+1
HI 1040 PLOT A+X1,B+Y1:PLOT A-X1,
   B+Y1
LF 1050 PLOT A+X1,B-Y1 PLOT A-X1,
   B-Y1
HE 1060 PLOT A+Y1,B+X1:PLOT A-Y1,
   B+X1
LB 1070 PLOT A+Y1,B-X1:PLOT A-Y1,
   B-X1
NJ 1080 FI=FIY:Y1=Y1+1
XG 1090 IF ABS(FIXY)<ABS(FIY) THE
   N FI=FIXY:X1=X1-1
KF 1100 IF X1>=Y1 THEN 1020
AE 1110 RETURN
```

Program można zmodyfikować tak, aby wprowadzać rozmiary pudełka, wielkość i początkowe położenie piłki.

Przemieszczenie punktu lub okręgu po zadanej trajektorii krzywoliniowej wykonuje się za pomocą obliczania bieżącego położenia według równania trajektorii. Ruch rozpoczyna się z określonego punktu początkowego i kończy w zadanym punkcie końcowym. Ruch punktu po okręgu lub elipsie można wykonać metodą tworzenia okręgów lub za pomocą operacji obrotu. W celu otrzymania równomiernego ruchu po okręgu, kolejno wyświetlane punkty powinny być rozmieszczane na trajektorii ze stałym krokiem. Oznacza to, że należy wykorzystać równania trygonometryczne dla obliczania bieżącego położenia przy stałym kroku kątowym. Stała zmiana promienia okręgu powoduje ruch po spirali. Obrót po okręgu jest podstawą podczas modelowania orbit sputników, planet systemu słonecznego, a także detali maszyn.

Ruch punktu po paraboli symuluje lot ciała rzuconego pod kątem. Długość lotu zależy od prędkości począt-

kowej i kąta rzutu zmierzonego od poziomu. Znając prędkość początkową S i kąt rzutu A, można obliczyć długość lotu R i maksymalną wysokość HT:

$$R = S \cdot S \cdot SIN(2 \cdot A)/G$$

$$HT = ((S \cdot SIN(A))^{2})/(2 \cdot G)$$

gdzie G — przyspieszenie ziemskie (980 cm/s·s).

Współrzędne punktów trajektorii ruchu można określić zmieniając X od 0 do R i obliczając odpowiadające wartości Y z równania:

$$Y = C1 \cdot X^{2} + C2 \cdot X + Y0$$

gdzie Y0 — dowolnie wybrane położenie początkowe na ekranie, a współczynniki C1 i C2 oblicza się według wzorów:

$$C1 = G/(2 \cdot (S \cdot COS(A))^{2})$$

$$C2 = -TAN(A)$$

Program nr 24 zawiera ilustrację ruchu punktu po paraboli. Współrzędne pozycji początkowej (X0, Y0), a także kąt A i prędkość początkowa S wprowadzane są za pomocą instrukcji INPUT. Wartość kąta A ograniczona jest zakresem 0—90°; z tego powodu ruch odbywa się tylko w prawo. Wartości S w zakresie 200—600 zapewniają trajektorię z maksymalną wysokością HT≈200 i długością lotu R≈200. Dla modelowania zbliżonych wariantów ruchu w programie można zmienić punkt początkowy i końcowy. Rzucanie przedmiotu z dachu domu lub szczytu góry różni się tym, że punkt początkowy i końcowy ma różne wartości współrzędnej Y.

```
QO 1 REM ****************
IB 2 REM *              *
QQ 3 REM * Program nr 24 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
OM 50 XM=319:YM=159:DEG
YU 60 GRAPHICS 8:COLOR 1
GL 70 ? CHR$(125);"Podaj x i y pu
   nktu startu ";:INPUT X0,Y0
LW 80 IF X0·0 OR X0=XM OR Y0<0 OR
    Y0>YM THEN 70
HH 110 ? CHR$(125);"Podaj kat (0-
   90) ",
IV 120 INPUT A
PC 130 ? "Podaj szybkosc (100-600
   ) ";
QP 140 INPUT S
PL 160 G=980
CM 170 R=S*S*SIN(2*A)/G
KP 180 IF X0+R>XM THEN 110
MH 210 HT=((S*SIN(A))^2)/(2*G)
AH 220 IF HT<=0 OR HT>YM THEN 110
EG 260 C1=G/(2*(S*COS(A))^2)
DZ 270 C2=-SIN(A)/COS(A)
HG 300 REM *** RUCH PUNKTU PO PAR
   ABOLI
KH 310 FOR X=0 TO R
UX 320 Y=C1*X*X+C2*X+Y0
SI 330 COLOR 1:PLOT X+X0,Y
LX 350 NEXT X
PY 360 GOTO 70
```

Modelowanie ruchu podskakującej piłki ilustrowane jest przez **program nr 25**. Piłka spada z pewnej wysokości H i skacze wzdłuż ekranu. Wysokość podskoków zmniejsza się. Dla uzyskania ruchów wahadłowych piłki używa się funkcji SIN(X), a dla stałego zmniejszania amplitudy — funkcji EXP(X). Jeśli chcemy, żeby ruch rozpoczynał się w punkcie (0, H), weźmiemy D = PI/2 dla funkcji SIN. Dalej wybieramy odległość między punktami uderzenia o ziemię równą 40, żeby na ekranie zmieściło się kilka takich punktów. W tym celu przyjmiemy W = PI/40. Naszym zamiarem jest, aby przy każdym podskoku piłka wyświetlała się dokładnie w chwili dotykania ziemi. Oznacza to, że krok zmiany współrzędnej X powinien być podzielnikiem liczby 40. W naszym przykładzie krok zmiany X równy jest 4, dlatego piłka obowiązkowo będzie się wyświetlać przy każdym uderzeniu o ziemię w punktach X = 20, 60, 100... itd.

Najprostsze ruchy linii i wielokątów wykonuje się podstawowymi metodami, rozpatrzonymi powyżej. Mu-

```
QO 1 REM ****************
IB 2 REM *              *
RL 3 REM * Program nr 25 *
ID 4 REM *             .*
QS 5 REM ****************
NL 6 REM
QS 60 XM=319:YM=191:P=7*16*16*16:
   GOSUB 28000
XJ 70 GRAPHICS 0:POSITION 0,10:?
   "Podaj wysokosc z Jakiej spada
    pilka ";
LT 80 INPUT H
ES 90 W=3.14159/40
UC 100 D=90*3.14159/180
DE 110 K=0.01
EM 120 GRAPHICS 8+16:SETCOLOR 2,0
   ,0
XP 130 COLOR 1
FO 140 PLOT 0,YM:DRAWTO XM,YM
QW 150 REM
VI 160 REM ********************
MY 170 FOR XN=0 TO XM-10 STEP 4
CO 180 YN=H*SIN(W*XN+D)*EXP(-K*XN
   )
NF 190 YN=YM-ABS(YN)-3
MD 195 TRAP 220
YL 200 I=USR(P,X,Y,2,0)
RX 210 I=USR(P,XN+10,YN,2,1)
SZ 220 X=XN+10
RM 230 Y=YN
SU 240 NEXT XN
ZI 245 I=USR(P,X,Y,2,0)
EE 250 IF PEEK(764)=255 THEN 250
WU 260 POKE 764,255
RZ 270 GOTO 70
AD 28000 ON PEEK(28845)=237 GOTO
   28010:FOR I=0 TO 758:READ A:PO
   KE 28672+I,A:NEXT I
OU 28004 RESTORE 29500
MT 28005 FOR I=1577 TO 1584:READ
   A:POKE I,A:NEXT I
DS 28010 RETURN
GW 28672 DATA 104,104,141,5,6,104
EG 28678 DATA 141,4,6,104,141,7
CY 28684 DATA 6,104,141,6,6,104
DA 28690 DATA 141,9,6,141,12,6
DT 28696 DATA 104,141,8,6,141,11
YR 28702 DATA 6,104,104,141,10,6
UB 28708 DATA 201,3,144,1,96,169
YD 28714 DATA 0,141,13,6,141,14
DX 28720 DATA 6,141,15,6,141,16
KB 28726 DATA 6,24,173,4,6,109
HL 28732 DATA 11,6,141,25,6,173
EH 28738 DATA 5,6,109,12,6,141
JG 28744 DATA 26,6,24,173,4,6
JS 28750 DATA 109,13,6,141,29,6
JG 28756 DATA 173,5,6,109,14,6
IZ 28762 DATA 141,30,6,56,173,4
IX 28768 DATA 6,237,11,6,141,27
IX 28774 DATA 6,173,5,6,237,12
QH 28780 DATA 6,141,28,6,56,173
GF 28786 DATA 4,6,237,13,6,141
FX 28792 DATA 31,6,173,5,6,141
EG 28798 DATA 14,6,141,32,6,24
FY 28804 DATA 173,6,6,109,11,6
IM 28810 DATA 141,33,6,173,7,6
FR 28816 DATA 109,12,6,141,34,6
IL 28822 DATA 24,173,6,6,109,13
NE 28828 DATA 6,141,37,6,173,7
LB 28834 DATA 6,109,14,6,141,38
PP 28840 DATA 6,56,173,6,6,237
JH 28846 DATA 11,6,141,35,6,173
FR 28852 DATA 7,6,237,12,6,141
PY 28858 DATA 36,6,56,173,6,6
MP 28864 DATA 237,13,6,141,39,6
LG 28870 DATA 173,7,6,237,14,6
JE 28876 DATA 141,40,6,173,25,6
GV 28882 DATA 141,0,6,173,26,6
KM 28888 DATA 141,1,6,173,37,6
LL 28894 DATA 141,2,6,173,38,6
CX 28900 DATA 141,3,6,32,106,114
DD 28906 DATA 173,27,6,141,0,6
ED 28912 DATA 173,28,6,141,1,6
QN 28918 DATA 32,106,114,173,25,6
GK 28924 DATA 141,0,6,173,26,6
KF 28930 DATA 141,1,6,173,39,6
EG 28936 DATA 141,2,6,173,40,6
DX 28942 DATA 141,3,6,32,106,114
ED 28948 DATA 173,27,6,141,0,6
FD 28954 DATA 173,28,6,141,1,6
UB 28960 DATA 32,106,114,173,29,6
CO 28966 DATA 141,0,6,173,30,6
FL 28972 DATA 141,1,6,173,33,6
IE 28978 DATA 141,2,6,173,34,6
EX 28984 DATA 141,3,6,32,106,114
AB 28990 DATA 173,31,6,141,0,6
CV 28996 DATA 173,32,6,141,1,6
SO 29002 DATA 32,106,114,173,29,6
BB 29008 DATA 141,0,6,173,30,6
FW 29014 DATA 141,1,6,173,35,6
GV 29020 DATA 141,2,6,173,36,6
DK 29026 DATA 141,3,6,32,106,114
YO 29032 DATA 173,31,6,141,0,6
BI 29038 DATA 173,32,6,141,1,6
ND 29044 DATA 32,106,114,173,14,6
DO 29050 DATA 205,12,6,240,3,144
LQ 29056 DATA 10,96,173,13,6,205
ML 29062 DATA 11,6,144,1,96,173
CW 29068 DATA 11,6,133,4,173,12
FX 29074 DATA 6,133,5,173,13,6
UE 29080 DATA 133,205,173,14,6,13
   3
MZ 29086 DATA 206,6,4,38,5,6
NT 29092 DATA 205,38,206,56,165,2
   05
JM 29098 DATA 109,15,6,141,17,6
XF 29104 DATA 165,206,109,16,6,14
   1
JV 29110 DATA 18,6,24,173,17,6
QS 29116 DATA 229,4,141,19,6,173
EB 29122 DATA 18,6,229,5,141,20
NC 29128 DATA 6,173,18,6,16,27
SG 29134 DATA 73,255,141,22,6,173
MW 29140 DATA 17,6,73,255,24,105
BQ 29146 DATA 1,141,21,6,173,22
XF 29152 DATA 6,105,0,141,22,6
GR 29158 DATA 24,144,9,141,22,6
EX 29164 DATA 173,17,6,141,21,6
LD 29170 DATA 173,20,6,16,27,73
WV 29176 DATA 255,141,24,6,173,19
ER 29182 DATA 6,73,255,24,105,1
IS 29188 DATA 141,23,6,173,24,6
AV 29194 DATA 105,0,141,24,6,24
MW 29200 DATA 144,9,141,24,6,173
JO 29206 DATA 19,6,141,23,6,173
IV 29212 DATA 17,6,141,15,6,173
GH 29218 DATA 18,6,141,16,6,24
BR 29224 DATA 173,13,6,105,1,141
EI 29230 DATA 13,6,173,14,6,105
CQ 29236 DATA 0,141,14,6,173,22
MY 29242 DATA 6,205,24,6,144,39
LV 29248 DATA 208,8,173,21,6,205
UQ 29254 DATA 23,6,144,29,173,19
DH 29260 DATA 6,141,15,6,173,20
OM 29266 DATA 6,141,16,6,56,173
UO 29272 DATA 11,6,233,1,141,11
BE 29278 DATA 6,173,12,6,233,0
LJ 29284 DATA 141,12,6,76,55,112
WE 29290 DATA 173,2,6,133,205,169
ML 29296 DATA 0,133,206,6,205,38
ND 29302 DATA 206,6,205,38,206,6
TA 29308 DATA 205,38,206,165,205,
   133
HO 29314 DATA 4,165,206,133,5,6
UP 29320 DATA 205,38,206,6,205,38
PG 29326 DATA 206,24,165,205,101,
   4
```

```
YO 29332 DATA 133,205,165,206,101
   ,5
LU 29338 DATA 133,206,173,0,6,133
DX 29344 DATA 4,173,1,6,133,5
AU 29350 DATA 70,5,102,4,70,5
WN 29356 DATA 102,4,70,5,102,4
OU 29362 DATA 24,165,205,101,4,13
   3
AC 29368 DATA 205,165,206,101,5,1
   33
LI 29374 DATA 206,24,165,205,101,
   88
WZ 29380 DATA 133,205,165,206,101
   ,89

GI 29386 DATA 133,206,173,0,6,41
CF 29392 DATA 7,170,160,0,173,10
NN 29398 DATA 6,208,10,189,41,6
MW 29404 DATA 73,255,49,205,145,2
   05
VS 29410 DATA 96,201,1,208,8,189
PS 29416 DATA 41,6,17,205,145,205
TU 29422 DATA 96,189,41,6,81,205
XK 29428 DATA 145,205,96,0,0,0
MN 29500 DATA 128,64,32,16,8,4,2,
   1
```

simy wyświetlić wszystkie linie, tworzące rysunek przedmiotu, wykasować je i ponownie wyświetlić w nowym miejscu ekranu. Wielokrotne powtórzenie tego ciągu operacji modeluje ruch przedmiotu. Dodatkowe trudności wynikają w tych wypadkach, kiedy chcemy jednocześnie z ruchem obracać lub skalować przedmiot, albo różnie przemieszczać różne części przedmiotu.

Odcinek utworzony między punktami (X, Y1) i (X, Y2) porusza się w kierunku pionowym tylko przy kolejnej zmianie współrzędnej Y. **Program nr 26** przedstawia wahadłowy ruch w górę i w dół linii pionowej między dwiema zadanymi granicami. Ruch tego odcinka można przyśpieszyć lub zwolnić wybierając odpowiednią wartość DY w przedziale od 1 do 20. Program nie pozwala odcinkowi osiągnąć lub przeciąć granicy ustanowionej od góry (YG) i od dołu (YD).

```
QO 1 REM ****************
IB 2 REM *              *
SG 3 REM * Program nr 26 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
EQ 50 XM=319:YM=159:GRAPHICS 8:SE
   TCOLOR 2,0,0:POKE 752,1
VX 60 ? CHR$(125);"Podaj gorna i
   dolna wartosc y ";
YD 70 INPUT YT,YB
FA 80 IF YT>=YB OR YT<0 OR YB>YM
   THEN 60
PJ 90 ? CHR$(125);"Podaj wielkosc
    zmiany y";
ZA 100 INPUT DY
WC 110 X=INT(XM/2)
MJ 120 Y1=YT+INT((YB-YT)/2)-20
KT 130 Y2=YT+INT((YB-YT)/2)+20
LI 140 ? CHR$(125);
YB 190 COLOR 1
IN 200 PLOT X,Y1:DRAWTO X,Y2
XK 210 IF DY>0 THEN 250
OR 220 REM
ZA 230 IF Y1+DY<=YT THEN DY=-DY
OY 240 GOTO 270
QX 250 REM
PN 260 IF Y2+DY>=YB THEN DY=-DY
RB 270 REM
XP 280 COLOR 0
JF 290 PLOT X,Y1:DRAWTO X,Y2
JS 300 Y1=Y1+DY
KJ 310 Y2=Y2+DY
PH 320 GOTO 190
```

Odcinek pionowy może poruszać się w kierunku poziomym, jeśli zamiast współrzędnej Y zmieniać współrzędną X. W tym wypadku powinniśmy ograniczyć jego ruch ściankami pionowymi, posiadającymi współrzędne XL i XP. Jednoczesna zmiana obu współrzędnych powoduje ruch po przekątnej. Oprócz tego można otrzymać wahadłowy ruch odcinka względem prostej, wykorzystując metody symulowania ruchu punktu. Ruch odcinków poziomych i wykreślonych pod dowolnym kątem, może być realizowany w analogiczny sposób. Przemieszczamy oba końce odcinka wykorzystując dla nich jednakowe przyrosty DX i DY. Odcinek przemieszcza się szybciej przy zwiększeniu przyrostów i zmienia kierunek ruchu przy zmianie znaku przyrostu jednej lub obu współrzędnych.

Odcinek można przemieszczać po trajektorii krzywoliniowej, zmieniając wartości DX i DY w czasie ruchu odcinka. Dla zapewnienia ruchu odcinka po zadanej krzywej niezbędne jest, aby wybrany punkt odcinka podczas jego ruchu zawsze poruszał się po tej krzywej. Np. program nr 27 ukazuje przemieszczanie jednego końca odcinka poziomego po okręgu. Odcinek porusza się do przodu i do tyłu, w górę i w dół.

```
QO 1 REM ****************
IB 2 REM *              *
TB 3 REM * Program nr 27 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
DW 60 XM=319:YM=191
PX 70 XC=INT(XM/2)-40
ZF 80 YC=INT(YM/2)
H  90 R=50
ME 100 DEG
IX 110 GRAPHICS 8+16
AP 120 REM RUCH ODCINKA
LL 130 FOR A=15 TO 360 STEP 15
XG 140 COLOR 0
AA 150 PLOT X,Y:DRAWTO X+80,Y
XA 160 X=XC+R*COS(A)
ZG 170 Y=YC+R*SIN(A)
XZ 180 COLOR 1
AI 190 PLOT X,Y:DRAWTO X+80,Y
IU 195 FOR W=1 TO 30:NEXT W
IG 200 NEXT A
MQ 210 GOTO 130
NX 220 END
```

W pewnych wypadkach wymaga się, aby odcinek zmieniał swój kierunek w miarę ruchu wzdłuż zadanej krzywej. Dla uzyskania takiego ruchu musimy odpowiednio zmieniać współczynnik nachylenia odcinka w każdym punkcie trajektorii lotu. Dla paraboli współczynnik nachylenia M stycznej do krzywej można obliczyć dla zadanej wartości X według wzoru:

$$M = 2 \cdot C1 \cdot X + C2$$

Współrzędne (X1, Y1) drugiego końca odcinka można określić znając współczynnik nachylenia M i długość odcinka L.

$$X1 - X = L \cdot COS(AS)$$
$$Y1 - Y = L \cdot SIN(AS)$$

gdzie: AS — kąt między kierunkiem odcinka a poziomem, określony z równania: AS = ATN(M).

Imitację lotu strzały wzdłuż paraboli przedstawia program nr 28. Grot strzały rysujemy za pomocą krótkich odcinków, które schodzą się w punkcie (X1, Y1) i mają współczynnik nachylenia nieznacznie różniący się od współczynnika nachylenia strzały. W programie współczynnik nachylenia dla jednego krótkiego odcinka wybieramy o 0.75 większy od M, a dla drugiego odcinka o 0.75 mniejszy od M. Długość krótkiego odcinka równa jest 8, a długość całej strzały — 40.

Przemieszczenie linii, stycznej do dowolnej krzywej, można wykonać

```
QO 1 REM ****************
IB 2 REM *              *
TW 3 REM * Program nr 28 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
ZD 100 XM=319:YM=159:POKE 82,0
UO 110 GRAPHICS 8:COLOR 1:SETCOLO
   R 2,0,0
YE 120 ? CHR$(125);"Podaj wspolrz
   edne poczatkowe X,Y";:INPUT XO
   ,YO
LP 130 IF XO<0 OR XO>XM OR YO<0 O
   R YO>YM THEN 120
JS 140 ? CHR$(125);"Podaj kat str
   zalu";:INPUT A
KG 150 A=A*3.14159/180
RB 160 ? CHR$(125);"Podaj predkos
   c (100-600)";:INPUT S
GN 170 G=980:RAD
RI 180 LA=40
GK 190 LT=8
JE 200 REM OBLICZENIE PULAPU I ZA
   SIEGU LOTU
CB 210 R=S*S*SIN(2*A)/G
XL 220 IF XO+R>XM THEN 140
ML 230 HT=((S*SIN(A))^2)/(2*G)
KW 240 IF HT<0 OR HT>YM THEN 140
EE 250 C1=G/(2*(S*COS(A))^2)
DX 260 C2=-SIN(A)/COS(A)
XR 270 ? CHR$(125);:POKE 752,1
KX 300 REM **RUCH STRZALY**
LW 310 FOR X=0 TO R STEP R/10
ST 320 Y=C1*X*X+C2*X+YO
YM 330 M=2*C1*X+C2
AS 340 A1=ATN(M)
UO 350 Y1=Y+LA*SIN(A1)
SG 360 X1=X+LA*COS(A1)
YI 370 IF X1>XM OR YM>YM THEN END
CN 380 GOSUB 700:REM KASOWANIE
FZ 390 M2=M+0.75
BG 400 A2=ATN(M2)
SJ 410 X2=X1-LT*COS(A2)
VA 420 Y2=Y1-LT*SIN(A2)
GM 430 M3=M-0.75
CH 440 A3=ATN(M3)
TQ 450 X3=X1-LT*COS(A3)
WH 460 Y3=Y1-LT*SIN(A3)
BE 470 GOSUB 600:REM KRESLENIE
SJ 480 XS=X
SY 490 YS=Y
EW 500 X1S=X1
FM 510 Y1S=Y1
LT 520 NEXT X
OA 530 END
TP 600 REM **KRESLENIE STRZALY**
TG 610 COLOR 1:TRAP 650
AC 620 PLOT X,Y:DRAWTO X1,Y1
ZE 630 DRAWTO X2,Y2

TQ 640 PLOT X1,Y1:DRAWTO X3,Y3
ZM 650 RETURN
UN 700 REM **KASOWANIE STRZALY**
TO 710 COLOR 0:TRAP 750
IR 720 PLOT XS,YS:DRAWTO X1S,Y1S
ZF 730 DRAWTO X2,Y2
BF 740 PLOT X1S,Y1S:DRAWTO X3,Y3
ZN 750 RETURN
```

za pomocą metod wykorzystanych w programie nr 28. Przemieszczamy jeden koniec odcinka po zadanej krzywej, obliczamy współczynnik nachylenia stycznej w każdym punkcie, znajdujemy kąt nachylenia, a następnie obliczamy współrzędne drugiego końca odcinka. Współczynnik nachylenia stycznej do krzywej w dowolnym punkcie określa się z równania krzywej. Np. dla wielomianu trzeciego stopnia, którego równanie ma postać:

Y = C1 · X^3 + C2 · X^2 + C3 · X + C4

współczynnik nachylenia jest równy:

M = 3 · C1 · X1^2 + 2 · C2 · X + C3

Dla sinusoidy opisanej równaniem:

Y = H · SIN(W · X · D)

współczynnik nachylenia stycznej w punkcie X jest równy:

M = H · W · COS(W · X · D)

Ruch stycznej do okręgu lub elipsy może być wykonany za pomocą operacji obrotu.

Obrót odcinka względem dowolnego punktu można wykonać powtarzając wielokrotnie operację obrotu. **Program nr 29** ilustruje obrót odcinka względem jego środka. Ten typ ruchu można wykorzystać dla zobrazowania szprych obracającego się koła. Równanie obrotu, wykorzystane w tym programie, zostało przekształcone dla skrócenia ilości obliczeń.

```
QO 1 REM ****************
IB 2 REM *              *
UR 3 REM * Program nr 29 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
DV 50 XM=319 YM=191
YY 60 XC=INT(XM/2)
ZO 70 YC=INT(YM/2)
QU 80 X1=XC-40
QI 90 X2=XC+40
HM 100 Y1=YC
HU 110 Y2=YC
ST 120 A=10
MK 130 DEG
ZZ 140 CA=COS(A)
EB 150 SA=SIN(A)
DK 160 XE=XC-XC*CA-YC*SA
CN 170 YE=YC-YC*CA+XC*SA
EL 210 GRAPHICS 8+16:SETCOLOR 2,0
   ,0
QR 220 REM
IN 230 X1R=XE+X1*CA+Y1*SA
KO 240 Y1R=YE+Y1*CA-X1*SA
KM 250 X2R=XE+X2*CA+Y2*SA
ME 260 Y2R=YE+Y2*CA-X2*SA
KE 270 REM KASOWANIE LINII
XP 280 COLOR 0
RX 290 PLOT X1,Y1 DRAWTO X2,Y2
IU 300 REM KRESLENIE LINII
XN 310 COLOR 1
GB 320 PLOT X1R,Y1R:DRAWTO X2R,Y2
   R
GF 330 X1=X1R
HE 340 Y1=Y1R
HI 350 X2=X2R
HX 360 Y2=Y2R
NN 370 GOTO 230
```

Wielokąt opisuje się współrzędnymi wierzchołków, które mogą być zapamiętane w takiej kolejności, w jakiej powinny być połączone. Ruch wielokąta oznacza ruch wszystkich jego wierzchołków z powtórnym wyświetlaniem boków wielokąta. **Program nr 30** demostruje przemieszczenie rysunku samochodu z lewej strony na prawą. Program ten zapewnia przemieszczenie każdego innego obiektu, opisanego w deklaracji DATA. Jednak im więcej odcinków w rysunku, tym wolniej wykonuje się ruch. Można uogólnić program, dla zapewnienia ruchu w dowolnym kierunku, wskazując przyrosty dla obu współrzędnych, tak jak w poprzednich programach.

```
QO 1 REM ****************
IB 2 REM *              *
OE 3 REM * Program nr 30 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
VI 10 REM ******************
WR 30 DIM X(20),Y(20),XT(20)
DU 40 XM=319:YM=191
BM 50 READ N
UB 60 FOR K=1 TO N
DZ 70 READ X,Y:X(K)=X:Y(K)=Y
AH 80 XT(K)=X(K)
JP 90 NEXT K
VO 110 H=15
EM 120 GRAPHICS 8+16:SETCOLOR 2,0
   ,0
TX 130 GOSUB 260
KX 140 REM PRZESUNIECIE
DZ 150 FOR K=1 TO N
WS 160 XT(K)=X(K)+H
GZ 170 NEXT K
SW 180 GOSUB 320
UJ 190 GOSUB 260
BH 200 IF XT(14)+H>XM THEN 430
DU 220 FOR K=1 TO N
JV 230 X(K)=XT(K)
GU 240 NEXT K
NJ 250 GOTO 140
NY 260 REM PROCEDURA KRESLENIA
AH 270 COLOR 1:PLOT XT(1),Y(1)
ER 280 FOR K=2 TO N
DD 290 DRAWTO XT(K),Y(K)
GN 300 NEXT K
ZB 310 RETURN
OO 320 REM PROCEDURA KASOWANIA
BC 330 COLOR 0:PLOT X(1),Y(1)
EK 340 FOR K=2 TO N
VT 350 DRAWTO X(K),Y(K)
GZ 360 NEXT K
ZN 370 RETURN
KK 380 REM DANE
LO 390 DATA 19
VB 400 DATA 10,90,20,90,20,93,22,
   97,30,97,32,93,32,90
VQ 410 DATA 78,90,78,93,82,97,88,
   97,90,93,90,90,100,90
CH 420 DATA 100,70,80,70,75,55,10
   ,55,10,90
TC 430 RESTORE :GOTO 50
```

Skalowanie wielokątów symuluje przybliżanie lub oddalanie obiektów. Ruch żaglówki na tle zachodu słońca realizuje **program nr 31**. Obraz łódki ciągle jest skalowany względem punktu linii horyzontu, symulując oddalanie się łódki.

Nieco inny sposób „ożywienia" rysunku przedstawia **program nr 32** ukazujący sylwetkę biegnącego człowieka. Każda faza biegu narysowana była na papierze milimetrowym, a następnie przeniesiona na współrzędne ekranu w celu przedstawienia położenia rąk i nóg w czasie biegu. Opisy dwóch faz ruchu przechowywane są w tablicy. Pozwala to łatwo przemieszczać każdy fragment

```
QO 1 REM ****************
IB 2 REM *              *
OZ 3 REM * Program nr 31 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
DI 60 DIM X(10),Y(10),XN(10),YN(1
   0)
OC 70 GRAPHICS 8+16:SETCOLOR 2,0,
   0:COLOR 1
OJ 80 REM TLO
SC 100 PLOT 0,60:DRAWTO 319,60
QL 110 REM SLONCE
XY 120 R=25
XK 130 XS=40
BC 140 YS=59
TV 150 DA=1/R
YX 160 C1=180*3.14159/180
ZD 170 C2=360*3.14159/180
UH 180 FOR A=C1 TO C2 STEP DA
FH 190 PLOT XS+R*COS(A),YS+R*SIN(
   A)
CQ 200 NEXT A
EG 210 REM ODCZYT DANYCH RYSUNKU
QW 220 FOR K=1 TO 9
OF 230 READ X,Y:X(K)=X:Y(K)=Y
ED 250 XN(K)=X
EV 260 YN(K)=Y
HA 270 NEXT K
TH 280 GOSUB 510
RF 290 REM
LP 300 XF=-30
VL 310 YF=70
OI 320 HS=0.9
RC 330 VS=0.9
QW 340 REM
VE 350 XE=XF*(1-HS)
YT 360 YE=YF*(1-HS)
WE 370 FOR T=1 TO 12
RJ 380 FOR K=1 TO 9
GD 390 XN(K)=X(K)*HS+XE+H
BH 400 YN(K)=Y(K)*VS+YE+V
GQ 410 NEXT K
VR 420 GOSUB 570
SZ 430 GOSUB 510
RC 440 FOR K=1 TO 9
HN 450 X(K)=XN(K)
IE 460 Y(K)=YN(K)
HC 470 NEXT K
KQ 480 NEXT T
JG 490 GOSUB 570:RESTORE :GOTO 22
   0
NP 500 REM PROCEDURA KRESLENIA
XO 510 COLOR 1:PLOT XN(1),YN(1)
QJ 520 FOR K=1 TO 8
IQ 530 DRAWTO XN(K+1),YN(K+1)
GX 540 NEXT K
ZL 550 RETURN
OY 560 REM PROCEDURA KASOWANIA
BM 570 COLOR 0:PLOT X(1),Y(1)
QV 580 FOR K=1 TO 8
DJ 590 DRAWTO X(K+1),Y(K+1)
GQ 600 NEXT K
ZE 610 RETURN
AJ 620 REM *******************
NE 630 DATA 250,140,290,140,250,8
   0,250,156,270,157
FP 640 DATA 250,182,215,162,210,1
   40,250,156
```

wzdłuż ekranu, wyświetlając je naprzemian, bez przeliczania względnego położenia części rysunku.

Podczas zwiększania ilości obliczeń i liczby wykreślanych linii wzrasta czas niezbędny dla utworzenia oddzielnego „kadru" ruchomego rysunku. Dla obiektów złożonych czas tworzenia obrazu może okazać się tak długi, że zanika wrażenie ruchu. Ruch przyśpieszyć możemy, jeśli będziemy „ożywiać" tylko proste figury, zawierające małą liczbę wykreślanych linii lub jeśli ograniczymy się do prostych typów ruchu. Inną efektywną metodą jest organizowanie ruchu tła dla stworzenia iluzji ruchu obiektu.

```
QO 1 REM ****************
IB 2 REM *              *
PU 3 REM * Program nr 32 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
SE 80 DIM X1(15),Y1(15),X2(15),Y2
   (15)
WO 90 XM=319
TC 100 FOR K=1 TO 13
WO 110 READ X,Y:X1(K)=X:Y1(K)=Y
GP 120 NEXT K
SR 130 FOR K=1 TO 12
YG 140 READ X,Y:X2(K)=X:Y2(K)=Y
GV 150 NEXT K
EW 160 XO=0
EW 170 GRAPHICS 8+16:SETCOLOR 2,0
   ,0
JF 180 IF XO+X1(12)>XM THEN 800
UV 190 GOSUB 270
VN 200 GOSUB 390
BX 210 XO=XO+15
IR 220 IF XO+X2(11)>XM THEN 800
SX 230 GOSUB 510
UI 240 GOSUB 630
AI 250 XO=XO+20
PD 260 GOTO 180
WD 270 REM KRESLENIE POZYCJI 1
YA 280 COLOR 1
RE 285 PLOT XO+X1(1),Y1(1)
NS 290 FOR K=1 TO 3
LS 300 DRAWTO XO+X1(K+1),Y1(K+1)
GP 310 NEXT K
DH 320 PLOT XO+X1(5),Y1(5):DRAWTO
   XO+X1(6),Y1(6)
OU 330 DRAWTO XO+X1(7),Y1(7)
LK 340 PLOT XO+X1(8),Y1(8):FOR K=
   9 TO 12
VG 350 DRAWTO XO+X1(K),Y1(K)
GZ 360 NEXT K
GS 370 X=XO+X1(13):Y=Y1(13):GOSUB
   1000
ZP 380 RETURN
XF 390 REM KASOWANIE POZYCJI 1
XB 400 COLOR 0
RN 410 GOTO 285
WY 510 REM KRESLENIE POZYCJI 2
XR 520 COLOR 1
SD 525 PLOT XO+X2(1),Y2(1)
NU 530 FOR K=2 TO 3
WS 540 DRAWTO XO+X2(K),Y2(K)
GZ 550 NEXT K
DJ 560 PLOT XO+X2(4),Y2(4):DRAWTO
   XO+X2(5),Y2(5)
PA 570 DRAWTO XO+X2(6),Y2(6)
JA 580 PLOT XO+X2(7),Y2(7):FOR K=
   8 TO 11
XC 590 DRAWTO XO+X2(K),Y2(K)
GQ 600 NEXT K
GD 610 X=XO+X2(12):Y=Y2(12):GOSUB
   1000
ZG 620 RETURN
SB 630 REM ***KASOWANIE POZYCJI 2
   ***
XL 640 COLOR 0
QM 650 GOTO 525
YW 750 REM ***DANE RYSUNKU***
MG 760 DATA 14,150,20,135,15,120,
   20,93,5,145,25,133,15,120
FE 770 DATA 20,115,10,110,19,92,2
   0,108,30,113,20,83
ZS 780 DATA 2,132,25,136,40,93,43
   ,150,50,130,30,120
ZR 790 DATA 30,111,22,107,38,95,4
   3,110,58,104,40,83
NX 800 END
KF 999 REM ***KRESLENIE GLOWNY***
VL 1000 PLOT X-5,Y+5:DRAWTO X-5,Y
   -5
EB 1010 DRAWTO X+5,Y-5
CU 1020 DRAWTO X+5,Y+5
DZ 1030 DRAWTO X-5,Y+5
AL 1040 RETURN
```

Program nr 33 demonstruje połączenie ruchu tła i prostego elementu rysunku w celu symulacji ruchu znacznie bardziej złożonego obiektu.

Przedstawione programy demonstrujące sposoby „ożywiania" rysunków wykazują, że język BASIC niezbyt nadaje się do takich zadań. Nieco lepsze efekty można uzyskać programując w języku asemblera. Bardziej interesujące rezultaty można otrzymać wykorzystując znacznie szybszy komputer. Zarówno w jednym, jak i w drugim wypadku zasady tworzenia takich efektów pozostają niezmienne.

```
QO 1 REM ****************
IB 2 REM *              *
QP 3 REM * Program nr 33 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
OD 80 GRAPHICS 8+16:SETCOLOR 2,0,
   0:COLOR 1
FY 90 REM KRESLENIE POCIAGU
JA 100 READ X1,Y1
VC 110 FOR K=1 TO 25
KD 120 READ X2,Y2
RK 130 PLOT X1,Y1:DRAWTO X2,Y2
BK 140 X1=X2
BZ 150 Y1=Y2
GX 160 NEXT K
WN 170 REM OKNO
HR 180 READ XL,XR,YT,YB
FK 190 FOR X=XL TO XR
PK 200 PLOT X,YT:DRAWTO X,YB
LO 210 NEXT X
DS 220 REM DODATKI
PC 230 FOR K=1 TO 6
VO 240 READ X1,Y1,X2,Y2
RP 250 PLOT X1,Y1:DRAWTO X2,Y2
GY 260 NEXT K
PK 270 REM KOLA
OG 280 FOR K=1 TO 4
YX 290 READ XC,YC,R
QK 300 GOSUB 1000
GP 310 NEXT K
QS 320 REM
LE 330 REM RUCH POCIAGU
QL 340 Y=152
XW 350 R=15
MS 360 DEG
CJ 370 FOR A=50 TO 360 STEP 50
NP 380 X1=XC+R*SIN(A)
MH 390 Y1=YC+R*COS(A)
XM 400 COLOR 1
EN 410 PLOT X1,Y1:DRAWTO X1-90,Y1
VG 420 FOR X=XS TO 319 STEP 35
SH 430 PLOT X,Y:DRAWTO X,Y-3:DRAW
   TO X+1,Y
LW 440 NEXT X
VM 450 FOR X=XS TO 319 STEP 35
JO 460 COLOR 0:PLOT X,Y:DRAWTO X,
   Y-3:DRAWTO X+1,Y
MC 470 NEXT X
IX 480 XS=XS+7
TZ 490 IF XS>=30 THEN XS=0
XC 500 COLOR 0
EO 510 PLOT X1,Y1:DRAWTO X1-90,Y1
CX 520 NEXT A
PJ 530 GOTO 370
LH 540 REM DANE DO WCZYTANIA
JS 550 REM POCIAG
NU 560 DATA 270,130,290,130,290,5
   0,300,50,300,30,220,30,220,70,
   200,70
SX 570 DATA 200,50,170,50,170,70,
   90,70,30,50,100,40,60,40,70,50
   ,70,70
FX 580 DATA 60,70,50,80,50,110,60
   ,120,40,120,20,150,50,150,50,1
   30,90,130
WV 590 REM OKNO
JP 600 REM
WO 610 DATA 230,280,40,70
CW 620 REM DODATKI
IU 630 DATA 220,70,220,100,220,10
   0,65,100
RZ 640 DATA 65,100,45,125,130,130
   ,140,130
BO 650 DATA 0,154,319,154,180,130
   ,230,130
PM 660 REM KOLA
WZ 670 DATA 65,140,10,110,130,20,
   160,130,20,250,130,20
EH 1000 RAD
QR 1010 FOR X=0 TO 3.14/4 STEP 1
   R
RN 1020 X1=INT(R*COS(X)+0.5)
JD 1030 Y1=INT(R*SIN(X)+0.5)
UE 1040 PLOT XC+X1,YC+Y1
VH 1050 PLOT XC-X1,YC+Y1
VW 1060 PLOT XC+X1,YC-Y1
WZ 1070 PLOT XC-X1,YC-Y1
UK 1080 PLOT XC+Y1,YC+X1
VH 1090 PLOT XC-Y1,YC+X1
VA 1100 PLOT XC+Y1,YC-X1
WD 1110 PLOT XC-Y1,YC-X1
LG 1120 NEXT X
AH 1130 RETURN
```

```
QO 1 REM ****************
IB 2 REM *              *
RK 3 REM * Program nr 34 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
FH 10 GOSUB 300:GOSUB 200
WH 20 K=PEEK(764):IF K<64 THEN 40
XG 30 FOR I=1536 TO 1575
DC 32 POKE I,PEEK(53770):NEXT I
SB 34 GOTO 20
OW 40 POKE 764,255
BC 50 REM
GC 52 IF K=50 THEN M=2:GOSUB 100:
   REM ..0
KH 54 IF K=31 THEN M=6:GOSUB 100:
   REM ..1
MF 56 IF K=30 THEN M=7:GOSUB 100:
   REM ..2
QM 58 IF K=26 THEN M=8:GOSUB 100:
   REM ..3
RH 60 IF K=24 THEN M=9:GOSUB 100:
   REM ..4
ZZ 62 IF K=29 THEN M=10:GOSUB 100
   :REM .5
BO 64 IF K=27 THEN M=11:GOSUB 100
   :REM .6
DK 66 IF K=51 THEN M=13:GOSUB 100
   :REM .7
HH 68 IF K=53 THEN M=15:GOSUB 100
   :REM .8
GL 70 IF K=63 THEN M=3:GOSUB 100:
   REM ..A
GP 72 IF K=21 THEN M=4:GOSUB 100:
   REM ..B
LG 74 IF K=18 THEN M=5:GOSUB 100:
   REM ..C
XW 76 IF K=58 THEN M=12:GOSUB 100
   :REM .D
YI 78 IF K=42 THEN M=14:GOSUB 100
   :REM .E
FW 80 IF K=62 THEN GOSUB 120:REM
   ......S
EZ 82 IF K=40 THEN GOSUB 140:REM
   ......R
RM 84 IF K=61 THEN GOSUB 150:REM
   ......G
RZ 90 GOTO 20
AZ 100 REM ZMIANA TRYBU
KI 102 POKE DL+3,64+M
TP 104 FOR I=DL+6 TO DL+14
HE 106 FOR W=0 TO 20:NEXT W
MS 108 POKE I,M:NEXT I
YZ 110 RETURN
SE 120 REM ZMIANA OBSZARU EKRANU
IH 122 ? :? :? :? :POKE 656,0
LJ 124 ? "OBSZAR EKRANU ZACZYNA S
   IE TERAZ OD      ";SA
BP 126 ? "WPROWADZ NOWY ADRES POC
   ZATKOWY"
VJ 128 INPUT SA
CV 130 SH=INT(SA/256):SL=SA-256*S
   H
QS 132 POKE DL+4,SL:POKE DL+5,SH
KP 134 GOSUB 250:RETURN
ME 140 REM ODTWORZENIE EKRANU
GH 142 SL=PEEK(88):SH=PEEK(89)
ID 143 POKE 623,0
IA 144 SA=SL+256*SH
RG 146 POKE DL+4,SL:POKE DL+5,SH
UI 147 GOSUB 250
FA 148 M=7:GOSUB 100:RETURN
MO 150 REM TRYBY GTIA
NP 152 I=PEEK(623)+64:POKE 712,48
YN 154 IF I>255 THEN I=I-256
UF 156 IF I>191 THEN POKE 712,0
HQ 158 POKE 623,I:RETURN
XD 200 REM INICJOWANIE
TP 202 GRAPHICS 2
FY 204 DL=PEEK(560)+256*PEEK(561)
FX 206 SA=PEEK(88)+256*PEEK(89)
```

```
PN 210 ? #6; " ...                "
OH 212 ? #6;" x      *****        "
LQ 214 ? #6;"    +++++++++++++    "
SB 216 ? #6;" x      tryby       x"
PB 218 ? #6;" x   ATARI 800 XL   x"
GG 220 ? #6;" x  -----  -------. "
OJ 222 ? #6;" x      *****        "
PH 224 ? #6;" x  ______________  x"
TU 226 ? #6;" x.      1988       "
TX 228 ? #6;" XXXXXXXXXXXXXXXXXX"
NZ 230 FOR I=0 TO 19 POKE 1616+I,
   I
VK 232 POKE 1636+I,I POKE 1656+I,
   I
HI 234 POKE 1676+I,I:POKE 1696+I,
   17*I
VB 236 POKE 1736+I,17*I POKE 1756
   +I,I
GW 238 NEXT I
GO 240 POKE 712,48 POKE 704 12*16
JT 242 POKE 705,6*16+6:POKE 706,1
   5*16+4
WP 244 POKE 707,166
PJ 250 REM GLOWNE MENU
II 252 ? :? :? :? :POKE 656,0:POK
   E 752,1
VF 254 ? " 0  GR.0  A  Antic3  E 
   GR.15"
TK 256 ? " 1  GR.1  B  GR.12   G 
   GTIA"
WL 258 ? " 2  GR.2  C  GR.13   S 
   Obszar ekranu"
LP 260 ? "...  etc.  D  GR.14   R 
   Restore";
ZM 270 RETURN
MD 300 GRAPHICS 0 POSITION 13,1
   "OPIS PROGRAMU" POKE 752,1
KL 310 ? :? "     Program demonst
   ruje 14 trybow" ? "graficznych
    komputerow ATARI serii"
IE 320 ? "XL i XE oraz cztery try
   by GTIA dla":? "kazdego z nich
   ."
DX 330 ? "     Tworzy on obraz w 
   trybie GR.2" "z mozliwym kurs
   orem w oknie tekstowym."
BI 335 ? "Wyswietlany tekst mozna
    obejrzec w ":? "56  mozliwych
     trybach wyswietlania."
WR 340 ? "Klawisze [0] - [8] wybi
   eraja tryby":? "wyswietlania,
   klawisze [A] - [E] wy-"
YJ 350 ? "bieraja tryby specyficz
   ne dla GTIA.":? "     Nacisnie
   cie [S] zmienia stan"
SJ 360 ? "GTIA. [S] zmienia odwzo
   rowany obs-" ? "zar pamieci.
    Proponujemy obejrze-"
BR 370 ? "pamieci od adresow: 153
   6, 53760,":? "53960, 0 i 40520
   . Klawisz [R] odtwa-"
ZS 380 ? "rza stan poczatkowy."
SK 390 ? :? :? "     NACISNIJ DOWO
   LNY KLAWISZ"
AO 400 IF PEEK(764)=255 THEN 400
ZC 410 RETURN
QO 1 REM *****************
IB 2 REM *               *
SF 3 REM * Program nr 35 *
ID 4 REM *               *
QS 5 REM *****************
NL 6 REM
XP 10 REM ANIMACJA WIELOMIANU
OI 40 GOTO 440
XX 50 REM POWTORZENIE
PI 60 GRAPHICS TRYB:POKE 710,0
VP 70 ON WYB GOTO 90,290
XF 80 REM CIAGLA PETLA
GT 90 LICZ=1
UU 100 FOR LICZNIK=1 TO 5 REM ILO
   SC MIEDZY KROKAMI
EJ 110 IF STRIG(0)<>1 THEN 1300
JE 120 PLOT TEMPX(1,LICZ),TEMPY(1
   ,LICZ):REM PUNKT POCZATKOWY DL
   A KAZDEGO WIELOMIANU
JL 130 FOR LL=2 TO 8
UW 140 DRAWTO TEMPX(LL,LICZ),TEMP
   Y(LL,LICZ):REM KONIEC WIELOMIA
   NU
NJ 150 NEXT LL
IZ 160 REM MIEDZY CZESCIAMI
JH 170 FOR LL=1 TO 8
RT 180 TEMPX(LL,LICZ)=TEMPX(LL,LI
   CZ)+STPSX(LL,LICZ):TEMPY(LL,LI
   CZ)=TEMPY(LL,LICZ)+STPSY(LL,LI
   CZ)
NR 190 NEXT LL
QW 200 ? #6;CHR$(125);
KA 210 NEXT LICZNIK
DF 220 LICZ=LICZ+1:IF LICZ<=NUM T
   HEN 100
IN 230 FOR LL=1 TO NUM
TJ 240 FOR LL2=1 TO 8
CP 250 TEMPX(LL2,LL)=X(LL2,LL)
DT 260 TEMPY(LL2,LL)=Y(LL2,LL)
HW 270 NEXT LL2:NEXT LL:GOTO 90
FE 280 REM POJEDYNCZA PETLA
VN 290 FOR LICZ=1 TO NUM-1
ME 300 FOR LICZNIK=1 TO 5
SS 310 PLOT TEMPX(1,LICZ),TEMPY(1
   ,LICZ)
MV 320 FOR LL=2 TO 8:DRAWTO TEMPX
   (LL,LICZ),TEMPY(LL,LICZ):NEXT
   LL
JB 330 FOR LL=1 TO 8
RN 340 TEMPX(LL,LICZ)=TEMPX(LL,LI
   CZ)+STPSX(LL,LICZ):TEMPY(LL,LI
   CZ)=TEMPY(LL,LICZ)+STPSY(LL,LI
   CZ)
NL 350 NEXT LL
RJ 360 ? #6;CHR$(125);
GN 370 NEXT LICZNIK:NEXT LICZ
CB 380 PLOT X(1,NUM),Y(1,NUM)
LX 390 FOR OST=1 TO 8
BF 400 DRAWTO X(OST,NUM),Y(OST,NU
   M)
EB 410 NEXT OST
FR 420 FOR OPOZ=1 TO 500 NEXT OPO
   Z
KW 430 GOTO 1300
VF 440 GOSUB 920
BT 450 DOL=0:X1=INT(XGOR/2+0.5):Y
   1=INT(YGOR/2) C=1
EC 460 WSK=0
RD 470 REM
SC 480 FOR LICZ=1 TO NUM
CR 490 GRAPHICS TRYB:POKE 710,0:C
   OLOR C
KA 500 LOCATE X1,Y1,KOLOR PLOT X1
   ,Y1 XST=X1 YST=Y1
NR 510 IF STRIG(0)<>1 THEN GOSUB
   730
PZ 520 IF WSK THEN 610
VR 530 GOSUB 840
ZZ 540 IF  NOT (TX OR TY) THEN 51
   0
EM 550 COLOR KOLOR:PLOT XST,YST:C
   OLOR C
ST 560 X1=X1+TX:IF X1>XGOR THEN X
   1=XGOR
RL 570 IF X1<DOL THEN X1=DOL
XS 580 Y1=Y1+TY:IF Y1>YGOR THEN Y
   1=YGOR
SR 590 IF Y1<DOL THEN Y1=DOL
MZ 600 GOTO 500
AL 610 X1=INT(XGOR/2+0.5):Y1=INT(
   YGOR/2) C=1
AG 620 WSK=0 LICZNIK=0
GF 630 FOR D=15 TO 0 STEP -0.15:S
   OUND 0,100,10,D NEXT D:SOUND 0
   ,0,0,0
KD 640 NEXT LICZ
PY 650 REM OBLICZENIE KROKOW
VE 660 FOR LICZ=1 TO NUM:FOR LL=1
    TO 8
DC 670 ROB=LICZ+1:IF ROB>NUM THEN
    ROB=1
TE 680 STPSX(LL,LICZ)=(X(LL,ROB)-
   X(LL,LICZ))/5
VF 690 STPSY(LL,LICZ)=(Y(LL,ROB)-
   Y(LL,LICZ))/5
BS 700 NEXT LL:NEXT LICZ
MT 710 GOTO 1160
NR 720 REM WSKAZNIK TABLICY
FB 730 FOR OPOZ=15 TO 0 STEP -0.7
   5 SOUND 0,60,10,OPOZ NEXT OPOZ
    SOUND 0,0,0,0
LL 740 LICZNIK=LICZNIK+1
ZI 750 X(LICZNIK,LICZ)=X1 Y(LICZN
   IK,LICZ)=Y1
AM 760 TEMPX(LICZNIK,LICZ)=X1:TEM
   PY(LICZNIK,LICZ)=Y1
AR 770 PLOT X(1,LICZ),Y(1,LICZ):I
   F LICZNIK=1 THEN 810
LL 780 FOR TMP=2 TO LICZNIK
JE 790 DRAWTO X(TMP,LICZ),Y(TMP,L
   ICZ)
BR 800 NEXT TMP
QV 810 IF LICZNIK+1>8 THEN WSK=1
ZI 820 RETURN
IW 830 REM OBSLUGA MANIPULATORA
ES 840 A=STICK(0)
KJ 850 TX=(A=5 OR A=6 OR A=7)-(A=
   9 OR A=10 OR A=11)
DU 860 TY=(A=5 OR A=9 OR A=13)-(A
   =6 OR A=10 OR A=14)
AC 870 IF PEEK(53279)<>6 THEN 900
MK 880 GRAPHICS TRYB+16:? CHR$(12
   5);"X = ";XST,"Y = ";YST
SF 890 FOR NN=0 TO 1:NN=(PEEK(532
   79)<>6):NEXT NN:GRAPHICS TRYB+
   32
ZF 900 RETURN
MN 910 REM INICJOWANIE I OTWARCIE
    EKRANU
NB 920 DIM X(8,10),Y(8,10),TEMPX(
   8,10),TEMPY(8,10),TM(8,2)
VV 930 DIM STPSX(8,10),STPSY(8,10
   )
YV 940 XL=1 TRAP 1430 GRAPHICS 15
HM 950 GRAPHICS 0:POKE 752,1
OR 960 POSITION 14,0
SO 965 ? "QRRRRRRRRRRE"
WU 970 POSITION 14,1:? "| ANIMACJ
   A |"
QD 980 POSITION 14,2
UA 985 ? "RRRRRRRRRRRC"
DF 990 POSITION 11,4:? " "
CZ 1000 ? :? :? :?
DZ 1010 ? "Kazdy utworzony wielob
   ok powinien":? "miec 8 punktow
   , ktore wybieramy"
EH 1020 ? "manipulatorem."
OB 1030 ? "Mozemy utworzyc do 10 
   wielobokow."
HU 1040 ? :? :? "    Ile wybieras
   z: ";:INPUT NUM
IF 1050 POSITION 2,17:? "Tryb gra
   ficzny? (3/5/7/8/15)";
OG 1070 ? ")";CHR$(158);CHR$(158)
   ;:INPUT TRYB
FR 1080 IF  NOT (TRYB=3 OR TRYB=5
    OR TRYB=7 OR TRYB=8 OR TRYB=1
   5) THEN ? CHR$(156),CHR$(156);
   CHR$(125):GOTO 1050
OJ 1090 IF TRYB=3 THEN XGOR=39:YG
   OR=23
AM 1100 IF TRYB=5 THEN XGOR=79:YG
   OR=47
ZX 1110 IF TRYB=7 THEN XGOR=159:Y
   GOR=95
KX 1120 IF TRYB=8 THEN XGOR=319:Y
   GOR=191
TT 1130 IF TRYB=15 THEN XGOR=159:
   YGOR=191
PN 1140 TRYB=TRYB+16:RETURN
SR 1150 REM PETLA-POJEDYNCZE POWT
   ORZENIE
GV 1160 GRAPHICS 0:POKE 752,1
GI 1170 POSITION 14,3
AW 1175 ? "QRRRRRRRRRE"
QR 1180 POSITION 14,4:? "| WYBIER
   Z |"
HY 1190 POSITION 14,5
CT 1195 ? "RRRRRRRRRRC"
DD 1200 ? :? :? :?
JG 1210 ? "       1) Ciagla petla"
NP 1220 ?
QW 1230 ? "       2) Pojedyncza pet
   la"
YA 1240 ? :? :?
IA 1250 ? "Wprowadz wybrana liczb
   e: ";:INPUT WYB
EC 1260 IF WYB=1 THEN POSITION 1,
   22 ? "DLA PRZERWANIA PETLI NAC
   ISNIJ PRZYCISK"
KZ 1270 FOR OPOZ=1 TO 400:NEXT OP
   OZ
RX 1280 GOTO 60
FP 1290 REM ZNOWU, ITD...
OH 1300 GRAPHICS 0:POKE 752,1
FU 1310 POSITION 14,3
AI 1315 ? "QRRRRRRRRRE"
QD 1320 POSITION 14,4:? "| WYBIER
   Z |"
HK 1330 POSITION 14,5
CF 1335 ? "RRRRRRRRRRC"
DR 1340 ? :? :? :?
BM 1350 ? "1) Wykonaj ponownie, t
   e same wieloboki";
KB 1360 ? "2) Wykonaj ponownie, i
   nne wieloboki."
```

```
FU 1370 ? "3) Koniec."
LY 1380 ? :? :? " Wprowadz wybran
   a liczbe: ";:INPUT WYB
UG 1390 ON WYB GOTO 1410,1470,149
   0
OG 1400 GOTO 1300
DL 1410 FOR LL=1 TO NUM
JE 1420 FOR LL2=1 TO 8
KG 1430 TEMPX(LL2,LL)=X(LL2,LL):T
   EMPY(LL2,LL)=Y(LL2,LL)
OS 1440 NEXT LL2
PV 1450 NEXT LL
RA 1460 GOTO 1160
NP 1470 RUN
RN 1480 POKE 752,0:END
WP 1490 XL=0:TRAP 1490:GOTO 1300
```

```
QO 1 REM ****************
IB 2 REM *              *
TA 3 REM * Program nr 36 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
UO 20 GRAPHICS 5:SETCOLOR 2,0,0
MX 30 COLOR 1:PLOT 0,0:DRAWTO 79,
   0:DRAWTO 79,39:DRAWTO 0,39:DRA
   WTO 0,0
MH 40 POKE 752,1:? CHR$(125);"WYN
   IK ";P;"          CZAS ";S;CHR$(
   30);
MX 50 POKE 559,46
PM 60 POKE 704,188
YM 70 I=PEEK(106)-8
BZ 80 POKE 54279,I
NY 90 POKE 53277,3
SI 100 POKE 53256,1
RW 110 PMB=I*256
IS 120 FOR I=PMB+512 TO PMB+640:P
   OKE I,0:NEXT I
QJ 130 POKE PMB+512+85,60:POKE PM
   B+513+85,60
LJ 140 FOR I=PMB+384 TO PMB+512:P
   OKE I,0:NEXT I
EM 150 POKE PMB+384,3:S=999
UX 160 YV=INT(1*RND(0)+2):XV=INT(
   -4*RND(0)+4)
BB 170 YM=19:XM=120
LG 180 ST=STICK(0):X=X+4*((ST=7)+
   2*(X<195)):X=X-4*((ST=11)+2*(X
   >45))
RP 190 POKE 53248,X
TV 200 POKE 53278,0
GD 210 POKE PMB+384+YM1,0:A=A+1:S
   =S-1
YT 220 IF S<10 THEN ? " ";
SW 230 IF S<100 THEN ? " ";
NF 240 POKE 752,1:? "";S;CHR$(30)
   ;CHR$(30);CHR$(30);
JC 250 XM=XM+XV:YM1=YM:YM=YM+YV:I
   F XM<50 OR XM>205 THEN XV=-XV:
   SOUND 0,20,10,15:SOUND 0,0,0,0
   :GOTO 210
RF 260 IF S=0 THEN GOTO 340
TM 270 IF YM<19 OR YM>92 THEN YV=
   -YV:SOUND 0,20,10,15:SOUND 0,0
   ,0,0:GOTO 210
NA 280 POKE PMB+384+YM,3:POKE 532
   52,XM
ZF 290 DF=0
OW 300 IF PEEK(53256)<>0 AND A>3
   THEN POKE 752,2:P=P+1:? CHR$(1
   25);"WYNIK ";P;"          CZAS "
   :YV=-YV:XV=-XV:DF=1
LA 310 IF DF=1 THEN POKE 53278,0:
   A=0:GOTO 330
OW 320 GOTO 180
ZA 330 XV=INT(-4*RND(0)+2):GOTO 2
   10
JO 340 POKE 53277,0:POKE 53265,0
   POKE 53261,0
ZN 350 GRAPHICS 18
JC 360 POSITION 2,3:? #6;"punkty
   : ";P
UN 370 POSITION 2,9:? #6;"START =
   nowa gra"
JY 380 POSITION 2,11:? #6;"SELECT
   = koniec"
DA 390 IF PEEK(53279)=6 THEN CLR
   :GOTO 20
NV 400 IF PEEK(53279)=5 THEN GRAP
   HICS 0:END
QA 410 GOTO 390
```

```
QO 1 REM ****************
IB 2 REM *              *
TV 3 REM * Program nr 37 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
EP 8 GOTO 10000
HE 10 FOR Y=0 TO R:X=SQR(RS-Y*Y):
   Y1=FT*(R-Y):X2=FT*(R+X):Y1=R
   Y-Y:Y2=RY+Y
ZH 12 IF Y1>YMAX THEN Y1=YMAX
AO 14 IF Y2>YMAX THEN Y2=YMAX
ME 16 X1=X1*(X1>0):IF X1>XMAX THE
   N X1=-1
YH 24 IF X2>XMAX THEN X2=XMAX
HJ 28 IF X1<0 OR X2<0 THEN 35
EO 30 IF Y1>=0 THEN PLOT X1,Y1:DR
   AWTO X2,Y1
HP 32 IF Y2>=0 THEN PLOT X1,Y2:DR
   AWTO X2,Y2
PE 35 NEXT Y:GOTO 100
JT 40 IF AY=BY THEN PLOT AX,AY:DR
   AWTO BX,BY:GOTO 60
TH 50 FOR Y=AY TO BY:PLOT CX-(CY-
   Y)*KCA,Y:DRAWTO BX-(BY-Y)*KBA,
   Y:NEXT Y
SW 60 IF BY=CY THEN PLOT BX,BY:DR
   AWTO CX,CY:GOTO 80
CQ 70 FOR Y=BY TO CY:PLOT CX-(CY-
   Y)*KCA,Y:DRAWTO CX-(CY-Y)*KCB,
   Y:NEXT Y
LX 80 GOTO 100
KF 100 REM GLOWNA PETLA
EE 110 IF PEEK(53279)=6 THEN 500
XD 120 ST=STICK(0)
SQ 130 IF ST=15 THEN IF PEEK(5327
   9)=3 THEN 700
TK 132 IF ST=15 THEN IF PEEK(5327
   9)=1 THEN 900
XK 135 IF ST=15 THEN IF PEEK(5327
   9)=5 THEN 800
SY 140 IF ST=15 THEN IF STRIG(0)=
   0 THEN 300
SU 150 XM(ZNA)=XM(ZNA)+SZYB*X(ST)
   :YNOW=YM(ZNA)+SZYB*Y(ST)
UL 160 IF XM(ZNA)>206 THEN XM(ZNA
   )=206
TS 170 IF XM(ZNA)<48 THEN XM(ZNA)
   =48
UT 180 IF YNOW<16 THEN YNOW=16
XE 190 IF YNOW>111 THEN YNOW=11
PW 200 POKE 53252+ZNA,XM(ZNA)
DG 204 IF YNOW=YM(ZNA) THEN 210
BF 205 POKE PMM+YM(ZNA),PEEK(PMM+
   YM(ZNA))-MK(ZNA):POKE PMM+YNOW
   ,PEEK(PMM+YNOW)+MK(ZNA):YM(ZNA
   )=YNOW
LJ 210 GOTO 100
MH 300 S$(3,10)=H1$(1,8)
JI 320 IF STICK(0)<>15 THEN S$(3,
   10)=H2$(1,8):GOTO 400
OK 1075 DATA 1,1,1,-1,1,0,0,0,-1,
   1,-1,-1,1,0,0,0,0,1,0,-1,0,0
AC 1100 DIM CIO$(6):FOR I=1 TO 6:
   READ A:CIO$(I,I)=CHR$(A):NEXT
   I
CE 1105 DATA 104,162,16,76,86,228
PV 1110 DIM SS$(80),ZBIOR$(15)
EL 1500 DIM DLI$(14):FOR I=1 TO 1
   4:READ A:DLI$(I,I)=CHR$(A):NEX
   T I
UQ 1501 DATA 72,173,111,2,41,3,14
   1,10,212,141,27,208,104,64
UR 1502 DL=PEEK(560)+256*PEEK(561
   )
CP 1514 POKE DL+93,66:POKE DL+96,
   2:POKE DL+97,PEEK(DL+101):POKE
   DL+98,PEEK(DL+102)
PH 1515 POKE DL+99,PEEK(DL+103):P
   OKE DL+94,SL:POKE DL+95,SH:GOT
   O 1520
DE 1520 POKE 513,INT(ADR(DLI$)/25
   6):POKE 512,ADR(DLI$)-256*PEEK
   (513)
CH 1521 POKE 54286,192
NO 1540 GOTO 100
CW 1550 DATA 0,1,2,2,1,0,0,2,1,1,
   2,0,1,0,2,2,0,1
RW 2000 A=PEEK(764):IF A=255 OR A
   =58 OR A=39 THEN 2000
GK 2005 GET #1,A:IF A=126 OR A=15
   5 THEN RETURN
DA 2010 IF A<32 OR A>90 THEN 200
   0
AX 2015 RETURN
RM 10000 GRAPHICS 17:POKE 752,1:V
   =1
KO 10001 DL=PEEK(560)+256*PEEK(56
   1)
SB 10002 POKE DL+3,71:FOR I=6 TO
   11:POKE DL+I,7:NEXT I
PF 10030 POSITION 1,3:? #6;"kszta
   lty i kolory"
UD 10040 POSITION 2,8:? #6;"OPTIO
   N   KOLORY"
MR 10050 POSITION 2,9:? #6;"selec
   t   KSZTALTY"
CA 10055 POSITION 2,10:? #6;"STAR
   T   KRESLENIE"
ZR 10056 POSITION 0,12:? #6;"OPTI
   ON+SELECT PAMIEC"
ZG 10060 POSITION 0,14:? #6;"joys
   tick - poczatek"
JQ 10066 IF STICK(0)<>15 THEN 100
   0
LS 10067 IF STRIG(0)=0 THEN 1000
FF 10068 IF STRIG(0)=0 THEN 10068
EL 10069 GOTO 10065
LO 990 B=USR(ADR(CIO$)):GOTO 999
ZH 997 TRAP 40000:POKE 54286,192:
   S$(42,71)="WYKRYTO BLAD SPROBU
   J JESZCZE  "
FU 998 FOR I=42 TO 71:S$(I,I)=CHR
   $(ASC(S$(I,I))-32):NEXT I:FOR
   I=1 TO 1000:NEXT I
FU 999 TRAP 40000:S$=SS$:CLOSE #2
   :CLOSE #1:GOTO 1520
HW 1000 REM
WD 1001 IF V=1 THEN GRAPHICS 23:P
   M=PEEK(106)-20:NKOLRS=4:XT=1:Y
   T=1:FT=1.25:XMAX=159:YMAX=95
EK 1003 POKE 54279,PM:PMM=PM*256+
   384:POKE 559,38:POKE 53277,1
BC 1004 POKE 623,1:FOR I=PMM TO P
   MM+127:POKE I,0:NEXT I
MM 1005 FOR I=0 TO 2:POKE 704+I,1
   4:NEXT I
AC 1010 DIM XM(2),YM(2),MK(2),KOL
   R$(96),KSZTA$(14)
ZK 1011 FOR I=0 TO 2:XM(I)=125:PO
   KE 53252+I,125:YM(I)=16:NEXT I
   :POKE PMM+16,255
WB 1012 MK(0)=3:MK(1)=12:MK(2)=48
   :REM MASKA POCISKU
JU 1014 KSZTA$="TROJKAT OKRAG   ":
   IF V=1 THEN KOLR$(1,36)=" CZAR
   NY  POMARANCZ ZIELONY NIEBIESK
   I"
JV 1015 FOR I=1 TO 36:KOLR$(I,I)=
   CHR$(ASC(KOLR$(I,I))-32):NEXT
   I
FG 1016 FOR I=1 TO 14:KSZTA$(I,I)
   =CHR$(ASC(KSZTA$(I,I))-32):NEX
   T I
BW 1020 KSZTA=0:KOLR=1:ZNA=0:SZYB
   =1:POKE 704,60
TF 1021 DIM S$(80):S=ADR(S$):SH=I
   NT(S/256):SL=S-SH*256
KY 1022 S$=" ":S$(80)=" ":S$(2)=S
   $
DE 1023 FOR I=1 TO 80:S$(I,I)=CHR
   $(ASC(S$(I,I))-32):NEXT I
FU 1025 DIM H2$(32):H2$="ZNACZN #
    SZYBK  KOLOR   KSZTALT "
WU 1026 DIM H1$(32):H1$="ZNACZN #
    SZYBK  KOLOR   KSZTALT "
BO 1027 FOR I=1 TO 32:H2$(I,I)=CH
   R$(ASC(H2$(I,I))-32):NEXT I
ZK 1028 FOR I=1 TO 32:H1$(I,I)=CH
   R$(ASC(H1$(I,I))-32):NEXT I
HV 1029 S$(3,34)=H2$:S$(46,46)=CH
   R$(ZNA+16):S$(55,55)=CHR$(SZYB
   +16):S$(58,66)=KOLR$(KOLR*9+1,
   KOLR*9+9)
LE 1030 S$(68,74)=KSZTA$(KSZTA*7+
   1,KSZTA*7+7)
SS 1034 DIM X(15),Y(15):FOR I=5 T
   O 15:READ A,B:X(I)=A:Y(I)=B:NE
   XT I
PX 680 GOTO 10
GH 699 TRAP 40000:FOR I=0 TO 30:P
   OKE 53279,0:NEXT I:GOTO 100
TF 700 S$(19,26)=H1$(17,24)
GK 720 IF PEEK(53279)=3 THEN 720
TI 730 KOLR=KOLR+1:IF KOLR=NKOLRS
   THEN KOLR=0
```

```
NO 740 S$(19,26)=H2$(1,24) [illegible]
LW 750 GOTO 100
QQ 800 S$(27,34)=H1$(25,32)
JA 820 IF PEEK(53279) [illegible]
ZZ 830 KSZTA=1 [illegible]
CK 840 S$(27,34)=H2$(25,32):S$(68,74)=[illegible]
 7)
LX 850 GOTO 100
KI 900 SS$=S$:POKE [illegible] 255 [illegible]
 #2 OPEN #2,4,0,[illegible] POKE [illegible]
 4 POKE 831,0
JF 910 S$="  NACISNIJ L DLA WPROWADZENIA EKRANU  [illegible] DLA PRZEGLADANIA EKRANU"
LI 920 FOR I=1 TO [illegible] S$(I,I)=[illegible]
 (ASC(S$(I,I)) [illegible]
FJ 925 GOSUB 2000 [illegible] THEN 99
NF 926 IF A=ASC("L") THEN [illegible]
 GOTO 940
CK 927 IF [illegible] ("S") THEN [illegible]
 1:GOTO 940
UC 930 FOR I=1 TO 20 POKE [illegible]
 :NEXT I:GOTO 925
WP 940 S$="NAZWA ZBIORU?

 ZBIOR$=S$(14,2?)
LV 943 FOR I=1 TO 80:S$(I,I)=CHR$
 (ASC(S$(I,I))-32):NEXT I
FU 950 I=0
FH 955 GOSUB 2000:IF A=155 AND I
 0 THEN 999
KJ 960 IF A=155 THEN 980
XT 965 IF A=126 AND I=0 THEN [illegible]
MR 968 IF A [illegible] THEN A=ASC([illegible]
 GOSUB 978 I=I-1 GOTO 955
VU 970 IF I=15 THEN 955
ZM 975 I=I+1:GOSUB 978:GOTO 955
FW 978 ZBIOR$(I,I)=[illegible]
 I,14+I)=CHR$(A-32):RETURN
MO 980 TRAP 997:POKE 54286,64
GI 981 IF ZBIOR$(1,1)=[illegible]
 [illegible] 53775,35 POKE 53[illegible] 40 [illegible]
 53764,0:POKE 53765,0 POKE [illegible]
 73,255
YI 982 OPEN #1,8,0,ZBIOR$
NQ 985 POKE 852,PEEK(88) POKE [illegible]
 ,PEEK(89) POKE 856,0:POKE [illegible]
 15[illegible] POKE 850,0
VN 330 IF STRIG(0)=0 THEN 320
IG 345 POKE 703:2NA 14
FP 350 ZNA=ZNA+1 IF [illegible]
 A=0
QP 355 S$(3,10)=[illegible]
 )=CHR$(ZNA+16) POKE 70[illegible]
LU 360 GOTO 100
AB 400 S$(11,18)=H1$(9,16)
JL 405 FOR W=1 TO 50:NEXT W
GU 410 IF STICK(0)=15 THEN S$(11,
 18)=H2$(9,16) GOTO 100
JY 420 IF STICK(0)=14 THEN SZYB=S
 ZYB+1:IF SZYB>9 THEN SZYB=9
DE 430 IF STICK(0)=13 THEN SZYB=S
 ZYB-1:IF SZYB<1 THEN SZYB=1
QE 435 S$(55,55)=CHR$(SZYB+16)
PD 440 GOTO 405
HT 500 COLOR KOLR:RESTORE 1[illegible]
NT 505 IF KSZTA=1 THEN 600
DF 507 TRAP 699:READ A,B,C
SG 510 IF YM(A)<=YM(B) AND YM(B)<
 =YM(C) THEN AY=YM(A) [illegible]
 BY=YM(B) BX=XM(B) [illegible]
 XM(C):GOTO 520
QJ 511 GOTO 507
HH 520 AX=(AX-48)*XT:BX=(BX-48)*X
 T:CX=(CX-48)*XT:AY=(AY-16)*YT:
 BY=(BY-16)*YT:CY=(CY-16)*YT
TL 530 TRAP 532:KCA=(CX-AX)/(CY-A
 Y)
SG 532 TRAP 534:KBA=(BX-AX)/(BY-A
 Y)
YJ 534 TRAP 536:[illegible]
 Y)
JS 536 TRAP 4[illegible]
QS 540 GOTO 40
CL 600 TRAP 699:READ A,B,C
NT 601 AX=(XM(A)-48)*XT/FT:AY=(YM
 (A)-16)*YT
OO 610 BX=(XM(B)-48)*[illegible]
 (B)-16)*YT

[illegible]
 2
XH 642 LBA=(BY+AY)/2^2-KBA*(BX+AX)/
 2
UM 650 TRAP 699:RY=(KBA*LCA-KCA*L
 [illegible]
CK 670 RS=(RY-AY)*(RY-AY)+(RX-AX)
 [illegible]

 1 REM ****************
 2 REM *              *
 3 REM * Program nr 38 *
 4 REM *              *
 5 REM ****************
 6 REM
 [illegible]
CB 190 MBL=PEEK(88):MBH=PEEK(89)
 [illegible]
 EEK(708+J)/16) HELL(J)=PEEK(70
 [illegible]
ZY 220 REM PODPROGRAM W ASSEMBLER
 [illegible]
UX 240 RESTORE 260
 [illegible]
 94
UR 270 DATA 157,66,3,104,157,69,3
 [illegible]
JB 310 REM  JOYSTICK + GRAFIKA
 [illegible]
FF 330 ST=1
FM 340 X=X+INT(ST*(((K=5 OR K=6 O
 [illegible]
DL 350 Y=Y+INT(ST*(((K=5 OR K=9 O
 R K=13) AND (Y<191))-((K=6 OR
 [illegible]
 THEN GET #1,M:M=M-47:ON M GOS
 [illegible]
AU 400 COLOR ABS(Z-1):PLOT X,Y IF
 [illegible]
LP 420 IF M=108 THEN GOSUB 1020
MU 430 GOTO 310
JY 440 REM Wlaczenie kreslenia
LL 450 COL=0:RETURN
 [illegible]
 XM,YM:XZ=X:YZ=Y:Z=COL RETURN
 [illegible]
 L

 [illegible]
RS 570 TRAP 580 XK=SIN(I)*R+XM YK
 [illegible]
 )*(X/(S-XM))):XMQ=0 GOTO 630
 [illegible]
FM 660 ? #6,CHR$(125);:Z=M RETURN
 [illegible]
GJ 710 K=STICK(0)
GO 720 FARB(F)=FARB(F)+((K=5 OR K
 [illegible]
HH 730 HELL(F)=HELL(F)+((K=5 OR K
 [illegible]
DS 750 FOR T=1 TO 20:NEXT T
 [illegible]
OT 770 GOTO 710
 [illegible]
RT 990 POKE 559,34
ZZ 1000 RETURN
PI 1010 REM Grafika i menu
 [illegible]
 TO 1040
 [illegible]
 :GR=0 RETURN
 [illegible]
RU 1060 REM Informacja - wydruk w
 [illegible]
OQ 1080 ? CHR$(28);
LO 1090 ? "X=";X;"  ";CHR$(127);
LL 1100 ? "Y=";Y;"  ";CHR$(127);
UC 1110 ? "COL=";COL
 [illegible]
JL 1130 POSITION X,Y:GOTO 420
 [illegible]
```

```
[illegible]
[illegible] FARB [illegible]
T J:IF INF=1 THEN GOSUB 1200:P
[illegible]
JY 1250 GR=8:RETURN
[illegible]
[illegible] CHR$(127) [illegible]
CHR$(127);CHR$(127);" MENU ":?
AP 1280 ? CHR$(127);"0 = KASOWANI
AO 1290 ? CHR$(127);"1 = KRESLENI
DL 1300 ? CHR$(127);"2 = KRESLENI
E LINII"
KZ 1310 ? CHR$(127);"3 = KRESLENI
E ROWNOLEGLE"
QR 1320 ? CHR$(127);"4 = WYPELNIA
NIE W PRAWO"
CG 1330 ? CHR$(127);"5 = KRESLENI
E OKREGU "
YK 1340 ? CHR$(127);"6 = KASOWANI
E EKRANU"
EJ 1350 ? CHR$(127);"7 = WYBRANIE
 KOLORU"
SX 1360 ? CHR$(127);"8 = INFORMAC
JA O POLOZENIU"
OV 1370 ? CHR$(127);"9 = MENU MAG
[illegible]
PF 1380 ? :? " RETURN  = MENU <->
 GRAFIKA - ZMIANA"
[illegible] NEXT [illegible]
[illegible]
[illegible]
[illegible]
YC 1420 COLOR Z:PLOT X1,Y1
TY 1430 GOSUB 2460:POKE 87,0
[illegible]
[illegible]
EL 1460 GRAPHICS 0:SETCOLOR 1,0,1
4:SETCOLOR 2,3,5:SETCOLOR 4,0,
[illegible]
[illegible] CHR$(29) [illegible]
[illegible] PAMIEC ZEWNETRZNA"
[illegible] RETURN
[illegible]
CD 1490 ? :? CHR$(127);"2 = WPROW
[illegible]
PO 1500 ? :? CHR$(127);"3 = PRZEC
[illegible]
KI 1510 ? :? CHR$(127);"4 = ZABEZ
[illegible]
DK 1520 ? :? CHR$(127);"5 = ODBEZ
PIECZENIE DANYCH"
EC 1530 ? :? CHR$(127);"6 = PRZEM
[illegible]
JE 1540 ? :? CHR$(127);"7 = KASOW
[illegible]
YN 1550 ? :? CHR$(127);"8 = MENU-
[illegible]
OH 1560 ?
[illegible] #1,MEN
JE 1580 REM
[illegible]
[illegible]
WE 1610 ON MEN GOTO 1630,1690,183
[illegible] 2400
[illegible]
[illegible]
[illegible] POSITION 2,20 ?
[illegible] CHR$(30);CHR$(30) CH
[illegible] 2290 ? CHR$(125);
[illegible]
[illegible] #2,6,0,A$
[illegible] #2,A$ ? A$:GOTO 166
0
DR 1670 CLOSE #2:? :? "  NACISNIJ
 DOWOLNY KLAWISZ ":GET #1,A:GO
TO 1470
ZU [illegible] REM Wprowadzanie obrazu
[illegible] POSITION 5,5 ? CHR$(27);C
HR$(31)
[illegible] GOSUB 2280
OH 1710 TRAP 2410
[illegible] THEN 1610
[illegible]
EW 1740 OPEN #2,4,0,A$
UO 1750 Q=USR(ADR(CIO$),32,7,SRAM
,7680)
[illegible] TO 4
QE 1770 GET #2,FARB:GET #2,HELL
ZZ 1780 FARB(J)=FARB:HELL(J)=HELL
[illegible] NEXT J
NI [illegible] #2:IF PEEK(867)<>1
THEN POKE 195,PEEK(867):GOTO 2
410
SQ 1810 GOTO 1470
TY 1820 REM Przechowanie obrazu
[illegible] 1830 POSITION 5,7 ? CHR$(27);C
HR$(31)
OF 1840 GOSUB 2280
OY 1850 TRAP 2410
TS 1860 IF A$="" THEN 1810
MZ 1870 SRAM=M8L+M8H*256
AO 1880 OPEN #2,8,0,A$
LH 1890 Q=USR(ADR(CIO$),32,11,SRA
M,7680)
YA 1900 FOR J=0 TO 4
PW 1910 PUT #2,FARB(J):PUT #2,HEL
L(J)
FY 1920 NEXT J
NO 1930 CLOSE #2:IF PEEK(867)<>1
THEN POKE 195,PEEK(867):GOTO 2
410
TB 1940 GOTO 1470
JW 1950 REM Zabezpieczenie danych
LS 1960 POSITION 5,9:? CHR$(27);C
HR$(31)
DW 1970 GOSUB 2280
PG 1980 TRAP 2410
ON 1990 IF A$="" OR A$(4)="" THEN
 1470
SZ 2000 XIO 35,#2,0,0,A$
SB 2010 GOTO 1470
KR 2020 REM Odbezpieczenie danych
[illegible] 2030 POSITION 5,11 ? CHR$(27);
CHR$(31)
[illegible] 2040 GOSUB 2280
OG 2050 TRAP 2410
[illegible] 2060 IF A$="" OR A$(4)="" THEN
 1470
UF 2070 XIO 36,#2,0,0,A$
SW 2080 GOTO 1470
VV 2090 REM Przemianowanie danych
GV 2100 POSITION 5,13:? CHR$(27);
CHR$(31)
CJ 2110 GOSUB 2280
NZ 2120 TRAP 2410
NG 2130 IF A$="" OR A$(4)="" THEN
 1470
SG 2140 XIO 32,#2,0,0,A$
[illegible] 2150 GOTO 1470
[illegible] 2160 REM Kasowanie danych
[illegible] 2170 POSITION 5,15 ? CHR$(27);
CHR$(31)
DE 2180 GOSUB 2280
OU 2190 TRAP 2410
MZ 2200 IF A$="" OR A$(4)="" THEN
 1470
[illegible] 2210 OPEN #2,[illegible] A$ CLOSE #2 ?
 CHR$(28);CHR$(156);CHR$(28);
XC 2220 ? "NACISNIJ -Y- W CELU WY
KASOWANIA..":? " --->";A$:?
KH 2230 ? "UWAGA  WSZYSTKIE DANE
[illegible] WYKAZU BEDA WYK
ASOWANE.";CHR$(30);
GH 2240 GET #1,A IF A<>89 THEN 14
70
[illegible] XIO 33,#2,0,0,A$
[illegible] GOTO 1470
PD 2270 REM Wprowadzanie nazwy zb
ioru
[illegible] 2280 POSITION 2,20 ? "D:";
[illegible] 2290 GET #1,A
MK 2300 IF A=156 OR A=157 OR A=12
5 OR A=28 OR A=29 THEN 2290
NP 2310 ? CHR$(A);
UF 2320 IF A<>155 THEN 2290
[illegible] 2330 POSITION 2,20
[illegible] 2340 INPUT #3,A$
[illegible] 2350 RETURN
[illegible] REM
NM 2370 GOSUB 1270:POKE 106,160
RS 2380 GOSUB 1240
PY 2390 POP :LOCATE 160,96,Z:GOTO
 300
VI 2400 REM Sygnalizacja bledu
WH 2410 CLOSE #2:POSITION 2,21:?
[illegible] CHR$(157);
KQ 2420 ? " ERROR - ";PEEK(195)
AK 2430 ? :? " NACISNIJ DOWOLNY KL
AWISZ"
IW 2440 GET #1,A:GOTO 1470
[illegible] REM
[illegible] POKE 88,M8L:POKE 89,M8H:P
[illegible]
BD 2470 RETURN
[illegible] 2480 POKE 88,M8L:POKE 89,M8H:P
[illegible]
BI 2490 RETURN
FT 3000 POKE 764,255
[illegible] 3010 K=PEEK(764):POKE 764,255
[illegible] 3015 [illegible]
ZC 3020 IF K=255 THEN K=15:Q=1:RE
TURN
WI 3030 IF K=13 THEN K=15:Q=0:RET
URN
AP 3040 IF K=14 THEN RETURN
PG 3050 IF K=15 THEN K=13:RETURN
BN 3060 IF K=6 THEN K=1:RETURN
YX 3070 IF K=7 THEN RETURN
AZ 3080 RETURN
PT 4000 ? PEEK(764):GOTO 4000

QU 1 REM ****************
IB 2 REM *              *
VL 3 REM * Program nr 39 *
ID 4 REM *              *
OS 5 REM ****************
NL 6 REM
UF 10 REM KRESLENIE W TRYBIE 10
XD 40 DIM C$(12),CIO$(7),ZBIOR$(2
0),A$(17),C1$(30),S(35):H=6:U1
=1:U1E=16
JK 45 FOR I=1 TO 27:READ A:C1$(I,
I)=CHR$(A):NEXT I
GZ 47 DATA 104,162,0,172,193,2,18
3,194,2,157,193,2,232,224,8,14
4,245,140,200,2,96,65,65,65,65
,65,65
FH 50 POKE 16,112:POKE 53774,112:
IF DA THEN RETURN
PB 52 POKE 764,255
MR 55 GOSUB 1000
YJ 56 IF PEEK(764)=255 THEN 56
PN 58 POKE 764,255
HL 60 GRAPHICS 17:DA=1:H=6:POKE 7
10,134:POKE 708,198:POKE 709,8
8:QQ=764:QQ1=255:RESTORE 370:F
OR I=1 TO 35:S(I)=U:T=U
PZ 65 FOR I=1 TO 35:S(I)=U:NEXT I
:T=U
NF 66 GOSUB 1100
IZ 100 FOR I=1 TO H:READ A:S(A)=U
1:NEXT I
MM 110 FOR I=U1 TO H:READ A:S(A)=
-U1:NEXT I:AL=U:SG=U1
OH 120 GRAPHICS 26:RESTORE 390:FO
R I=705 TO 712:READ A:POKE I,A
:NEXT I:X=38:Y=100
HO 125 C=U1:Z=U:COLOR C:I=U:F=U:E
=U
YB 130 POKE QQ,QQ1:GOSUB 50:IF T
THEN RETURN
ND 140 POKE 77,U:S=PEEK(632):SG=P
EEK(644):IF AL AND S=15 THEN A
=AL:SG=R
BY 150 IF AL AND S<>15 THEN AL=S:
R=SG
SA 160 A=S(S):H=S(S+20):X=X+A*((X
<79 AND Q=U1) OR (X>U AND Q=-U
1)):Y=Y+H*((Y<184 AND H=U1) OR
 (Y>U AND H=-U1))
GZ 170 IF E THEN LOCATE X,Y,C:COL
OR U1:PLOT X,Y
KN 180 IF SG=U THEN C=U1+C*(C<9):
E=U:COLOR C:IF I=U1 THEN I=U:C
=B
UY 190 IF I=U1 THEN PLOT X,Y:COLO
R U1:PLOT X,Y:GOTO 220
RV 200 IF F THEN TRAP 200:POSITIO
N X,Y:POKE 765,C:XIO 18,#6,12,
0,"S:"
HU 210 PLOT X,Y:COLOR U:PLOT X,Y:
IF Z THEN Y1=USR(ADR(C1$))
IQ 220 COLOR C:PLOT X,Y:A=PEEK(QQ
):IF A=QQ1 THEN 140
EM 230 POKE QQ,QQ1:IF A=44 AND Z=
U THEN Z=U1:A=QQ1
FZ 240 IF A=44 AND Z THEN Z=U:RES
TORE 390:FOR I=705 TO 712:READ
 J:POKE I,J:NEXT I
PT 250 IF A=1 THEN E=U1:[illegible]
LU 260 IF A=0 THEN C$="Wprowadzan
ie":GOTO 410
PG 270 IF A=62 THEN C$="przechowa
nie":GOTO 410
AO 280 IF A=56 AND F=U1 THEN F=U:
A=QQ1
```

```
ER 290 IF A=56 AND F=U THEN F=U1
   E=U:I=U
QC 300 IF A=42 THEN I=U1:E=U:F=U:
   B=C:C=U
DE 310 IF A=54 OR A=118 THEN 60
NI 320 IF A=18 AND AL THEN A=QQ1:
   AL=U R=U1
CI 330 IF A=18 THEN AL=S:R=SG
HV 340 PLOT X,Y:COLOR U:PLOT X,Y:
   IF Z THEN Y1=USR(ADR(C1$))
FP 350 COLOR C:PLOT X,Y:GOTO 140
MB 360 REM DANE RUCHU MANIPULATOR
   A
SR 370 DATA 5,6,7,25,29,33,9,10,1
   1,26,30,34
GX 380 REM DANE KOLORU DLA GR.10
LN 390 DATA 6,36,70,102,132,166,2
   26,42
IP 400 REM WPROWADZANIE LUB PRZEC
   HOWANIE RYSUNKU
FK 410 ZBIOR$="" GOSUB 570:GRAPHI
   CS 2:POSITION 4,1:? #6;C$:? #6
   ;"NAZWA ZBIORU ?":INPUT ZBIOR$
SK 415 IF LEN(ZBIOR$)<3 AND ZBIOR
   $(1,1)="C" THEN 500
OH 417 IF LEN(ZBIOR$)>U1 THEN 470
IZ 420 GRAPHICS 2:POKE 710,130:?
   #6;"KATALOG ZBIOROW":TRAP 460:
   OPEN #U1,6,U,"D1:*.PIC" FOR I=
   U1 TO 9:INPUT #U1,B$
RA 430 IF ASC(B$)<>32 THEN FOR I=
   U1 TO LEN(B$):? #6;CHR$(ASC(B$
   (I,I))+128);:NEXT I:GOTO 460
GN 440 IF ASC(B$)=32 THEN B$=B$(2
   )
KN 450 TRAP 460:? #6;B$:NEXT I
TS 460 CLOSE #U1:? "File to ";C$;
   :INPUT ZBIOR$:IF LEN(ZBIOR$)<2
    THEN 420
PT 470 IF ZBIOR$(1,2)<>"D:" THEN
   B$=ZBIOR$ ZBIOR$="D:":ZBIOR$(3
   )=B$
JS 480 A=LEN(ZBIOR$):TRAP 490:IF
   ZBIOR$(A-3)=".PIC" THEN 500
XE 490 ZBIOR$(A+1)=".PIC"
BE 500 TRAP 60:CLOSE #2:IF C$="sa
   ve" THEN I=8 CMD=11:POKE 559,0
   :GOSUB 520:POKE 559,34:GOTO 12
   0
PY 510 CMD=7:T=1:GOSUB 120:I=4:GO
   SUB 520:GOTO 140
RA 520 OPEN #2,I,128,ZBIOR$
NJ 530 J=STADR GOSUB 560:POKE 868
   ,B:POKE 869,A J=BYTES:GOSUB 56
   0:POKE 872,B:POKE 873,A
GZ 540 POKE 866,CMD:ERROR=USR(ADR
   (CIO$),32)
ZQ 550 ERROR=PEEK(867):CLOSE #2:R
   ETURN
ZP 560 A=INT(J/256):B=INT(J-A*256
   ) RETURN
NW 570 POKE 764,255:CIO$="hhh*LVd
   ":BYTES=7400:STADR=PEEK(88)+25
   6*PEEK(89):RETURN
HW 1000 REM
RA 1010 GRAPHICS 0:DA=1:GOSUB 50:
   POSITION 13,3:? "OPIS PROGRAMU
   .":? :? :?
FX 1020 ? "      Prezentowany prog
   ram umozliwia":? "kreslenie ry
   sunku, za pomoca manipu-"
MV 1025 ? "latora, w dziewiecioko
   lorowym trybie"
YU 1030 ? "graficznym (10). Uzywa
    on stacji dys-":? "kow lub ma
   gnetofonu do przechowania":? "
   rysunku."
QP 1040 ? "      Po uruchomieniu p
   rogramu pojawia":? "sie zestaw
    dostepnych instrukcji."
UW 1050 ? "Mozemy powrocic do nie
   go w dowolnej":? "chwili po na
   cisnieciu klawisza"
HI 1060 ? "<CLEAR>, lecz kasuje t
   o rysunek.":? "Mozemy zmienic
   kolor kreslenia nacis-"
KQ 1070 ? "kajac przycisk manipul
   atora."
BB 1080 ? :? "Dalsze informacje p
   o nacisnieciu":? "dowolnego kl
   awisza."
BA 1090 RETURN
CY 1100 GRAPHICS 0:GOSUB 50:POSIT
   ION 10,0:? "INSTRUKCJE KRESLEN
   IA"
OD 1110 ? :? " manipulator   kres
   lenie":?
BT 1120 ? " przycisk      zmiana
    koloru":?
BA 1130 ? " tab key       zamiana
    kolorow":?
AO 1140 ? " clear         wylacze
   nie kreslenia":?
TU 1150 ? "  I            kursor
   niewidoczny":?
MN 1160 ? "  E            tryb ka
   sowania":?
NC 1170 ? "  C            kontynu
   owanie ruchu":?
YP 1180 ? "  F            wypelni
   anie w prawo":?
WD 1190 ? "  L            wprowad
   zanie (LOAD)":?
JZ 1200 ? "  S            przecho
   wanie (SAVE)":?
MF 1210 ? "      NACISNIJ DOWOLNY K
   LAWISZ"
TL 1220 IF PEEK(764)=255 THEN 122
   0
BO 1230 POKE 764,255 RETURN
QO 1 REM ****************
IB 2 REM *              *
OY 3 REM * Program nr 40 *
ID 4 REM *              *
QS 5 REM ****************
NL 6 REM
KG 10 GRAPHICS 0:DIM PIER(22,14),
   DRU(22,14)
UR 15 GOTO 45
LZ 20 FOR X=1 TO 22:FOR Y=1 TO 14
AP 30 PIER(X,Y)=0:DRU(X,Y)=0
DG 40 NEXT Y:NEXT X
ZY 42 RETURN
MM 45 GOSUB 10000
GU 50 GRAPHICS 2:POKE 712,148:L=0
BN 60 STX=0:STY=0:AKTX=0:AKTY=0:M
   AXX=1:MINX=22
EJ 65 MX=1:MI=22
IK 70 COLOR 10:PLOT AKTX,AKTY
ZP 80 D=STICK(0)
CF 81 IF STRIG(0) THEN 90
PM 82 PIER(AKTX+2,AKTY+2)= NOT (P
   IER(AKTX+2,AKTY+2))
DJ 83 IF AKTX+2>MAXX THEN MAXX=AK
   TX+2
YP 84 IF AKTX+2<MINX THEN MINX=AK
   TX+2
EY 87 FOR CZ=1 TO 100:NEXT CZ
MD 90 IF D/2=INT(D/2) THEN AKTY=A
   KTY-1
JG 100 IF D=9 OR D=13 OR D=5 THEN
    AKTY=AKTY+1
JA 110 IF D>8 AND D<12 THEN AKTX=
   AKTX-1
HE 120 IF D>4 AND D<8 THEN AKTX=A
   KTX+1
RB 130 AKTX=AKTX*(AKTX>=0):AKTY=A
   KTY*(AKTY>=0)
DL 140 IF AKTX>19 THEN AKTX=19
SG 150 IF AKTY>11 THEN AKTY=11
FU 160 COLOR 0:PLOT STX,STY:IF PI
   ER(STX+2,STY+2)=1 THEN COLOR 1
   20:PLOT STX,STY
ZD 165 COLOR 10:PLOT AKTX,AKTY
ZA 170 STX=AKTX:STY=AKTY
YL 175 IF PEEK(53279)=6 THEN COLO
   R 0:PLOT STX,STY:D=1:GOTO 200
SA 180 GOTO 70
HK 200 POKE 709,30:POKE 711,0:POK
   E 752,1:L=L+1:? CHR$(125);"Pop
   ulacja nr ";L
PD 210 FOR X=2 TO 13:FOR Y=MINX-1
    TO MAXX+1
SM 221 GOSUB 500
JB 222 TRAP 40000
DI 230 IF PIER(Y,X)=1 THEN IF SAS
   =2 OR SAS=3 THEN DRU(Y,X)=1:GO
   SUB 300
AS 232 IF SAS=3 THEN DRU(Y,X)=1:G
   OSUB 300
EH 240 COLOR (120+128*D)*DRU(Y,X)
YF 242 PLOT Y-2,X-2
WM 250 NEXT Y:NEXT X:MINX=MI:MAXX
   =MX
CC 260 K=0:FOR X=2 TO 13:FOR Y=2
   TO 21
AS 265 IF DRU(Y,X)<>0 THEN K=K+1
DT 270 PIER(Y,X)=DRU(Y,X):DRU(Y,X
   )=0
UP 280 NEXT Y:NEXT X
KJ 285 IF K=0 THEN RUN
TB 290 D= NOT (D):GOTO 200
NI 300 IF Y>MX THEN MX=Y
VK 310 IF Y<MI THEN MI=Y
ZD 320 RETURN
XO 500 POKE 77,0:SAS=0:FOR A=-1 T
   O 1:FOR B=-1 TO 1:IF A=0 AND B
   =0 THEN 520
VN 505 IF Y+A>22 OR X+B>14 THEN 5
   20
EF 510 SAS=SAS+PIER(Y+A,X+B)
RV 520 NEXT B:NEXT A:RETURN
XI 1000 GRAPHICS 0:POSITION 13,2:
   ? "OPIS PROGRAMU"
GL 1005 POKE 764,255
QO 1010 ? :? :? "      Zasady tej
   gry stworzyl John ":? "Horton
   Conway, matematyk z Cambridge,
   "
PE 1020 ? "w 1970 roku. Rozpowsze
   chnil ja Martin":? "Gardner, r
   edaktor 'Scientific Ameri-'":?
   "can."
GN 1030 ? "      Jest to gra niety
   powa. Gracz ":? "praktycznie n
   ie istnieje. Mamy"
BM 1040 ? "uproszczona symulacje
   rozwoju kolonii":? "mikroorgan
   izmow ogladana pod mikro-"
YK 1050 ? "skopem. W przeciwienst
   wie do rzeczy-":? "wistego zyc
   ia, jego symulacja ma "
BC 1060 ? "trzy proste reguly. Ka
   zdy ruch (lub ":? "'populacja'
   ) zmienia ksztalt komorek"
NR 1070 ? "zgodnie z tymi regulam
   i. Generacje":? "zmieniaja kol
   ory miedzy zlotym a "
WV 1080 ? "czarnym, na niebieskim
    tle."
LE 1090 ? :? "      NACISNIJ DOWOL
   NY KLAWISZ."
PS 1100 IF PEEK(764)=255 THEN 110
   0
FW 1110 POKE 764,255
MM 1120 GRAPHICS 0:? :? :? "1. Re
   gula smierci. Komorka z cztere
   -":? "    ma lub wiecej sasiada
   mi umiera"
DU 1130 ? "     z zageszczenia. Kom
   orka z jednym ":? "    lub bez
    sasiadow umiera z samot-"
XH 1140 ? "     nosci."
MQ 1150 ? "2. Regula przetrwania.
    Jesli komorka":? "    ma dwoch
    lub trzech sasiadow, zy-"
OT 1160 ? "     je do nastepnej run
   dy,":? "3. Regula narodzin. Je
   sli dokladnie":? "    trzech sa
   siadow jest w bezposred-"
LM 1170 ? "     niej bliskosci pust
   ego obszaru,":? "    rodzi sie
    nowa komorka."
NE 1180 ? :? :? "      Po uruchomi
   eniu na ekranie poja-":? "wi s
   ie migajaca gwiazdka [*]. Uzyw
   a-"
TF 1190 ? "jac manipulatora, prze
   suwamy ja do ":? "czesci ekran
   u, gdzie chcemy umiescic"
ZJ 1200 ? "pierwsza komorke i nac
   iskamy przy-":? "cisk. Komorka
    wyglada jak X. "
TY 1210 ? :? "        NACISNIJ DOWO
   LNY KLAWISZ.";
TL 1220 IF PEEK(764)=255 THEN 122
   0
GV 1225 POKE 764,255
OX 1230 GRAPHICS 0:? :? :? "
   W podobny sposob tworzymy wzor
   ":? "komorek na ekranie. Mozem
   y utworzyc"
RD 1240 ? "dowolny ksztalt. Pamie
   tajmy jednak,":? "ze im mniej
   komorek, tym szybsza re-"
UA 1250 ? "generacja i wzrost. Po
    zakonczeniu":? "naciskamy kla
   wisz <START>"
XI 1260 ? :? :? "      NACISNIJ D
   OWOLNY KLAWISZ.";
ZU 1270 IF PEEK(764)=255 THEN 127
   0
```

```
RM 1280 GOTO 50
MW 10000 GRAPHICS 2:POSITION 7,4:
   ? #6;"ZYCIE":POKE 710,0:POKE 7
   52,1
AM 10005 GOSUB 20
YZ 10010 ? "Czy chcesz opis progr
   amu (t/n)? ";
WS 10020 A=PEEK(764):IF A=255 THE
   N 10020
JF 10030 IF A=45 THEN 1000
DV 10060 RETURN
```

```
1 REM ******************
2 REM *                *
3 REM * Program nr 19a *
4 REM *  dla komputera *
5 REM *    AMSTRAD     *
6 REM *                *
7 REM ******************
8 REM
30 INK 1,9:INK 2,26:PEN 2:PEN
#2,2
40 INK 0,9
50 MODE 1
60 WINDOW #2,1,40,22,25:CLS #2
70 REM
80 ya=1
90 xa=1
100 xm=639
110 ym=399
120 GRAPHICS PEN 3
125 FOR kk=1 TO 100:NEXT kk
126 SOUND 1,1,0,15,1,.1
130 GOSUB 830: REM tworzenie m
enu
140 aa=0: REM
150 REM
160 LOCATE 1,1
170 PRINT "          ";
180 FOR k=1 TO 300:NEXT k
190 LOCATE 1,1
200 PRINT"wybierz:";
210 FOR k=1 TO 300:NEXT k
220 a$=INKEY$: REM w a$ wybor
ksztaltu
230 IF a$="" OR a$<"1" OR a$>"
4" THEN 160
240 ro=VAL(a$)*2+1
250 n=0:REM n wskazuje, czy ak
tualny zarys figury powinien b
yc wykasowany, czy nie
260 LOCATE #2,1,1
270 PRINT #2,"wybierz:":REM
280 LOCATE 1,ro
290 PRINT" ";
300 FOR k=1 TO 300:NEXT k
310 LOCATE #2,1,1
320 PRINT #2;"          ";
330 LOCATE 1,ro
340 PRINT a$;
350 FOR k=1 TO 300:NEXT k
360 b$=INKEY$:REM w b$ jest wy
bor przeksztalcenia
370 IF b$="" OR b$<>"p" AND b$
<>"s" AND b$<>"o" AND b$<>"k"
AND b$<>"n" AND b$<>"e" THEN 2
60
380 IF b$="e" THEN 3100
390 IF b$<>"n" THEN 440
400 RESTORE
410 GOSUB 990
420 REM kasowanie
430 GOTO 140
440 IF b$<>"k" THEN 530
450 GRAPHICS PEN 0:REM Kreslen
ie w kolorze tla
460 IF b$="k" AND a$="1" AND n
=1 THEN GOSUB 1070
470 IF b$="k" AND a$="2" AND n
=1 THEN GOSUB 1100
480 IF b$="k" AND a$="3" AND n
=1 THEN GOSUB 1160
490 IF b$="k" AND a$="4" AND n
=1 THEN GOSUB 1430
500 RESTORE
510 GOSUB 990
520 GOTO 140
530 LOCATE 1,23:PRINT "     ";
540 LOCATE 1,24:PRINT "
";
550 REM Przemieszczenie?
560 IF b$<>"p" THEN 640
570 GOSUB 1850:REM Wydruk inst
rukcji do przemieszczenia
580 GRAPHICS PEN 0:REM ustawie
nie koloru do kasowania
590 IF b$="p" AND a$="1" THEN
GOSUB 1510
600 IF b$="p" AND a$="2" THEN
GOSUB 1590
610 IF b$="p" AND a$="3" THEN
GOSUB 1690
620 IF b$="p" AND a$="4" THEN
GOSUB 1760
630 REM Skalowanie
640 IF b$<>"s" THEN 720
650 GOSUB 2330:REM Wydruk inst
rukcji do skalowania
660 GRAPHICS PEN 0
670 IF b$="s" AND a$="1" AND n
=1 THEN GOSUB 1920
680 IF b$="s" AND a$="2" AND n
=1 THEN GOSUB 2020
690 IF b$="s" AND a$="3" AND n
=1 THEN GOSUB 2140
700 IF b$="s" AND a$="4" AND n
=1 THEN GOSUB 2220
710 REM Obrot
720 IF b$<>"o" THEN 790
730 GOSUB 3000: REM Wydruk ins
trukcji do obrotu
740 GRAPHICS PEN 0
750 IF b$="o" AND a$="1" AND n
=1 THEN GOSUB 2400
760 IF b$="o" AND a$="2" AND n
=1 THEN GOSUB 2540
770 IF b$="o" AND a$="3" AND n
=1 THEN GOSUB 2740
780 IF b$="o" AND a$="4" AND n
=1 THEN GOSUB 2830
790 n=1:REM od teraz kasowanie
 starego obrazu
800 GOSUB 960:REM Ponowne wysw
ietlenie menu przeksztalcania
810 GOTO 260
830 REM Wyswietlenie menu
840 LOCATE 1,1:PRINT "wybierz:
";
850 LOCATE 1,3:PRINT "1.";
860 LOCATE 1,5:PRINT "2.";
870 LOCATE 1,7:PRINT "3.";
880 LOCATE 1,9:PRINT "4.";
890 GRAPHICS PEN 3
900 GOSUB 990
910 GOSUB 1080
920 GOSUB 1100
930 aa=0
940 GOSUB 1160
950 GOSUB 1430
960 CLS #2:LOCATE #2,1,1:PRINT
 #2,"wybierz:                 "
970 LOCATE #2,1,2:PRINT #2,"P
- przesuniecie  S - zmiana ska
li   O - obrot  K - kasowanie
N - nowy obiekt   E - koniec p
racy";
980 RETURN
990 REM Czytanie danych o punk
tach
1000 READ xp,yp,xq,yq
1010 READ xu,yu,xv,yv,xw,yw,xx
,yx
1020 READ xc,yc,rx,ry
1030 READ xr,yr,xs,ys,xt,yt
1040 RETURN
1050 PRINT""
1060 REM ***Procedury kreslen
a***
1070 REM Kreslenie linii
1080 PLOT xp,ym-yp:DRAW xq,ym-
yq
1090 RETURN
1100 REM Kreslenie prostokata
1110 PLOT xu,ym-yu:DRAW xv,ym-
yv
1120 DRAW xw,ym-yw
1130 DRAW xx,ym-yx
1140 DRAW xu,ym-yu
1150 RETURN
1160 REM Kreslenie okregu
1170 IF rx<ry THEN 1200
1180 r=ry
1190 GOTO 1210
1200 r=rx
1210 IF aa<>0 THEN 1310:REM
1220 FOR a=0 TO 1.5708 STEP 1/
r:REM Kreslenie czterech symet
rycznych punktow
1230     dx=rx*COS(a)
1240     dy=ry*SIN(a)
1250     PLOT xc+dx,ym-yc+dy*ya
1260     PLOT xc+dx,ym-yc-dy*ya
1270     PLOT xc-dx,ym-yc+dy*ya
1280     PLOT xc-dx,ym-yc-dy*ya
1290 NEXT a
1300 GOTO 1420
1310 cx=SIN(aa)*xa:REM Oblicze
nie stalych czesci rownania
1320 cy=SIN(aa)*ya
1330 FOR a=0 TO 3.14159 STEP 1
/r
1340 x=rx*COS(a):REM Rownania
okregu
1350 y=ry*SIN(a)*ya
1360 xh=x
1370 x=x*COS(aa)+y*cx:REM Rown
ania obrotu
1380 y=y*COS(aa)-xh*cy
1390 PLOT xc+x,ym-yc+y
1400 PLOT xc-x,ym-yc-y:REM Kre
sli 2 symetryczne punkty naraz
1410 NEXT a
1420 RETURN
1430 REM Kreslenie trojkata
1440 PLOT xr,ym-yr:DRAW xs,ym-
ys
1450 DRAW xt,ym-yt
1460 DRAW xr,ym-yr
1470 RETURN
1480 REM
1490 REM ***Przemieszczenie***
1500 REM Przemieszczenie linii
1510 IF n=0 THEN 1530:REM Jesl
i n jest zero, ksztalt jest je
szcze w menu
1520 GOSUB 1070:REM kasowanie
1530 xp=xp+ht:yp=yp+vt
1540 xq=xq+ht:yq=yq+vt
1550 GRAPHICS PEN 3
1560 GOSUB 1070:REM Kreslenie
1570 RETURN
1580 REM Przemieszczenie prost
okata
1590 IF n=0 THEN 1610
1600 GOSUB 1100:REM Kasowanie
1610 xu=xu+ht:yu=yu+vt
1620 xv=xv+ht:yv=yv+vt
1630 xw=xw+ht:yw=yw+vt
1640 xx=xx+ht:yx=yx+vt
1650 GRAPHICS PEN 3
1660 GOSUB 1100:REM Kreslenie
1670 RETURN
1680 REM Przemieszczenie okreg
u
1690 IF n=0 THEN 1710
1700 GOSUB 1160:REM Kasowanie
1710 xc=xc+ht:yc=yc+vt
1720 GRAPHICS PEN 3
1730 GOSUB 1160:REM Kreslenie
```

```
1740 RETURN
1750 REM Przemieszczenie trojk
ata
1760 IF n=0 THEN 1780
1770 GOSUB 1430:REM Kasowanie
1780 xr=xr+ht:yr=yr+vt
1790 xs=xs+ht:ys=ys+vt
1800 xt=xt+ht:yt=yt+vt
1810 GRAPHICS PEN 3
1820 GOSUB 1430:REM Kreslenie
1830 RETURN
1840 REM Instrukcje
1850 CLS #2:LOCATE #2,1,1:PRI
NT #2,"          ";
1860 LOCATE #2,1,1
1870 PRINT #2,"przesuniecie x
i y ";
1880 INPUT #2,ht,vt
1890 RETURN
1900 REM
1910 REM ***Skalowanie***
1920 REM Skalowanie linii
1930 IF n=0 THEN 1950
1940 GOSUB 1070:REM Kasowanie
1950 xf=(xp+xq)/2
1960 yf=(yp+yq)/2
1970 xp=xp*hs+xf*(1-hs):yp=yp*
vs+yf*(1-vs)
1980 xq=xq*hs+xf*(1-hs):yq=yq*
vs+yf*(1-vs)
1990 GRAPHICS PEN 3
2000 GOSUB 1070:REM Kreslenie
2010 RETURN
2020 REM Skalowanie pudelka
2030 IF n=0 THEN 2050
2040 GOSUB 1100:REM Kasowanie
2050 xf=(xu+xv+xw+xx)/4
2060 yf=(yu+yv+yw+yx)/4
2070 xu=xu*hs+xf*(1-hs):yu=yu*
vs+yf*(1-vs)
2080 xv=xv*hs+xf*(1-hs):yv=yv*
vs+yf*(1-vs)
2090 xw=xw*hs+xf*(1-hs):yw=yw*
vs+yf*(1-vs)
2100 xx=xx*hs+xf*(1-hs):yx=yx*
vs+yf*(1-vs)
2110 GRAPHICS PEN 3
2120 GOSUB 1100:REM Kreslenie
2130 RETURN
2140 REM Skalowanie okregu
2150 IF n=0 THEN 2170
2160 GOSUB 1160:REM Kasowanie
2170 rx=rx*hs
2180 ry=ry*vs
2190 GRAPHICS PEN 3
2200 GOSUB 1160:REM Kreslenie
2210 RETURN
2220 REM Skalowanie trojkata
2230 IF n=0 THEN 2250
2240 GOSUB 1430:REM Kasowanie
2250 xf=(xr+xs+xt)/3
2260 yf=(yr+ys+yt)/3
2270 xr=xr*hs+xf*(1-hs):yr=yr*
vs+yf*(1-vs)
2280 xs=xs*hs+xf*(1-hs):ys=ys*
vs+yf*(1-vs)
2290 xt=xt*hs+xf*(1-hs):yt=YT*
VS+YF*(1-vs)
2300 GRAPHICS PEN 3
2310 GOSUB 1430:REM Kreslenie
2320 RETURN
2330 REM Instrukcje
2340 CLS #2:LOCATE #2,1,1
2350 PRINT #2,"wprowadz wspolc
zynniki skali x i y";
2360 INPUT #2,hs,vs
2370 RETURN
2380 REM
2390 REM ***Obrot***
2400 REM Obrot linii
2410 IF n=0 THEN 2430
2420 GOSUB 1070:REM Kasowanie
2430 xo=(xp+xQ)/2:REM Szukanie
 srodka linii
2440 yo=(yp+yq)/2
2450 xh=xp:REM Zachowanie wart
osci XP
2460 xp=xo+(xp-xo)*COS(ar)+(yp
-yo)*SIN(ar)*xa
2470 yp=yo+(yp-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2480 xh=xq
2490 xq=xo+(xq-xo)*COS(ar)+(yq
-yo)*SIN(ar)*xa
2500 yq=yo+(yq-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2510 GRAPHICS PEN 3
2520 GOSUB 1070:REM Kreslenie
2530 RETURN
2540 REM Obrot pudelka
2550 IF n=0 THEN 2570
2560 GOSUB 1100:REM Kasowanie
2570 xo=(xu+xv+xw+xx)/4
2580 yo=(yu+yv+yw+yx)/4:REM Sz
ukanie srodka prostokata
2590 xh=xu:REM Zachowanie biez
acej wartosci xu
2600 xu=xo+(xu-xo)*COS(ar)+(yo
-yo)*SIN(ar)*xa
2610 yu=yo+(yu-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2620 xh=xv
2630 xv=xo+(xv-xo)*COS(ar)+(yv
-yo)*SIN(ar)*xa
2640 yv=yo+(yv-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2650 xh=xw
2660 xw=xo+(xw-xo)*COS(ar)+(yw
-yo)*SIN(ar)*xa
2670 yw=yo+(yw-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2680 xh=xx
2690 xx=xo+(xx-xo)*COS(ar)+(yx
-yo)*SIN(ar)*xa
2700 yx=yo+(yx-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2710 GRAPHICS PEN 3
2720 GOSUB 1100:REM Kreslenie
2730 RETURN
2740 REM Obrot okregu
2750 IF n=0 THEN 2780
2760 aa=as
2770 GOSUB 1160:REM Kasowanie
2780 aa=ar+as
2790 GRAPHICS PEN 3
2800 GOSUB 1160:REM Kreslenie
2810 as=aa
2820 RETURN

2830 REM Obrot trojkata
2840 IF n=0 THEN 2860
2850 GOSUB 1430:REM Kasowanie
2860 xo=(xr+xs+xt)/3:REM Szuka
nie srodka trojkata
2870 yo=(yr+ys+yt)/3
2880 xh=xr
2890 xr=xo+(xr-xo)*COS(ar)+(yr
-yo)*SIN(ar)*xa
2900 yr=yo+(yr-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2910 xh=xs
2920 xs=xo+(xs-xo)*COS(ar)+(ys
-yo)*SIN(ar)*xa
2930 ys=yo+(ys-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2940 xh=xt
2950 xt=xo+(xt-xo)*COS(ar)+(yt
-yo)*SIN(ar)*xa
2960 yt=yo+(yt-yo)*COS(ar)-(xh
-xo)*SIN(ar)*ya
2970 GRAPHICS PEN 3
2975 GOSUB 1430:REM Kreslenie
2980 RETURN
2990 REM Instrukcje
3000 CLS #2:LOCATE #2,1,1
3010 PRINT #2,"kat obrotu";
3020 INPUT #2,ar
3030 ar=ar*3.14159/180:REM Zam
iana stopni na radiany
3040 RETURN
3050 REM Dane
3060 DATA 30,37,80,37
3070 DATA 80,50,80,80,30,80,30
,50
3080 DATA 55,100,16,16
3090 DATA 30,140,80,140,55,120
3100 END
```

ATARI

ALFABET MORSE'A

```
1 POKE 752,1
2 ? CHR$(125):POSITION 12,5:? "M
 O R S E 'A":POSITION 9,9:? "PRO
GRAM EDUKACYJNY"
3 POSITION 31,20:? "& JB.TB.":PO
SITION 33,21:? "1986"
4 FOR Y=0 TO 200 STEP 3:SOUND 0,
Y,10,10:SOUND 1,Y/4,10,10:SETCOL
OR 2,2,Y/4:NEXT Y
5 FOR Y=200 TO 0 STEP -2:SOUND 0
,Y,10,10:SOUND 1,Y/2/3,10,10:SET
COLOR 2,2,Y
6 SOUND 0,0,0,0:SOUND 1,0,0,0:SE
TCOLOR 2,0,0
8 DIM A(999),IL$(3),E$(1),R$(4):
GOTO 30000
10 ? CHR$(125):POSITION 10,1:? "
TEKST PRZYGOTOWANY":REM TEKST PR
ZYGOTOWANY
11 S=0:POKE 752,0
12 CLOSE #1:OPEN #1,4,0,"K:":GET
 #1,A
13 IF A=30 THEN ? CHR$(30);CHR$(
32);CHR$(30);:IF A(S)<>0 THEN A(
S)=0:S=S-1:GOTO 12
```

```
16 IF A=155 THEN 25
17 IF A>=48 AND A<=57 OR A>=65 A
ND A<=94 OR A=61 OR A=32 OR A=45
 THEN GOTO 20
19 GOTO 12
20 S=S+1:A(S)=A:? CHR$(A);
22 GOTO 12
25 ? CHR$(125):POKE 752,1
26 FOR A=1 TO S:GOSUB A(A):NEXT
A
27 IF PEEK(53279)=3 THEN CLR :GO
TO 10
28 IF PEEK(53279)=5 THEN CLR :GO
TO 300
29 GOTO 27
30 POKE 752,1
31 GOSUB 122:GOSUB 121:GOSUB 121
:GOSUB 121:GOSUB 122:GOSUB 123:R
ETURN
32 ? " ";:RETURN
45 ? "-";:GOSUB 123:RETURN
48 ? "0";:GOSUB 122:GOSUB 122:GO
SUB 122:GOSUB 122:GOSUB 122:GOSU
B 124:RETURN
49 ? "1";:GOSUB 121:GOSUB 122:GO
SUB 122:GOSUB 122:GOSUB 122:GOSU
B 124:RETURN
50 ? "2";:GOSUB 121:GOSUB 121:GO
SUB 122:GOSUB 122:GOSUB 122:GOSU
B 124:RETURN
51 ? "3";:GOSUB 121:GOSUB 121:GO
SUB 121:GOSUB 122:GOSUB 122:GOSU
B 124:RETURN
52 ? "4";:GOSUB 121:GOSUB 121:GO
SUB 121:GOSUB 121:GOSUB 122:GOSU
B 124:RETURN
53 ? "5";:GOSUB 121:GOSUB 121:GO
SUB 121:GOSUB 121:GOSUB 121:GOSU
B 124:RETURN
54 ? "6";:GOSUB 122:GOSUB 121:GO
SUB 121:GOSUB 121:GOSUB 121:GOSU
B 124:RETURN
55 ? "7";:GOSUB 122:GOSUB 122:GO
SUB 121:GOSUB 121:GOSUB 121:GOSU
B 124:RETURN
56 ? "8";:GOSUB 122:GOSUB 122:GO
SUB 122:GOSUB 121:GOSUB 121:GOSU
B 124:RETURN
57 ? "9";:GOSUB 122:GOSUB 122:GO
SUB 122:GOSUB 122:GOSUB 121:GOSU
B 124:RETURN
61 ? "=";:GOSUB 122:GOSUB 121:GO
SUB 121:GOSUB 121:GOSUB 122:GOSU
B 124:RETURN
65 ? "A";:GOSUB 121:GOSUB 122:GO
SUB 123:RETURN
66 ? "B";:GOSUB 122:GOSUB 121:GO
SUB 121:GOSUB 121:GOSUB 124:RETU
RN
67 ? "C";:GOSUB 122:GOSUB 121:GO
SUB 122:GOSUB 121:GOSUB 124:RETU
RN
68 ? "D";:GOSUB 122:GOSUB 121:GO
SUB 121:GOSUB 124:RETURN
69 ? "E";:GOSUB 121:GOSUB 124:RE
TURN
70 ? "F";:GOSUB 121:GOSUB 121:GO
SUB 122:GOSUB 121:GOSUB 124:RETU
RN
71 ? "G";:GOSUB 122:GOSUB 122:GO
SUB 121:GOSUB 124:RETURN
72 ? "H";:GOSUB 121:GOSUB 121:GO
SUB 121:GOSUB 121:GOSUB 124:RETU
RN
73 ? "I";:GOSUB 121:GOSUB 121:GO
SUB 124:RETURN
74 ? "J";:GOSUB 121:GOSUB 122:GO
SUB 122:GOSUB 122:GOSUB 124:RETU
RN
75 ? "K";:GOSUB 122:GOSUB 121:GO
SUB 122:GOSUB 124:RETURN
76 ? "L";:GOSUB 121:GOSUB 122:GO
SUB 121:GOSUB 121:GOSUB 124:RETU
RN
77 ?. "M";:GOSUB 122:GOSUB 122:GO
SUB 124:RETURN
78 ? "N";:GOSUB 122:GOSUB 121:GO
SUB 124:RETURN
79 ? "O";:GOSUB 122:GOSUB 122:GO
SUB 122:GOSUB 124:RETURN
80 ? "P";:GOSUB 121:GOSUB 122:GO
SUB 122:GOSUB 121:GOSUB 124:RETU
RN
81 ? "Q";:GOSUB 122:GOSUB 122:GO
SUB 121:GOSUB 122 GOSUB 124:RETU
RN
82 ? "R";:GOSUB 121:GOSUB 122:GO
SUB 121:GOSUB 124:RETURN
83 ? "S";:GOSUB 121:GOSUB 121:GO
SUB 121:GOSUB 124:RETURN
84 ? "T";:GOSUB 122:GOSUB 124:RE
TURN
85 ? "U";:GOSUB 121:GOSUB 121:GO
SUB 122:GOSUB 124:RETURN
86 ? "V";:GOSUB 121:GOSUB 121:GO
SUB 121:GOSUB 122:GOSUB 124:RETU
RN
87 ? "W";:GOSUB 121:GOSUB 122:GO
SUB 122:GOSUB 124:RETURN
88 ? "X";:GOSUB 122:GOSUB 121:GO
SUB 121:GOSUB 122:GOSUB 124:RETU
RN
89 ? "Y";:GOSUB 122:GOSUB 121:GO
SUB 122:GOSUB 122:GOSUB 124:RETU
RN
90 ? "Z";:GOSUB 122:GOSUB 122:GO
SUB 121:GOSUB 121:GOSUB 123:RETU
RN
```

```
91 ? "C";:GOSUB 122:GOSUB 122:GO
SUB 122:GOSUB 121:GOSUB 124:RETU
RN :REM CZ
92 ? "\";:GOSUB 122:GOSUB 122:GO
SUB 122:GOSUB 122:GOSUB 124:RETU
RN :REM SZ
93 ? "J";:GOSUB 121:GOSUB 122:GO
SUB 121:GOSUB 122:GOSUB 124:RETU
RN :REM JA
94 ? "^";:GOSUB 121:GOSUB 121:GO
SUB 122:GOSUB 122:GOSUB 124:RETU
RN :REM JU
121 FOR I=1 TO K:SOUND 0,F,10,10
:NEXT I:SOUND 0,0,0,0:FOR I=1 TO
 K:NEXT I:RETURN
122 FOR I=1 TO K+K+K:SOUND 0,F,1
0,10:NEXT I:SOUND 0,0,0,0:FOR I=
1 TO K:NEXT I:RETURN
123 FOR I=1 TO K+K+K:NEXT I:RETU
RN
124 FOR I=1 TO K+K:NEXT I:RETURN
125 FOR I=0 TO 100:NEXT I:RETURN
130 FOR I=0 TO 300:NEXT I:RETURN
142 INPUT A$:N=0:T=LEN(A$)
149 REM SYGNAL
150 ? CHR$(125):POKE 752,1:POSIT
ION 14,4:? "S Y G N A L"
151 POSITION 7,7:? "TON   1":POS
ITION 7,9:? "TON   2":POSITION 7
,11:? "TON   3":POSITION 7,13:?
"TON   4"
152 POSITION 7,15:? "TON   5":PO
SITION 7,17:? "TON   6":POSITION
 7,19:? "TON   7":POSITION 7,21:
? "TON   8"
153 CLOSE #1:OPEN #1,4,0,"K:":GE
T #1,A
154 IF A>=49 AND A<=56 THEN 156
155 GOTO 153
156 IF A=49 THEN POSITION 23,7:?
 "1":F=80:GOSUB 125:GOTO 255
157 IF A=50 THEN POSITION 23,9:?
 "2":F=75:GOSUB 125:GOTO 255
158 IF A=51 THEN POSITION 23,11:
? "3":F=75:GOSUB 125:GOTO 255
159 IF A=52 THEN POSITION 23,13:
? "4":F=60:GOSUB 125:GOTO 255
160 IF A=53 THEN POSITION 23,15:
? "5":F=50:GOSUB 125:GOTO 255
161 IF A=54 THEN POSITION 23,17:
? "6":F=40:GOSUB 125:GOTO 255
162 IF A=55 THEN POSITION 23,19:
? "7":F=30:GOSUB 125:GOTO 255
163 IF A=56 THEN POSITION 23,21:
? "8":F=20:GOSUB 125:GOTO 255
200 ? CHR$(125):REM NAGLOWEK
202 POSITION 2,0:? "TEMPO    GRU
P/MIN      ILOSC TLG      ":POSI
TION 8,0:? TEMPO
203 POSITION 1,[illegible]
ADAWANYCH    TLG NR       ":POSI
TION 35,0:? VB1
205 VB=VB+1:IF VB>LICZ THEN LICZ
=VB:POSITION 35,1:? LICZ:IF VB-1
=VB1 THEN 300
206 IF PEEK(53279)=3 THEN 300
207 GOSUB 31:GOSUB 130
210 GOSUB 7[illegible]:GOSUB [illegible]:GOSUB [illegible]
OSUB 123:GOSUB NR:GOSUB 1000:GOS
UB Z
211 GOSUB 1000:GOSUB Z:GOSUB [illegible]
GOSUB 123:GOSUB 1100:GOSUB [illegible]
SUB 123:GOSUB 1010:GOSUB Z
215 IF Z=48 THEN GOSUB 1020:GOSU
B Z:GOTO 220
216 IF Z=51 THEN GOSUB 10[illegible]:GOSU
B Z:GOTO 220
219 GOSUB 1000:GOSUB Z
220 GOSUB 32:GOSUB 123:GOSUB 1[illegible]
0:GOSUB Z
240 IF Z=48 OR Z=49 THEN GOSUB 1
000:GOSUB Z:GOTO 242
241 IF Z=50 THEN GOSUB 1010:GOSU
B Z
242 GOSUB 1050:GOSUB Z:GOSUB 100
0:GOSUB Z:GOSUB 123:GOSUB 61:GOS
UB 123:GOSUB 123
243 GOSUB 123:REM ADRES NAGLOWKA
244 GOSUB 90:GOSUB 1060:GOSUB Z:
GOSUB 1060:GOSUB Z:GOSUB [illegible]
B 123
245 GOSUB 90:GOSUB 1060:GOSUB Z:
GOSUB 1060:GOSUB Z:GOSUB [illegible]
B 123

246 GOSUB 90:GOSUB 10[illegible]:GOSUB Z:
GOSUB 1060:GOSUB Z:GOSUB [illegible]
B 123
247 GOSUB 90:GOSUB 1060:GOSUB Z:
GOSUB 1060:GOSUB Z:GOSUB [illegible]
B 123
248 GOSUB 90:GOSUB 1060:GOSUB Z:
GOSUB 1060:GOSUB Z:GOSUB [illegible]
B 123:GOSUB 123:GOSUB 123
249 GOSUB 3[illegible]
250 POKE 1[illegible],0:POKE [illegible]:POKE [illegible]
0
251 RETURN
254 RETURN
255 ? CHR$(125):POKE 731,1:POKE
752,1:REM TEMPO
256 POSITION 14,4:? "T E M P O"
257 POSITION [illegible]:? "TEMPO NADAWA
NIA  6 GRUP/MIN:   0"
258 POSITION 3,10:? "TEMPO NADAW
ANIA  8 GRUP/MIN:   1"
259 POSITION 3,11:? "TEMPO NADAW
ANIA  9 GRUP/MIN:   2"
```

```
[illegible]
CLOSE #1:OPEN #1,4,0,"K:":GE
T #1,TE
[illegible]
? "6":POSITION 33,9:? "0":TEMPO=
[illegible]
:? "8":POSITION 33,10:? "1":TEMP
[illegible]
:? "9":POSITION 33,11:? "2":TEMP
[illegible]
:? "10":POSITION 33,12:? "3":TEM
[illegible]
274 IF TE=52 THEN POSITION 19,13
:? "11":POSITION 33,13:? "4":TEM
PO=11:K=8.7:GOSUB 130:GOTO 300
[illegible]
:? "12":POSITION 33,14:? "5":TEM
[illegible]
276 IF TE=54 THEN POSITION 19,15
:? "14":POSITION 33,15:? "6":TEM
PO=14:K=6.7:GOSUB 130:GOTO 300
[illegible]
:? "16":POSITION 33,16:? "7":TEM
PO=16:K=5.66:GOSUB 130:GOTO 300
[illegible]
:? "18":POSITION 33,17:? "8":TEM
[illegible]
PO=20:K=4.3:GOSUB 130:GOTO 300
300 VB=0:LICZ=0:LG=0:LX=0:LY=4:R
[illegible]
301 ? CHR$(125):POKE 752,1:POSIT
ION 12,4:? "---------------"
303 POSITION 12,5:? " RODZAJ PRA
CY "
[illegible]
[illegible] NADAWANIE
 KLAWIATURY              1  "
320 POSITION 2,11:? " NADAWANIE
TEKSTU PRZYGOTOWANEGO  2  "
330 POSITION 2,13:? " NADAWANIE
TEKSTU LOSOWE          3  "
335 CLOSE #1:OPEN #1,4,0,"K:":GE
T #1,A
[illegible] AND A<=51 THEN 370
360 GOTO [illegible]
370 IF A=49 THEN POSITION 36,9:?
 "1":GOSUB 125:GOTO 900
371 IF A=50 THEN POSITION 36,11:
? "2":GOSUB 125:GOTO 10
[illegible] THEN POSITION 36,13:
? "3":GOSUB 125:GOTO 500
401 U=0:L=0:J=0:H=0:POSITION 33,
7:? "1":NR=49:GOSUB 130:GOSUB 20
0:GOTO 600
402 U=0:L=0:J=0:H=0:POSITION 33,
9:? "2":NR=50:GOSUB 130:GOSUB 20
0:GOTO 650
[illegible] POSITION 33
,13:? "3":NR=51:GOSUB 130:GOSUB
200:GOTO 750
404 U=0:IL=0:J=0:H=0:POSITION 33
,11:? "4":NR=52:GOSUB 130:GOSUB
[illegible]
405 U=0:IL=0:J=0:H=0:POSITION 33
,15:? "5":NR=53:GOSUB 130:GOSUB
200:GOTO 800
[illegible]:GOTO 490:REM ZMIANA
[illegible] DLA [illegible]
[illegible] REM ZMIANA [illegible] DLA CYFR
472 IF TE=48 THEN K=K+6:GOTO 490
[illegible] IF TE=49 THEN [illegible]:GOTO 4
[illegible]
474 IF TE=50 THEN K=K+3.2:GOTO 4
90
[illegible] IF TE=51 THEN [illegible]:GOTO 4
90
[illegible] IF TE=52 THEN K=K+2.7:GOTO 4
90
477 IF TE=53 THEN K=K+2.2:GOTO 4
90
478 IF TE=54 THEN K=K+2.15:GOTO
[illegible]
479 IF TE=55 THEN K=K+1.66:GOTO
[illegible]
[illegible] IF TE=56 THEN K=K+1:GOTO 490
481 IF TE=57 THEN K=K+0.8:GOTO 4
90
[illegible] GOSUB 65:GOSUB 82:
[illegible]
[illegible](PEEK(18)*256*PEEK
(19)+PEEK(20))/50:? :? " CZAS NA
D. ";SEK;" SEK."
493 ? :? "TEMPO NAD. ";INT(IL/(S
EK/60)+0.5);" GRUP/MIN."
[illegible] LY=4:LG=0
[illegible] GOSUB 130:GOSUB [illegible]:GOSUB 13
[illegible] GOSUB 130:GOTO [illegible] REM [illegible] D
```

```
0 PODPR. WYBIERANIA Z ,IL,
500 POKE 731,1 REM WYBOR TEKSTU
501 POSITION 5,1:? "ILE GRUP :";
:INPUT IL
502 POSITION 27,1:? "ILE TLG";:I
NPUT VB1:IF IL=0 OR VB1=0 THEN ?
 CHR$(125):GOTO 500
503 IF IL>100 OR VB1>100 THEN ?
CHR$(125):GOTO 500:T=LEN(IL$)
504 IL$=STR$(IL)
505 T=LEN(IL$)
506 ? CHR$(125):POSITION 14,4:?
"WYBOR TEKSTU"
507 POSITION 2,7:? "TEKST LITERO
WY (lp+lr)          1"
508 POSITION 2,9:? "TEKST MIESZA
NY (lp+cyfry)       2"
509 POSITION 2,11:? "TEKST CYFRO
WY                  3"
510 POSITION 2,13:? "TEKST LITER
OWY (lp)            4"
511 POSITION 2,15:? "TEKST MIESZ
ANY (lp+lr+cyfrY)   5"
512 POSITION 9,19:? "POWROT DO S
PISU   [OPTION]"
532 L=0:J=0:H=0:U=0
535 CLOSE #1:OPEN #1,4,0,"K:":GE
T #1,A
540 IF A>=49 AND A<=53 THEN 544
543 GOTO 535
544 IF A=49 THEN POSITION 33,7:?
 "1":NR=49:GOSUB 130:GOSUB 200:G
OTO 600
545 IF A=50 THEN POSITION 33,9:?
 "2":NR=50:GOSUB 130:GOSUB 200:G
OTO 650
546 IF A=51 THEN POSITION 33,11:
? "3":NR=51:GOSUB 130:GOSUB 200:
GOTO 750
547 IF A=52 THEN POSITION 33,13:
? "4":NR=52:GOSUB 130:GOSUB 200:
GOTO 700
548 IF A=53 THEN POSITION 33,15:
? "5":NR=53:GOSUB 130:GOSUB 200:
GOTO 800
600 ? :REM LOS. LP.+LR.
605 Z=65+INT(RND(3)*30)
606 IF Z>=65 AND Z<=94 THEN GOSU
B Z
610 L=L+1:J=J+1:H=H+1:U=U+1:LX=L
X+1
612 IF L=5 THEN ? " ";:L=0:GOSUB
 1300:GOSUB 123:LX=LX+1:POSITION
 LX+2,LY
614 IF J=25 THEN ? "          ":J=0
:LX=0:LY=LY+1
616 IF H=50 THEN GOSUB 123:GOSUB
 123:H=0
617 IF U=IL*5 THEN GOTO 490
620 GOTO 605
650 ? :REM LOS. MIESZANY LP.+CYF
RA
652 K=K-0.5
655 Z=50+INT(RND(3)*41)
660 IF Z=50 OR Z=53 OR Z=55 OR Z
=56 OR Z>=65 AND Z<=90 THEN GOSU
B Z:GOTO 670
665 GOTO 655
670 L=L+1:J=J+1:H=H+1:U=U+1:LX=L
X+1
675 IF L=5 THEN ? " ";:L=0:GOSUB
 1300:GOSUB 123:LX=LX+1:POSITION
 LX+2,LY
680 IF J=25 THEN ? "          ";:J
=0:LX=0:LY=LY+1
685 IF H=50 THEN GOSUB 123:GOSUB
 123:H=0
687 IF U=IL*5 THEN 450
688 IF PEEK(53279)=3 THEN 300
690 GOTO 655
700 ? :REM LOS. LIT.POL.
705 Z=65+INT(RND(1)*26)
706 GOSUB Z
710 L=L+1:J=J+1:H=H+1:U=U+1:LX=L
X+1:A(U)=Z
712 IF L=5 THEN ? " ";:L=0:GOSUB
 1300:GOSUB 123:LX=LX+1:POSITION
 LX+2,LY
714 IF J=25 THEN ? "          ";:J
=0:LX=0:LY=LY+1
716 IF H=50 THEN GOSUB 123:GOSUB
 123:H=0
717 IF U=IL*5 THEN GOTO 490
718 IF PEEK(53279)=3 THEN 300
720 GOTO 705
750 ? :REM LOS. CYFRY
752 IF TE=48 THEN K=K-6
753 IF TE=49 THEN K=K-3.9
754 IF TE=50 THEN K=K-3.2
755 IF TE=51 THEN K=K-3.2
756 IF TE=52 THEN K=K-2.7
757 IF TE=53 THEN K=K-2.2
758 IF TE=54 THEN K=K-2.15
759 IF TE=55 THEN K=K-1.66
760 IF TE=56 THEN K=K-1
761 IF TE=57 THEN K=K-0.8
762 IF L=5 THEN ? " ";:L=0:GOSUB
 1300:GOSUB 123:LX=LX+1:POSITION
 LX+2,LY
767 Z=48+INT(RND(1)*10)
768 GOSUB Z
770 L=L+1:J=J+1:H=H+1:U=U+1:LX=L
X+1
772 IF L=5 THEN ? " ";:GOSUB 123
:L=0:GOSUB 1300:LX=LX+1:POSITION
 LX+2,LY
774 IF J=25 THEN ? "          ";:J
=0:LX=0:LY=LY+1
776 IF H=50 THEN GOSUB 123:GOSUB
```

```
 123:H=0
777 IF U=IL*5 THEN 470
778 IF PEEK(53279)=3 THEN 300
779 GOTO 767
800 ? :REM LP+LR+CYFRY
805 Z=48+INT(RND(4)*47)
806 IF Z>=48 AND Z<=57 OR Z>=65
AND Z<=94 THEN GOSUB Z:GOTO 810
807 GOTO 805
810 L=L+1:J=J+1:H=H+1:U=U+1:LX=L
X+1
812 IF L=5 THEN ? " ";:L=0:GOSUB
 1300:GOSUB 123:LX=LX+1:POSITION
 LX+2,LY
814 IF J=25 THEN ? "           ";:J
=0:LX=0:LY=LY+1
816 IF H=50 THEN GOSUB 123:GOSUB
 123:H=0
817 IF U=IL*5 THEN GOTO 490
818 IF PEEK(53279)=3 THEN 300
820 GOTO 805
867 IF U=IL*5 THEN GOTO 490
900 POSITION 9,19:? "POWROT DO S
PISU  [?]":GOSUB 125
902 ? CHR$(125):POKE 731,1:REM N
ADAW. Z KLAWIATURY
905 CLOSE #1:OPEN #1,4,0,"K:":GE
T #1,A
910 IF A>=65 AND A<=94 OR A=32 O
R A>=48 AND A<=57 THEN GOSUB A
911 IF A=63 THEN 300
912 GOTO 905
914 RETURN
999 SAVE "D:MORSE1":STOP
1000 Z=48+INT(RND(1)*10):REM LOS
.0-9 DLA NAGL.
1010 Z=48+INT(RND(1)*4):REM LOS.
0-3
1012 RETURN
1020 Z=49+INT(RND(1)*9):REM LOS.
1-9
1022 RETURN
1030 Z=48+INT(RND(1)*2):REM LOS.
0,1
1032 RETURN
1040 Z=48+INT(RND(1)*3):REM LOS.
0,1,2
1042 RETURN
1050 Z=48+INT(RND(1)*6):REM LOS.
0-5
1052 RETURN
1060 Z=65+INT(RND(1)*26):REM LOS
. LITER ADR.
1062 RETURN
1070 Z=401+INT(RND(1)*3):REM LOS
. WYBOR TEKSTU
1072 IF Z=352+NR THEN 1070
1076 GOTO Z
1100 N=0:REM ODCZYT ILOSCI GRUP
Z INSTR. INPUT ;IL;
1110 C=0:B=0:A=0
1120 N=N+1:E$=IL$(N)
1125 GOSUB ASC(E$):IF T=N THEN 1
135
1130 GOTO 1120
1135 FOR N=1 TO T
1160 R$=IL$(N):GOSUB 1206+INT(T/
N)
1200 NEXT N
1201 GOTO 1250
1207 A=-48+ASC(R$):RETURN
1208 B=-48+ASC(R$):RETURN
1209 C=-48+ASC(R$):RETURN
1250 IL=100*C+10*B+A
1270 RETURN
1300 LG=LG+1:POSITION 9,1:? LG:R
ETURN
3000 CLOSE #1:OPEN #1,8,0,"P:":S
1=0
3010 FOR I=1 TO A:S1=S1+1
3020 ? #1;CHR$(A(I));:IF S1=69 T
HEN S1=0:? #1
3030 NEXT I
30000 REM ZMIANY GEN. ZNAKOW
30001 NRZ=1
30002 IF NRZ=0 THEN ? CHR$(125):
POSITION 7,10:? "REJESTR ZMIAN J
EST PUSTY":GOTO 30500
30003 ? CHR$(125):POSITION 4,5:?
 "WPROWADZENIE ZMIAN GEN. ZNAKOW
"
30004 POSITION 8,10:? "PROSZE CH
WILE POCZEKAC"
30005 N=57344:M=32768:LK=33
30006 FOR I=0 TO 1023
30007 POKE M+I,PEEK(N+I)
30008 NEXT I:POKE 756,128
30011 POKE 33071,6
30012 POKE 33248,1
30013 POKE 33249,0
30014 POKE 33250,0
30015 POKE 33251,6
30016 POKE 33252,0
30017 POKE 33253,1
30018 POKE 33254,0
30019 POKE 33255,255
30020 POKE 1,0
30021 POKE 2,219
30022 POKE 33248,0
30023 POKE 33249,219
30024 POKE 33250,219
30025 POKE 33251,219
30026 POKE 33252,219
30027 POKE 33248,64
30028 POKE 33249,96
30029 POKE 33250,48
30030 POKE 33251,24
30031 POKE 33252,12
30032 POKE 33253,6
30033 POKE 33254,0
```

```
30034 POKE 33255,0
30035 POKE 33248,0
30036 POKE 33249,73
30037 POKE 33250,73
30038 POKE 33251,73
30039 POKE 33252,73
30040 POKE 33253,73
30041 POKE 33254,127
30042 POKE 33265,57
30043 POKE 33266,69
30044 POKE 33267,69
30045 POKE 33268,71
30046 POKE 33269,69
30047 POKE 33270,57
30048 POKE 33257,30
30049 POKE 33258,34
30050 POKE 33259,34
30051 POKE 33260,30
30052 POKE 33261,34
30053 POKE 33262,66
30054 POKE 33257,28
30055 POKE 33241,126
30056 POKE 33242,98
30057 POKE 33243,12
30058 POKE 33244,48
30059 POKE 33245,96
30060 POKE 33245,32
30061 POKE 33244,16
30499 ? CHR$(125):POSITION 7,5
 "ZMIANY ZOSTALY WPROWADZONE"
30500 FOR T=1 TO 200 NEXT T GOTO
 150
```

* SUMA KONTROLNA / ETYKIETA *

EL	1	PH	2	CG	3	AZ	4
FK	5	US	[illegible]	BB	9	IT	10
OP	11	CH	12	MG	13	ZM	16
ZO	17	SU	19	KR	20	SH	22
MK	25	UL	26	RL	27	NK	28
VC	29	FZ	30	IH	31	BO	32
OI	45	BN	48	BP	49	ZO	50
YM	51	WG	52	VA	53	WL	54
XU	55	ZK	56	RO	57	AB	61
HP	65	OF	66	PK	67	JT	68
NV	69	[illegible]	70	LE	71	PH	72
[illegible]	73	UM	74	MY	75	PU	76
MW	77	ML	78	PL	79	TY	80
VI	81	NN	82	MT	83	SH	84
OT	85	PH	86	UT	87	WS	88
TO	89	IO	90	V[illegible]	91	RD	92
WV	93	LM	94	LS	121	TT	122
BW	123	HG	124	IF	125	IS	130
DP	142	WC	149	LT	150	GM	151
SZ	152	FC	153	ZS	154	PS	155
TY	156	HU	157	PF	158	IV	159
LM	160	UF	161	UT	162	IX	163
FB	200	IH	202	PD	203	EK	205
AL	206	MR	207	FP	210	WH	211
XT	215	WH	216	LO	219	DB	220
AN	240	MH	241	YY	242	XZ	243
AH	244	HD	245	AG	246	HJ	247
EL	248	VU	249	EF	250	MZ	251
NI	254	UL	255	LN	256	BY	257
JL	258	ST	259	FI	260	JP	261
NY	262	TS	[illegible]	JN	[illegible]	FG	[illegible]
JM	266	[illegible]	[illegible]	FF	[illegible]	[illegible]	[illegible]
KZ	270	FI	271	ZU	272	NP	273
PT	274	JL	275	IF	276	MH	277
FL	278	XH	279	TF	300	FS	301
FE	303	IC	304	EU	310	KJ	320
SF	330	FG	335	SM	350	UD	360
DH	370	MK	371	YK	372	SD	401
WK	402	XT	403	KC	404	VU	405
CY	450	GP	470	PM	472	XT	473
NK	474	NZ	475	SM	476	OG	477
KK	478	PX	479	KK	480	TM	481
YA	490	EF	492	YF	493	AL	494
KI	496	UB	500	JN	501	OU	502
CW	503	FL	504	DD	505	LF	506
YH	507	ON	508	VP	509	EN	510
OA	511	OO	512	VK	532	FI	535
XS	540	PE	543	YJ	544	ZF	545
SY	546	LG	547	TX	548	[illegible]	600
OP	605	ZU	606	NG	610	SN	612
FM	614	MU	616	[illegible]	[illegible]	[illegible]	[illegible]
PN	650	GD	652	OV	[illegible]	[illegible]	[illegible]
SV	665	NS	670	TJ	[illegible]	NH	[illegible]
NF	685	TU	687	EL	[illegible]	SM	[illegible]
TI	700	PN	705	UP	706	HS	[illegible]
SU	712	MZ	714	MV	[illegible]	DU	[illegible]
HY	718	UG	720	DL	750	IH	[illegible]
KM	[illegible]	HD	754	AS	[illegible]	FF	[illegible]
AZ	757	BF	758	HL	759	HF	[illegible]
GF	761	ST	762	MS	[illegible]	PH	[illegible]
NT	[illegible]	KE	[illegible]	NL	[illegible]	NH	[illegible]
VI	777	BK	[illegible]	[illegible]	[illegible]	VE	[illegible]
WM	805	OB	806	PJ	[illegible]	[illegible]	[illegible]
SP	812	NA	814	[illegible]	[illegible]	[illegible]	[illegible]
AJ	818	OP	820	EA	[illegible]	[illegible]	[illegible]
LH	902	FG	905	PI	[illegible]	[illegible]	[illegible]
RG	912	NH	914	HK	[illegible]	NK	1000
ZB	1010	PE	1012	NL	1020	PH	[illegible]
TP	1030	FK	[illegible]	UU	1040	NH	[illegible]
DZ	1050	PU	1052	[illegible]	[illegible]	FI	[illegible]
UW	1070	YZ	[illegible]	LU	[illegible]	FF	[illegible]
HR	1110	LA	1120	[illegible]	[illegible]	OL	[illegible]
VL	1135	EH	[illegible]	GU	1200	QH	[illegible]
UL	1207	UM	[illegible]	[illegible]	[illegible]	JL	[illegible]
PS	[illegible]	SN	[illegible]	EH	[illegible]	EU	[illegible]
LU	[illegible]	FA	[illegible]	FY	[illegible]	HL	[illegible]
TC	30002	OG	[illegible]	FF	30004	LH	[illegible]
TD	30006	XF	[illegible]	XF	30008	JT	[illegible]
MT	30012	MW	30013	IL	30014	NE	30015
JS	30016	LF	30017	LI	30018	PA	30019
IW	30020	WY	30021	MF	30022	QB	[illegible]
LH	30024	MC	30025	MX	30026	DI	[illegible]
HR	30028	AX	30029	VO	30030	UH	[illegible]
OH	30032	FR	30033	LM	30034	MY	30035
DW	30036	ZL	30037	ZX	30038	HS	30039
ZT	30040	LZ	30041	LM	30042	GL	3004[illegible]
HG	30044	CY	30045	IW	30046	B[illegible]	30047
YU	30048	CN	30049	BU	30050	TJ	[illegible]
XP	[illegible]	BY	30053	DJ	30054	[illegible]	[illegible]
EX	30056	VU	[illegible]	DE	[illegible]	FW	[illegible]
XF	30060	YI	30061	CD	30499	KM	30500

Jan BARSZCZEWSKI

OILS WELL

Gdy przegladam czasami album ze starymi fotografiami, wzrok mój zatrzymuje sie na pożółkłym zdjeciu przedstawiajacym pracujacych mezczyzn. Jednym z nich jest mój pradziadek. Za nim stoi wieza do wiercen geologicznych. Z opowiesci dziadków wynika, że pradziadek, wierzac w swoja szczesliwa gwiazde, ruszył w swiat w poszukiwaniu pieniedzy. Wiodło mu sie róznie. Jako farmer zarobił troche pieniedzy. Stracił je, gdy zajał sie poszukiwaniem złota. Zmieniał zawód jeszcze pare razy, aż zajał sie poszukiwaniem i wydobyciem ropy naftowej. Dzieki temu dożył spokojnej i dostatniej starosci. Nie był jednak przewidujacy, pozostawiajac liczne grono spadkobierców, którzy po jego smierci energicznie wzieli sie do wydawania zgromadzonego kapitału. Nie trzeba było długo czekac na to, by po fortunie pozostały tylko wspomnienia oraz kilka pożółkłych fotografii. Historie te przytaczam nie po to, by zniechecic do poszukiwania przygód, a wrecz przeciwnie. Mój czteroletni syn również wydobywa rope naftowa, robi to w programie pt.: " OILS WELL ". Nie jest on skomplikowany graficznie ani pod wzgledem fabuły, dlatego też nie powinien wywoływac stresów.

Po uruchomieniu otrzymasz menu. Wynika z niego, że do uruchomienia gry można użyc dwóch klawiszy;

OPTION - w celu wybrania stopnia trudnosci;

START - rozpoczecia gry.

Stopien trudnosci (CHANGE DIFFICULTY), dzieli sie na trzy poziomy

REGULAR - to poziom sredni, który jest ustawiany przez komputer po uruchomieniu sie programu,

UNLEADED i PREMIUM - sa odpowiednio trudniejszymi. Stworki, poruszajace sie w wydrażonych otworach sa bardziej energiczne i jest ich wiecej.



Sam szyb naftowy jest niezwykły, jak niezwykłe potrafia byc programy komputerowe. Składa sie z szesciu poziomów. Im głebiej, tym wiecej można uzyskac baryłek ropy naftowej. Za baryłke otrzymasz 10 punktów. Za pochwycenie stworka otrzymasz również premie, ale jej wysokosc zależy od poziomu, na którym został złapany. I tak liczac od góry, punktacja wyglada nastepujaco: 20, 50, 80, 110, 140, 170 punktów. Rodzaj stworka jest nieistotny. Sa one bardzo niebezpieczne. W przypadku gdyby napotkały na swojej drodze ramie urzadzenia, którym wydobywana jest ropa, spowoduja jego uszkodzenie. Rozpoczyna sie odliczanie czasu od poczatku z jednoczesna utrata jednego urzadzenia. Tych urzadzen w sumie masz trzy. Łatwo je możesz stracic, gorzej ze zdobyciem nowych. Urzadzenie takie otrzymasz jako premie gdy zdobedziesz 10000 punktów. Ułatwic to może pojawiajacy sie co pewien czas pulsujacy kielich. Po chwyceniu go do wyniku dodawane jest 1000 punktów premii. Najwieksza jednak premia jest otrzymanie nastepnego szybu do eksploatacji. Stanie sie tak, gdy zostana zebrane wszystkie baryłki przy posiadaniu jednoczesnie co najmniej jednego urzadzenia wydobywczego. Na jednym z najniższych poziomów szybu znajduje sie pulsujaca baryłka. Warto ja wziac, gdyż w ten sposób na pewien czas nastepuje spowolnienie poruszania sie stworków. Umożliwi to zebranie baryłek na najniższych poziomach. Od czasu do czasu bedzie sie pojawiał pulsujacy obiekt, prawdopodobnie podłożony przez konkurencje. Ramienia urzadzenia wydobywczego nie uszkodzi, ale może zrobic to z jego uchwytem, eliminujac urzadzenie z eksploatacji.

Jak każde urzadzenie tak i to ma swój okres używalnosci. Wytrzymuje ono 1000 jednostek pracy, tj. ok. 100 sekund, po czym nastepuje awaria. Jest to nieuchronne, chyba że szybciej zbierzesz wszystkie baryłki, aniżeli zużyje sie urzadzenie. Wtedy dodatkowa premia jest regeneracja i czas rusza od poczatku.

Życze Wam zostania królami nafty.



INFORMATYKA
KOMPUTERY
SYSTEMY

Dodatek „Żołnierza Wolności" Cena 200 zł

zeszyty programów komputerowych

nr 4 /88

JĘZYK 'C'

CZĘŚĆ 2



DEEP BLUE C

Z. Dyjak
H. Krasuski

Kompilator DEEP BLUE C nie posiada zaimplementowanych liczb zmiennoprzecinkowych (dane typu float w standardowym C), co uniemożliwia, między innymi, wykorzystanie tego języka w poważniejszych obliczeniach matematyczno - technicznych czy w poważnych zastosowaniach graficznych, które są jedną z mocniejszych stron "małych ATARI". Niedostatek ten wypełniają biblioteki funkcji matematycznych i trygonometrycznych (MATHLIB i TRIG). Niniejszy zeszyt szczegółowo je przedstawia. Zamieszczone w końcowej części dwa programy: SPIRALE i OBLICZ ilustrują sposób posługiwania się funkcjami z tych bibliotek. Pokazują one również sposób łączenia procedur w kodzie maszynowym z programem w języku C oraz wykorzystanie jednego z ciekawszych mechanizmów tego języka - funkcji rekurencyjnych. Stanowią ponadto przykłady tworzenia stosunkowo dużych i skomplikowanych programów w "C".

SPIS TRESCI

DODATEK B

B1. Wstep.................str. 3
B2. Spis funkcji zawartych w bibliotece MATHLIB...str. 3
B3. Pliki dolaczone do biblioteki MATHLIB.........str. 4
B4. Wykorzystanie biblioteki matematycznej........str. 4
B4.1. Inicjalizacja biblioteki matematycznej.......str. 4
B4.2. Deklaracja zmiennych zmiennoprzecinkowych.....str. 4
B4.3. Tworzenie stalych zmiennoprzecinkowych.......str. 5
B4.4. Drukowanie liczby zmiennoprzecinkowej........str. 5
B4.5.1. Konwersja liczby calkowitej na liczbe zmiennoprzecinkowa..............str. 5
B4.5.2. Forma opisu funkcji z biblioteki MATHLIB......str. 6
B4.6. Funkcje z biblioteki MATHLIB..................str. 6
B4.6.1. Inicjalizacja biblioteki.....................str. 7
B4.6.2. Kowersja kodu ATASCII na liczbe zmiennoprzecinkowa....................str. 7
B4.6.3. Konwersja liczby zmiennoprzecinkowej na liczbe w kodzie ATASCII......str. 7
B4.6.4. Zamiana liczby calkowitej bez znaku na liczbe zmiennoprzecinkowa.......str. 8
B4.6.5. Zamiana liczby calkowitej ze znakiem na liczbe zmiennoprzecinkowa.......str. 8
B4.6.6. Zamiana liczby zmiennoprzecinkowej na liczbe calkowita bez znaku........str. 9
B4.6.7. Zamiana liczby zmiennoprzecinkowej na liczbe calkowita ze znakiem.......str. 9
B4.6.8. Dodawanie liczb zmiennoprzecinkowych.......str. 9
B4.6.9. Odejmowanie liczb zmiennoprzecinkowych......str. 10
B4.6.10. Dzielenie liczb zmiennoprzecinkowych......str. 10
B4.6.11. Mnozenie liczb zmiennoprzecinkowych..........str. 11
B4.6.12. Logarytm liczby zmiennoprzecinkowej........str. 11
B4.6.13. Antylogarytm liczby zmiennoprzecinkowej.......str. 12
B4.6.14. Funkcja wykladnicza liczby zmiennoprzecinkowej.str. 12
B4.6.15. Pierwiastek kwadratowy z liczby zmiennoprzecinkowej.....................str. 13
B4.6.16. Czesc calkowita liczby zmiennoprzecinkowej.str. 13
B4.6.17. Czesc ulamkowa liczby zmiennoprzecinkowej.str. 14
B4.6.18. Porownanie dwoch liczb zmiennoprzecinkowych..........................str. 14
B4.6.19. Obliczanie modulu liczby zmiennoprzecinkowej.str. 14
B4.6.20. Zmiana znaku liczby zmiennoprzecinkowej......str. 15
B4.6.21. Ustawienie liczby zmiennoprzecinkowej na zero..........................str. 15
B4.6.22. Przesuniecie liczby zmiennoprzecinkowej......str. 15
B4.7. Funkcje trygonometryczne zawarte w pliku TRIG.CCC....................str. 16
B4.7.1. Inicjalizacja funkcji trygonometrycznych.....str. 16
B4.7.2. Ustawienie trybu wykonywania obliczen w stopniach lub radianach......str. 16
B4.7.3. Zamiana radianow na stopnie..................str. 16
B4.7.4. Zamiana stopni na radiany....................str. 17
B4.7.5. Zamiana stopni, minut, sekund na stopnie dziesietne...............str. 17
B4.7.6. Zamiana stopni dziesietnych na stopnie, minuty, sekundy..........str. 18
B4.7.7. Obliczanie wartosci funkcji sinus..........str. 18
B4.7.8. Obliczanie wartosci funkcji cosinus.........str. 18
B4.7.9. Obliczanie funkcji tangens.................str. 19
B4.7.10. Obliczanie wartosci funkcji arctangens......str. 19

PROGRAMY.

Spirale...............................str. 20
Oblicz................................str. 24

OPISY GIER KOMPUTEROWYCH.

Aztec.................................str. 28
Dimension X...........................str. 29
Spy vs Spy............................str. 31



B1. WSTĘP

MATHLIB jest biblioteką tych funkcji języka C, które zapewniają realizację obliczeń zmiennoprzecinkowych z wykorzystaniem kompilatora DEEP BLUE C. Język DEEP BLUE C nie posiada zmiennoprzecinkowych typów danych. Uniemożliwia to, między innymi, wykorzystanie tego języka w poważnych zastosowaniach graficznych, które są mocną stroną komputerów Atari. Biblioteka MATHLIB wypełnia ten niedostatek DEEP BLUE C.

Wykorzystywanie funkcji biblioteki MATHLIB nie daje takiej zwięzłości jak używanie normalnych operatorów arytmetycznych, ale zapewnia pełny zakres możliwości obliczeniowych.

Program demonstracyjny, dołączony do biblioteki MATHLIB, ilustruje jak przy pomocy jej funkcji można zaimplementować grafikę żółwia.

Aby móc korzystać z funkcji biblioteki MATHLIB należy posiadać:

-kompilator DEEP BLUE C
-edytor ekranowy
-makroasembler Atari

Funkcje biblioteki MATHLIB działają na standardowych liczbach zmiennoprzecinkowych w formacie Atari i zapewniają dostęp do procedur zawartych w pamięci ROM. Dodatkowo dołączono funkcje trygonometryczne, stanowiące znaczne rozszerzenie możliwości procedur zawartych w ROM. W sumie MATHLIB zapewnia dodatkowe 32 funkcje.



B2. SPIS FUNKCJI ZAWARTYCH W BIBLIOTECE MATHLIB

MATHLIB zapewnia realizację następujących funkcji:

- konwersja liczby podanej w kodach ATASCII na liczbę zmiennoprzecinkową i liczby zmiennoprzecinkowej na kody ATASCII;
- konwersja liczby całkowitej, bez lub ze znakiem, na liczbę zmiennoprzecinkową i liczby zmiennoprzecinkowej na liczbę całkowitą;
- dodawanie, odejmowanie, mnożenie i dzielenie liczb zmiennoprzecinkowych;
- pierwiastek kwadratowy;
- funkcje trygonometryczne (sinus, cosinus, tangens i arctangens) dla kątów podanych w stopniach i radianach;
- konwersja radianów na stopnie i stopni na radiany;
- zamianę stopni podanych w systemie dzesiętnym na stopnie, minuty, sekundy i na odwrót;
- wyodrębnianie części całkowitej i ułamkowej liczby zmiennoprzecinkowej;
- porównanie dwóch liczb zmiennoprzecinkowych;
- obliczanie modułu liczby zmiennoprzecinkowej;
- zmianę znaku liczby zmiennoprzecinkowej;
- nadanie liczbie zmiennoprzecinkowej wartości 0;
- przenoszenie wartości liczby zmiennoprzecinkowej z jednego miejsca pamięci w inne.

Zakładamv, że posiadasz kompilator DEEP BLUE C. Oto pliki DEEP BLUE C, które są wymagane podczas pracv z MATHLIB (nie są one dołączone do MATHLIB):

- CC.COM - kompilator DEEP BLUE C;
- CLINK.COM - konsolidator DEEP BLUE C;
- DBC.OBJ - moduł uruchomieniowy DEEP BLUE C;
- AIO.CCC - kod wynikowy funkcji we/wy.

Aby wygenerować wykonywalną postać programu demonstrującego grafikę żółwia (znajduje się na dyskietce zawierającej MATHLIB) należy dołączyć następujące pliki DEEP BLUE C (nie są dołączone do MATHLIB):

- GRAPHICS.CCC - kod wynikowy biblioteki funkcji graficznych;
- PRINTF.CCC - kod wynikowy funkcji wyprowadzania sformatowanego.



B3. PLIKI DOŁĄCZONE DO BIBLIOTEKI MATHLIB

Na dyskietce z biblioteką matematyczną znajdują się następujące pliki:

- MATHLIB.C – kod źródłowy funkcji matematycznych MATHLIB;
- MATHLIB.CCC – kod wynikowy dla MATHLIB.C, musi być zawsze wymieniony w pliku sterującym konsolidacją;
- TRIG.C – kod źródłowy funkcji trygonometrycznych;
- TRIG.CCC – kod wynikowy dla TRIG.C, musi być wymieniony w pliku sterującym konsolidacją, gdy korzystamy z funkcji trygonometrycznych;
- MATHLIB.OBJ – kod wynikowy dla napisanego w asemblerze interfejsu do procedur operacji zmiennoprzecinkowych zawartych w pamięci ROM. Musi być zawsze wymieniony w pliku sterującym konsolidacją. Plik MATHLIB.OBJ ładuje się do pamięci RAM w obszar od $2dco do $2fff. Jeżeli wykorzystujesz jednocześnie swoje własne procedury napisane w asemblerze, to upewnij się czy ich położenie nie koliduje z lokalizacją MATHLIB.OBJ. Możesz także, przy pomocy makroasemblera Atari, zmienić lokalizację MATHLIB.OBJ. W tym przypadku należy również zmienić adresy punktów wejściowych do MATHLIB.OBJ znajdujących się w MATHLIB.C i ponownie skompilować ten plik.
- MATHLIB.ASM – kod źródłowy w asemblerze dla MATHLIB.OBJ (wymaga użycia makroasemblera CX8121);
- TURTLE.C – kod źródłowy programu demonstracyjnego. Wykorzystuje funkcje matematyczne do realizacji podstawowych ruchów grafiki żółwia.
- TURTLE.CCC – kod wynikowy dla TURTLE.C;
- CLIPPER.C – kod źródłowy algorytmu Cohena-Southerlanda, adaptowanego dla języka DEEP BLUE C. Zawiera funkcje wywoływane przez TURTLE.
- CLIPPER.CCC – kod wynikowy dla CLIPPER.C;
- TURTLE.LNK – plik sterujący konsolidacją programu demonstracyjnego;
- TURTLE.COM – kod wykonywalny programu demonstrującego grafikę żółwia.



B4. WYKORZYSTANIE BIBLIOTEKI MATEMATYCZNEJ

W tym rozdziale przedstawiamy sposób wykorzystania biblioteki matematycznej. Pokażemy jak ją inicjować, jak deklarować zmienne zmiennoprzecinkowe, jak zamieniać liczby całkowite na zmiennoprzecinkowe i odwrotnie. Wszystkie nazwy funkcji zawartych w bibliotece zaczynają się przedrostkiem "c_" aby uniknąć konfliktów z nazwami użytkownika.

B4.1 Inicjalizacja biblioteki matematycznej.

Przed wykorzystaniem jakiejkolwiek funkcji z biblioteki matematycznej należy wywołać funkcję *c_iml*. Jeżeli wykorzystujemy funkcje trygonometryczne należy wywołać także *c_trig*. Funkcje te inicjują stałe i zmienne, których używają funkcje MATHLIB podczas realizacji obliczeń. Żadna z tych funkcji nie ma parametrów i są one wywoływane w sposób następujący:

```
        c_iml();
        c_trig();
```

Funkcja *c_trig* może być wywoływana tylko wówczas, gdy wykorzystywane są funkcje trygonometryczne, a plik TRIG.CCC został włączony do programu. Funkcja *c_iml* musi zostać wywołana przed *c_trig*.

B4.2 Deklaracja zmiennych zmiennoprzecinkowych.

Biblioteka MATHLIB wykorzystuje liczby zmiennoprzecinkowe przedstawione w standardowym formacie Atari. Aby móc wykorzystywać funkcje MATHLIB nie musisz znać formatu liczb zmiennoprzecinkowych w komputerze Atari i nie będziemy go tutaj omawiać. Jeżeli Cię to interesuje, to kierujemy do książki *"De Re Atari"*, rozdział 8-45 i 8-46. Jedyne co należy wiedzieć to fakt, że liczby zmiennoprzecinkowe w Atari zajmują 6 bajtów. Funkcje biblioteki MATHLIB wykorzystują do przechowywania liczb zmiennoprzecinkowych tablice znakowe. Aby

zadeklarować zmienną, która będzie służyć do przechowywania liczby zmiennoprzecinkowej, napisz (zakładając, że nazwą tej zmiennej będzie *fpvar*):

```
char fpvar[6];
```

Wszystkie zmienne zmiennoprzecinkowe wykorzystywane prze funkcje biblioteki MATHLIB muszą być zadeklarowane w podany sposób, a wszystkie argumenty funkcji dotyczące liczb zmiennoprzecinkowych są wskaźnikami znakowymi. Oznacza to, że gdy w jakiejś funkcji z MATHLIB odwołujesz się do liczby zmiennoprzecinkowej używaj tylko nazwy zmiennej, bez indeksu. Na przykład jeżeli chcesz dodać dwie liczby zmiennoprzecinkowe *fa* i *fb* oraz chcesz aby wynik został zapamiętany w *fb*, to zmienne te powinny zostać zadeklarowane w sposób następujący:

```
char fa[6], fb[6];
```

Funkcja biblioteczna, która dodaje dwie liczby zmiennoprzecinkowe nazywa się *c_fadd()*. Wymaga podania trzech argumentów: dwie dodawane liczby i wynik.

Tak więc aby dodać dwie liczby w naszym przykładzie należy napisać:

```
c_fadd(fa, fb, fb);
```

Zwróć uwagę, że podajemy tylko nazwy zmiennych, bez indeksów.

B4.3 Tworzenie stałych zmiennoprzecinkowych.

DEEP BLUE C nie zapewnia standardowego typu danych *float*, a więc niemożliwe jest zadeklarowanie w tym języku stałej zmiennoprzecinkowej. Z tego też powodu stałe nie mogą być wprost przekazywane jako argumenty funkcji MATHLIB. Stała taka musi zostać najpierw stworzona przez odpowiednią funkcję biblioteczną, która faktycznie wpisuje ją do sześciobajtowej tablicy znakowej zadeklarowanej w programie użytkowym.

Stała zmiennoprzecinkowa jest początkowo łańcuchem znaków ATASCII, który następnie w jednym kroku zamieniany jest na standardową reprezentację zmiennoprzecinkową Atari. Zadanie to realizuje funkcja biblioteczna *c_afp()* (ATASCII to floating point). Funkcja *c_afp()* wykorzystuje jako argumenty dwa wskaźniki do tablic znakowych. Pierwszy wskazuje na łańcuch znaków ATASCII reprezentujący stałą, a drugi na sześciobajtową tablicę, która zawierać będzie zmiennoprzecinkową reprezentację tej stałej. Przypuśćmy, że chcemy stworzyć zmiennoprzecinkową reprezentację stałej "*pi*" (3.14159265) i zapamiętać ją w tablicy *fppi*. Należy napisać:

```
char fpi[6], *pntr;
pntr = "3.14159265";
c_afp( pntr, fppi);
```

W tym przykładzie *pntr* jest wskaźnikiem do łańcucha znaków ATASCII, a funkcja *c_afp()* przekształca go na reprezentację zmiennoprzecinkową, zapamiętaną następnie w sześciobajtowej tablicy znakowej *fppi*. *fppi* może być wykorzystywana w wywołaniach innych funkcji bibliotecznych i służyć do przekazywania wartości stałej "*pi*".

B4.4 Drukowanie liczby zmiennoprzecinkowej.

Wymiar tablicy dla reprezentacji liczby w kodzie ATASCII powienien być określony. Liczby zmiennoprzecinkowe w komputerze Atari zapewniają dziesięciocyfrową dokładność obliczeń. Dodatkowo liczba może być poprzedzona znakiem minus, zawierać kropkę, a także dwucyfrowy wykładnik w formie "E-xx". Na końcu liczby zmiennoprzecinkowej w kodzie ATASCII będzie występował znak *null* (standardowe zakończenie łańcucha w języku C). Dodając wszystko otrzymujemy 17 znaków i tak długa powinna być tablica znakowa służąca do przechowywania liczby zmiennoprzecinkowej w kodzie ATASCII.

Liczba zmiennoprzecinkowa w formacie Atari nie może być drukowana bezpośrednio. Musi zostać najpierw przekształcona na łańcuch znaków ATASCII. Tę czynność wykonuje funkcja biblioteczna *c_fasc()*. Posiada ona dwa argumenty będące wskaźnikami do łańcuchów znakowych. Pierwszy wskazuje na liczbę zmiennoprzecinkową a drugi na tablicę znakową, do której zostanie przekazana reprezentacja tej liczby w kodzie ATASCII. Tablica znakowa zawierająca przekształconą liczbę może być później wykorzystywana jako argument funkcji *printf()*.

Załóżmy na przykład, że mamy sześciobajtową liczbę w tablicy znakowej *fpnbr* i chcemy ją wydrukować wykorzystując funkcję *printf()*. Można tego dokonać w sposób następujący:

```
char fpnbr[6]; /* liczba zmiennoprzecinkowa */
char arfpnbr[17]; /* reprezentacja liczby w
ATASCII */

c_fasc( fpnbr, arfpnbr); /* przekształcenie
liczby zmiennoprzecinkowej na kod ATASCII */
printf("%s", arfpnbr); /* wydruk reprezentacji
w kodzie ATASCII */
```

B4.5.1 Konwersja liczby całkowitej na liczbę zmiennoprzecinkową.

Biblioteka MATHLIB zawiera także funkcje zapewniające konwersję liczb całkowitych na zmiennoprzecinkowe i na odwrót. Wyróżnia się dwa rodzaje liczb całkowitych: 16-bitowe bez znaku, z przedziału od 0 do 65535, i liczby ze znakiem

z przedziału od -32768 do +32767. Liczby bez znaku w DEEP BLUE C nie występują (liczba bez znaku większa niż 32767 traktowana jest jako ujemna). Jednakże jeżeli wiesz, że liczba zmiennoprzecinkowa nie jest ujemna to lepiej wykorzystywać funkcje biblioteki MATHLIB gdyż procedury ROM działają bezpośrednio na liczbach całkowitych bez znaku. Liczby całkowite ze znakiem wymagają dodatkowego przetwarzania.

Jeżeli wywołasz funkcję zamiany liczby zmiennoprzecinkowej bez znaku na liczbę całkowitą i liczba zmiennoprzecinkowa jest większa od 65535, to funkcja sygnalizuje błąd obliczeń. Taki sam błąd da próba zamiany liczby zmiennoprzecinkowej ze znakiem, większej niż 32767. Pamiętaj, że liczba całkowita bez znaku większa niż 32767 będzie traktowana przez DEEP BLUE C jak liczba ujemna.

Istnieją cztery procedury konwersji:

ifp() — konwersja liczby całkowitej bez znaku na liczbę zmiennoprzecinkową;
sifp() — konwersja liczby całkowitej ze znakiem na liczbę zmiennoprzecinkową;
c_fpi() — konwersja liczby zmiennoprzecinkowej bez znaku na liczbę całkowitą;
c_sfpi() — konwersja liczby zmiennoprzecinkowej ze znakiem na liczbę całkowitą.

B4.5.2 Forma opisu funkcji z biblioteki MATHLIB.

Kolejny rozdział zawiera opisy funkcji biblioteki MATHLIB. Każdy opis składa się z siedmiu części.

Zastosowanie.
Jedno lub dwa zdania precyzujące zastosowanie funkcji.

Wywołanie funkcji.
Przykład wywołania funkcji. Zdefiniowanie wszystkich parametrów jakie powinny zostać wprowadzone przed wywołaniem funkcji. Wyjątek stanowi *status* będący wartością zwracaną przez funkcję. Wartość zwracana określa czy dana operacja została wykonana poprawnie. Główną przyczyną błędu może być przekroczenie zakresu, tzn. wynikiem działania funkcji może być liczba przekraczająca standardowy zakres liczby zmiennoprzecinkowej w formacie Atari (od 10^{-99} do 10^{+98}). Jeżeli mamy pewność, że wynik będzie mieścił się w tym przedziale to nie musimy kontrolować wartości *status* po wywołaniu funkcji.

Parametry wejściowe.
Opisujemy tutaj wszystkie parametry przekazywane funkcji z programu wywołującego. Pierwszy wiersz każdego opisu zawiera nazwę parametru i jego typ. Potem następuje werbalny opis parametru.

Parametry wyjściowe.
Opisujemy tutaj wszystkie parametry przekazywane z funkcji do programu wywołującego.

Opis.
Przedstawiany jest tutaj opis czynności, jakie realizuje dana funkcja w odniesieniu do parametrów wejściowych i wyjściowych.

Wykorzystane funkcje.
W tym miejscu podajemy listę funkcji wykorzystanych do implementacji funkcji opisywanej. Mogą to być inne funkcje z biblioteki MATHLIB lub z biblioteki AIO. Jeżeli nie wykorzystywano innych funkcji oprócz standardowych konstrukcji języka, wystąpi w tym miejscu napis "Brak".

Przykład.
W tej części prezentujemy opcjonalnie jeden ze sposobów wykorzystania danej funkcji. Jeżeli opis funkcji jest bardzo czytelny tę część pomijamy. Przykłady zwykle rozpoczynają się od przedstawienia łańcuchów znaków ATASCII reprezentujących liczby zmiennoprzecinkowe, następnie obejmują przekształcenie ich na liczby zmiennoprzecinkowe w standardzie Atari, wywołanie funkcji, przekształcenie wyniku na kod ATASCII i jego wydruk przy pomocy funkcji

printf()

Jeżeli nazwa parametru używana jest w komentarzu, to pisana jest pismem pochyłym. Jeżeli parametr jest wskaźnikiem i brana jest pod uwagę wskazywana wielkość, to nazwę zmiennej poprzedza gwiazdka.

B4.6 Funkcje z biblioteki MATHLIB.

Funkcje opisane w tym rozdziale znajdują się w pliku MATHLIB.CCC. Wymieniamy je w kolejności ich występowania w dalszej części tego rozdziału.

c_ini — inicjalizacja biblioteki MATHLIB.CCC;
c_afp — konwersja z ATASCII na zmienny przecinek;
c_fasc — konwersja ze zmiennego przecinka na ATASCII;
c_ifp — konwersja liczby całkowitej bez znaku na zmiennoprzecinkową;
c_sifp — konwersja liczby całkowitej ze znakiem na zmiennoprzecinkową;
c_fpi — konwersja liczby zmiennoprzecinkowej na liczbę całkowitą bez znaku;
c_fpsi — konwersja liczby zmiennoprzecinkowej

na liczbę całkowitą ze znakiem;

c_fadd dodawanie liczb zmiennoprzecinkowych;
c_fsub odejmowanie liczb zmiennoprzecinkowych;
c_fmul mnożenie liczb zmiennoprzecinkowych;
c_fdiv dzielenie liczb zmiennoprzecinkowych;
c_log logarytm naturalny liczby zmiennoprzecinkowej;
c_log10 logarytm dziesiętny liczby zmiennoprzecinkowej;
c_alog antylogarytm naturalny liczby zmiennoprzecinkowej;
c_alog10 antylogarytm dziesiętny liczby zmiennoprzecinkowej;
c_exp funkcja wykładnicza;
c_sqrt pierwiastek kwadratowy z liczby zmiennoprzecinkowej;
c_int część całkowita liczby zmiennoprzecinkowej;
c_frac część ułamkowa liczby zmiennoprzecinkowej;
c_cmp porównanie dwóch liczb zmiennoprzecinkowych;
c_abs moduł liczby zmiennoprzecinkowej;
c_chs zmiana znaku liczby zmiennoprzecinkowej;
c_zero ustawienie liczby zmiennoprzecinkowej na zero;
c_move przesunięcie liczby zmiennoprzecinkowej w pamięci;

B4.6.1 Inicjalizacja biblioteki.

Zastosowanie.

Do inicjalizacji stałych i zmiennych wymaganych przez bibliotekę do realizacji jej funkcji.

Wywołanie funkcji.

```
c_ini();
```

Parametry wejściowe.

Brak.

Parametry wyjściowe.

Brak.

Opis.

Funkcja ta musi zostać wywołana tylko raz przed wykorzystaniem jakiejkolwiek innej funkcji z biblioteki.

Wykorzystane funkcje.

c_afp;

B4.6.2 Konwersja kodu ATASCII na liczbę zmiennoprzecinkową.

Zastosowanie.

Przekształcenie liczby zmiennoprzecinkowej podanej w postaci łańcucha znaków ATASCII na format liczby zmiennoprzecinkowej w standardzie Atari.

Wywołanie funkcji.

```
status = c_afp( acs, fpn);
```

Parametry wejściowe.

acs char array

Wskaźnik do łańcucha znaków zawierającego reprezentację ATASCII liczby zmiennoprzecinkowej.

Parametry wyjściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na liczbę zmiennoprzecinkową w formacie Atari odpowiadającą liczbie wejściowej w podanej kodzie ATASCII.

status integer scalar

Zwracany status:

0 gdy liczba przekształcona poprawnie.

-1 gdy pierwszy bajt liczby w kodzie ATASCII jest niepoprawny.

Opis.

Funkcja ta pobiera kolejne bajty z **acs*, aż do napotkania bajtu, który nie może być częścią liczby. Bajty wprowadzone do tego momentu są przekształcane na liczbę zmiennoprzecinkową, która zostanie zapamiętana w tablicy **fpn* o długości sześciu bajtów. Jeżeli pierwszy bajt napotkany w **acs* jest niepoprawny to *status* będzie miał wartość -1. W przypadku przeciwnym przyjmuje wartość 0.

Wykorzystane funkcje.

Funkcja odwołuje się tylko do procedur zawartych w pamięci ROM.

Przykład.

```
char pntr, fpn[6];
pntr = "56.789";
c_afp( pntr, fpn);
```

B4.6.3 Konwersja liczby zmiennoprzecinkowej na liczbę w kodzie ATASCII.

Zastosowanie.

Przekształcenie liczby zmiennoprzecinkowej w standardzie Atari na łańcuch znaków w formacie odpowiednim do drukowania.

Wywołanie funkcji.

```
c_fasc( fpn, acs);
```

Parametry wejściowe.

fpn char arry

Wskaźnik do sześciobajtowej tablicy znaków,

która zawiera liczbę zmiennoprzecinkową w standardzie Atari.

Parametry wyjściowe.
acs char array
Wskaźnik do siedemnastobajtowej tablicy znaków, w której znajdzie się gotowa do wydruku reprezentacja liczby zmiennoprzecinkowej w kodzie ATASCII.

Opis.
Funkcja ta przekształca liczbę zmiennoprzecinkową *fpn* na postać gotową do wydruku (ATASCII) i umiesza ją w tablicy znakowej *acs*, która musi mieć przynajmniej 17 bajtów długości. Funkcja ta nie wykrywa żadnych błędów.

Wykorzystane funkcje.
Funkcja wykorzystuje bezpośrednio procedury zmiennoprzecinkowe zawarte w pamięci ROM.

Przykład.
```
char pntr, fpn[6], output[17];
pntr = "56.789";
c_afp( pntr, fpn);
c_fasc( fpn, output);
printf("%s", output);

56.789
```

B4.6.4 Zamiana liczby całkowitej bez znaku na liczbę zmiennoprzecinkową.

Zastosowanie.
Zamiana liczby całkowitej bez znaku (od 0 do 65535) na liczbę zmiennoprzecinkową w formacie Atari.

Wywołanie funkcji.
c_ifp(usint, fpn);

Parametry wejściowe.
usint integer scalar
Liczba całkowita bez znaku (od 0 do 65535), która ma zostać zamieniona na liczbę zmiennoprzecinkową.

Parametry wyjściowe.
fpn integer array
Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na liczbę zmiennoprzecinkową.

Opis.
Funkcja ta zamienia liczbę całkowitą bez znaku *usint* na liczbę zmiennoprzecinkową w formacie Atari i zapamiętuje ją w sześcibajtowej tablicy znakowej wskazywanej przez *fpn*. Funkcja nie wykrywa żadnych błędów. Zwróć uwagę, że DEEP BLUE C nie dopuszcza istnienia liczb całkowitych bez znaku. Wszystkie liczby całkowite większe niż 32767 są traktowane jak ujemne.

Wykorzystane funkcje.
Funkcja ta wykorzystuje bezpośrednio procedury zmiennoprzecinkowe zawarte w pamięci ROM.

Przykład.
```
int integer;
char fpn[6], output[17];
integer = -5000;
c_ifp( integer, fpn);
c_fasc( fpn, output);
printf("%s", output);

60536
```

B4.6.5 Zamiana liczby całkowitej ze znakiem na liczbę zmiennoprzecinkową.

Zastosowanie.
Zamiana liczby całkowitej ze znakiem na liczbę zmiennoprzecinkową w formacie Atari.

Wywołanie funkcji.
c_sifp(sint, fpn);

Parametry wejściowe.
sint integer scalar
Liczba całkowita ze znakiem (od -32768 do +32767), która ma zostać zamieniona na liczbę zmiennoprzecinkową.

Parametry wyjściowe.
fpn char array
Wskaźnik do sześciobajtowej tablicy znakowej, w której znajduje się wynikowa liczba zmiennoprzecinkowa.

Opis.
Funkcja ta przekształca liczbę całkowitą ze znakiem *sint* na liczbę zmiennoprzecinkową w formacie Atari i zapamiętuje ją w sześcibajtowej tablicy znakowej wskazywanej przez *fpn*. Funkcja nie wykrywa błędów.
Wykorzystane funkcje.
Funkcja wykorzystuje funkcję *c_ifp*, przez co jest od niej mniej efektywna w zastosowaniu do liczb dodatnich.

Przykład.
```
int integer;
char fpn[6], output[17];
integer = -5000;
c_sifp( integer, fpn);
```

```
     c_fasc( fpn, output);
     printf("%s", output);

     -5000
```

B4.6.6 Zamiana liczby zmiennoprzecinkowej na liczbę całkowitą bez znaku.

Zastosowanie.

Zamiana standardowej liczby zmiennoprzecinkowej podanej w formacie Atari na liczbę całkowitą bez znaku.

Wywołanie funkcji.

```
     status = c_fpi( fpn, &usint);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej standardową liczbę zmiennoprzecinkową w formacie Atari, która ma zostać przekształcona na liczbę całkowitą bez znaku.

Parametry wyjściowe.

usint integr scalar

Zmienna całkowita przeznaczona na otrzymaną liczbę całkowitą bez znaku.

status integer scalar

Zwracany status:

0 - zamiana została wykonana poprawnie.

-1 - liczba zmiennoprzecinkowa jest >= 65535.5.

-2 - liczba zmiennoprzecinkowa jest ujemna.

Opis.

Funkcja ta zamienia standardową liczbę zmiennoprzecinkową w formacie Atari zawartą w *fpn* na liczbę całkowitą bez znaku. Jeżeli liczba zmiennoprzecinkowa jest ujemna zwracana jest wartość -2 i nie następuje żadna konwersja. Jeżeli liczba zmiennoprzecinkowa jest większa lub równa 65535.5 zwracana jest wartość -1 i, jak wyżej, konwersja się nie dokonuje. Funkcja ta realizuje prawdziwe zaokrąglanie, a nie obcinanie.

Wykorzystane funkcje.

Funkcja wykorzystuje bezpośrednio procedury działające na liczbach zmiennoprzecinkowych zawarte w pamięci ROM.

Przykład.

```
     char pntr, fpn[6];
     int integer;
     pntr = "6000";
     c_afp( pntr, fpn);
     c_fpi(fpn, &integer);
     printf("%d", integer);

     -5536.
```

B4.6.7 Zamiana liczby zmiennoprzecinkowej na liczbę całkowitą ze znakiem.

Zastosowanie.

Zamiana standardowej liczby zmiennoprzecinkowej w formacie Atari na liczbę całkowitą ze znakiem.

Wywołanie funkcji.

```
     status = c_fpsi( fpn, &sint);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową.

Parametry wyjściowe.

sint integer scalar

Zmienna całkowita przeznaczona na otrzymaną w wyniku konwersji liczbę całkowitą ze znakiem.

status integer scalar

Zwracany status:

0 - zamiana dokonała się poprawnie,

-1 - moduł liczby zmiennoprzecinkowej jest większy niż 32767.5.

Opis.

Funkcja zamienia liczbę zmiennoprzecinkową zawartą w *fpn* na liczbę całkowitą ze znakiem. Jeżeli moduł liczby jest większy lub równy 32767.5, to zwracana jest wartość -1, a zamiana się nie dokonuje.

Wykorzystane funkcje.

Omawiana funkcja wywołuje *c_fpi*. W stosunku do liczb dodatnich jest ona mniej efektywna niż bezpośrednie wywołanie funkcji *c_fpi*.

Przykład.

```
     char pntr, fpn[6];
     int integr;
     pntr = "60000";
     c_afp( pntr, fpn);
     c_fpsi(fpn, &integer);
     printf("%d", integer);

     -5536
```

B4.6.8 Dodawanie liczb zmiennoprzecinkowych.

Zastosowanie.

Dodanie dwóch liczb zmiennoprzecinkowych podanych w formacie Atari.

Wywołanie funkcji.

```
status = c_fadd( fpn1, fpn2, fpsum);
```

Parametry wejściowe.

fpn1 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej pierwszą liczbę.

fpn2 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej drugą liczbę.

Parametry wyjściowe.

fpsum char array

Wskaźnik do sześciobajtowej tablicy, w której zostanie zapamiętana suma dwóch pierwszych liczb.

status integer scalar

Zwracany status:

0 - dodawanie wykonane poprawnie,

-1 - wynik przekroczył dopuszczalny zakres.

Opis.

Funkcja ta dodaje zawartość **fpn1* do **fpn2* i zapamiętuje wynik w **fpsum*. Jeżeli wynik przekroczy zakres formatu liczb zmiennoprzecinkowych Atari, to zwracany jest status o wartości -1. Jeżeli operacja wykonana została poprawnie, zwracana jest wartość 0. *fp1*, *fp2* i *fpsum* mogą być tymi samymi wskaźnikami.

Wykorzystane funkcje.

Funkcja wykorzystuje w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp(pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_add( fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);

342.243
```

B4.6.9 Odejmowanie liczb zmiennoprzecinkowych.

Zastosowanie.

Odjęcie jednej liczby zmiennoprzecinkowej od drugiej.

Wywołanie funkcji.

```
status = c_fsub( odjemna, odjemnik, różnica);
```

Parametry wejściowe.

odjemna char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej odjemną.

odjemnik char arry

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej odjemnik.

Parametry wyjściowe.

różnica char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której umieszczona zostanie obliczona różnica.

status integer scalar

Zwracany status:

0 - odejmowanie wykonane poprawnie,

-1 - wynik przekroczył dopuszczalny zakres.

Opis.

Funkcja ta odejmuje **odjemnik* od **odjemna* i zapamiętuje wynik w *różnica*. Jeżeli wynik przekracza dopuszczalny zakres liczb zmiennoprzecinkowych w formacie Atari, zwracana jest wartość -1. 0 oznacza prawidłowe wykonanie odejmowania.

Wykorzystane funkcje.

Funkcja ta wykorzystuje w sposób bezpośredni procedury znajdujące się w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp(pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_fsub(fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);

299.997
```

B4.6.10 Dzielenie liczb zmiennoprzecinkowych.

Zastosowanie.

Dzielenie jednej liczby zmiennoprzecinkowej przez drugą.

Wywołanie funkcji

```
status = c_fdiv( dzielna, dzielnik, iloraz);
```

Parametry wejściowe.

dzielna char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej dzielną.

dzielnik char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej dzielnik.

Parametry wyjściowe.

iloraz char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na wynik obliczeń.

status integer scalar

Zwracany status:

0 poprawne zakończenie obliczeń.

1 przekroczenie zakresu lub dzielenie przez zero

Opis.

dzielna jest dzielona przez **dzielnik*, a wynik jest zapisywany w *iloraz*. Jeżeli wynik przekroczył dopuszczalny zakres lub wystąpiło dzielenie przez zero, to zwracana jest wartość -1. Jeżeli obliczenia są poprawne, zwracana jest wartość 0.

Wykorzystane funkcje.

Funkcja wykorzystuje w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp( pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_div( fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);

15.202386
```

B4.6.11 Mnożenie liczb zmiennoprzecinkowych.

Zastosowanie.

Pomnożenie przez siebie dwóch liczb zmiennoprzecinkowych.

Wywołanie funkcji.

status = c_fmul(mnożna, mnożnik, iloczyn);

Parametry wejściowe.

mnożna char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej mnożną.

mnożnik char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej mnożnik.

Parametry wyjściowe.

iloczyn char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której zostanie zapamiętany wynik mnożenia.

status integer scalar

Zwracany ststus:

0 - mnożenie wykonane poprawnie,

-1 - przekroczenie dopuszczalnego zakresu.

Opis.

Funkcja mnoży **mnożna* przez **mnożnik* i zapamiętuje wynik w *iloczyn*. Jeżeli nastąpi przekroczenie zakresu, zwrócana będzie wartość -1. 0 oznacza poprawne zakończenie obliczeń.

Wykorzystane funkcje.

Funkcja wykorzystuje w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp( pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_fmul( fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);

6783.01776
```

B4.6.12 Logarytm liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczanie logarytmu liczby zmiennoprzecinkowej.

Wywołanie funkcji.

status = c_log(nbr, log);

status = c_log10(nbr, log);

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której logarytm chcemy obliczyć.

Parametry wyjściowe.

log char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której zostanie zapamiętany logarytm liczby **nbr*.

status integer scalar

Zwracany status:

0 - obliczenia poprawne,

1 - liczba ujemna lub przekroczenie zakresu.

Opis.

Funkcja *c_log()* daje w wyniku logarytm naturalny, a *c_log10()* logarytm zwykły. Jeżeli **nbr* jest ujemna lub powoduje przepełnienie, to status jest równy -1. Gdy obliczenia są poprawne, to funkcja zwraca wartość 0.

Wykorzystane funkcje.

Obydwie funkcje wykorzystują w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, nbr[6], log[6], answer[17];
pntr = "256.512";
c_afp( pntr, nbr);
c_log( nbr, log);
c_fasc( log, answer);
printf("%s", answer);

5.5471754
```

B4.6.13 Antylogarytm liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczanie antylogarytmu liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
status = c_alog( nbr, antilog);
status = c_alog10( nbr, antilog);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową w formacie Atari, której antylogarytmu poszukujemy.

Parametry wyjściowe.

antilog char array

Wskaźnik do sześcibajtowej tablicy znakowej przeznaczonej na antylogarytm liczby *nbr.

status integer scalar

Zwracany status:

0 — antylogarytm obliczony poprawnie,

-1 — przepełnienie.

Opis.

Funkcja *c_alog* oblicza antylogarytm naturalny, a *c_alog10* antylogarytm zwykły. Antylogarytm naturalny to e (2.7182818) podniesione do potęgi *nbr. Antylogarytm zwykły to 10 podniesione do potęgi *nbr. Jeżeli wystąpi przepełnienie, funkcja zwraca wartość -1. Jeżeli obliczenia zostały wykonane poprawnie, to funkcja zwraca wartość 0. *nbr* i *antilog* mogą być tym samym wskaźnikiem.

Wykorzystane funkcje.

Obydwie funkcje korzystają bezpośrednio z procedur zawartych w pamięci ROM.

Przykład.

```
char *pntr, nbr[6], log[6], answer[17];
pntr = "5.5471754";
c_afp( pntr, nbr);
c_alog( nbr, log);
c_fasc( log, answer);
printf( "%s", answer);

256.512
```

B4.6.14 Funkcja wykładnicza liczby zmiennoprzecinkowej.

Zastosowanie.

Podniesienie standardowej liczby zmiennoprzecinkowej do potęgi będącej również liczbą zmiennoprzecinkową w formacie Atari.

Wywołanie funkcji.

```
status = c_exp( podstawa, wykładnik, wynik);
```

Parametry wejściowe.

podstawa char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową w formacie Atari, której żądaną potęgę mamy znaleźć.

wykładnik char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową w formacie Atari, która ma być wykładnikiem potęgowania.

Parametry wyjściowe.

wynik char array

Wskaźnik do sześciobajtowej tablicy, w której zapamiętany zostanie wynik potęgowania liczby *podstawa.

status integr scalar

Zwracany status:

0 — obliczenia wykonane poprawnie,

-1 — przekroczenie zakresu.

Opis.

Liczba zawarta w *podstawa* zostanie podniesiona do potęgi *wykładnik* a wynik tej operacji umieszczony zostanie w *wynik*. *podstawa* i *wykładnik* mogą być tą samą zmienną, a *wynik* może być tą samą zmienną co *podstawa* lub *wykładnik*. Jeżeli wynik nie mieści się w zakresie liczb zmiennoprzecinkowych, funkcja zwraca wartość -1, a w przypadku przeciwnym wartość 0.

Wykorzystane funkcje.

c_fmul

c_log10

c_alog10

Przykład.

```
char *pntr, podstawa[6], wykladnik[6],
wynik[6], answer[17];
pntr = "2.37";
c_afp( pntr, podstawa);
pntr = "7.95";
c_afp( pntr, wykladnik);
c_exp( podstawa, wykladnik, wynik);
c_fasc( wynik, answer);
printf( "%s", answer);

953.34337
```

B4.6.15 Pierwiastek kwadratowy z liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczyć pierwiastek kwadratowy ze standardowej liczby zmiennoprzecinkowej w formacie Atari.

Wywołanie funkcji.

status = c_sqrt(nbr, pierwiastek);

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę, której pierwiastek chcemy obliczyć.

Parametry wyjściowe.

pierwiastek char array

Wskaźnik do sześciobajtowej tablicy znakowej, do której zostanie zapisany pierwiastek kwadratowy liczby **nbr*.

status integer scalar

Zwracany status:

 0 – obliczenia wykonane poprawnie,
 -1 – przekroczenie zakresu,
 -2 – liczba **nbr* jest ujemna.

Opis.

Funkcja oblicza pierwiastek kwadratowy z dodatniej liczby zmiennoprzecinkowej znajdującej się w *nbr* i przesyła go do *pierwiastek*. Jeżeli pierwiastek obliczony został poprawnie to funkcja zwraca wartość 0, jeżeli liczba **nbr* jest ujemna wartość -2, a jeżeli wynik obliczeń przekroczy dopuszczalny zakres wartość -1.

Wykorzystane funkcje.

c_fmul
c_log10
c_alog10

Przykład.

```
char *pntr, nbr[6], pierw[6], answer[17];
pntr = "256.512";
c_afp( pntr, nbr);
c_sqrt( nbr, pierw);
c_fasc( pierw, answer);
printf( "%s", answer);

16.015992001
```

B4.6.16 Część całkowita liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczenie części całkowitej liczby zmiennoprzecinkowej. Wynik jest także liczbą zmiennoprzecinkową.

Wywołanie funkcji.

status = c_int(nbr, intpor);

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której część całkowitą chcemy uzyskać.

Parametry wyjściowe.

intpor char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której znajdzie się część całkowita liczby **nbr*.

status integer scalar

Zwracany status:

 0 – liczba normalna,
 -1 – brak części ułamkowej, do **intpor* przesłana jest **nbr*,
 -2 – **nbr* < 1, **intpor* jest ustawiana na zero.

Opis.

Część ułamkowa liczby **nbr* zostaje obcięta, a wynik zapamiętywany jest w *intpor*. Zwracana wartość niezerowa nie oznacza błędu tylko specjalne przypadki. *nbr* i *intpor* mogą być tą samą zmienną.

Wykorzystane funkcje.

move (biblioteka AIO)

Przykład.

```
char *pntr, nbr[6], intp[6], answer[17];
pntr = "1234.5678";
c_afp( pntr, nbr);
c_int( nbr, intp);
c_fasc( intp, answer);
printf( "%s", answer);

1234
```

B4.6.17 Część ułamkowa liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczenie części ułamkowej liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
status = c_frac( nbr, fracpor);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której część ułamkową należy obliczyć.

Parametry wyjściowe.

fracpor char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na część ułamkową liczby **nbr*.

status integer scalar

Zwracany status:

0 – liczba normalna,

-1 – **nbr* < 1, brak części całkowitej,

-2 – brak części ułamkowej, **fracpor* zostaje ustawiona na zero.

Opis.

Część całkowita liczby **nbr* zostaje obcięta, a wynik zapamiętywany jest w **fracpor*.

Niezerowa wartość statusu nie oznacza błędu, ale sygnalizuje specjalne przypadki wymienione wcześniej. **nbr* i **fracpor* mogą być tą samą zmienną.

Wykorzystane funkcje.

move (biblioteka AIO)
c_int
c_fsub

Przykład.

```
char *pntr, nbr[6], fracp[6], answer[17];
pntr = "1234.5678";
c_afp( pntr, nbr);
c_frac( nbr, fracp);
c_fasc( fracp, answer);
printf( "%s", answer);

0.5678
```

B4.6.18 Porównanie dwóch liczb zmiennoprzecinkowych.

Zastosowanie.

Porównanie dwóch liczb zmiennoprzecinkowych.

Wywołanie funkcji.

```
result = c_cmp( fpn1, fpn2)
```

Parametry wejściowe.

fpn1 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej pierwszą liczbę zmiennoprzecinkową.

fpn2 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej drugą liczbę zmiennoprzecinkową.

Parametry wyjściowe.

result char array

Wynik porównania:

-1 – **fpn1* mniejsza niż **fpn2*,

0 – **fpn1* równa **fpn2*,

+1 – **fpn1* większa niż **fpn2*.

Opis.

Liczba **fpn1* jest porównywana z **fpn2*. Jeżeli **fpn1* jest mniejsza niż **fpn2*, to *result* przyjmuje wartość -1. Jeżeli liczby są sobie równe, to result zostaje ustawiony na 0. Jeżeli **fpn1* jest większa niż **fpn2*, to *result* przyjmuje wartość +1.

Wykorzystywane funkcje.

c_fsub

Przykład.

```
char *pntr, nbr1[6], nbr2[6];
int status;


pntr = "-27.45";
c_afp( pntr, nbr1);
pntr = "14.55";
c_afp( pntr, nbr2);
status = c_cmp( nbr1, nbr2);
printf( "%s", status);

-1
```

B4.6.19 Obliczanie modułu liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczanie modułu liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
c_abs( fpn, absfpn);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której moduł chcemy obliczyć.

Parametry wyjściowe.

absfpn char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na moduł liczby **fpn*.

Opis.

Następuje obliczenie modułu liczby **fpn* i zapamiętanie go w **absfpn*. *fpn* i *absfpn* mogą być tą samą zmienną.

Wykorzystane funkcje.

Brak.

Przykład.

```
char *pntr, nbr[6], absnbr[6], answer[17];
pntr = "-15.789";
c_afp( pntr, nbr);
c_abs( nbr, absnbr);
c_fasc( absnbr, answer);
printf ("%s", answer);

15.789
```

B4.6.20 Zmiana znaku liczby zmiennoprzecinkowej.

Zastosowanie.

Zmiana znaku liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
c_chs( fpn, negfpn);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę, którą chcemy zanegować.

Parametry wyjściowe.

negfpn char array

Wskaźnik do sześcio bajtowej tablicy znakowej przeznaczonej na liczbę będącą negacją **fpn*.

Opis.

Zmieniony zostaje znak **fpn*, a wynik zostaje zapamiętany w **negfpn*. *fpn* i *negfpn* mogą być tą samą zmienną.

Wykorzystywane funkcje.

Brak.

Przykład.

```
char *pntr, nbr[6], output[6], answer[17];
pntr = "15.7895";
c_afp( pntr, nbr);
c_chs( nbr, output);
c_gasc( output, answer);
printf( "%s", answer);

-15.7895
```

B4.6.21 Ustawienie liczby zmiennoprzecinkowej na zero.

Zastosowanie.

Ustawienie liczby zmiennoprzecinkowej na zero.

Wywołanie funkcji.

```
c_zero( fpn);
```

Parametry wejściowe.

Brak.

Parametry wyjściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której wartość wynosi zero.

Opis.

Do liczby **fpn* ładowane jest zero w formacie liczby zmiennoprzecinkowej Atari.

Przykład.

```
char fpn[6], answer[17];
c_zero( fpn);
c_fasc( fpn, answer);
printf ( "%s", answer);

0
```

B4.6.22 Przesunięcie liczby zmiennoprzecinkowej.

Zastosowanie.

Przepisywanie liczb zmiennoprzecinkowych z jednego miejsca pamięci w inne.

Wywołanie funkcji.

```
c_move( fpn1, fpn2);
```

Parametry wejściowe.

fpn1 char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową, którą należy skopiować.

Parametry wyjściowe.

fpn2 char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na kopiowaną liczbę.

Opis.

Liczba **fpn1* jest kopiowana do **fpn2*.

Wykorzystywane funkcje.

move (biblioteka AIO)

Przykład.

```
char *pntr, fpn1[6], fpn2[6], answer[17];
```

```
pntr = "66";
c_afp( pntr, fpn1);
c_move( fpn1, fpn2);
c_fasc( fpn2, answer);
printf( "%s", answer);

66
```

B4.7 Funkcje trygonometryczne zawarte w pliku TRIG.CCC.

W tym rozdziale omówimy wszystkie funkcje trygonometryczne biblioteki MATHLIB zawarte w pliku TRIG.CCC. Oto ich lista:

c_trig	inicjalizacja funkcji trygonometrycznych;
c_rad	ustawienie trybu obliczeń;
c_rd	zamiana radianów na stopnie;
c_dr	zamiana stopni na radiany;
c_dmsd	zamiana stopni, minut, sekund na stopnie dziesiętne;
c_ddms	zamiana stopni dziesiętnych na stopnie, minuty, sekundy;
c_sin	obliczanie wartości funkcji sinus;
c_cos	obliczanie wartości funkcji cosinus;
c_tan	obliczanie wartości funkcji tangens;
c_atan	obliczanie wartości funkcji arctangens.

Określenie "stopnie dziesiętne" oznacza wartość podaną w stopniach, będącą liczbą zmiennoprzecinkową. Na przykład 30 stopni 25 minut i 37 sekund to 30.42694444 w stopniach dziesiętnych.

B4.7.1 Inicjalizacja funkcji trygonometrycznych.

Zastosowanie.
Inicjalizacja funkcji trygonometrycznych biblioteki MATHLIB.

Wywołanie funkcji.
c_trig();

Parametry wejściowe.
Brak.

Parametry wyjściowe.
Brak.

Opis.
Funkcja inicjalizuje stałe i zmienne używane przez funkcje trygonometryczne biblioteki MATHLIB. Musi zostać wywołana przed użyciem jakiejkolwiek funkcji trygonometrycznej. Pominięcie tej funkcji spowoduje, że obliczenia realizowane przez funkcje trygonometryczne będą błędne. Funkcja *c_trig* ustawia tryb wykonywania obliczeń w stopniach.

Wykorzystywane funkcje.
- c_afp

B4.7.2 Ustawienie trybu wykonywania obliczeń w stopniach lub radianach.

Zastosowanie.
Wybranie trybu realizacji obliczeń przez funkcje trygonometryczne. Obliczenia mogą być realizowane w stopniach lub radianach.

Wywołanie funkcji.
c_rad(flag);

Parametry wejściowe.
flag integer scalar
Parametr *flag* oznacza wybrany rodzaj pracy:
równy 0 – stopnie,
nie równy 0 – radiany.

Parametry wyjściowe.
Brak.

Opis.
Funkcja ta informuje MATHLIB, że obliczenia trygonometryczne będą realizowane w stopniach lub radianach. Funkcja ta może być wywoływana w dowolnym momencie. Wywołanie funkcji *c_trig* ustawia tryb pracy w radianach.

Wykorzystywane funkcje.
Brak.

B4.7.3 Zamiana radianów na stop nie.

Zastosowanie.
Zamiana radianów na stopnie.

Wywołanie funkcji.
status = c_rd(radiany, stopnie);

Parametry wejściowe.
radiany char array
Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową określającą kąt w radianach.

Parametry wyjściowe.
stopnie char array
Wskaźnik do sześciobajtowej tablicy znakowej, do której przesłany zostanie kąt w stopniach dziesiętnych równy kątowi podanemu w *radiany*.

```
        status integer scalar
     Zwracany status:
        0 - zamiana została dokonana poprawnie,
       -1 - przekroczenie zakresu.
```

Opis.

Funkcja ta zamienia radiany na stopnie dziesiętne. *radiany* i *stopnie* mogą być tą samą zmienną.

Wykorzystywane funkcje.

```
        c_fdiv
```

Przykład.

```
     char *pntr, radiany[6], stopnie[6],
     answer[17];
     pntr = "0.78539816";
     c_afp( pntr, radiany);
     c_rd( radiany, stopnie);
     c_fasc( stopnie, answer);
    printf( "%s", answer);

     45
```

B4.7.4 Zamiana stopni na radiany.

Zastosowanie.

Zamiana stopni dziesiętnych na radiany.

Wywołanie funkcji.

```
        status = c_dr( stopnie, radiany);
```

Parametry wejściowe.

```
        stopnie char array
```

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową oznaczającą kąt w stopniach dziesiętnych.

Parametry wyjściowe.

```
        radiany char array
```

Wskaźnik do sześciobajtowej tablicy znakowej, do której zostaje wpisany kąt w radianach.

```
        status integer scalar
     Zwracany status:
        0 - zamiana wykonana poprawnie,
       -1 - przekroczenie zakresu.
```

Opis.

Funkcja zamienia stpnie dziesiętne na radiany. *stopnie* i *radiany* mogą być tą samą zmienną.

Wykorzystywane funkcje.

```
        c_fmul
```

Przykład.

```
     char *pntr, radiany[6], stopnie[6],
     answer[17];
     pntr = "45";
     c_afp( pntr, stopnie);
     c_dr( stopnie, radiany);
     c_fasc( radiany, answer);
     printf( "%s", answer);


  0.78539816
```


B4.7.5 Zamiana stopni, minut, sekund na stopnie dzesiętne.

Zastosowanie.

Zamiana stopni. minut i sekund na stopnie dziesiętne.

Wywołanie funkcji.

```
 status = c_dmsd( stopnie, minuty, sekundy, sd);
```

Parametry wejściowe.

```
        stopnie char array
```

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę stopni.

```
        minuty char array
```

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę minut.

```
        sekundy char array
```

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę sekund.

Parametry wyjściowe.

```
        sd char array
```

Wskaźnik do sześciobajtowej tablicy znakowej, do której zostanie załadowana liczba określająca kąt w stopniach dziesiętnych.

```
        status integer scalar
     Zwracany status:
        0 - zamiana dokonana poprawanie,
       -1 - przekroczenie zakresu
```

Opis.

Kąt wyrażony w stopniach, minutach i sekundach zostaje zamieniony na stopnie dziesiętne.

Wykorzystane funkcje.

```
        c_fdiv
        c_fadd
```

Przykład.

```
     char st[6], min[6], sek[6], std[6], *aux,
     output[17];
     aux = "30";
     c_afp( aux, st);
     aux = "25";
     c_afp( aux, min);
     aux = "37";
     c_afp( aux, sek);
```

```
c_dmsd( st, min, sec, std);
c_fasc(ddeg, output);
printf("%s", output);

30.42694444
```

B4.7.6 Zamiana stopni dziesiętnych na stopnie, minuty, sekundy.

Zastosowanie.

Zamiana stopni dziesiętnych na stopnie, minuty i sekundy.

Wywołanie funkcji.

```
status = c_ddms( sd, stopnie, minuty, sekundy);
```

Parametry wejściowe.

sd char array

Wskaźnik do szściobajtowej tablicy zawierającej liczbę zmiennoprzecinkową określającą kąt w stopniach dziesiętnych.

Parametry wyjściowe.

stopnie char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę stopni.

minuty char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę minut.

sekundy char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę sekund.

status integer scalar

Zwracany status:

0 – zamiana wykonana została poprawnie,

-1 – przekroczenie zakresu.

Opis.

Kąt podany w stopniach dziesiętnych zamieniany jest na kąt przedstawiony w stopniach, minutach i sekundach.

Wykorzystane funkcje.

c_int
c_fsub
c_fmul

Przykład.

```
char st[6], min[6], sek[6], std[6], *aux;
char out1[17], out2[17], out3[17];
aux = "30.42694444";
c_afp( aux, std);
c_DDms( std, st, min, sek);
c_fasc( st, out1);
c_fasc( min, out2);
c_fasc( sek, out3);
printf("%s %s %s", out1, out2, out3);

30 25 27
```

B4.7.7 Obliczanie wartości funkcji sinus.

Zastosowanie.

Obliczanie wartości funkcji sinus dla podanej wartości kąta.

Wywołanie funkcji.

```
status = c_sin( kąt, sinus);
```

Parametry wejściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową określającą wartość kąta.

Parametry wyjściowe.

sinus char array

Wskaźnik do sześcobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową będącą wartością funkcji dla danego kąta.

status integer scalar

Zwracany status:

0 – obliczenia zostały wykonane poprawnie,

1 – przekroczenie zakresu.

Opis.

Kąt *kąt zostaje zredukowany do wartości z przedziału [0, +pi/2]. Wartość funkcji sinus, dzięki rozłożeniu jej w szereg Tayolra, obliczna jest z dokładnością do ośmiu miejsc po przecinku.

Wykorzystane funkcje.

move (biblioteka AIO)
c_fmul
c_fdiv
c_frac
c_fsub
c_fadd

Przykład.

```
char *pntr, nbr[6], sinnbr[6], answer[17];
rad(0); /* obliczenia w stopniach */
pntr = "30";
c_afp( pntr, nbr);
c_sin( nbr, sinnbr);
c_fasc( sinnbr, answer);
printf( "%s", answer);

0.5
```

B4.7.8 Obliczanie wartości funkcji cosinus.

Zastosowanie.

Obliczenie wartości funkcji cosinus dla zadanej wartości kąta.

Wywołanie funkcji.

status = c_cos(kąt, cosinus);

Parametry wejściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej kąt podany w stopniach dzisiętnych.

Parametry wyjściowe.

cosinus char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość funkcji cosinus dla podanego kąta.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zkresu.

Opis.

Oblicza wartość funkcji cosinus dla kąta *k t i zapamiętuje ją w *cosinus*.

Wykorzystane funkcje.

move (biblioteka AIO)
c_fmul
c_fsub
c_sin

Przykład.

```
char *pntr, nbr[6], cosnbr[6], answer[17]
rad(0);  /* obliczenia w stopniach */
pntr = "30";
c_afp( pntr, nbr);
c_cos( nbr, cosnbr);
c_fasc( cosnbr, answer);
printf( "%s", answer);

0.8660254
```

B4.7.9 Obliczanie funkcji tangens.

Zastosowanie.

Obliczenie wartości funkcji tangens dla podanego kąta.

Wywołanie funkcji.

ststus = c_tan(kąt, tangens);

Parametry wejściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość kąta podaną w stopniach dziesiętnych.

Parametry wyjściowe.

tangens char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość funkcji dla zadanego kąta.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zakresu.

Wykorzystane funkcje.

c_sin
c_cos
c_fdiv

Przykład.

```
char *pntr, nbr[6], tannbr[6], answer[17];
rad(0);  /* obliczenia w stopniach */
c_afp( pntr, nbr);
c_tan( nbr, tannbr);
c_fasc( tannbr, answer);
printf( "%s", answer);

0.57735027
```

B4.7.10 Obliczanie wartości funkcji arctangens.

Zastosowanie.

Obliczanie wartości funkcji arctangens dla danej liczby rzeczywistej.

Wywołanie funkcji.

status = c_atan(tangens, kąt);

Parametry wejściowe.

tangens char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę, dla której należy policzyć wartość funkcji arctangens.

Parametry wyjściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość obliczonego kąta.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zakresu.

Opis.

Obliczenia realizowane są z dziewięciocyfrową dokładnością. Wynik mieści się w przedziale (-90, +90) w stopniach lub w przedziale (-pi/2, +pi/2) w radianach. Zależy to od wybranego trybu realizacji obliczeń.

Wykorzystane funkcje.

move (biblioteka AIO)
c_fdiv
c_cmp
c_fmul
c_fadd

```
c_fsub
```

Przykład.

```
char *pntr, deg45[6], actan45[6], answer[17];
rad(0);   /* obliczenia w stopniach */
pntr = "1";
c_afp( pntr, deg45);
c_atan( deg45, atan45);
c_fasc( atan45, answer);
printf( "%s", answer);

45
```



PROGRAMY

*** SPIRALE ***

Program ten ilustruje wykorzystanie funkcji z biblioteki trygonometrycznej TRIG. Rozwiązuje on problem przedstawiony w książce Charlesa Corge p.t. ELEMENTY INFORMATYKI, PWN, Warszawa 1981r. Oddajmy głos autorowi tej pracy: "Cztery biedronki o imionach Afrodyta, Busiris, Cybela i Danaos znajdują się w danym momencie w czterech rogach kwadratu o boku a, oznaczonych odpowiednio ich inicjałami. Afrodyta i Cybela są samiczkami, Busiris i Danaos samcami. Każda z biedronek, pałając czułym, lecz niestety nie odwzajemnionym uczuciem, ożywiona jest nieodpartą chęcią spotkania się z wybrańcem swego serca. Decydują się na ten krok w tym samym momencie i wyruszają z identyczną prędkością; Afrodyta w kierunku Busirisa, Busiris w kierunku Cybeli, Cybela w kierunku Danaosa i wreszcie Danaos w kierunku Afrodyty". Problem polega na wykreśleniu dróg, które przebędą biedronki. Program SPIRALE kreśli te drogi. Z matematycznego punktu widzenia drogi te są krzywymi, które zakreślają wierzchołki kwadratu o boku a i środku znajdującym się w początku układu współrzędnych XY, poddanego serii przekształceń, będących złożeniem obrotu o kąt alfa z jednokładnością o stosunku exp(-alfa). Program obok funkcji sterującej main() składa się również z funkcji:

obl_jedn() - oblicza stosunek jednokładności, tj. wartość wyrażenia exp(-alfa) dla różnych wartości kąta alfa;

rek_kat() - dokonuje redukcji kąta z przedziału 0<=alfa<=360 stopni do kąta z przedziału 0-90 stopni;

obroc_kw() - kreśli kwadrat;

p_alfa() - wczytuje przyrost kąta obrotu alfa;

ut_tabs() - oblicza wartości funkcji sinus dla kątów od 0 do 90 stopni i umieszcza je w tabeli sinusów;

dr_ekran()- drukuje wykreślone drogi biedronek na drukarce ATARI 1029.

Ponadto program SPIRALE wykorzystuje procedurę w kodzie maszynowym (jest ona wywołana w funkcji dr_ekran()) drukującą graficzną zawartość ekranu. Przedstawiony kod źródłowy tej procedury należy zasemblować (na przykład asemblerem EASMD) i dołączyć kod maszynowy do programu podczas fazy linkowania.

Program SPIRALE

Funkcja sterujaca main()

```
/*********************************
 *                               *
 *       program "SPIRALE"       *
 *                               *
 *********************************

 * demonstruje wykorzystanie bibliotek
 * zmiennoprzecinkowej i
trygonometrycznej
 * kompilatora DBC w tworzeniu grafiki
 * w trybie wysokiej rozdzielczosci.
 * Program kresli figure powstala w
 * wyniku zlozenia przeksztalcen obrotu
 * o kat alfa z jednokladnoscia o
stosunku
 * exp(-alfa) kwadratu o boku a i srodku
 * w poczatku ukladu wspolrzednych
kartezjanskich

 * Wazniejsze zmienne programu:
 * sint - tablica sinusow katow z
przedzialu
 *          0<=alfa<=90 stopni
 * kzred- tablica umozliwiajaca odczyt
 *          sinusa i cosinusa kata z w/w
 *          przedzialu
 * wspw - tablica wspolrzednych
wierzcholkow
 *          kwadratu
 * bok  - dlugosc boku kwadratu
 * alfa - kat obrotu kwadratu
 */


char sint[546];
```

```
main()
$(
 int wspw[8],kzred[4];
 char arg[20],c,c1;
 int
i,j,bok,czytaj,status,alfa,beta,xi,yi;
 char bokf[6],dwa[2],p2[6],pp2[6],sj[6];
 char rob[6],x[6],y[6],promien[6],e[6];
 char *const;
 int palfa,kont;
 kont=1;

/* inicjacja bibliotek */
 c_iml();
 c_itrig();

/* utworz tabele sinusow */
 ut_tabs();

/* obliczenie wartosci stalej
0.5*sqrt(2) */
 const="2.0";
 c_afp(const,dwa);


 c_sqrt(dwa,p2);
 c_fdiv(p2,dwa,pp2);

 const="2.7183";  /* liczba e */
 c_afp(const,e);

 while(kont)
 $(

/* wczytaj dlugosc boku kwadratu */
 graphics(0);
 czytaj=1;
 while(czytaj)
  $(
   printf("\npodaj dlugosc boku
kwadratu\n");
   printf("40<=bok<=160\n");
   printf("bok = ");
   status=gets(arg);
   if (status<=0)
    $(
     printf("\g\nblad przy
wprowadzaniu\n");
     continue;
    $)
   bok=val(arg);
   if(bok<40 || bok>160)
    $(
     printf("\g\ndlugosc poza dozwolonym
zakresem");
     continue;
    $)
   czytaj=0;
  $)    /* koniec petli while  */

/* oblicz dlugosc polowy przekatnej
kwadratu */
/* tj. promien obrotu     */
 c_ifp(bok,bokf);
 c_fmul(bokf,pp2,promien);

/* wczytaj przyrost kata obrotu */
 palfa=p_alfa();

/* ustaw kolory  */
 graphics(24);
 color(1);
 poke(710,16);
 poke(712,16);
 /* przeksztalcenie kwadratu i rysowanie
kwadratu
  * przeksztalconego  */
 for(j=0;j<180;j+=palfa)
  $(
   obl_jedn(e,j,sj);
   for(i=0;i<=1;i++)
    $(
     beta=j+45+i*90;
     red_kat(beta,kzred);
     obroc_kw(kzred,promien,x,y);
     /* zmniejsz wspol. w skali sj */
     c_fmul(x,sj,rob);
     /* zamien wspol. na integer */
     c_fpsi(rob,&xi);
     wspw[2*i]=xi;

     c_fmul(y,sj,rob);
     c_fpsi(rob,&yi);
     wspw[2*i+1]=yi,
    $)
   wspw[4]=-wspw[0];
   wspw[5]=-wspw[1];
   wspw[6]=-wspw[2];
   wspw[7]=-wspw[3];
   rys_kw(wspw);
  $)
  while(peek(53279)!=5);
  dr_ekran();
  graphics(0);
  printf("chcesz jeszcze raz (T/N)");
  c=toupper(getchar());
  c1=getchar();
  if(c=='N') kont=0;
  $)
$)
```

Funkcja obl_jedn()

```
/*  FUNKCJA obl_jedn(e,alfa,stjedn)
 * oblicza wartosc wyrazenia
 *    stjedn=exp(-alfa)
 * ktora jest wykorzystana jako skala
 * zmniejszenia dlugosci boku kwadratu
 * przy jego obrocie o kat alfa
 */

obl_jedn(e,alfa,stjedn)
int alfa;
char *stjedn,*e;
$(
 char alfaf[6],alfar[6];
 c_ifp(alfa,alfaf);    /* zamien int na
fp */
 c_dr(alfaf,alfar);    /* zamien stopnie
na radiany */
 *alfar = *alfar^0x80; /* zmien znak  */
 c_exp(e,alfar,stjedn);
$)   /* koniec funkcji obl_jedn() */
```

Funkcja red_kat()

```
/*     Funkcja red_kat(kat,kzred)
 * dokonuje redukcji kata z przedzialu
 * 0<=kat<=360 stopni do kata z
przedzialu
 * 0<=kat<=90 stopni.
 * miara zredukowanego kata nie ulega
zmianie
 * funkcja umieszcza cztery wielkosci
typu
```

```
 * integer w czteroelementowej tablicy
KZRED
 * ich znaczenie jest nastepujace:
 * kzred[0] - liczba wskazujaca element
tablicy sint
 * zawierajacy sinus kata
 * kzred[1] - j.w. dotyczy cosinusa kata
 * kzred[2] - znak sinusa (kzred[2]=1
gdy sin<0,
 *                            kzred[2]=0

gdy sin>=0)
 * kzred[3] - znak cosinusa (podobnie
jak wyzej)
 */


red_kat(kat,kzred)
int kat;
int *kzred;
$(
 int m,n,znaks,znakc;
 if (kat>359) kat=kat%360;
 if (kat<91)
  $(
   m=kat;
   znaks=0;
   n=90-kat;
   znakc=0;
  $)
 else if (kat<181)
  $(
   m=180-kat;
   znaks=0;
   n=kat-90;
   znakc=1;
  $)
 else if (kat<271)
  $(
   m=kat-180;
   znaks=1;
   n=270-kat;
   znakc=1;
  $)
 else
  $(
   m=360-kat;
   znaks=1;
   n=kat-270;
   znakc=0;
  $)
 *kzred=m;
 *(kzred+1)=n;
 *(kzred+2)=znaks;
 *(kzred+3)=znakc;
$)  /* koniec funkcji red_kat()  */
```

Funkcja obroc_kw()

```
/* FUNKCJA obroc_kw(kzred,promien,x,y)
 * wyznacza wspolrzedne jednego
wierzcholka
 * kwadratu po obrocie o kat, ktorego
 * sinus i cosinus (i ich znaki)
okreslane sa
 * przez elementy tablicy KZRED
 * promien - zawiera dlugosc polowy
 * przekatnej kwadratu w postaci liczby
 * zmiennoprzecinkowej.
 * x,y - wspolrzedne po obrocie
 */


obroc_kw(kzred,promien,x,y)
 int *kzred;
 char *promien,*x,*y;
 $(


 char *wsk;
 int j;
 char sin[6],cos[6];
 j = *kzred;
 wsk=sint + j*6;
 move(wsk,sin,6);
 j = *(kzred+2);
 if(j) *sin = *sin^0x80;
 j = *(kzred+1);
 wsk=sint + j*6;
 move(wsk,cos,6);
 j = *(kzred+3);
 if(j) *cos = *cos^0x80;
 c_fmul(cos,promien,x);
 c_fmul(sin,promien,y);
$)  /* koniec funkcji obroc_kw()  */
```

Funkcja p_alfa()

```
/*        FUNKCJA P_ALFA()
 * wczytuje przyrost kata obrotu alfa
 */

p_alfa()
$(
 int znak;
 int iocb,pob;
 printf("\n\nWybierz przyrost kata
obrotu alfa\n");
 printf("\262  - dwa stopnie\n");
 printf("\263  - trzy stopnie\n");
 printf("\264  - cztery stopnie\n");
 printf("\265  - piec stopni\n");
 printf("\266  - szesc stopni\n");
 printf("\267  - siedem stopni\n");
 position(2,18);
 printf("po narysowaniu figury nacisnij
\323\305\314\305\303\324");
 iocb=copen("K:",'r');
 pob=1;
 while(pob)
  $(
   znak=cgetc(iocb);
   if(znak<'2' || znak>'7')
    $(
     printf("\g");
     continue;
    $)
   pob=0;
  $)
 cclose(iocb);
 return(znak-48);
$)
```

Funkcja dr_ekran()

```
/*    funkcja dr_ekran()
 * drukuje zawartosc pamieci ekranu
 * (w 8 trybie graficznym) na drukarce
 * funkcja wykorzystuje podprogram
 * w kodzie maszynowym rezydujacy na
 * szostej stronie pamieci (adres 1536
 * lub 0x600)
 */

 dr_ekran()
 $(
```

```
int l,iocb;
char *ekran;
ekran=dpeek(88);
for(l=0;l<28;l++)
 $(
/*
 * otwarcie kanalu dla drukarki
 */
  iocb=copen("P:",'w');
  if(iocb<0)
   $(
    graphics(0);
    printf("\g\n\ndrukarka odlaczona");
    return;
   $)
/* ustaw tryb graficzny pracy drukarki
*/
  cputc('\033',iocb);
  cputc('9',iocb);
  cputc('\033',iocb);
  cputc('A',iocb);
  cputc('\001',iocb);
  cputc('\030',iocb);
/*
 * wywolanie podprogramu rezydujacego
 * na szostej stronie pamieci
 */

  usr(1536,iocb,ekran);

  cclose(iocb);
  ekran=ekran+280;
 $)
$)      /* koniec funkcji dr_ekran() */
```

Funkcja ut_tabs()

```
/*
 * FUNKCJA ut_tabs()
 * oblicza wartosci funkcji sinus
 * dla katow od 0 do 90 stopni i
 * umieszcza je w postaci szescio-
 * bajtowych liczb zmiennoprzecinkowych
 * w tablicy sint[594]
 */

ut_tabs()
$(
 int i;
 char kat[6],sinasc[17],
 char *wsksin;
 printf("\f");  /* czysc ekran */
 poke(752,1);   /* schowaj kursor */
 position(10,3),
 printf("********************");
 position(10,4);
 printf("*                  *");
 position(10,5);
 printf("  program SPIRALE  *");
 position(10,6);
 printf("*                  *");
 position(10,7);
 printf("********************\n\n");
 printf(" kresli cztery spirale
logarytmiczne\n");
 printf("    stanowiace rozwiazanie
problemu\n");
 printf(" przedstawionego w pracy C.
Corge\n");
 printf("     p.t. Elementy
Informatyki\n");
 printf("     PWN, Warszawa 1981r. str.
340\n\n");
 printf("Prosze chwile
poczekac...\n\n");
 printf("tworzenie tabeli sinusow");
 i=0;
 c_rad(0);     /* miary katow - stopnie
*/
 while(i<91)
  $(
   c_ifp(i,kat);     /* zamien integer na
fp */
   wsksin=sint+6*i; /* wskaznik do
odpowiedniego */
                     /* elementu tablicy
sint     */
   c_sin(kat,wsksin);
   c_fasc(wsksin,sinasc);
   position(1,20);
   printf("kat %-3d stop.    sin =
%s",i,sinasc);
   ++i;
  $)       /* koniec while */
$)         /* koniec funkcji ut_tabs() */
```

```
01 ;PR1029.ASM
05 .OPT OBJ
10 ;Podprogram wyprowadza na drukarke
20 ;ATARI 1029 zawartosc fragmentu
30 ;pamieci ekranu w 8 trybie graficznym
40 ;Sposob wywolania z jezyka DBC
50 ;
60 ; usr(adr,iocb,adres)
70 ; gdzie adr,iocb sa zmiennymi
80 ; typu int i oznaczaja:
90 ; adr   - poczatek obszaru pamieci
RAM,

0100 ;        w ktorym rezyduje ten
podprogram
0110 ;iocb - numer IOCB, z ktorym
zostala
0120 ;        otworzona drukarka.
Podprogram nie
0130 ;        przeprowadza operacji
otwarcia.
0140 ;adres- adres pamieci ekranu,
ktorego
0150 ;        zawartosc ma byc
wydrukowana.

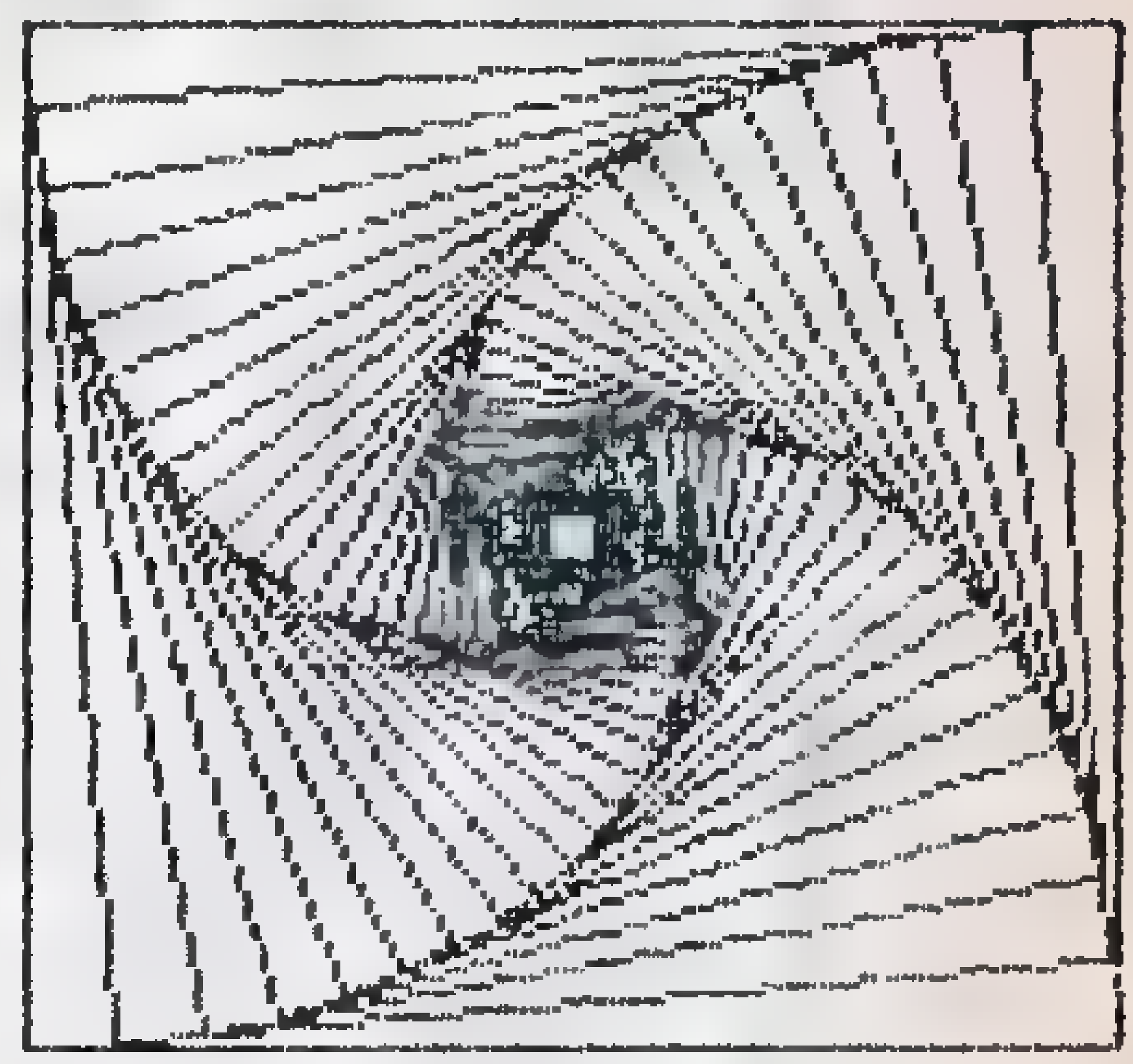
0152 ;        "adres jest wskaznikiem
typu char.
0160 ;UWAGA: podprogram wyprowadza
jedynie
0170 ;        280 bajtow pamieci ekranu
poczawszy
0180 ;        od adresu "adres"
0190 ;
0200 ;
0210 ;Wykorzystane zmienne systemu
operacyjnego

0220 ICBCOM = $0342
0230 ICBAL  = $0344
0240 ICBAH  = $0345
0250 ICBLL  = $0348
0260 ICBLH  = $0349
0270 CIOV   = $E456
0280 ;Zmienne podprogramu
```

```
0290 IOCB    = $06E0
0300 ADRES   = $06E2
0305 EKRAN   = $FFFF
0310 OBSZ    = $06E4
0320 ;
0330 ;Poczatek podprogramu
0340 ;
0350         *= $0600
0360 ;
0370 START   PLA     ;zwieksz wskaznik
stosu
0380         PLA     ;st. bajt nr IOCB
0390         PLA     ;ml. bajt nr IOCB
0400         ASL A   ;pomnoz
0410         ASL A   ;przez
0420         ASL A   ;szesnascie
0430         ASL A   ;
0440         STA IOCB
0450         LDA #0      ;zeruj starszy
bajt
0460         STA IOCB+1 ;numeru IOCB
0470         PLA     ;pobierz ADRES
0480         STA ADRES+1
                        PR1029 str.: 2

0490         PLA
0500         STA ADRES
0510 DALEJ   JSR AKTAD ;aktualizuj adresy
0520         CLC         ;i drukuj po 8
bajtow
0530         LDA PRZES+1
0540         ADC #1
0550         STA ADRES
0560         LDA PRZES+2
0570         ADC #0
0580         STA ADRES+1
0590         INC IOCB+1
0600         LDA IOCB+1
0610         CMP #40
0620         BNE DALEJ
0630 KONIEC  RTS   ;wydrukowano 280 bajtow
0640 AKTAD   LDX #0
0650 AKTAD1  CLC
0660         LDA ADRES+1
0670         STA PRZES+2,X
0680         LDA ADRES
0690         STA PRZES+1,X
0700         ADC #40
0710         STA ADRES
0720         LDA ADRES+1
0730         ADC #0
0740         STA ADRES+1
0750         CLC
0760         TXA
0770         ADC #6
0780         TAX
0790         CPX #37
0800         BCC AKTAD1
0810         LDX #0
0820 PRZES   ASL EKRAN
0830         ROL OBSZ,X
0840         ASL EKRAN
0850         ROL OBSZ,X
0860         ASL EKRAN
0870         ROL OBSZ,X
0880         ASL EKRAN
0890         ROL OBSZ,X
0900         ASL EKRAN
0910         ROL OBSZ,X
0920         ASL EKRAN
0930         ROL OBSZ,X
0940         ASL EKRAN
0950         ROL OBSZ,X
0960         INX
0970         CPX #8
0980         BNE PRZES
0982 ;
0984 ;drukuj osiem bajtow
0986 ;
0990 DRUK    LDX IOCB
1000         LDA #11
1010         STA ICBCOM,X
1020         LDA #OBSZ&255
1030         STA ICBAL,X
1040         LDA #OBSZ/256
1050         STA ICBAH,X
1060         LDA #8
1070         STA ICBLL,X
1080         LDA #0
1090         STA ICBLH,X
1100         JSR CIOV
                        PR1029 str.: 3

1110         RTS
1120         .END
```



*** OBLICZ ***

Ostatni z prezentowanych programów ilustruje sposób tworzenia i wykorzystania w języku C funkcji rekurencyjnych. Funkcja rekurencyjna jest to funkcja, która rozwiązując problem, sama siebie wywołuje. Program OBLICZ oblicza wartość złożonego wyrażenia matematycznego (ciąg liczb połączonych znakami operacji matematycznych np: 2.0+2.0 1+1000).

Dozwolonymi znakami operacji matematycznych są: +. -.*,:. Funkcją rekurencyjną jest funkcja obl(). Analizuje ona napisane na ekranie wyrażenie i po napotkaniu znaku operacji matematycznej wywołuje sama siebie w celu zanalizowania części wyrażenia, stojącego po prawej stronie operatora. Funkcja obl() wykonuje najpierw mnożenie, następnie dzielenie, następnie dodawanie, a na końcu odejmowania, stąd wartość wymienionego wyżej przykładowego wyrażenia obliczona przez funkcję obl() jest równa -997. Używając nawiasów (), można zmienić powyższą hierarchię działań.

Funkcje rekurencyjne są bardzo wygodnym narzędziem oferowanym przez język C. Dzięki nim można napisać krótką i zwartą funkcję, rozwiązującą jednak stosunkowo skomplikowany problem (proszę spróbowac napisać podobną do funkcji obl() procedurę w języku BASIC i porównać je). Pisząc funkcję rekurencyjną, należy zawsze szacować liczbę jej wywołań, konieczną do rozwiązania problemu, dla którego funkcja ta jest tworzona. Zbyt duża liczba wywołań tzw. "nieskończona rekurencja", może doprowadzić do "przepełnienia" stosu i zawieszenia się programu wynikowego.

Program OBLICZ

Program "OBLICZ"

Funkcja sterujaca main()

```
/**********************************
 *       PROGRAM "OBLICZ"         *
 * demonstruje dzialanie funkcji  *
 * rekurencyjnej w jezyku C       *
 **********************************/

main()
$(
char
wiersz[39],kom1[35],kom2[35],kom3[35];
char kom4[35];
char wynik[17],wynikf[6];
char *vram,*kom;
char c,c1;
int stat,dalej;
c_iml();
kom="Nieparzysta liczba nawiasow ()";
strcpy(kom1,kom);
kom="   Blad w podanym wyrazeniu   ";
strcpy(kom2,kom);
kom="Dlugosc przekroczyla 38 znakow";
strcpy(kom3,kom);
kom="   Chcesz liczyc dalej T/N    ";
strcpy(kom4,kom);
poke(752,1);
printf("\f\g");    /* czysc ekran */
pisz(5,1,"Podaj dowolne wyrazenie",0);
pisz(1,2,"zbudowane z liczb, nawiasow ()
oraz",0);
pisz(1,3,"symboli operacji
matematycznych tj.",0);
pisz(15,4,"+, -, *, :",0);
pisz(1,5,"mozesz uzywac liczb z kropka
dzies.",0);
pisz(15,7,"WYRAZENIE",128);
ryskon(0,9,39,11,'c');
pisz(15,13,"WARTOSC",128);
ryskon(10,15,31,17,'g');
vram=dpeek(88)+401;
/*
 * wczytaj wyrazenie
 */
dalej=1;
while(dalej)
 $(
  stat=wczytaj_w(vram,wiersz);
  if(stat)
   $(
    pisz(2,20,kom3,0);
    printf("\g");
    continue;
   $)
  stat=spr_naw(wiersz);
  if(stat)
   $(
    pisz(2,20,kom1,0);
    printf("\g");
    continue;
   $)
 /*
  * oblicz wartosc wyrazenia


 */
  stat=obl(wiersz,wiersz+strlen(wiersz)-
1,wynikf);
  if(stat)
   $(
    pisz(2,20,kom2,0);
    printf("\g");
    continue;
   $)
 /*
  * wyprowadz wartosc wyrazenia
  */
  c_fasc(wynikf,wynik);
  position(12,16);
  printf("                ");
  position(12,16);
  printf("%s",wynik);
  pisz(2,20,kom4,0);
  c=toupper(getchar());
  c1=getchar();
  if(c!='T') dalej=0;
 $)  /* koniec petli while */
$)   /* koniec funkcji main() */
```

Funkcja spr_znak()

```
#define DEL 126
#define RET 155

/**********************************
 *     funkcja spr_znak(znak)     *
 * sprawdza, czy wczytany znak jest*
 * jednym ze znakow DEL, RETURN, -,*
 * +,*,:,.,(,) lub cyfra. Jezeli  *
 * tak funkcja zwraca zero do     *
 * miejsca wywolania, w przeciwnym *
 * wypadku zwraca -1              *
 **********************************/

spr_znak(znak)
char znak;
$(
 if(znak==DEL) return(0);
```

```
  if(znak==RET) return(0);
  if(znak=='-') return(0);
  if(znak=='+') return(0);
  if(znak=='*') return(0);
  if(znak==':') return(0);
  if(znak=='.') return(0);
  if(znak=='(') return(0);
  if(znak==')') return(0);
  if(znak>='0' && znak<='9') return(0);
  return(-1);
$)
```

Funkcja wczytaj_w()

```
/**********************************
 *  funkcja wczytaj_w(adr,wier)    *
 * wczytuje wiersz (wyrazenie),    *
 * wartosc ma byc obliczona. Zwraca *
 * 0 gdy wiersz zostal wprowadzony *
 * poprawnie, -1 gdy jego dlugosc  *
 * przekroczyla 38 znakow          *
 **********************************/

wczytaj_w(adr,wier)
char *adr,*wier;
$(
 char znak;
 int czyt,klaw,rz,lpz,i;
/*
 * umieszczenie 38 znakow spacji w
inwersie
 * w pamieci obrazu od adresu adr
 */
 for(i=0;i<38;i++)
     poke(adr+i,128);
 klaw=copen("K:",'r');
 czyt=1;
 lpz=0;
 while(czyt)
  $(
if(lpz>37) return(-1);
znak=cgetc(klaw);
rz=spr_znak(znak);
if(rz)  /* gdy zly znak */
 $(
  printf("\g");
  continue;
 $)
if(znak==DEL && lpz>0)
 $(
  adr--;
  wier--;
  lpz--;
  poke(adr,128);
  continue;
 $)
if(znak==DEL && lpz==0)
 $(
  printf("\g");
  continue;
 $)
if(znak==RET)
 $(
  *wier='\0';
  cclose(klaw);
  return(0);
 $)
poke(adr,(znak-32)|128);
*wier=znak;
adr++;
wier++;
lpz++;
  $)    /* koniec petli while */
$)      /* koniec funkcji wczytaj_w() */
```

Funkcja spr_naw()

```
/**********************************
 *     funkcja spr_naw(wiersz)     *
 * sprawdza, czy w podanym wyrazeniu*
 * wystepuje parzysta liczba nawia- *
 * sow zamykajacych ')' i otwieraja-*
 * cych '('. Zwraca 0 gdy jest tak  *
 * a -1 w przeciwnym wypadku        *
 **********************************/

spr_naw(wiersz)
char *wiersz;
$(
int lnaw;
lnaw=0;
while(*wiersz)
  $(
   if(*wiersz=='(') lnaw++;
   if(*wiersz==')') lnaw--;
   wiersz++;
  $)
if(lnaw) return(-1); else return(0);
$)  /* koniec funkcji spr_naw() */
```

Funkcja obl()

```
/**********************************
 *  funkcja obl(pocz,kon,liczba)   *
 * funkcja rekurencyjna obliczjaca *
 * wartosc wyrazenia.              *
 * argumenty:                      *
 * pocz - wskaznik do poczatku     *
 *        tablicy znakowej zawiera-*
 *        jecej wyrazenie          *
 * kon  - wskaznik do konca wyraze-*
 *        nia; kon=pocz+dlugosc-1  *
 * liczba-wskaznik do tablicy prze-*
 *        znaczonej na wartosc wyr.*
 **********************************/

 obl(pocz,kon,liczba)
 /* zmienne zewnetrzne */

 char *pocz,*kon,*liczba;

 $(
 char liczba1[6],liczba2[6];
 char *i;
 int poziom;

 poziom=0;
 if(pocz>kon+1) return -1;
 if(pocz==kon+1)$(
   c_zero(liczba);
   return 0; $)
 for(i=pocz;i<=kon;i++)$(
   if(*i=='(')$(
     poziom++;
     continue; $)
   if(*i==')')$(
     poziom--;
     continue; $)
   if(!poziom && (*i=='+' || *i=='-'))$(
     if(obl(pocz,i-1,liczba1)) return
-1;
     if(obl(i+1,kon,liczba2)) return -1;
     if(*i=='+') return
```

```
c_fadd(liczba1,liczba2,liczba);
      else return
c_fsub(liczba1,liczba2,liczba);
   $)
 $) /* for */
 if(poziom) return -1;
 for(i=pocz;i<=kon;i++)$(
   if(*i=='(')$(
     poziom++;
     continue; $)

   if(*i==')')$(
     poziom--;
     continue; $)
   if(!poziom && (*i=='*' || *i=='/'))$(
     if(obl(pocz,i-1,liczba1)) return
-1;
     if(obl(i+1,kon,liczba2)) return -1;
     if(*i=='*') return
c_fmul(liczba1,liczba2,liczba);
     else return
c_fdiv(liczba1,liczba2,liczba);
   $)
 $) /* for */
 if((*pocz=='(')&&(*kon==')')) return
obl(pocz+1,kon-1,liczba);
 return c_afp(pocz,liczba);
 $) /* koniec funkcji obl() */
```

Funkcja ryskon()

```
/************************************
 *       FUNKCJA ryskon()           *
 *   rysuje kontur - ramke          *
 * argumenty funkcji:               *
 * x1,y1 - lewy gorny rog ramki     *
 * x2,y2 - prawy dolny rog ramki    *
 * rk    - rodzaj kreski (linii),   *
 *         ktora ma byc narysowana  *
 *         ramka                    *
 * rk ='c' linia cienka             *
 * rk ='g' linia gruba              *
 ************************************/

/* kody znakow uzytych do
 * kreslania konturu - ramki
 *       linia gruba
 *
 * lewy gorny rog            ATASCII 137
 * kreska pozioma gorna      ATASCII 149
 * kreska pozioma dolna      ATASCII 21
 * prawy gorny rog           ATASCII 143
 * kreska pionowa lewa       ATASCII 25
 * kreska pionowa prawa      ATASCII 153
 * lewy dolny rog            ATASCII 139
 * prawy dolny rog           ATASCII 140
 *
 *       linia cienka
 *
 * lewy gorny rog            ATASCII 17
 * kreska pozioma gorna      ATASCII 18
 * kreska pozioma dolna      ATASCII 18
 * prawy gorny rog           ATASCII 5
 * kreska pionowa lewa       ATASCII 124
 * kreska pionowa prawa      ATASCII 124
 * lewy dolny rog            ATASCII 26
 * prawy dolny rog           ATASCII 3
 */

 ryskon(x1,y1,x2,y2,rk)
 int x1,y1,x2,y2;
 char rk;
  $(
   int i;
   char lgr,kpg,kpd,pgr,kpl,kpp,ldr,pdr;
   switch(rk)
    $(
     case 'c' : lgr=17; /* kreska cienka
  */
              kpg=18;
              kpd=18;
              pgr=5;
              kpl=124;
              kpp=124;
              ldr=26;
              pdr=3;
              break;
     case 'g' : lgr=137; /* kreska gruba
  */
              kpg=149;
              kpd=21;
              pgr=143;
              kpl=25;
              kpp=153;
              ldr=139;
              pdr=140;
              break;
    $)    /* koniec switch */
   position(x1,y1);
   putchar(lgr);
   for(i=0;i<x2-x1-1;i++) putchar(kpg);
   putchar(pgr);
   for(i=y1+1;i<y2;i++)
    $(
     position(x1,i); putchar(kpl);
     position(x2,i); putchar(kpp);
    $) /* koniec petli for        */
   position(x1,y2);
   putchar(ldr);
   for(i=0;i<x2-x1-1;i++) putchar(kpd);
   putchar(pdr);
  $)   /* koniec funkcji ryskon() */
```

Funkcja pisz()

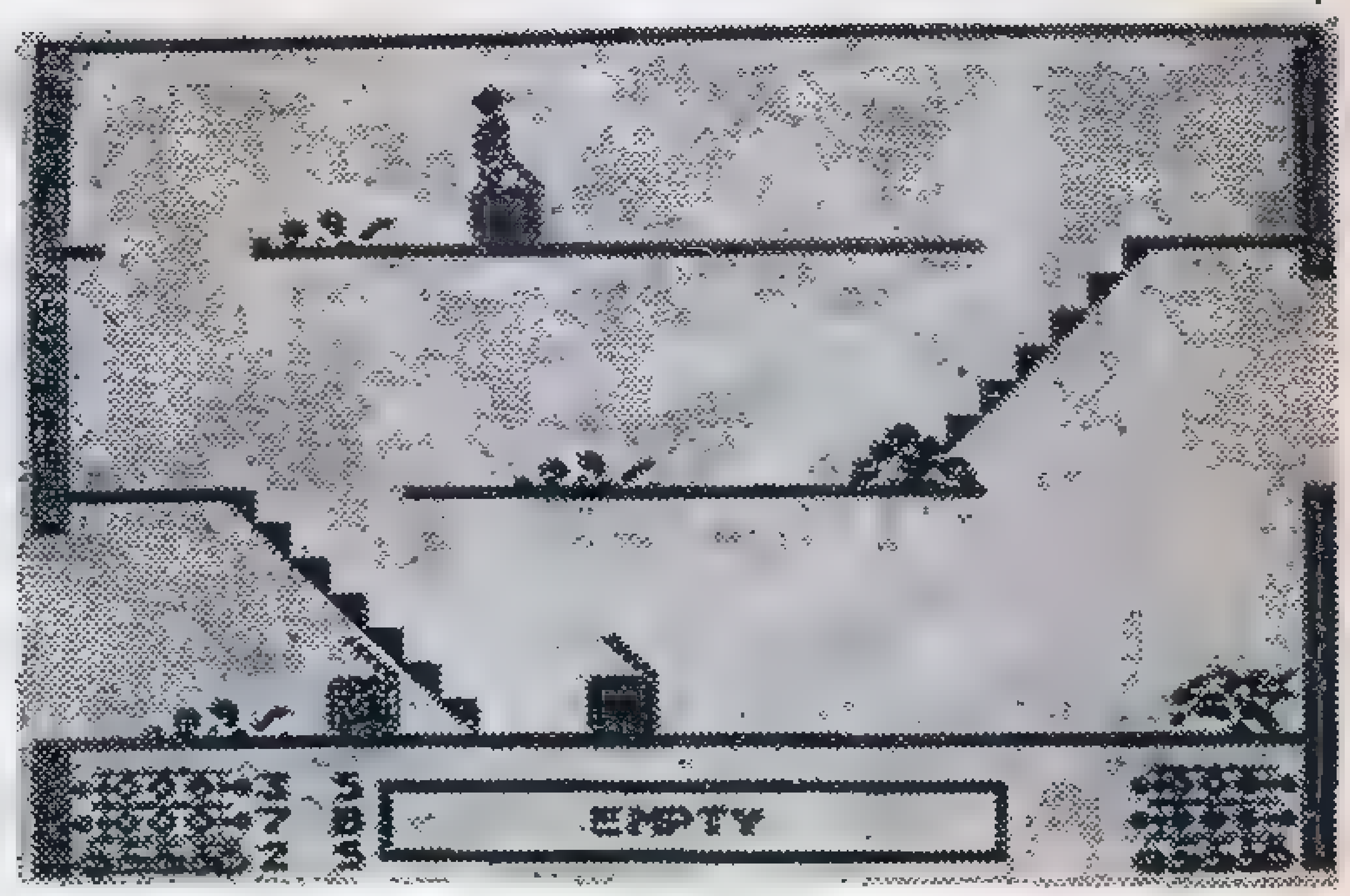
```
/************************************
 *      FUNKCJA pisz()              *
 * wyswietla na ekranie lancuch zna- *
 * kow w miejscu okreslonym argumen- *
 * tami x,y.                        *
 * Ponadto jesli argument inv=128   *
 * to lancuch wyswietlany jest w    *
 * inwersie, inna wartosc inv nie   *
 * zmienia postaci wyswietlanego    *
 * lancucha.                        *
 * tekst - wyswietlany lancuch      *
 ************************************/

pisz(x,y,tekst,inv)
int x,y,inv;
char *tekst; /* wskaznik do lancucha */
$(
 char *kont; /* wskaznik do konca
              * lancucha           */
 int z;
 z=strlen(tekst);
 kont=tekst+z;
 position(x,y);
 for(;tekst<kont;tekst++)
  if(inv==128) putchar(*tekst|inv);
   else putchar(*tekst);
$)   /* koniec pisz */
```

AZTEC

Stara budowla z labiryntami tuneli ciągnących się bez końca to magnes przyciągający ludzi żądnych przygód, sławy a także bogactwa. Nie zważają na klątwy rzucone w zamierzchłych czasach przez kapłanów na intruzów, mogących zakłócić spokój prochom władców pochowanych w tych mauzoleach. Przykładów na to, że nie jest to znowu taka sobie fikcja, mamy wiele. Ginący w niewyjaśnionych okolicznościach odkrywcy grobowców faraonów egipskich, uśmiercani przez dziwnych strażników awanturnicy, odnajdujący (w poszukiwaniu złota) miejsca kultu religijnego Indian amerykańskich, przypadki podobnych zdarzeń w Azji, Środkowej Afryce i Europie. Sądzę, że zaskoczeniem będzie sytuacja, jaka wytworzy się w momencie pojawienia się tego programu w Twoim mieszkaniu. Niebezpieczeństwo skarcenia Ciebie przez rodziców, np. z takiego powodu, że lekcje jeszcze nie odrobione lub kolacja już wystygła, jest znaczne. Do tego dochodzi również znaczne ryzyko " utraty życia " na skutek nieostrożności lub zbyt małej wprawy w poruszaniu się po podziemiach.

Po uruchomieniu programu ukaże Ci się menu. Dzięki niemu możesz urozmaicić i tak już bogatą w przygody grę. Początkowe litery określają, jaki klawisz należy nacisnąć aby uruchomić daną opcję. W przeciwieństwie do wielu programów (jeżeli oczywiście podołasz zadaniu) możesz wybrać wszystkie opcje jednocześnie. Nie zalecałbym jednak tego. Szczególnie przydatna opcja jest pierwsza (**ETERNAL STRENGTH**). Dzięki niej stajesz się niezniszczalny. Aby przejść do następnej części programu należy nacisnąć klawisz **START**. Komputer zapyta się wtedy o stopień trudności w skali od 1 do 8. Naciśnij któryś z tych klawiszy, a rozpocznie się przygoda. Na dole po lewej i prawej stronie ekranu umieszczone są Twoje zasoby i znaleziska: bomba (**STICKS**) - w momencie rozpoczęcia gry masz ich trzy. Maksymalna ilość, jaką możesz ich zgromadzić, to siedem; naboje (**BULLETS**) musisz, niestety szukać, zresztą tak samo jak pistoletu (**PISTOL**), maczety (**MACHETE**) czy bożka (**THE IDOL**). Przy odrobinie szczęścia możesz znaleźć również dzban z życiodajną wodą, dzięki czemu powiększy się Twój zapas siły (**STRENGTH**). Wszystkie te przedmioty ukryte będą w skrzyniach lub pod stertami śmieci. Aby sprawdzić czy one tam się znajdują, należy zatrzymać się, następnie nacisnąć przycisk raz, a wieko skrzyni otworzy się oraz ukaże się napis, co jest w środku, np. pusta (**EMPTY**), czaszki (**SKULLS**). Naciśnięcie jeszcze raz przycisku zapisuje przedmiot (jeżeli był) na Twoje konto. Sterte śmieci należy po prostu zebrać, naciskając przycisk manipulatora dotąd, aż miejsce nie oczyści się i pokaże czy coś się tam znajduje. Gdy tak będzie, naciśnij przycisk jeszcze raz, a staniesz się właścicielem tego przedmiotu. Jest, oczywiście, od tego odstępstwo, to znaczy bomba z zapalonym lontem. Zostaniesz ostrzeżony napisem (**DID WERE BLOWN UP !**). Odsuń się wtedy na bezpieczną odległość i poczekaj, aż nastąpi eksplozja.



Awanturnik, w którego się wcielasz może wykonywać szereg czynności. Wszystkie są wykonywane przy pomocy manipulatora. Poznanie i zapamiętanie uruchamiania poszczególnych czynności ułatwi Ci penetrację podziemi, a także uchroni przed czyhającymi niebezpieczeństwami. Oto niektóre z nich:

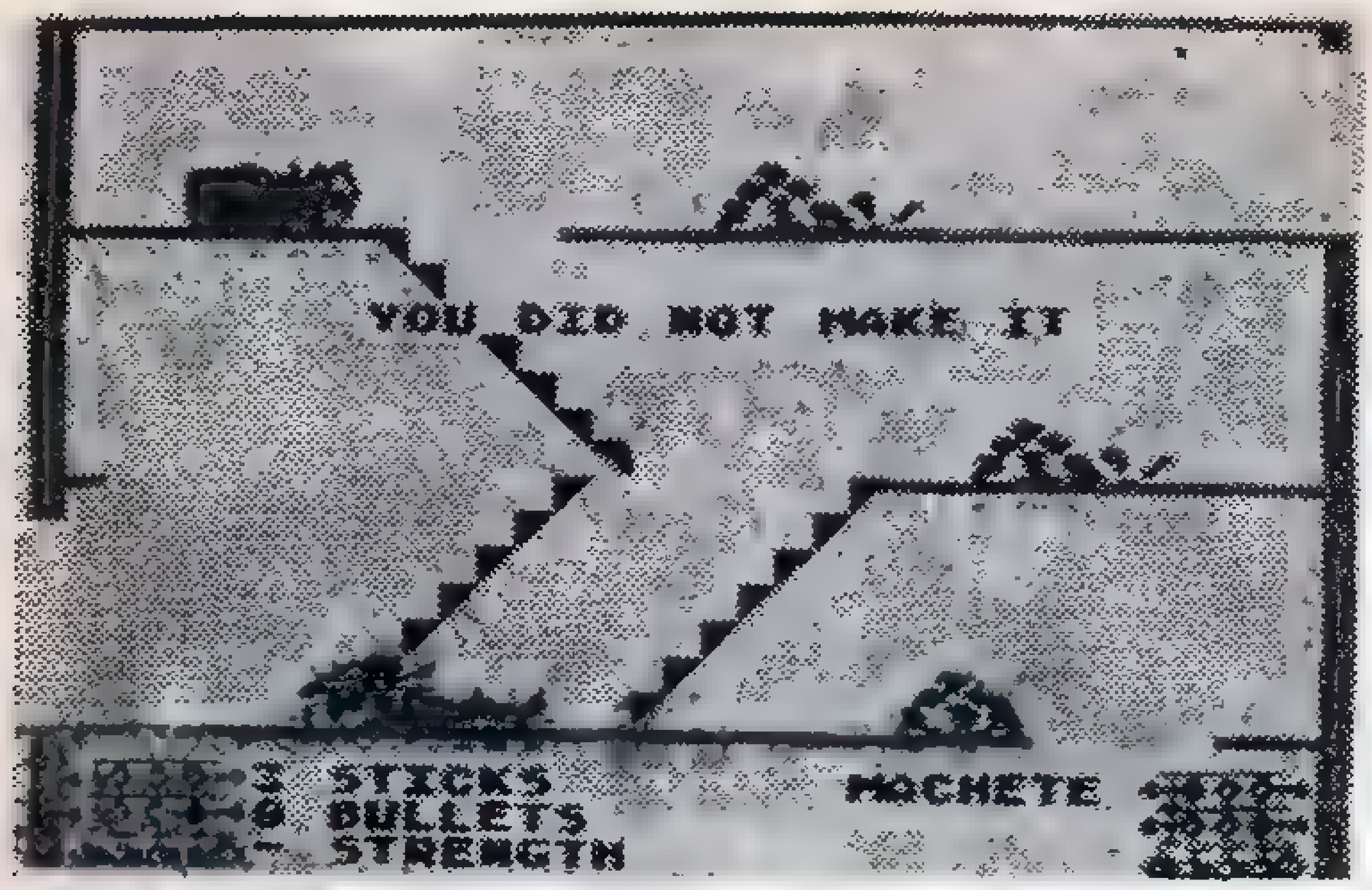
- Skierowanie manipulatora w lewo powoduje ruszenie w tę stronę postaci (kowboja).
- Skierowanie manipulatora w prawo - kowboj rusza w prawo.
- Drążek manipulatora do góry - kowboj zatrzymuje się lub w przypadku, gdyby klęczał, wstaje.
- Drążek manipulatora do dołu - kowboj przykuca.
- Przytrzymanie drążka podczas poruszania się kowboja w lewo lub prawo sprawia, że zaczyna biec.
- Gdy kowboj idzie, a Ty naciśniesz przycisk manipulatora, zostanie przez niego wykonany skok.
- Gdy kowboj jest przykucnięty, a drążek skierujemy w lewo lub prawo, kowboj rozpocznie poruszać się na czworakach w daną stronę.
- Na schody będziesz mógł wejść, gdy skierujesz drążek manipulatora pod kątem do góry.

We wszystkich swoich wędrówkach po korytarzach, gdy dopisze Ci szczęście, możesz znaleźć między innymi bombę, pistolet lub maczetę. Przedmioty te bardzo się przydadzą. Używa się ich w następujący sposób:

- Aby postawić bombę, kowboja należy ustawić w pozycji klęczącej z wyciągniętą ręką, przodem do miejsca, gdzie ma być podłożona bomba, a następnie nacisnąć przycisk manipulatora. Gdy pojawi się bomba z palącym się lontem, należy wyjść poza miejsce rażenia odłamkami bomby, ale pozostając w tym samym pomieszczeniu.

Stawianie bomby przydaje się do przebijania ścian lub posadzek w celu przedostania się do innego tunelu. Czasami mogą one uratować Ciebie od niechybnej śmierci, gdy wejdziesz do tunelu, z którego nie ma wyjścia.

- Gdy masz pistolet (bo znalazłeś go np. w skrzyni) i chcesz go użyć , sprawdź w pierwszej kolejności , czy posiadasz do niego naboje (**BULLETS**) . Informacje o



nich masz w lewej dolnej części ekranu . Skierowanie manipulatora do góry i jednoczesne naciśniecie przycisku sprawi, że kowboj stanie w pozycji gotowej do oddania strzału .

- Oddanie strzału odbywa się przez naciśniecie przycisku manipulatora .

- Aby schować pistolet , należy skierować drążek manipulatora do góry oraz nacisnąć przycisk . Z wyciągniętą bronią kowboj może poruszać się tylko w poziomie .

- Gdy masz maczetę i chcesz jej użyć , skieruj drążek manipulatora do góry i naciśnij jednocześnie przycisk. Bronią tą możesz zadawać dwa rodzaje ciosów . Pierwszy to pchnięcie na wprost z wysunięciem nogi, wygląda to jak atak szpadą . Wykonuje się go przez naciskanie przycisku. Drugi to cięcie obronne w dół , szczególnie przydatny podczas ataku małych zwierząt ; należy w tym przypadku skierować uchwyt manipulatora do siebie .

- W przypadku , gdy posiadasz zarówno pistolet jak i maczetę , do walki możesz używać jednej z tych broni . Gdy walczysz maczetą i w pewnej chwili dochodzisz do wniosku, że lepiej użyć jest pistoletu, skieruj drążek manipulatora do góry , a nastąpi zmiana broni .

W czasie przeszukiwania korytarzy napotkasz wiele groźnych zwierząt ; lwa , zwierzę prehistoryczne , ośmiornice , a także Indianina będziesz mógł " uspokoić " , strzelając do nich z pistoletu . Kobrę i tygrysa możesz uśmiercić z pistoletu , jak również siekąc maczetą . Pozostałych zwierząt możesz pozbyć się przy pomocy maczety . W przypadku , gdy nie dysponujesz jeszcze żadną bronią , jedyną radą jest ucieczka lub jeżeli jest to : wąż , skorpion lub mały pająk , możesz po prostu je przeskoczyć . Długość korytarzy i liczba ich poziomów jest ograniczona . Stąd i ilość atrakcji ma swoja granicę . Czas, który uda Ci się w nich spędzić, wystarczy aż nadto , aby zaspokoić Twoją potrzebę wrażeń i przyjaciół . Gdy ukaże się napis **YOU DID NOT MAKE IT** , będzie oznaczało , że już nic w tej grze nie zrobisz, a więc możesz wcisnąć klawisz **START** i rozpocząć grę od nowa .

DIMENSION X

Nie będę rozpisywał się , jak to się stało , że na Twą niedużą planetę o nazwie Jaraloba "zawitali" wrogowie . Skoro jednak tak się stało , siadaj za sterami . Nim wystartujesz , przypomnij sobie kilka istotnych informacji .

Po pierwsze · które klawisze oraz do czego są używane w grze . Wierzę ,że odpowiesz następująco :

OPTION -uzyskanie na ekranie menu,
SELECT - wybór opcji w menu ;
START - start programu ;
HELP - przejście od gry do menu;
SPACE - zatrzymanie gry . Powtórzenie tego klawisza ponownie uruchamia program (kontynuowanie gry) ;

Po drugie : tablica przyrządów zawiera szereg wskaźników . Wszystkie są ważne. Rozmieszczenie ich przedstawia poniższy rysunek .

TABLICA PRZYRZĄDÓW :

1 -Tablica komunikatów komputera pokładowego ;
2 -Informacja o ilości ekranów ochronnych ;
3-Wskaźnik ilości energii statku;
4 -Wskaźnik odległości od nieprzyjaciela ;
5 - Ekran radaru ;
6 - Mapa sektorów planety .

W celu ułatwienia posługiwania się poszczególnymi wskaźnikami zostaną one kolejno omówione .

Podczas lotu należy zwracać uwagę na tablicę komunikatów . Ukazujące się na niej informacje w istotny sposób mogą wpłynąć na całokształt gry, jeżeli będziesz się do nich dostosowywał . A oto ich wykaz wraz z objaśnieniami :

- AUTOMATIC BOOSTERS ENGAGED - automaty włączyły silniki biplazmowe. Komunikat ten ma miejsce tylko podczas przechodzenia statku przez sektory , w których nic nie występuje . Ma to na celu szybsze przejście przez sektory . W czasie działania tych silników wyłączone są ekrany ochronne .

- BOOSTERS DISENGAGED - nastąpiło wyłączenie silników biplazmowych . Można do tego doprowadzić przez poruszanie drążka manipulatora .

- DESORT MAP HIT - nastąpiło uszkodzenie mapy .

- EDGE OF SECTOR AHEAD - pojazd zbliża się do końca sektora . Równocześnie z tym komunikatem uruchamiany jest sygnał dźwiękowy.

- END OF TUNNEL AHEAD - statek zbliża się do końca lotu przez dziesiąty wymiar . Tak jak w przypadku zbliżania się do końca sektora , tak i tu dodatkowo informowani jesteśmy sygnałem dźwiękowym .

- FUEL TANKS HIT - przeciwnik usz-

łodził zbiorniki paliwa . Paliwo wydobtaje się z nich , co grozi katastrofą .

- LOW ON FUEL -zaczyna wyczerpywać się paliwo . Dodatkowym ostrzeżeniem jest sygnał dźwiękowy .
RIGILLAN APPROACHING - do statku

Odporność uderzeniowa ekranów ochronnych dzieli się na trzy grupy :
- silne (STRONG) wytrzymujące trzy trafienia z działa przeciwnika ;

- normalne (NORMAL) wytrzymujące dwa trafienia z działa ;

uszkodzenie , naprawi je , o czym zostaniesz powiadomiony . Gdy statek zostanie doprowadzony do pełnej zdolności bojowej , rozpoczyna się odliczanie czasu , po upływie którego statek Twój zostanie wystrzelony do migającego pomarańczowego sektora. Jeżeli chcesz być przeniesiony w inny niż wskazany sektor , ustaw jego nowe położenie przy pomocy manipulatora . Musisz jednak zdążyć z tą czynnością w czasie odliczania .

W czasie lotu przez sektor możesz napotkać statki wroga . Zniszczenie ich jest Twoim obowiązkiem. Przed dojściem do styczności wzrokowej znasz odległość do nieprzyjaciela dzięki wskaźnikowi odległości . Odległość do wroga jest o tyle istotna , że możesz dokonać manewru wyprzedzającego ich uderzenie .
Liczba ujemna na liczniku informuje, że statki nieprzyjaciela znajdują się za Tobą .



zbliża się nieprzyjaciel .
- SCANNER HIT - nastąpiło uszkodzenie , a tym samym unieruchomienie głównego czujnika (scannera) .
- SECTOR EMPTY - komunikat ten informuje , że w sektorze , w którym aktualnie się znajdujesz nie ma nic .
- SECTOR SECURED - chwilowo w sektorze jest bezpiecznie , mimo że znajdują się w nim statki przeciwnika .
- SHIELD DIENTEGRATING -ze względu na trafienia osłabiona jest moc ekranu ochronnego .
- THE CAPITOL IS SURROUNDED - wróg otoczył stolicę . Jeżeli go teraz nie zniszczysz , wedrze się do miasta , a tym samym będzie to oznaczało Twój koniec .
- WELCOME ABOARD WARRIOR - komunikat ten występuje na początku gry. Jest to grzecznościowy zwrot powitania na pokładzie statku .
- ZONE DEFLECTION - Twój statek znajduje się w strefie neutralnej. Tracisz jeden poziom ekranu ochronnego .

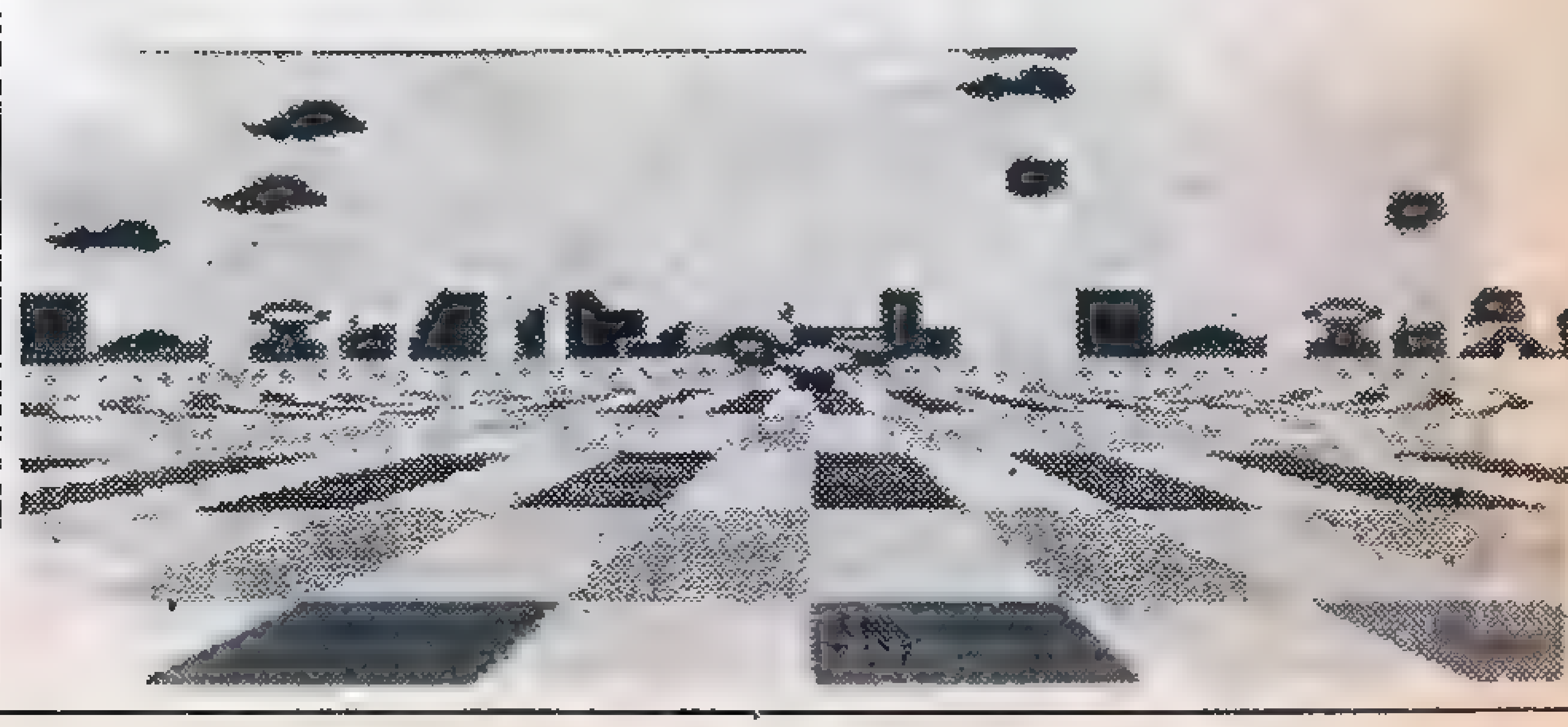
Wskaźnik informujący o ilości ekranów ochronnych da Ci początkowo luksus wzrokowego bezpieczeństwa , lecz w miarę , jak będziesz inkasował celne strzały wroga , spojrzenia "kierowane" na ten element tablicy będą coraz bardziej nerwowe .

- słabe (WEAK) wytrzymujące tylko jedno trafienie z działa przeciwnika .
Rozpoczynając grę , dysponujesz pięcioma ekranami : zielonym , żółtym , czerwonym , krytycznym i zerowym . W trakcie gry , gdy otrzymujesz trafienia od przeciwnika , przechodzisz na coraz niższe poziomy ekranów ochronnych. Aktualną sytuację otrzymujesz na tablicy przyrządów .

Energia Twojego statku w miarę lotu powoli wyczerpuje się . Będziesz mógł to obserwować na wskaźniku ilości energii statku . Gdy zauważysz , że zapas jej wyczerpuje się , odszukaj na mapie sektor, w którym znajduje się baza paliwa . Spróbuj do niej dotrzeć . Gdy uda Ci się dobrnąć do sektora, w którym ona się znajduje , pojawi się na ekranie cały szereg informacji . Komputer kolejno sprawdza zespoły statku , jeżeli wykryje

Podobną rolę do wskaźnika odległości pełni radar pokładowy (QUADRA SCANNER).
Obraz na monitorze radaru pokazuje rozmieszczenie statków przeciwnika (błyskające białe kropki) w stosunku do Twojego pojazdu (żółta kropka).

Podstawowej informacji na temat położenia Twojego statku na planecie udziela mapa sektorów. Składa się ona z 25 pól (5x5) . Wyświetlane są na niej informacje ważne dla dalszych Twoich poczynań . Poznaczane są różnymi kolorami , literami i cyframi .
- Pole koloru **pomarańczowego** szybko pulsujące z czarną ramką - oznacza sektor, w którym właśnie się znajdujesz .
- Pole koloru **pomarańczowego wolno pulsujące** - wskazuje sektor , do którego zostaniesz przeniesiony .



- oznacza sektor pus-

...eskie - jest wskaźni...ra, który do tej pory ...ał jeszcze zbadany. Stąd ...o nim jakichś konkretnych ...domości.

...ole białe w kropki - przedstawia ...obą strefę neutralną.
- Litera C na polu - ozn. sektor, w którym znajduje się stolica.
- Litera F na polu - określa miejsce lokalizacji bazy paliwowej dla Twego statku.

Cyfra na polu - informuje, ile statków nieprzyjaciela znajduje się w danym sektorze.

Stopień utrudnienia gry możesz sam ustalić tak, byś nie zniechęcił się do niej po paru minutach zabawy. Czynności tych powinieneś dokonać przed uruchomieniem gry. Służy do tego menu, w którym występują trzy opcje:
- Wyszkolenie pilota (PILOT SKILL). Dzieli się na: nowicjusza (novice), zaawansowany (advanced) i mistrz (expert);
- Odporność uderzeniowa ekranów ochronnych (SHIELD STRENGTH) została omówiona wcześniej;
- Liczba przeciwników w sektorze (ALIENS PER SECTOR). Masz możliwość wyboru jednego z trzech wariantów: niewielu (few), normalna liczba (normal) i wielu (many).

Szczególnie w pierwszych podejściach do gry nie za bardzo będziesz zadowolony ze swoich umiejętności. Niech Cię to nie zraża, spróbuj jeszcze raz.

Gdy nie uda Ci się przebyć przez dziesiąty wymiar (rozbijesz się) lub zestrzeli Cię wróg, otrzymasz na ten temat stosowny komunikat. Dowiesz się z niego między innymi o tym, że planeta Jaraloba, tc jest ta, której broniłeś została opanowana przez Rigillians, niestety, z Twojej winy. Poinformowany zostaniesz również o ocenie, jaką wystawił Ci za całość zmagań komputer np. NOVICE WARRIOR CLASS 2.

SPY vs SPY

Wyścig zbrojeń trwa i taki stan będzie istniał, dopóki zło nie zostanie wyplenione z myśli i czynów ludzi. Powtarzanie tego zdania nie robi już wrażenia, gdyż jest ono prawdą powtarzaną od wieków. Przytaczam je dlatego, że będzie ono po części dotyczyło i Ciebie. Ministerstwo do spraw wojskowych w Twojej ojczyźnie prowadzi doświadczenia z nowym typem rakiet międzykontynentalnych. Jest to odpowiedź na takie same doświadczenia prowadzone przez państwo, mające złe zamiary w stosunku do Twej ojczyzny. Jak zwykle zdarzają się próby nieudane. Tak właśnie się stało. Rakieta po przelocie nad oceanem kilkuset kilometrów nieoczekiwanie spadła na niewielką tropikalną wyspę należącą do Archipelagu Smoka, rozpadając się na trzy części. Natychmiast zostały podjęte środki mające na celu odzyskanie rakiety, a także uniemożliwienie dostępu do niej szpiegów wroga. Jako fachowiec wysokiej klasy otrzymujesz zadanie odzyskania rakiety. W tym celu zostajesz przetransportowany na archipelag. Na wyspę dostajesz się, zostając zrzuconym na spadochronie. Jednocześnie w kierunku wyspy została wysłana łódź podwodna do odebrania rakiety, oczywiście po znalezieniu i zmontowaniu wszystkich części przez Ciebie. Przeciwnik też nie śpi. Wysłał na wyspę szpiega, zrzucając go na spadochronie z takim samym zadaniem, z jakim Ty się tam znalazłeś. Obaj jesteście wyposażeni w najnowocześniejszy sprzęt, jaki wyprodukowano dla prowadzenia działalności szpiegowskiej. Będziecie walczyć ze sobą oraz z czasem, gdyż na wyspie znajduje się wybuchający co pewien czas wulkan. Wybuch jego zniszczy wyspę, a więc uniemożliwi wykonanie zadania.

Po uruchomieniu programu zobaczysz po prawej stronie zestaw opcji:
- Zegar podający czas do erupcji wulkanu;
- Liczba graczy (**PLAYERS**) 1 lub 2;
- Poziom trudności (**LEVEL**) od 1 do 7;
- Stopień "inteligencji" komputera (**I.Q.**) od 1 do 5;
- Ujawnienie łodzi podwodnej tak (Y) lub nie (N).

Opcje wybiera się przez ruch manipulatora do lub od siebie. Wybrana opcja pulsuje. Poziom trudności wybiera się przez skierowanie manipulatora w lewo lub prawo. Gdy uznasz, że wszystkie elementy ustawiłeś właściwie, naciśnij przycisk manipulatora lub klawisz spacji i będziesz mógł rozpocząć grę.

W czasie gry będziesz mógł, a nieraz i musiał, używać niektórych klawiszy:

A - ruch w górę lub w dołu;
Z - ruch do dołu lub do przodu;
K - ruch w lewo;
L - ruch w prawo;

Q - włączenie urządzenia przenoszącego w inny wymiar;
S - włączenie lub wyłączenie muzyki;
ESC - zatrzymanie programu oraz powtórne jego uruchomienie;
SPACE - rozpoczęcie gry, podniesienie lub opuszczenie przedmiotu, wybór i odbezpieczenie miny;
RESET - ponowne ładowanie programu;

OPTION - powrót programu do ekranu z opcjami.

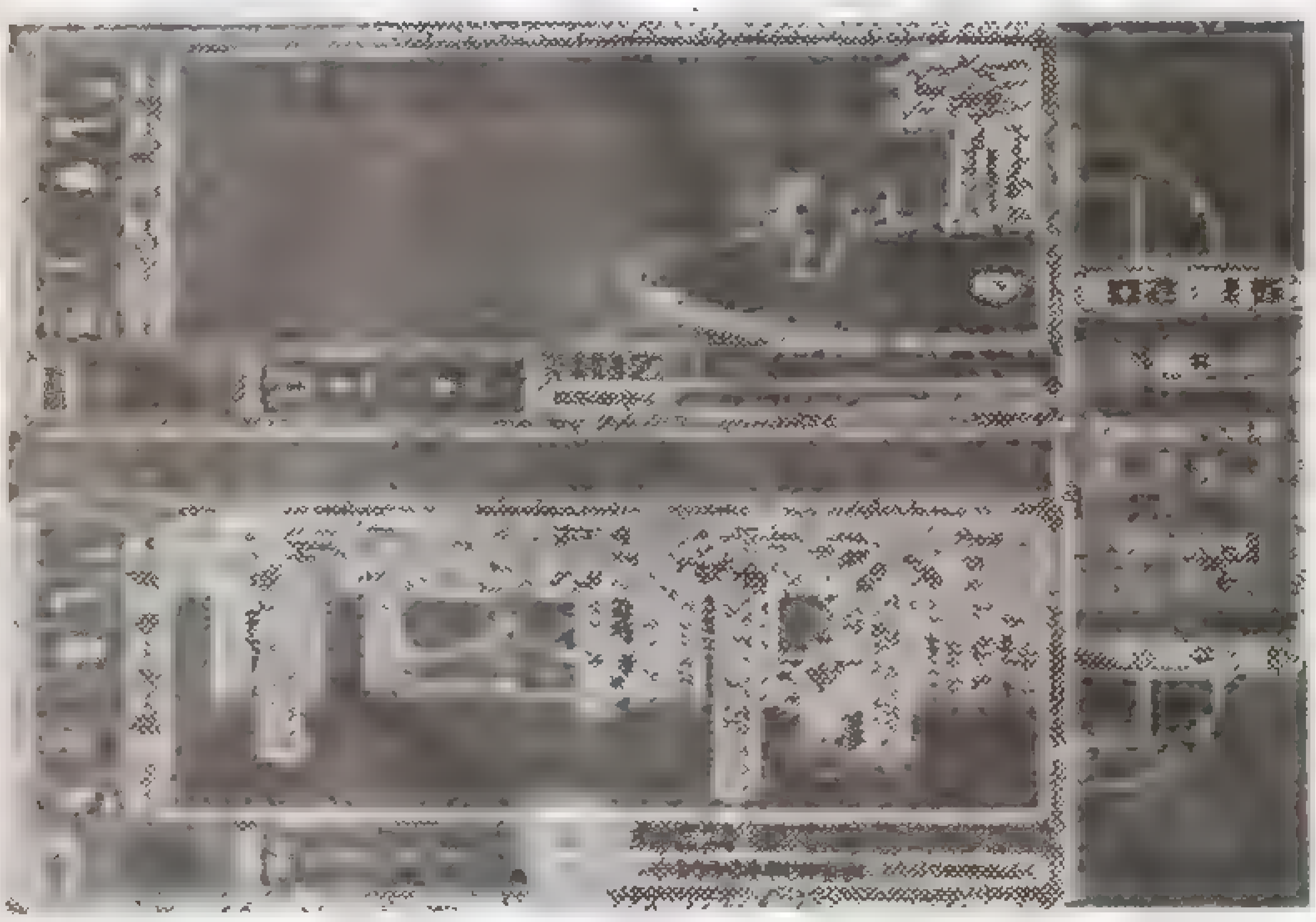
Niektórych klawiszy używa się również przy wciśniętym jednocześnie klawiszu CONTROL, a mianowicie: (podczas zadawania ciosów szablą)
Klawisz **A** - podniesienie ręki z szablą do góry;
Klawisz **K** - ręka do siebie;
Klawisz **L** - pchnięcie szablą;
Klawisz **Z** - cięcie szablą.

Bardzo ważnym przedmiotem w grze jest urządzenie, przy pomocy którego możesz przechowywać przedmioty znalezione na wyspie. Działa ono na zasadzie przenoszenia w inny wymiar i z powrotem. W ten sposób możesz gromadzić wszystko, co przedstawiają symbole po lewej stronie okna. Przenoszenie w inny wymiar odbywa się za pomocą przycisku manipulatora. Pierwsze przyciśnięcie powoduje podniesienie przedmiotu, drugie przeniesienie do innego wymiaru. Obecność w nim tego przedmiotu sygnalizowana jest zapalonym wskaźnikiem obok symbolu. Użyć któregoś z przed-

lotów można również przez naciśnięcie przycisku manipulatora następnie wybranie przez ruch drążkiem do dołu lub góry i ponownym naciśnięcia jego przycisku . Szczególnym przedmiotem w tym wykazie jest mapa wyspy . Poruszając się po wyspie w poszukiwaniu części rakiety , możemy w każdej chwili skorzystać z tej mapy , aby zlokalizować ich miejsca . Mapa będzie dzieliła wyspę na sektory . Ilość sektorów , których może być do 22, będzie zależna od wielkości wyspy , a więc od stopnia trudności. Części rakiety będą widoczne na wyspie w postaci białych kwadracików . Będzie również zaznaczona twoja droga .

Części rakiety będą leżały na wyspie ukryte pod pryzmami piasku, wyglądającymi każda jak trzy małe trójkąty. Dopiero ich podniesienie ujawni nam przedmiot , jaki znajduje się pod nimi . Nieraz może to być przykre, gdyż może okazać się, że odkryty przedmiot to mina która za chwilę wybuchnie . Warto dlatego zaglądać do mapy .
Również czas skłania do tego, gdyż na odnalezienie i wywiezienie rakiety jest go niewiele . W zależności od stopnia trudności czas ten do wybuchu wulkanu wynosi (licząc od stopnia najłatwiejszego) : 6, 8, 11, 14, 18 za 27 minut. Za stopniem trudności związanych jest również wiele zmian , gdyż ma się do czynienia z coraz większą wyspą, ilością przedmiotów na wyspie , a także ilością wysp .

Poruszanie się po wyspie można, oczywiście , realizować także za pomocą manipulatora. Gdy dojdziesz do lewego lub prawego skraju ekranu , następuje jego zmiana (pokazany jest kolejny sektor wyspy). Inaczej ma się sprawa z przechodzeniem sektorów w górę lub dół . Nie w każdym sektorze możesz to zrobić . Umożliwią Ci to sektory , w których będą większe luki między drzewami lub przerwa w dolnej części okna .



Na wyspie czyha na Ciebie szereg pułapek wymienione wcześniej zranienie przez granat ; można wpaść w bagno (miejsca o ciemniejszym tle) . Wydostać możesz się z niego przez szybki pionowy ruch manipulatora lub kręcenie nim w kółko . Czeka Cię jednak pewna utrata czasu .
Możesz wpaść również w pułapkę , którą sam założyłeś .
Gdy złożysz całą rakietę , przyjdzie kolej na przetransportowanie jej na łódź podwodną . Na brzegu nie ma łódki , musisz zrobić to wpław . Pływanie zabiera Ci dużo energii , dlatego zb... odpływaj od brzegu w ... okrętu , gdyż możesz ... Drugie niebezpieczeństwo ... może spotkać Ciebie w wodzie ... rekiny . Szybko musisz wtedy ... nać , by ujść ich szczękom .

Wojowniczość szpiega jest ... tępnym niebezpieczeństwem . Gdy znajdzie się w tym samym sektorze wyspy , może zaatakować , strzelając do Ciebie z pistoletu lub siekać szablą . Stopień zmęczenia masz ukazany po prawej stronie pod oknem . Twoje zmęczenie to **WHITE STRENGTH**, a wroga **BLACK STRENGTH** . Przy odrobinie szczęścia możesz szpiega dzięki inicjatywie , uporczywości i zręczności wysłać na tamten świat , a następnie postawić mu pomnik .
Gra umożliwia Ci zastawianie pułapek na przeciwnika . Służą do tego następujące przedmioty : lina , łopata , napalm , bomba . W zależności od stopnia trudności niektóre z tych przedmiotów możesz mieć ze sobą dzięki urządzeniu przenoszącemu w inny wymiar , inne dopiero będziesz mógł znaleźć na wyspie . Metoda wydobywania ich z innego wymiaru oraz sposób szukania na wyspie zostały omówione wcześniej .

Sposób zastawiania pułapek jest prosty .

Lina jest wykorzystywana do wykonania potrzasku . Należy wejść na drzewo , zaczepić jeden koniec liny . Następnie zejść z drzewa i zaczepić ją w dowolnym miejscu na ziemi (naciskać przycisk) , lina stanie się niewidoczna .
Ukrywanie bomb oraz napalmu jest jeszcze prostsze . Po położeniu na ziemi upodabniają się one do górek z piasku (trzy trójkąty) .
Przy pomocy łopaty możesz wykopać szereg dołów .
Pułapki te są groźne dla wroga , a także i dla Ciebie , a więc pamiętaj o tym , gdzie je zastawiłeś .

W trakcie wędrówki po wyspie tracisz ogólną energię . Jedyną

" IKS " - dodatek " Żołnierza Wolności ". Redaguje Wiesław Cetera (kierownik zespołu), Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski , Ireneusz Miernik , Ludwik Piela , Jacek Szaniawski . Adres redakcji : 00-950 Warszawa ul . Grzybowska 77 telefon centrali 20-12-61 w. 486. Telex 313664 .

Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów . Nakładem Wydawnictwa " Czasopisma Wojskowe ", Warszawa ul Grzybowska 77. Druk offsetowy - Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego.

Nr Zam 1981 Nr ind 396281, U-28

KUP BĘDZIESZ WIEDZIAŁ

INFORMATYKA
KOMPUTERY
SYSTEMY

zeszyty programów komputerowych

Dodatek „Żołnierza Wolności" Cena 200 zł

nr 1 '89



Szybki rozwój informatyki, a zwłaszcza jej najmłodszej gałęzi — mikrokomputerów osobistych spowodował duże zainteresowanie literaturą fachową z tej dziedziny. Znikoma liczba tytułów na rynku krajowym zmusza większość użytkowników sprzętu mikroinformatycznego do korzystania z wydawnictw zagranicznych. Z kolei nie zawsze znajomość języków obcych oraz szybkie powstawanie nowych terminów powoduje niezbyt precyzyjne ich rozumienie. Celem tej publikacji jest próba wyjaśnienia i uporządkowania terminologii obcojęzycznej, stosowanej do określania w opisach, instrukcjach i czasopismach, nowoczesnych środków automatyzacji procesów dyskretnych, a szczególnie mikrokomputerów. Leksykon Mikrokomputerów zawiera ponad siedemset haseł, podaje objaśnienia około 100 skrótów oraz przy 140 synonimach odsyła czytelnika do właściwego hasła. Odsyłacze (w postaci →) zastosowano tylko tam, gdzie dalsze wyjaśnienie może być pomocne do właściwego zrozumienia hasła. Obok terminu (hasła) głównego, leksykon zawiera często wiele innych terminów niezależnych lecz spotykanych w literaturze czy mowie potocznej, aby ułatwić ich identyfikację. Tematyka haseł leksykonu jest tak dobrana, aby mogli z niego korzystać zarówno profesjonaliści, jak i osoby po raz pierwszy stykające się z tą dziedziną, np. czytające książki lub czasopisma w najpopularniejszych językach: angielskim, niemieckim i rosyjskim. Tam, gdzie brak odpowiednika obcojęzycznego (którego nie udało się znaleźć autorom) zostawiono wolne miejsca do wpisania.

Część haseł dotycząca informatyki, a ściślej programowania i przetwarzania danych, zawarta w leksykonie jest niezbędna do zrozumienia zasad oprogramowania tych środków automatyzacji.

Skróty polskie oraz skróty angielskie, powszechnie używane również w innych językach, podano w części głównej w postaci haseł wraz z ich rozwinięciem. Skróty stosowane w innych językach podano w objaśnieniach obcojęzycznych części głównej i w słowniczkach obcojęzycznych. Terminy hasłowe mające kilka znaczeń objaśniono kolejno, podając ich znaczenie po cyfrach: 1., 2., itd.

W objaśnieniach (komentarzach) haseł ujęto tylko ten zakres znaczeniowy, który wiąże się tematycznie z leksykonem. Synonimy i terminy bliskoznaczne lub potoczne w języku polskim podano w części głównej obok hasła (termin zalecany — podkreślony, wytłuszczony) po przecinku, a w treści objaśnienia w nawiasie. W wersjach obcojęzycznych natomiast — w części głównej po przecinku.

Leksykon z pewnością nie zaspokoi w pełni potrzeb i oczekiwań profesjonalistów wymienionych branż, a zwłaszcza projektantów i producentów sprzętu elektronicznego, informatyków i studentów odpowiednich kierunków. Autorzy postawili sobie za cel: zbudować pierwszy pomost na tej trudnej drodze, w ciągle dynamicznie rozwijającej się dziedzinie. Podjęli próbę uporządkowania stosowanej terminologii i ułatwienia korzystania z niej. Leksykon ten jest rozszerzoną wersją drukowanego w odcinkach na łamach tygodnika „Przegląd Techniczny" w latach 1985—1986.

Autorzy oczekują na listy i uwagi Czytelników, umożliwiające uzupełnienie i doskonalenie przedstawionego opracowania.

Roman GRABOWSKI
Bronisław HYNOWSKI
Ewa MAŃKIEWICZ-CUDNY



ABIOS → BIOS
ACIA — Asynchronous Communication Interface Adapter — układ towarzyszący
ADC — Analogy Digital Converter — przetwornik analogowo-cyfrowy
administrator bazy danych → system zarządzania bazą danych
adres — identyfikator zapisu w pamięci; nazwy adresów pochodzą od metod adresowania (→ adresowanie) bądź funkcji, jakie spełniają
A. address
N. Adresse (f)
R. адрес
adres aktualny — wynikający z przeprowadzonej uprzednio modyfikacji lub obliczenia adresu
A. actual address
N. aktuelle Adresse
R. действительный адрес
adres bazowy — taki, który w połączeniu (adres bazowy + przesunięcie adresowe) z adresem względnym daje poszukiwany adres bezwzględny argumentu
A. base address
N. Basisadresse (f)
R. основной адрес
adres bezwzględny → adres rzeczywisty
adres efektywny — występuje w adresowaniu pośrednim względnym i zostaje obliczony dopiero w trakcie realizacji programu.
A. effective address
N. effektive Adresse (f)
R. исполнительный адрес
adres maszyny → adres rzeczywisty
adres przemieszczalny — wartość jego ustalana jest w sposób względny do innego adresu danego programu, stanowiącego adres odniesienia. Adresem odniesienia jest na ogół pierwszy rozkaz danego programu.
A. relocatable address
N. relokatible Adresse (f), verschiebbare Adresse (f)
R. настраиваемый перемещаемый адрес
adres rozkazu — adres komórki pamięci, w której znajduje się określony rozkaz
A. instruction address
N. Befehlsadresse (f), Instruktionsadresse (f)
R. адрес команды (инструкции)
adres rozkazu skoku powrotnego — występuje w programie wywołującym podprogram; jest to adres rozkazu powrotu do programu przechowywany w pamięci programu tuż za rozkazem wywołującym podprogram
A. return address
N. Rücksprungadresse (f), Rückkehradresse (f)
R. адрес возврата
adres rzeczywisty — faktyczny adres miejsca pamięci; dla danego komputera jest to wartość stała
A. absolute address, machine address
N. absolute Adresse (f), Maschinenadresse (f)
R. абсолютный адрес
adres symboliczny — adres komórki pamięci zastąpiony symbolem, występuje tylko w programach źródłowych
A. symbolic address
N. symbolische Adresse (f)
R. символический адрес

adres względny — część adresu, która przy adresowaniu względnym jest zawarta w rozkazie
A. relative address
N. Relativadresse (f)
R. относительный адрес

adresowanie — zaopatrywanie danych, rozkazów lub części programów w → adresy; zasadnicze trzy klasy metod adresowania (bezpośrednie, pośrednie i względne) można podzielić na następujące rodzaje adresowania:

1. adresowanie bezpośrednie — z pełnym adresem argumentu
A. direct addressing
N. direkte Adressierung
R. прямая адресация

1.1. adresowanie bezpośrednie uwikłane — adres argumentu zawarty jest w kodzie operacji
A. implicite direct addressing
N. implizit direkte Adressierung
R. неявная адресация

1.2. adresowanie bezpośrednie rejestru — w rozkazie zawarty jest adres rejestru, w którym zapamiętany jest argument
A. register direct addressing
N. register-direkte Adressierung
R. прямая адресация регистра

1.3. adresowanie bezpośrednie symboliczne — w rozkazie zawarty jest pełny adres pamięci (komórki pamięci) zawierającej argument
A. symbolic direct addressing
N. Speicher-direkte Adressierung
R. символическая адресация

1.4. adresowanie bezpośrednie natychmiastowe — argument jest zawarty bezpośrednio w rozkazie, zamiast jego adresu
A. immediate addressing
N. unmittelbare (unmittelbar-direkte) Adressierung
R. прямая адресация

2. adresowanie pośrednie — w rozkazie zawarty jest adres specyfikujący rejestr lub miejsce w pamięci, w którym znajduje się adres argumentu lub znów adres pośredni
A. indirect addressing
N. indirekte (interierte) Adressierung
R. косвенная (непрямая) адресация

2.1. adresowanie pośrednie rejestru — w rozkazie zawarty jest adres rejestru, w którym znajduje się adres argumentu lub znów adres pośredni
A. register indirect addressing
N. Register indirekte Adressierung
R. косвенное адресование регистра

2.2. adresowanie pośrednie rejestru z autoinkrementem — po znalezieniu argumentu w pamięci, jak w 2.1. zawartość rejestru adresowanego w rozkazie zostaje zwiększona automatycznie o stałą wartość (inkrement = przyrost dodatni)
A. register indirect with autoincrement
N. Register indirekt mit Autoinkrement
R. косвенное адресование регистра с автоинкрементом

2.3. adresowanie pośrednie rejestru z autodeinkrementem — po znalezieniu argumentu w pamięci, jak w 2.1. zawartość rejestru adresowanego w rozkazie zostaje automatycznie zmniejszona o stałą wartość (deinkrement = przyrost ujemny)
A. register indirect with autodeincrement
N. Register indirekt mit Autodeinkrement
R. косвенное адресование регистра с автодеинкрементом

3. adresowanie względne — w rozkazie zawarty jest adres częściowy (adres względny), który musi być połączony z innym adresem częściowym (adres bazowy, adres odniesienia) aby otrzymać adres argumentu. Adresy częściowe mogą być również podane w sposób pośredni
A. relative addressing
N. relative Adressierung
R. относительная адресация

3.1. adresowanie względne indeksowane — zawarty w rozkazie adres względny jest dodawany lub odejmowany od adresu bazowego zapamiętanego w rejestrze indeksowym
A. indexed relative addressing
N. indizierte relative Adressierung
R. индексированная относительная адресация

3.2. adresowanie względne z podziałem pamięci na bloki — adres bazowy określa blok pamięci, która została podzielona na bloki równej długości; adres względny podaje słowo (słowo pamięciowe) w obrębie jednego bloku
A. mapping, relative addressing (with mapping)
N. mapping, relative Adressierung (mit mapping)
R.

3.3. adresowanie względne ze stronicowaniem — adres bazowy określa stronę pamięci, która została podzielona na równej wielkości „strony"; adres względny podaje słowo (słowo pamięci) w obrębie tej strony
A. paging (pageing) relative addressing (with paging)
N. paging relative Adressierung (mit paging)
R.

3.4. adresowanie względne z licznikiem rozkazów — jako adres bazowy służy chwilowa zawartość licznika rozkazów (licznik programu), adres względny jest dodawany do adresu bazowego lub od niego odejmowany. Wynik jest wprowadzany do licznika rozkazów.
A. program counter relative addressing
N. Programmschrittzähler relative Adressierung
R. относительная адресация со счетчиком команд

AGAT — radziecki mikrokomputer o pamięci wewnętrznej 64, 128 lub 256 kB. Jako pamięć zewnętrzną można stosować dyski elastyczne lub kasety magnetofonowe. Wyposażony jest w klawiaturę o układzie alfanumerycznym. Przystosowany do współpracy z monitorem czarno-białym lub kolorowym oraz do prezentowania danych w postaci graficznej.

akumulator — główny rejestr → jednostki arytmetyczno-logicznej służący do przechowywania argumentów i wyników operacji arytmetyczno-logicznych.
A. accumulator, accu
N. Akkumulator, Akku
R. аккумулятор

alfaskop → monitor ekranowy

algorytm — przepis wykonania jednej lub ciągu operacji do osiągnięcia określonego celu
A. algorithm
N. Algorithmus (m)
R. алгоритм

ALU — Arithmetic Logic Unit → jednostka arytmetyczno-logiczna.

AMM — automatyczna maszyna manipulacyjna zwana popularnie → robotem.
A. automatic manipulations machine
N. automatische Manipulierungsmaschine
R. автоматическая манипулирующая машина

AMPL — Advanced Microprocessor Prototyping Laboratory — system projektowania firmy Texas Instruments → systemy uruchomieniowo-rozwojowe.

AMSTRAD — angielska firma znana głównie z produkcji średniej klasy sprzętu hi-fi. Jesienią 1984 r. wypuściła na rynek brytyjski i zachodnioniemiecki pierwsze egzemplarze mikrokomputera **Amstrad Schneider CPC 464**. W roku następnym sprzedawano już trzy modele: 464, 664 i 6128. Ich cechy wspólne, to: procesor Z80A, grafika o maksymalnej rozdzielczości 640×200 punktów

(2 kolory), 320×200 punktów (4 kolory) i 160×200 punktów (16 z palety 27 kolorów), dźwięk — 3 generatory po 7 oktaw z możliwością pełnej modyfikacji brzmienia oraz z wyjściem stereo (tzn. 1 generator — kanał lewy, 2 generator — kanał prawy, 3 generator pracujący w obu kanałach), możliwość dołączenia 240 modułów pamięci ROM, po 16 kB każdy.

analizator stanów obwodów logicznych — przyrząd (lub zestaw przyrządów) do badań funkcjonalnych w czasie rzeczywistym zespołów logicznych w postaci układów cyfrowych i całych systemów mikroprocesorowych, których stany logiczne i/lub czasowe są sygnalizowane na ekranie lampy kineskopowej w sposób kroczący
 A. logic (circuit) analyzer, digital analyzer
 N. Logikanalysator (m): Analyzator
 R. шемный анализатор

aplikacja — stosowanie i/lub wdrażanie (zastosowań) sprzętu lub technik (technologii)
 A. application
 N. Anwendung (f), Einsatz (m), Auwendbarkeit (f), Einsatzmöglichkeit (f)
 R. применения

APD — automatyczne przetwarzanie danych

APU — Arithmetic Processing Unit → arytmometr

architektury — zasadnicze cechy funkcjonalne budowy mikrokomputerów (mikroprocesorów), decydujące o sposobie ich działania → podział. Dokumentacyjnie (rysunkowo) architekturę przedstawia się w postaci funkcjonalnego schematu blokowego zawierającego między innymi: ALU, rodzaje i pojemności pamięci, magistrale adresów i danych, rejestry, zegar we/wy urządzeń peryferyjnych. Zasadniczą cechą jest to, w jaki sposób za pomocą rozkazu komputerowego można wykonać operację na argumentach i wyprowadzić wyniki.
 A. architectures
 N. Architekturen (pl)
 R. архитектуры

Rozróżnia się architektury o następujących możliwościach operacji:

— **z rejestru do rejestru** — argumenty znajdują się w rejestrach, wynik przesyłany jest do rejestru
 A. register to register architecture
 N. Register zu Register Architektur
 R.

— **z rejestru do pamięci** — jeden argument znajduje się w rejestrze, drugi w pamięci, wynik przesyłany jest do rejestru. Architektura ta zawiera w sobie poprzednią
 A. register to memory architecture
 N. Register zu Speicher Architektur
 R.

— **z pamięci do pamięci** — argumenty znajdują się w pamięci, wynik przesyłany jest do pamięci. Architektura ta zawiera obie poprzednie
 A. memory to memory architecture
 N. Speicher zu Speicher Architektur
 R.

architektura mikroprocesorowa — architektura, która określa współpracę grup funkcjonalnych mikroprocesora, tj. jednostki sterującej i arytmometru, ich budowę, sposób działania, właściwości oraz powiązania między nimi
 A. microprocessor architecture
 N. Mikroprozessorarchitektur (f), Mikroprozessoraufbau (m)
 R. архитектура микропроцессора

argument — wartość lub informacja potrzebna do wykonania rozkazu, którą na podstawie adresu zawartego w rozkazie należy sprowadzić z pamięci, lub też która zawarta jest bezpośrednio w rozkazie (niektórzy autorzy tę ostatnią nazywają operandem)
 A. operand
 N. Operand (m)
 R. операнд

argument operacji → argument

arytmetyka adresów → obliczanie adresu

arytmometr — część jednostki centralnej względnie mikroprocesora (procesora) do wykonywania operacji arytmetycznych, logicznych, przesunięć, porównań, zaokrągleń itd., w skład a. wchodzą: ALU, akumulatory, rejestry
 A. arithmetic unit, APU — Arithmetic Processing Unit
 N. Rechenwerk (n), Recheneinheit (f)
 R. арифметическое устройство

ASCII — American Standard Code for Information Interchange — kod alfanumeryczny siedmiobitowy z 8 bitem do kontroli parzystości (parity bit), zwany również Kodem USASCII

asembler — 1. translator asemblera — translator z języka typu asembler na język komputera, czyli program używany do asemblowania; 2. język typu asembler — zorientowany maszynowo, symboliczny język programowania.
 A. 1. assembler, 1. assemblery program
  2. assembler language
 N. 1. Assembler (m), 1. Assemblierer (m),
  2. Assemblersprache (f)
 R. 1. ассемблер
  2.

asembler jednokrotny — translator asemblera, który tłumaczy program w jednej operacji (jednokrotne wczytywanie programu źródłowego).
 A. one pass assembler
 N.
 R.

asembler makro — asembler, który uwzględnia wywołania makro zawarte w programie źródłowym napisanym w języku typu asembler
 A. Macro assembler
 N. Makroassembler (m)
 R. ассемблер макро

asembler mikro kodowy → mikro asembler

asembler skrośny → krosasembler

asemblowanie — 1. przebieg translacji (tłumaczenie, przekształcanie, translacje asemblera przez translator) programu opracowanego w określonym języku typu asembler na program w języku (kodzie) określonego komputera zwanym również językiem maszynowym (z jednoczesnym sprawdzaniem błędów syntaktycznych), przy czym adresy symboliczne zastępowane są adresami rzeczywistymi lub względnymi typu przemieszczalnego (relocatable); 2. potocznie: łączenie podprogramów
 A. assemble, assembling
 N. Assemblieren, Assemblieren (n), Assemblierung (f)
 R. 1. ассемблировать
  2. компоновать

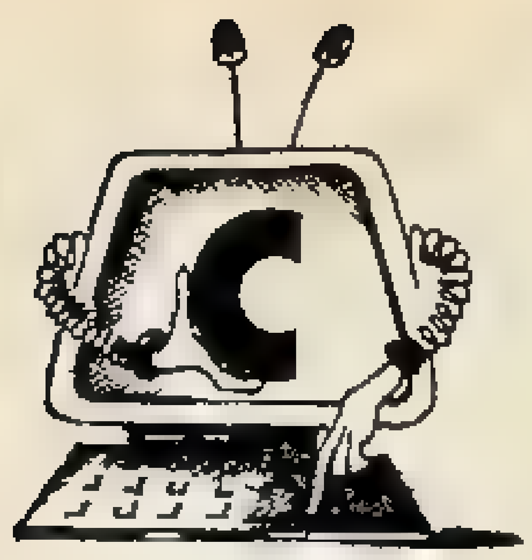
Atari — firma Atari Corporation należy do czołowych wytwórców gier telewizyjnych. W 1981 r. weszły na rynek mikrokomputery tej firmy Atari 400 i 800 (dziś już nie produkowane). W 1983 r. ich następcy, to seria XL złożona z 3 modeli: 600XL, 800XL i 1200XL, o pojemności pamięci odpowiednio — 16, 64 i 64 Kb. Najpopularniejszy z tej serii to 800XL.

Atari 800XL oparty jest na 8-bitowym mikroprocesorze 6502, który adresuje 64 Kb pamięci RAM i 24 Kb ROM. W pamięci ROM zawarty jest system operacyjny i interpreter języka BASIC (dla programów pisanych w BASIC-u dostępne jest ok. 38 Kb pamięci RAM).
800XL ma bardzo rozbudowane możliwości graficzne, obejmujące m.in.: 16 podstawowych trybów zorganizowania ekranu (5 tekstowych i 11 graficznych), możliwość

wyboru barwy obiektu w jednym z 256 kolorów, możliwość prostej animacji przy użyciu tzw. grafiki player missile.
Auto-CAD — program wspomagający projektowanie w systemie CAD na sprzęcie IBM PC.



bajt — umowna jednostka przetwarzanych informacji, w organizacji komputerów obejmuje ona 8 bitów, z tego 7 informacyjnych i 1 bit kontrolny (dwie tetrady), wielokrotności tej jednostki to 2^{10} czyli K bajt (KB) = 1024 bajtów i M bajt = 1024×1024 bajtów
A. byte
N. Byte (n)
R. байт
bajt stanu urządzenia — w niektórych komputerach znajduje się w pamięci operacyjnej i umożliwia kontrolę stanu urządzeń przyłączonych, jak też stan układów wewnętrznych i przebiegu poszczególnych operacji. B.s.u. składa się z bitów stanów poszczególnych układów
A. standard device byte
N. Gerätebyte (n)
R. байт состояния устройств
bank danych — całość informacji potrzebnych do eksploatacji określonego systemu przetwarzania danych
A. data bank, data base
N. Datenbank (f)
R. банк данных
BASIC — Beginners All Purpose Symbolic Instruction Code — bardzo prosty, łatwy do nauki, problemowy (programowanie wysokiego poziomu) język programowania
baza danych — element → banku danych stanowiący zbiór danych określonego obszaru problemowego
bezpośredni dostęp do pamięci — komunikacja danych przez urządzenia we/wy do pamięci z pominięciem mikroprocesora (jednostki centralnej), co umożliwia szybkie przemieszczenie danych
A. direct memory access
N. direkter Speicherzugriff (m)
R. произвольный (прямой) доступ памяти
BIOS — Basic Imput/Output System — zawarty w ROM zbiór procedur bezpośrednio współpracujących ze sprzętem, obejmuje m.in. zestaw pamięci RAM, handlery (procedury obsługi) we/wy, usługowe programy systemowe pierwotnego wprowadzania itp. Stanowi system operacyjny PS/2. Jego odmiany, to ABIOS — Advanced BIOS i CBIOS — Compability BIOS, stosowane w większych systemach PS (od M50 wzwyż).
bit — cyfra w zapisie dwójkowym (b)
A. bit, binary digit
N. Bit (n)
R. бит
bit kontrolny parzystości → bit parzystości i → bit nieparzystości
A. parity bit
N. Paritätsbit (n)
R. бит чётности
bit najbardziej znaczący — bit znajdujący się na lewym skraju liczby binarnej
A. most significat bit
N. höchstwertiges Bit (n)
R. самый старший бит
bit najmniej znaczący — bit znajdujący się na prawym skraju liczby binarnej
A. least significat bit
N. niedrigstwertiges Bit (n)
R. самый младший бит, младший бит
bit nieparzystości — bit kontrolny parzystości, który przyjmuje wartość 1, jeżeli w części informacyjnej bajtu jest parzysta suma jedynek binarnych (bajt taki nazywa się nieparzystym — odd byte — bo suma bitów łącznie z kontrolnym jest nieparzysta, np. w kodzie EIA8B)
A. odd parity bit
N. ungrades Parität — Bit
R. бит контроля на нечётность
bit parzystości — bit kontrolny parzystości, który przyjmuje wartość 1, jeżeli w części informacyjnej bajtu jest nieparzysta suma jedynek binarnych (bajt taki nazywa się parzystym — even byte — bo suma bitów łącznie z kontrolnym jest parzysta, np. w kodzie ISO/R840)
A. even parity bit
N. gerades Parität — Bit
R. бит чётности
bit stanu — bit charakteryzujący aktualny stan chwilowy układów mikrokomputera (komputera), określony wynik lub spełnienie (niespełnienie) określonych warunków, **b.s.** rejestruje się w rejestrach stanu (np. w postaci bajtów stanu) i stan ich może być sprawdzany specjalnym rozkazem lub programem → zapytań
A. status bit
N. status — Bit (n), Bedingungs — bit (n)
R. бит состояния
bit zgodności → oznacznik zgodności
blok — 1. zapis słów, znaków lub cyfr, które mogą być potraktowane jako jednostka w programie lub pamięci zewnętrznej; 2. grupa sąsiednich znaków teletransmisji danych
A. block
N. Block (m)
R. блок
blok funkcjonalny → układ funkcjonalny o budowie modułowej
A. function block
N. Funktionsblock (n)
R. функцийный блок
bod — jednostka szybkości transmisji danych, jeden bod równa się 1 bit/s
A. baud
N. Baud (n)
R. бод
BPI — Bits PER Inch — jednostka gęstości zapisu na taśmie magnetycznej (liczba bitów na długości jednego cala)
brama → rejestr translacyjny → port
bramka → elementy logiczne
BS/2 → system operacyjny OS/2
budowa modułowa — system konstrukcji blokowej — celowe zgrupowanie wzajemnie ze sobą powiązanych zunifikowanych zespołów ułatwiających produkcję i eksploatację (np. wymianę w wypadku awarii)
A. modular design, modular concept
N. Baukastenprinzip (n)
R. модультная система
budowa rozkazu — struktura rozkazu, format rozkazu — podaje długość rozkazu (w bitach), jak również jego układ i liczbę bitów przeznaczonych na argumenty lub ich adresy
A. instruction format, instruction structure, command structure
N. Befehlsformat (n), Befehlstruktur (f), Befehlsaufbau (m)
R. структура команды (инструкции), формат команды



CAD — Computer Aided Desing → komputerowe wspomaganie projektowania
CADAM-CAD/CAM — Computer Aided Design and Manufacturing — komputerowe wspomaganie projektowania i wytwarzania
CAE — Computer Aided Engineering — komputerowe wspomaganie konstruowania
CAI — Computer Aided Industry (instruction); 1 → komputerowe wspomaganie wytwarzania przemysłowego; 2 → komputerowe wspomaganie w instruowaniu
CAM — Computer Aided Manufacturing → komputerowe wspomaganie wytwarzania
CAMAC — Computer Automated (Application for) Meacurement And Control — znormalizowany system złączowy, stosowany w układach sterowania i automatyki → układ złączowy
CBIOS → BIOS
CD-ROM — Compact Disc ROM — pamięć dyskowa o zapisie i odczycie laserowym i o bardzo dużej pojemności (500 MB), o dostępie swobodnym i czasie dostępu 1 s.
Cechy charakterystyki użytkowej mikroprocesorów — zasadnicze parametry mikroprocesorów
technologia: bipolarna (ECL, J²L, STTL, TTL), unipolarna MOS (CMOS, MMOS, NMOS, PMOS).
  A. process technology
  N.
  R.
długość słowa (danych/instrukcji): od 1/4 do 32/32
  A. word size (data/instruction)
  N.
  R.
pojemność adresowania (liczba słów): do 16 MB
  A. direct addressing range (words)
  N.
  R.
maksymalna częstotliwość zegara (MHz) faz: od 0,5/1 do 30/30
  A. maximum clock frequency (MHz) phases
  N.
  R.
liczba instrukcji: od 16 do kilkuset
  A. number of basic instruction
  N.
  R.
czas wykonania instrukcji (μs): od 0,2 (2OMIPS) do 780
  A. instruction time (μs)
  N.
  R.
obudowa (liczba wyprowadzeń): od 16 do 132
  A. package size (pins)
  N.
  R.
CGA — Color Graphies Adapter — karta/pakiet grafiki kolorowej
CIM — Computer Integrated Manufacturing → kompleksowe skomputeryzowane wytwarzanie
CODEC — koder — dekoder
CLK — Clock — zegar taktujący
CNC — Computer (ised) Numerical Control — komputerowe (skomputeryzowane) sterowanie numeryczne
Commodore — najpopularniejszy w RFN mikrokomputer (ze względu na liczbę sprzedanych), używany często jako najmniejszy business computer Commodore 64. Firma Commodore związana jest ściśle z historią mikrokomputerów domowych — wypuściła jeden z pierwszych mikrokomputerów — PET. Po C 64 pojawiły się na rynku C 16, C 116 i Plus 4, ale nie miały one tradycji C 64 i nie zostały „zaakceptowane" przez nabywców. Dużą zgodność z C 64 wykazuje C 128, którego procesor 8502 jest kompatybilny z procesorem C 64, może on pracować ze wszystkimi urządzeniami zewnętrznymi C 64.
CPC — Continuous Path Control — sterowanie kształtowe (ciągłe)

CP/M — Control Program for Microcomputers — system operacyjny mikrokomputerów
CPU — Central Processing Unit → jednostka centralna
CRC — Computer Robot Control — komputerowe sterowanie robotów
Cross support — pojęcie obejmujące zastosowanie programów, które wspomagają („wspomaganie wskrośne") opracowywanie i testowanie programów do mikrokomputera (komputera) za pomocą innego mikrokomputera (komputera), jest to metoda powszechnie stosowana do sprzętu opartego na mikroprocesorach i programach w postaci cross — assemblerów, — kompilatorów, — symulatorów
  A. cross support
  N. cross support
  R. кросс-суппорт
CRT — Cathode Ray Tube — lampa elektrono-promieniowa
  N. Katodenstrahlröhre (f)
  R. электронно-лучевая трубка
CRU — Communication Register Unit — programowalne złącze we/wy

CU — Control Unit — jednostka sterująca
cyfra — znak ze zbioru znaków określonego systemu liczbowego, w zależności od liczby cyfr w zbiorze danego systemu, system nazywany dla:
  10 cyfr (0..9) — dziesiętnym
  dwu cyfr (0,1) — dwójkowym itp.
  A. digit
  N. Ziffer (f)
  R. цифра
czas dostępu — czas opóźnienia podania danych od chwili podania adresu, np. dla pamięci MOS o dostępie bezpośrednim wynosi on 250...450 ns, dla pamięci statycznej typu RAM na układach scalonych z GaAs — 1 ns, pamięci MCY 7102 — 100 ns
  A. access time
  N. Zugriffszeit (f)
  R. время доступа, время выборки
cyfrowe układy → układy cyfrowe

czas realizacji — 1. czas, jaki potrzebuje mikrokomputer (komputer) na wykonanie jakiegoś rozkazu, mierzony np. liczbą taktów przy danej częstotliwości taktów (zegara), 2. czas, jaki potrzebuje mikrokomputer (komputer) na wykonanie programu
  A. execution time
  N. Ausführungszeit (f)
  R. время выполнения
czas rzeczywisty → przetwarzanie w czasie rzeczywistym
część sterująco-nadzorcza programów organizacyjnych — program lub część programu do specjalnych zadań wchodząca na ogół w skład systemu operacyjnego
  A. supervisor
  N. Supervisor (m), Überwacher (m)
  R. супервизор



DAC — Digital Analog Converter — przetwornik analogowo-cyfrowy

dane — 1. informacje przeznaczone do przetwarzania; 2. przedstawienie faktów, koncepcji lub instrukcji w tak uporządkowany sposób, że nadają się one do przesyłania, interpretowania lub przetwarzania przez ludzi lub maszyny; 3. każdy sposób przedstawiania informacji w postaci znaków lub wielkości analogowych, którym może być lub jest przypisane określone znaczenie
 A. data
 N. Daten (f)
 R. данные

dane standardowe — takie dane w rozkazie, które mogą być opuszczone, ponieważ komputer uzupełni je (o **d.s.**) w trakcie wykonania rozkazu
 A. default
 N. Standardwert (m)
 R. стандарт

DBMS — Data Base Management System — system (program) zarządzania bazami danych

deasembler — program umożliwiający przeprowadzenie odwrotnego procesu do asemblowania, tzn. istniejący w języku komputera program (program wynikowy) zostaje przetłumaczony na poprzednią jego formę (program źródłowy) w języku typu asembler
 A. de-assembler, disassembler
 N. Disassembler (m), Deassembler (m)
 R.

dekoder rozkazów — deszyfrator, który interpretuje rozkazy komputera, wartościowuje wyniki badań (zapytań) i dostarcza częściom wykonawczym mikroprocesora (procesora) sygnały sterujące i adresy niezbędne do wykonania rozkazu; na ogół w skład tych czynności wchodzi również rozdział rozkazu komputera na mikrorozkazy
 A. instruction decoder, command decoder
 N. Befehlsdekoder (m), Instruktionsdekoder (m)
 R. дешифратьр команд

digitizer — urządzenie zmieniające informacje analogowe na cyfrowe np. zapis rysunku w pamięci komputera
 A. digitizer
 N. Digitalisierer (m)
 R.

długość bloku — liczba znaków określająca wymiar bloku
 A. block size, block length
 N. Blocklänge (f)
 R. длина блока

długość słowa (komputera) — liczba bitów (w słowie), którą komputer jest w stanie przetworzyć w jednym ciągu
 A. word length, word size
 N. Wortlänge (f)
 R. длина слова

DMA — Direct Memory Access — bezpośredni (swobodny) dostęp do pamięci

DNC — Direct Numerical Control — bezpośrednie (z centralnego komputera) sterowanie numeryczne (urządzeniami i procesami technologicznymi).

DOS — Disk Operating System — system operacyjny dyskowy

dostęp — możliwość odczytania lub wpisania informacji do pamięci
 A. access
 N. Zugriff (m)
 R. доступ, выборка

dostęp bezpośredni — dostęp swobodny, dostęp dowolny — możliwość bezpośredniego dostępu do każdego miejsca pamięci zarówno przy odczycie, jak i zapisie danych, czas dostępu jest tu niezależny od adresu
 A. random access, direct access
 N. wahlfreier Zugriff (m), direkter Zugriff (m), Direktzugriff (m)
 R. произвольная выборка, свободный (прямой) доступ

dostęp sekwencyjny — dostęp do informacji wg kolejności ich przechowywania w pamięci, a zapis wg kolejności napływu
 A. sequential access, serial access
 N. sequentieller Zugriff, serialler Zugriff
 R. последовательный доступ

dostęp swobodny → dostęp bezpośredni

DSC — Digital Supervisory Control — sterowanie cyfrowe nadrzędne przez komputer

DTP — Desk Top Publishing — stanowisko mikrokomputerowe z oprogramowaniem do prac wydawniczych

dupleks → transmisja jednoczesna

dyrygent → program dyrygent

dyrektywa — wskazówka, wytyczna, komentarz (np. komentarz w j. typu asembler)
 A. directive
 N. Direktive (f)
 R.

dysk elastyczny — dyskietka — tzw. miękki dysk, odmiana dysku magnetycznego o bardzo krótkim czasie dostępu (średni czas dostępu 0,25 s) przy małych rozmiarach charakteryzuje się dużą pojemnością (dwudyskietkowa jednostka do 1 MB). Stosowane dyskietki mają średnice 3"; 3,5"; 5,25" i 8".
 A. floppy dysk, minidysk
 N. Diskette (f), Minidisk (f)
 R. маленький дыск

dyskietka → dysk elastyczny



EAPROM — Electrically Alterable PROM → podział pamięci

EAROM — Electrically Alterable ROM — elektrycznie przeprogramowalna pamięć typu „wyłącznie odczyt", kasowanie odbywa się tu na drodze elektrycznej (np. na drodze lawinowego wstrzykiwania dziur) w odróżnieniu od pamięci EPROM (→ podział pamięci), którą kasuje się promieniami nadfioletowymi (2537 Å)

EBCDI → kod EBCDI

EEPROM — Electrically Erasable PROM → podział pamięci

EGA — Enhanced Graphics Adapter — pakiet (karta) ulepszonego sterownika grafiki komputera osobistego IBM PC

edytor — program redagujący, umożliwiający przez monitor ekranowy wczytywanie tekstu do pamięci komputera, a następnie jego przeredagowanie
 A. editor
 N. Editor (m), Texbearbeitungsprogram (n), Texteditor (m)
 R. редактующая программа

ekranopis → monitor ekranowy
elastyczne kompleksy wytwórcze — zautomatyzowane wielozadaniowe wytwórnie o dużej elastyczności adaptacyjnej do produkcji różnych typowielkości części i wyrobów finalnych oraz dużym zakresie stosowanych technologii (od magazynu surowców do magazynu wyrobów gotowych przez obróbkę części, montaż, malowanie i pakowanie), w których praca człowieka ogranicza się wyłącznie do funkcji nadzorczych kompleksowo skomputeryzowanego sterowania procesów wytwarzania
  A. flexible manufacturing complex, FMC
  N. flexibles Fertigungssystem (n)
  R.
elektroniczna maszyna cyfrowa → komputer

Elektronika BK-0010 — radziecki mikrokomputer 16-bitowy, pamięć operacyjna RAM 32 kB, ROM — 32 kB, grafika 512×256 punktów (przy dwóch stopniach szarości) lub 256×256 punktów (przy czterech stopniach szarości). Przewidywany jako standardowe wyposażenie szkół radzieckich.
elementy binarne → elementy dyskretne
elementy cyfrowe → elementy dyskretne
elementy dyskretne — elementy podstawowe, których wejścia i wyjścia mogą przyjmować określoną wartość (poziom, stan) z pewnego przeliczalnego zbioru, najczęściej dwuskładnikowego (elementy binarne). Przykładami **e.d.** są: elementy logiczne (kombinacyjne, elementy bez pamięci) i przerzutniki (elementy pamięciowe). Określenia **e.d.** używa się często (bezzasadnie) przeciwstawnie do układów scalonych zamiast określenia elementy podstawowe
  A. discrete elements komponent, digital elements
  N. digitales Element (n), Ziffernelement (n), diskrete Bauelement (n)
  R. цифровые элементы
elementy funkcjonalne → elementy przełączające oraz pomocnicze
elementy kombinacyjne → elementy logiczne
elementy logiczne, funktory — logiczne układy podstawowe — elementy realizujące podstawowe funkcje logiczne (np. NIE, LUB, I), służące do budowy układów logicznych; **e.l.** zwano wcześniej bramkami
  A. logic (logical) elements
  N. Verknüpfungsglied (n), logisches Element (n), Logikglied (n)
  R. логические элементы

elementy pamięciowe — elementy przełączające, które mogą przyjmować różne stany wyjść nie tylko w zależności od aktualnego własnego stanu wewnętrznego
  A. memory elements
  N. Speicherelements (pl., n)
  R. запоминающие элементы
element podstawowy — najprostsza część składowa, w elektronice są to np. kondensatory, rezystory, cewki, diody, tranzystory. Z wielu **e.p.** można stworzyć → układy elementarne
  A. basic element, structural element
  N. Bauelement (n), Baustein (m)
  R. основная часть
elementy przełączające — jednostki przełączające złożone z elementów podstawowych; **e.p.** dzielą się na kombinacyjne i pamięciowe jak np. przerzutniki
  A. switching (circuit) element
  N. Schaltelement (n)
  R. переключающий элемент
emiter danych — część urządzenia do przetwarzania lub teletransmisji danych, która może sygnały danych emitować
  A. data source
  N. Datenquelle (f)
  R. источник данных
emulacja — symulacja w czasie rzeczywistym, naśladowanie (lub odwzorowywanie) jakiegoś mikrokomputera (komputera) na innym niewymiennym pod względem rozkazów czy formatu danych; **e.** w przeciwieństwie do symulacji realizowana jest nie tylko programowo, ale i sprzętowo (np. przez zmianę pamięci stałych, kart układów)
  A. emulation
  N. Emulierung (f), Emulation (f)
  R. эмуляция
emulacja „in circuit" — funkcjonalne, elektryczne i mechaniczne odwzorowywanie (emulacja) jakiegoś mikrokomputera lub mikroprocesora w postaci jednego układu. Podłączenie takiego emulatora następuje przez łącze. Odwzorowywanie (czasów reagowania i realizacji) zapewnione jest w tym wypadku w szerokim zakresie; często stosowane w systemach projektowych mikrokomputerów.
  A. in circuit emulation, ICE
  N. In-Circuit-Emulierung (f), Testadapter (m)
  R.
emulator — system lub część systemu, która umożliwia emulacje
  A. emulator, in-circuitemulator
  N. Emulator (m), Testadapter (m)
  R. эмулятор
ELWRO — 1) Zakłady Elektroniczne „Elwro" we Wrocławiu, największy w kraju producent sprzętu komputerowego; 2. mikrokomputer różnych serii, np. Elwro 500, Elwro 600, Elwro 700 i Elwro 800.
ELWRO 500 — mikrokomputer zaprezentowany po raz pierwszy w 1983 r. Przygotowanie jego produkcji trwało bardzo długo i w tym czasie powstały kolejne prototypy Elwro.
ELWRO 600 — jest praktycznie kontynuacją mikrokomputera Elwro 500, a podstawowa różnica tkwi w budowie modułowej. Moduł jednostki centralnej ma pamięć operacyjną RAM 64 KB, pamięć stałą ROM 8 KB. Klawiatura alfanumeryczna, wydzielona klawiatura numeryczna dziesiętna i heksadecymalna oraz klawisze funkcjonalne. **E. 600** jest typowym mikrokomputerem biurowym, podobnie jak **E. 500**.
ELWRO 700 Solum — powstał jako dzieło brygady racjonalizatorów koła Stowarzyszenia Elektryków Polskich w ZE ELWRO. Może być wyposażony w pamięć ROM od 8 ... 16 KB oraz RAM od 16 ... 48 KB.
Klawiatura alfanumeryczna w układzie z alfabetem polskim i dodatkowymi klawiszami funkcyjnymi. Solum miał być wg założeń polskim mikrokomputerem edukacyjnym. W 1986 r. miało się znaleźć na rynku ok. 10 tys. szt. tego mikrokomputera, a w następnych latach znacznie więcej.
ELWRO 800 — mikrokomputer opracowany wspólnie przez Instytut Automatyki Politechniki Poznańskiej oraz Instytut Komputerowych Systemów, Automatyki i Pomiarów we Wrocławiu. Oparty na mikroprocesorach 8- i 16-bitowych umożliwiających tworzenie systemów od jednoprocesorowych 8-bitowych przez jedno- i wieloprocesorowe 16-bitowe, aż do wieloprocesorowych mieszanych. Może współpracować z wieloma urządzeniami zewnętrznymi dostępnymi na rynku krajowym (produkcji polskiej, bułgarskiej, NRD, Węgierskiej Republiki Ludowej. Struktura modułowa **E. 800**, przy szerokim zestawie oferowanych modułów, pozwala na konfigurowanie systemów specjalizowanych o bardzo zróżnicowanym stopniu złożoności.
Na 57 MTP w Poznaniu **E. 800** wyróżniono Złotym Medalem.

ELWRO 800 Junior — mikrokomputer edukacyjny opracowany przez Instytut Automatyki Politechniki Poznańskiej w 1986 r. jako 8-bitowy mikrokomputer edukacyjny, mogący pracować w sieci zarówno z takim samym sprzętem, jak też z popularnymi już w polskich szkołach ZX SPECTRUM.

EOF — END OF FILE — koniec kartoteki
N. Datenende (n)
R. конец картотеки

EOJ — END OF JOB — koniec programu lub przebiegu programu
N. Programmende (n)
R. конец программы

EOT — 1. END OF TAPE — koniec taśmy magnetycznej; 2. End of Transmission — „koniec teledacji".
N. 1. Magnetbandende (n)
2. Übertaungsende (n)
R. конец ленты
2. конец передачи

EPD — elektroniczne przetwarzanie danych
A. EPD, electronic data processing
N. EDV, elektronische Datenverarbeitung (f)
R. электронная обработка данных

EPROM — Erasable Programmable ROM — pamięć reprogramowalna kasowana w całości lub przez odpowiednie maski promieniami ultrafioletowymi (→ podział pamięci)

etykietka nagłówkowa — etykietka początku (zbioru danych)
A. header, header label
N. Überschrift (f), Kopf (m), Vorsatz (m)
R. национальная метка, метка заголовка

etykietka w programie — symbol identyfikujący w sposób symboliczny rozkazy, adresy lub dane w programie źródłowym
A. label
N. Marke (f)
R. метка



FIFO — FIRST — IN — FIRST — OUT — sposób organizacji pamięci, w którym informacje wpisane jako pierwsze są również jako pierwsze odczytywane (typu silos) → podział pamięci
A. FIFO — memory
N. Silosspeicher (m)

Flip-Flop → przerzutnik

flag — 1. sygnalizator, znacznik lub bit w przerzutniku lub rejestrze sygnalizujący określony stan, wynik, spełnienie lub niespełnienie warunku (bity warunków) czyli błąd; 2. pojedynczy przerzutnik (lub jego stan) w rejestrze statusowym
A. flag
N. Kennzeichen (n), Markierung (f), Flagge (f)
R. признак, флажок, флаг

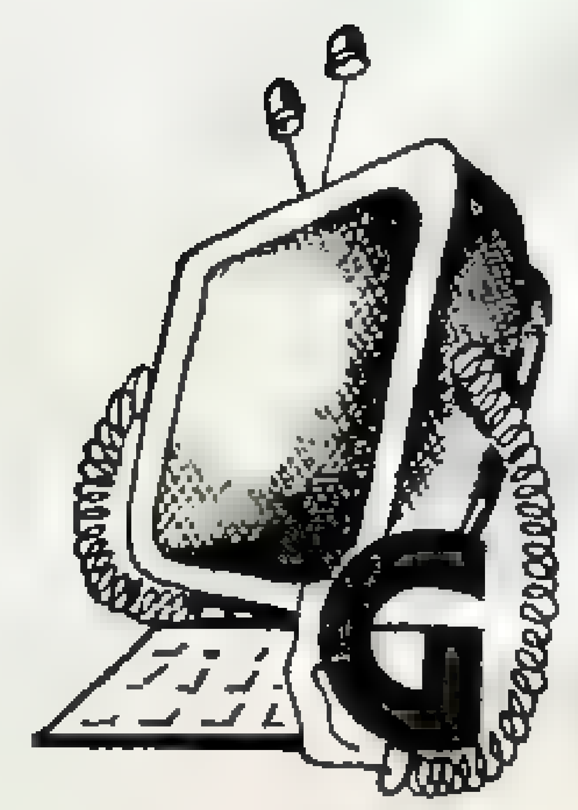
format danych, układ danych, struktura danych — 1. podaje w bitach, jak długie może być słowo danych, aby mogło być jeszcze przetworzone za pomocą pojedynczego rozkazu mikrokomputera; 2. ogólnie — wzór lub przepis odnośnie formalnego układu danych
A. data format
N. Datenformat (n)
R. формат данных

Format rozkazu → budowa rozkazu

format upakowany — rodzaj przedstawienia informacji, w którym dwie cyfry dziesiętne przedstawione są w jednym bajcie
A. packed format
N. gepacktes Format (n)
R. упакованный формат

FPLA — Field Programmable Logic Array — programowalne (u użytkownika) zespoły logiczne (układy logiczne)

funktory logiczne → elementy logiczne



GEM — Graphies Enviroment Manger — system operacyjny zorientowany graficznie z zastosowaniem symboli graficznych (WIMP — Widdow, Ikou, Mous Pushdown Menu) zamiast rozkazów dla ułatwienia posługiwania się mikrokomputerem

generator — 1. układ wytwarzający wielkość wyjściową o zadanych parametrach; 2. program, który generuje określone językiem programowania programy lub ciągi funkcji standardowych
A. generator, AG — Applikations Generator
N. Generator (m), Erzeuger (m)
R. генератор, программ-генератор

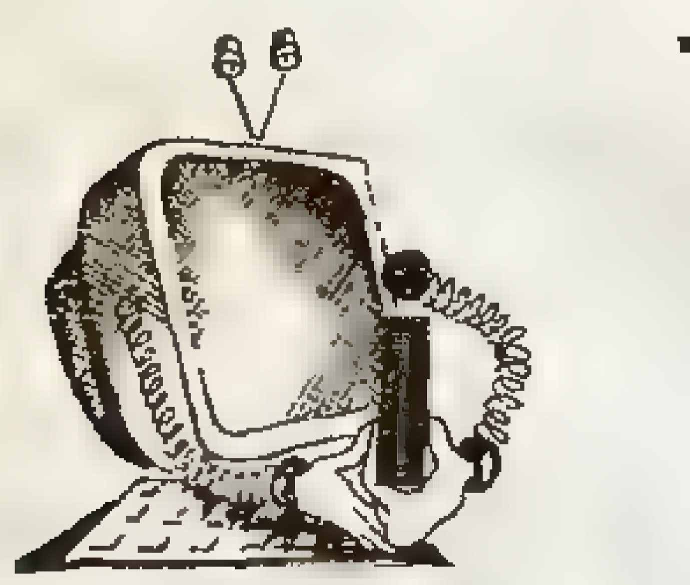
generator zegarowy, generator taktujący — generator impulsów napięciowych o wartościach odpowiadających 0 lub 1, synchronizujących (taktujących) przebieg procesów w układach synchronicznych na bazie cyklu podstawowego (generator cyklu podstawowego) lub cyklów pochodnych od tego ostatniego
A. clock generator, timing generator
N. Taktgeber (m), Taktgenerator (m)
R. генератор тактов

gęstość upakowania — 1. zagęszczenie elementów podstawowych w gotowym do pracy urządzeniu lub zespole; 2. gęstość zapisu danych
A. packing density
N. Packungsdichte (f)
R. 1. плотность упаковки
2. плотность записи

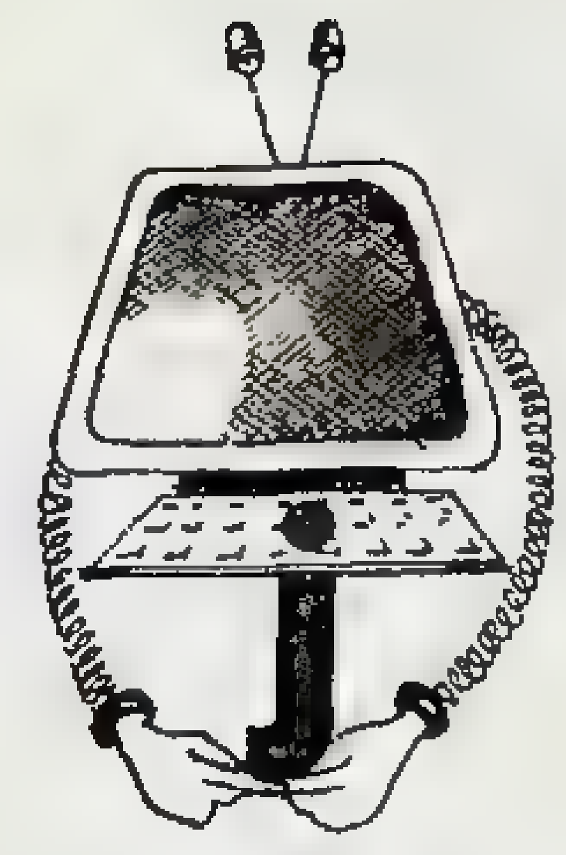
grafoskop → monitor ekranowy, który za pomocą promienia katodowego może wyświetlać dowolne krzywe, może on być wyposażony również w pióro świetlne (pisak świetlny) do przekazywania danych do komputera.
A. graphic display unit
N. grafisches Sichtanzeigegerät (n)
R. графическое иникоторное устройство



H — Hexidecimal — znak zapisu szesnastkowego
hardware → sprzęt komputerowy
Helwett-Packard — firma z Palo Alto (Kalifornia) produkująca systemy komputerowe, wprowadziła na rynek przenośny komputer osobisty pod nazwą HP-IPC (HP-Integral Personal Computer) — pierwszy na rynku amerykańskim z systemem operacyjnym UNIX. Mikroprocesor 16-bitowy, pamięci 256kB ROM i 512kB RAM oraz 32 kB do wyświetlania danych na ekranie, pamięć zewnętrzna — dyskietki 3 i 1/2-calowe o pojemności użytkowej 710kB.
HGC — Hercules Graphies Cart — pakiet (karta) sterowania grafiką firmy Hercules.



IBM — International Business Machines — największy producent sprzętu komputerowego na świecie. Wielu producentów mikrokomputerów dąży do tego aby ich sprzęt był kompatybilny z urządzeniami IBM.
IBM PC — IBM Personal Computer — komputer osobisty IBM (kolejne wersje XT i AT) stał się nieoficjalnym standardem mikrokomputerów profesjonalnych.
IC — Integrated Circuit — obwód scalony
ICL — International Computers Limited — największa komputerowa firma brytyjska, mająca swoje oddziały w ponad 80 krajach świata. W Polsce od ponad dwudziestu lat działa Biuro Handlowe ICL oraz ponad 400 systemów komputerowych ICL.
IEEE — Institute of Electrical and Electronic Engineers — Stowarzyszenie Inżynierów Elektryków i Elektroników USA
informatyka — nauka o systemach przetwarzania informacji
A. computing (computer) science, computer Theory
N. Informatik (f) Datenerfaassung — verarbeitung u. Programmierung
R. информатика
inicjacja — uruchamianie pierwotne np. programu, danych, pamięci zewnętrznej itp. (**i.** ma miejsce zazwyczaj natychmiast po włączeniu danego urządzenia)
A. initialization, initiation
N. Initialisierung (f) Initierung (f)
R. инициация, (за) пуск программы
instrukcja — przepis działania sformułowany w języku programowania, **i.** może się składać z **i.** cząstkowych; **i.**, której nie może rozłożyć na cząstkowe nazywa się rozkazem.
A. statement, instruction
N. Anweisung (f), Instruktion (f)
R. оператор
interfejs → złącze
integracja → skala integracji
interpreter — 1. program, który w sposób kroczący (sekwencyjny) interpretuje instrukcje programu napisanego w języku innym niż język komputera i które są przez komputer na bieżąco realizowane. Interpreter różni się tym między innymi od kompilatora i → translatora, że nie tworzy nowego, pośredniego programu (programu wynikowego), a interpretowane instrukcje są realizowane bezpośrednio przez komputer; 2. przyrząd, który czytając dane zawarte w nośnikach perforowanych rozszyfrowuje je i pisze tekstem czytelnym.
A. interpreter, interpretive program
N. Interpreter, (m), Interpretierer (m)
1. interpretierendes Program (n)
2. Lochschriftübersetzer (m)
R. 1. интерпретирующая программа
2. устройство для распечатки перфокарм
interpretator → interpreter

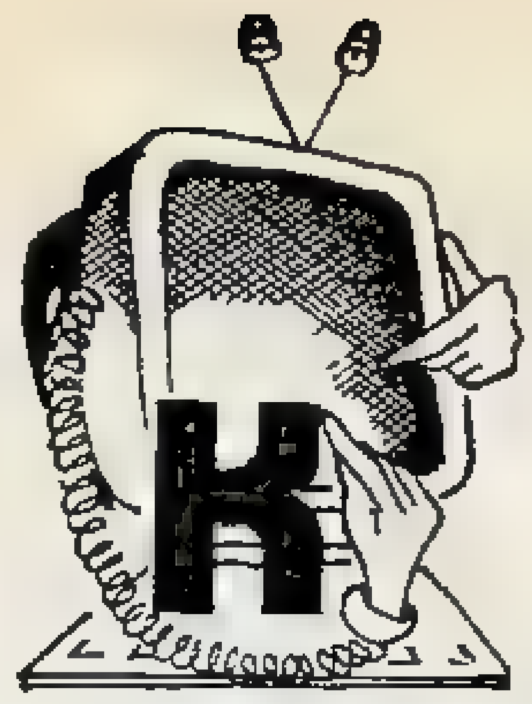


jednokierunkowa — przymiotnik magistrali, na której przesył danych może odbywać się tylko w jednym kierunku
A. uni-directional
N. unidirektional
R.

jednostka arytmetyczno-logiczna — wyróżniany w arytmometrze lub mikroprocesorze (procesorze) podstawowy układ funkcjonalny, który sygnały wyjściowe kojarzy w sposób logiczny lub arytmetyczny w zależności od stanu wejść sterujących
A. arithmetic-logic unit, ALU
N. arithmetisch-logische Einheit (f)
R. арифметический и логический блок
jednostka centralna, procesor centralny — część komputera, na którą składa się jednostka sterująca, arytmometr, urządzenie we/wy i pamięc operacyjna, a w której dokonuje się operacji logiczno-arytmetycznych oraz koordynuje i steruje działaniem innych jednostek.
A. central processing init. CPU
N. Zentraleinheit (f)
R. центральное устройство
jednostka sterująca — część jednostki centralnej wzgl. mikroprocesora (procesora), w której są inicjowane i kontrolowane niezbędne przebiegi do zrealizowania danego rozkazu, w mikrokomputerze do **j.s.** zalicza się: licznik rozkazów (programu), dekoder rozkazów, rejestr rozkazów, zegar (generator taktów).
A. control unit
N. Steuerwerk (n), Leitwerk (n), Kommandowerk (n), Kontrolleinheit (f)
R. устройство управления, блок управления
język asemblerowy → język typu asembler
język komputera — język programowania, w którym stosuje się wyłącznie rozkazy objęte listą rozkazów danego mikrokomputera (komputera) oraz adresy bezwzględne komórek pamięci ułożone zgodnie ze składnią tego języka
A. computer language, machine language
N. Maschinensprache (f), Rechnersprache (f)
R. машинный язык
język maszynowy → język komputera
język niezależny od komputera — język programowania, który nie jest ukierunkowany na żaden sprzęt i w zasadzie może być realizowany na każdym mikrokomputerze (komputerze), wymaga jednak w praktycznej realizacji tłumaczenia przez kompilator lub interpreter
A. computer independent language
N. maschinenunabhängige Sprache (f)
R. машинно-независимый язык

język pierwotny → język źródłowy
język proceduralnie zorientowany → język zorientowany problemowo
języki programowania wyższego poziomu — języki programowania maszynowo niezależne, których translacja odbywa się przez kompilator lub interpreter; porównaj język zorientowany problemowo
A. high level language
N. hohere Programmiersprache (f), Hochsprache (f)
R. язык высокого уровня
język sterowania zadaniami — język do opisywania części systemu operacyjnego zawierającego informacje sterowania zadaniami
A. job control language
N. Job-Kontrollsprache (f), Job-Steuersprache (f)
R. язык управления заданиями
język symboliczny → język typu asembler
język typu asembler — zorientowany maszynowo symboliczny język programowania, w którym rozkazy i większość adresów oparta jest na kodzie mnemotechnicznym; jest to język z grupy języków niskiego poziomu programowania (zorientowanych sprzętowo)
A. assembly (program) language, assembler language
N. Assembl(ier), ersprache (f)
R. язык ассемблера
język typu kompilator — język programowania na ogół zorientowany problemowo, a maszynowo niezależny, czyli wymagający kompilatora (programu kompilującego) do przetwarzania na program w kodzie maszyny, synonim języków programowania wyższego rzędu (wyższego poziomu)
A. compiler language
N. Compiler-Sprache (f)
R.
język ukierunkowany maszynowo → język zorientowany maszynowo
język wewnętrzny → język komputera
język zorientowany maszynowo — język programowania, którego instrukcje są podobnej lub takiej samej struktury, jak rozkazy danego komputera, a liczba instrukcji odpowiada liczbie rozkazów: do **j.z.m.** należą języki typu asembler
A. computer oriented language
N. maschinenorientierte Sprache (f), MOPS, maschinennahe Programmiersprache (f), rechnerorientierte Sprache (f)
R. машинно-ориентированный язык
język zorientowany problemowo — język programowania przystosowany do wygodnego formułowania określonej klasy problemów do rozwiązania na komputerze bliżej nie określonym (język niezależny od komputera), np. ALGOL, COBOL, FORTRAN itp.
A. problem oriented language
N. problemorientierte Sprache (f), genutzerorientierte Sprache (f), PPS — Problemorientierte Programmier Sprache
R. проблемно-ориентированный язык

język zorientowany proceduralnie → język zorientowany problemowo
język źródłowy — język programowania użyty przez programistę do sformułowania lub zakodowania programu, który przed użyciem go w komputerze musi być tłumaczony jednym z programów translacyjnych
A. source language
N. Quellsprache (f), Ursprungsprache (f)
R. исходный язык, входной язык
JS EMC — Jednolity System Elektronicznych Maszyn Cyfrowych wytwarzanych w krajach RWPG (komputery serii RIAD)
Junior → Elwro 800 Junior



kanał — logiczne połączenie jednostki centralnej z urządzeniami we/wy do optymalizowania przepływu danych; może być wyposażony w procesor
A. channel
N. Kanal (n)
R.
karta (płyta) obwodu drukowanego — płyta z materiału izolacyjnego, na której wykonano z cienkiej warstwy metalu obwód elektryczny i zamontowano układ funkcjonalny w postaci → modułu, karty takie zaopatrzone w elementy styków wsuwane są (bezpośrednio lub w pakietach czy kasetach) np. do szyn konsoli i (lub) gniazd wtykowych
A. printed circuit board (card)
N. Karte mit gedrukter Schaltung (f), gedruckte Leiterplatte (f), Kartenbaugruppe (f), Platine (f)
R. модуль (пластинка) с печатным монтажом, печатная плата
karta wtykowa — karta (płyta) obwodu drukowanego z układem funkcjonalnym w postaci wsuwki o znormalizowanych gabarytach (np. Eurocard 160×112×84)
A. plug-in (circuit) card
N. steskbare Schaltungskarte (f), Einsteckkarte (f), Einschub (f), Karte (f), Platine (f)
R. съемная карта
kaseta — wsuwka, oprawka (element zbiorczy) stosowana w systemach kasetowych pamięci i w budowie modułowej układów
A. cassette, stacker, casket
N. Kasette (f)
R. кассета
kaseta taśmy magnetycznej — tani system pamięci zewnętrznych o dostępie sekwencyjnym (stosunkowo wolnym)
A. cartridge, magnetic — tape casette (casket) store
N. Magnetbandkassette (f), Magnetbandkassettenspeincher (m)
R. кассета магнитной ленты
kasownik pamięci — przyrząd służący do kasowania pamięci reprogramowalnych, należy do → sprzętu pomocniczo-projektowego
A. erasel head
N. PROM-Lóschgerat
R.
klawiatura — składa się z klawiszy funkcyjnych i klawiszy danych (literowych i cyfrowych)
A. keybord
N. Tastatur (f)
R. клавиатура
kod — system umownych znaków (sygnałów) umożliwiający przedstawienie określonych informacji w sposób jednoznaczny i odpowiedni do danego celu; w komputerze każda informacja (instrukcja) musi być kolejno kodowana (język programowania → asemblery → kompilatory → interpretery) aż do możliwości jego przyswojenia, tj. rozpoznawania tylko dwóch stanów (0 — 1; L — H; przepływ prądu lub jego brak)
A. code
N. Kode (m)
R. код
kod BCD — BINARY CODED DECIMAL → kod dwuwartościowy dla cyfr dziesiętnych
kod dziesiętno-binarny → kod dwuwartościowy dla cyfr dziesiętnych
kod binarno-dziesiętny → kod dwuwartościowy dla cyfr dziesiętnych

kod dwuwartościowy dla cyfr (liczb) dziesiętnych — kod, w którym stosuje się sygnały w postaci dwuwartościowej (binarnej, bo praktycznie tylko takie mogą być w układach cyfrowych stosowane), w której to zakodowane są poszczególne cyfry (pozycje) a nie całe liczby zapisu dziesiętnego. Na każdą cyfrę (pozycję) przeznacza się grupę czterobitową (tetradę) o wartości (wadze) poszczególnych pozycji stanowiących kolejne wartości potęgi o podstawie 2, a więc np 8—4—2—1 (stąd niesłuszne częste utożsamianie kodu BCD z kodem „8 4 2 1", a jest to tylko jedna z wersji kodów BCD)
A. binary coded decimal code, binary-decimal code, decimal binary code, BCD-code
N. Binärcode für Dezimalziffern, Binär-Dezimal-Code (m), Dezimal-Binär-Code (m), BCD — Code (m)
R. двоично-десятичный код
kod dziesiętny → zapis dziesiętny
kod EBCDI, KOI-8 — rozszerzony 8-bitowy → kod BCD, do kodowania znaków alfanumerycznych, stosowany w sprzęcie IBM, podobny do kodu ASCII
A. EBCDI — code, extended binary coded decimal interchangecode
N. EBCDI — Code, EBCDIC
R. расширенный двоично-десятичный код обмена
kod heksadecymalny — kod dwuwartościowy (binarny wykorzystujący → zapis szesnastkowy (o podstawie 16) w celu skrócenia długości zapisu słów binarnych przy wykorzystaniu 16 cyfr, tj. 0...9, A=10, B=11, C=12, D=13, E=14, F=15 umieszczonych na pozycjach, którym przyporządkowano kolejne potęgi liczby 16.
A. hexadecimal (sedecimal) code, hex code
N. hexadezimal (sedezimal) Kode (m)
R. шестнадцатеричный код
kod mnemotechniczny — kod uwzględniający zasady mnemotechniki, tłumaczenie **k.m.** na kod komputera dokonywane jest w translatorach
A. mnemonic code
N. mnemotechnischer Kode (m)
R. мнемонический код
kod operacji — część kodu rozkazowego (część operacyjna) określająca rodzaj przewidzianej do wykonania operacji
A. operation code, opcode
N. Operationskode (m), Befehlokode (m), Operations — (Befehls) Schlüssel (m)
R. код операции
kod wewnętrzny — kod wewnętrzny (maszynowy) komputera
kod wynikowy — kod docelowy, czyli ostateczna forma przedstawiania rozkazów komputera, w jakiej je może akceptować dany mikrokomputer (komputer), a przynajmniej jego program ładowania, tzn. mogą one być zakodowane w postaci ciągów 0/1 (zero-jedynkowych)
A. object code
N. Objektkode (m)
R. 'объект код
kody ISO — odpowiadają standardowi ISO, jak np. KOI-7 stosowany w komputerach JS i IBM (7 — bitów informacyjnych 8 — ścieżka ma znak parzystości (lub kod ISO) R 111 z września 1969 r. i 150/R840 z października 1968 r. stosowane głównie w układach sterowania cyfrowego np. OSN
A. ISO — code
N. ISO — Code (m)
R. код ИСО
kolumna → stos
komentarz — objaśnienia (słowa, znaki) zawarte w programie (opracowanym w języku typu asembler) służą wyłącznie do jego objaśnienia a nie stanowią rozkazów, instrukcji, czy makrorozkazów do mikrokomputera (komputera) i są przez niego ignorowane
A. comment
N. kommentar (m)
R. комментарии
kompatybilność → wymienność
kompilator — 1. translator programu opracowanego w języku problemowo zorientowanym tworzący program w kodzie komputera (jednoetapowo) lub w języku asembler, a następnie przeprowadza się translację na program w kodzie komputera (wieloetapowo); 2. układ do kompilacji (na ogół z wykorzystaniem mikroprocesora)
A. 1. compiler, compiling program
2. compiler
N. 1. Kompilierer (m), Compilierer (m)
2. Compiler (m), Übersetzer (m)
R. транслятор, компилятор
kompilator skrośny → kroskompilator

kompleksowo skomputeryzowane wytwarzanie, CIM → elastyczne kompleksy wytwórcze
komputer — zestaw urządzeń (lub urządzenie) o programowalnej pamięci do przetwarzania danych, które po wprowadzeniu do pamięci programu oraz danych przetwarza je wg programu w sposób automatyczny bez potrzeby ingerencji człowieka. (Charakter tego opracowania nie pozwala jednak nawet na ogólną charakterystykę — uwzględniającą budowę, przeznaczenie, sposób działania itd. — wszystkich odmian tego sprzętu). Przyjęty podział główny, to:
a) z uwagi na sposób działania:
komputer analogowy
A. analog computer
N. Analogrechner (m)
R. аналоговая вычислительная машина
komputer cyfrowy
A. Digital computer
N. Digitalrechner (m), Digitalanlage (f)
R. цифровая вычислительная машина
b) z uwagi na podstawowe dziedziny zastosowań:
komputer do przetwarzania danych, k. do EPD
A. data processing computer (machine)
N. Datenverarbeitungsanlage (f), DVA, EDVA, Datenverarbeitungsmaschine (f)
R. машина для обработки данных
komputer do obliczeń naukowo-technicznych
A. scientific and engineering information processing computer
N. wissenschaftlicher Rechner (m)
R. машина для обработки научно-технической информации
komputer do sterowania procesów, k. sterujący (kontrolno-sterujący)
A. process control computer
N. Prozess(leit)rechner (m)
R.
A. computer
N. Rechner (m), Rechenanlage (f), Computer (m)
R. вычислительная машина, вычислительное устройство, ЭВМ (электронная вычислительная машина)
komputer biurowy, osobisty — w pełni samodzielny mikrokomputer zbudowany najczęściej w postaci jednego bloku (jednostki) dzięki zastosowaniu obwodów scalonych o wysokim stopniu integracji. W niektórych zastosowaniach ustępuje komputerom tradycyjnym (pojemnością pamięci wewnętrznej, szybkością operacji), w innych je przewyższa (bezpośredniość dostępu i oprogramowania pod użytkownika — stąd przymiotnik „osobisty", prostota wykonania — stąd niska cena). Główne przeznaczenie, to zastosowanie do prac inżynierskich (wyższy stopień kalkulatorów biurowych), w systemach testujących i sterujących, do emulacji różnych

procesów i systemów, może być również wykorzystywany jako terminal w dużych systemach komputerowych. Składa się, poza systemem wewnętrznym, na ogół z urządzeń we/wy, plotera i ekranu monitorowego
A. office desktop computer, personal computer, PC
N. Tischrechner (m)
R.

komputer wspomagający, komputer gościnny — komputer wspomagający (cross support) opracowanie i testowanie programów do innych komputerów
A. host computer
N. Gast-computer (m)
R.

komputerowe wspomaganie projektowania (konstruowania) — automatyzacja prac projektowych, komputeryzacja projektowania; projektowanie wspomagane komputerowo można podzielić na dwa poziomy wynikające z rodzaju sprzętu komputerowego, jakiego wymaga rozwiązanie zadania projektowego: 1. typowe obliczenia inżynierskie do zadań dających się ująć w stosunkowo proste algorytmy i nie wymagających zbyt obszernych pamięci operacyjnych, wykonywane na tzw. komputerach biurowych lub osobistych (mini- lub mikrokomputery); 2. zadania wymagające dużych pamięci i/lub specjalnego oprogramowania (software, firmwarę), jak również wymagające rozległej wiedzy, metod i technik komputerowych, np. modelowanie, symulacja i optymalizacja w projektowaniu układów scalonych bardzo wielkiej skali integracji lub konstruowaniu części o złożonych kształtach, gdzie tzw. ręczne projektowanie jest zbyt czasochłonne, a przede wszystkim za mało dokładne. Zadanie takie rozwiązuje się za pomocą dużych, o rozbudowanej peryferii komputerów.
A. Computer Aided Design, CAD, Computer Aided Engineering, CAE
N. rechnerunterstützter Entwurf (m), — Konstruktion (f)
R.

komputerowe wspomaganie produkcji — sterowanie produkcji za pomocą komputera, obejmuje zarówno techniczne przygotowanie produkcji (planowanie: technologiczne, produkcyjne, zaopatrzenia materiałowego, narzędziowego itd.), jak i sterowanie procesem przebiegu produkcji (rozdzielnictwo robót, transport wewnętrzny, nadzór stanu technicznego maszyn i urządzeń, kontrola jakości, informacja i rejestracja przebiegu produkcji), a docelowo przejście na całkowicie zautomatyzowane → elastyczne kompleksy wytwórcze, tzw. fabryki bezzałogowe.
A. Computer Aided Manufacturing, CAM
N. rechnerunterstützte Fertigung (f)
R.

komputerowe wspomaganie projektowania i produkcji — połączenie projektowania i sterowania produkcji (CAD i CAM) w jeden kompleksowy system komputerowy
A. CAD/CAM, CADAM
N. CAD/CAM, CADAM
R.

konfiguracja — zestaw (układ) sprzętu komputerowego (hardware) do określonych zadań, ale również jego wydajność i możliwości funkcjonalne, możliwości rozbudowy itd. W odniesieniu do mikrokomputerów konfiguracja wynika częściowo z ich architektury
A. configuration
N. Konfiguration (f)
R. конфигурация

końcówka → urządzenie końcowe

konsola — 1. wydzielony pulpit operatora zwany też pulpitem sterowania, na ogół z monitorem operatora; 2. wspornik zespołów i elementów.
A. console
N. Konsole (f), Bedienungskonsole (f)
R. консоль

kopia zawartości pamięci — 1. kopia lub wyciąg zawartości pamięci; 2. proces wykonywania kopii lub wyciągu pamięci
A. dump, dumping
N. Dump
R. перезапись

korezydentne → programy korezydentne

kontroler — sterownik, programator, patrz też: → programowalne układy sterowania oraz → podział funkcjonalny mikroprocesorów

koordynatograf → urządzenie rysujące

koprocesor arytmetyczny → NDP

kostka — 1. popularna nazwa jednego („jednostki") układu scalonego, w ścisłym znaczeniu o kostkach mówi się tylko w pewnej fazie produkcji półprzewodnikowych układów scalonych, przy cięciu większych płytek (ang. wafer) na poszczególne kostki; 2. kompletny mikroukład zamknięty w obudowie
A. chip
N. Chip (m)
R. цип

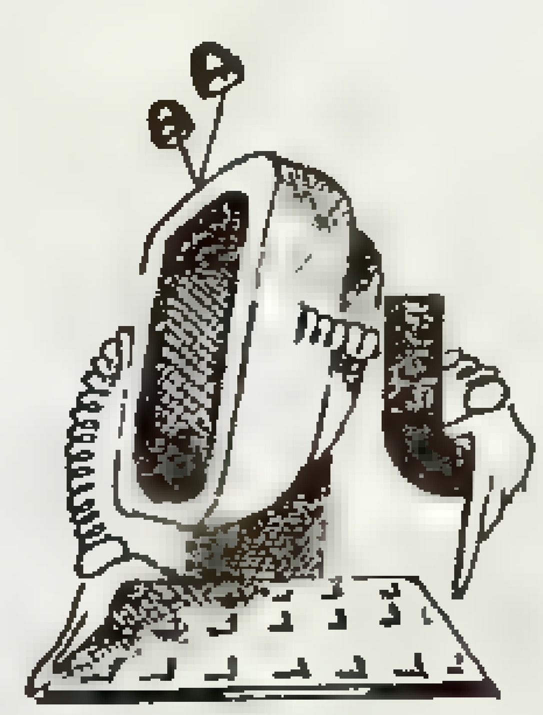
kość → kostka

krosasembler, asembler skrośny — program, którym dokonuje się asemblowania programów z przeznaczeniem do innych mikrokomputerów (komputerów) niż ten, na którym je asemblowano
A. cross — assembler
N. Cross — Assembler (m)
R. кросс — ассемблер

kroskompilator, kompilator skrośny — przeznaczenie jak → krosasemblera, lecz do języków programowania problemowo zorientowanych
A. cross — compiler
N. Cross — Compiler (m)
R. кросс — компилятор

kumulacja → kumulowanie danych

kumulowanie danych — redukowanie pojedynczych informacji do treści istotnej w dalszym przetwarzaniu lub wykorzystaniu, stosowane do zaoszczędzenia miejsca w pamięci lub bardziej skoncentrowanego informowania osób
A. data reduction
N. Datenverdichtung (f), Verdichtung (f), Datenreduktion (f)
R. сжатие (упломнение) данных



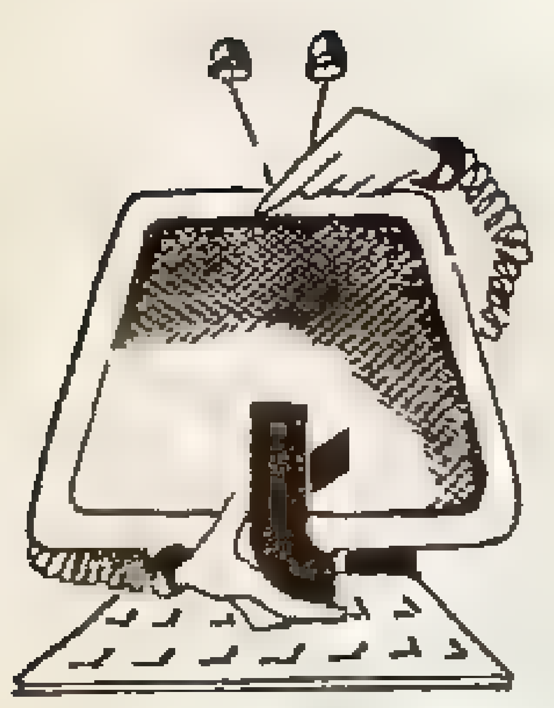
LCD — Liquid Cristal Display — wskaźnik ciekłokrystaliczny

LED — Light Emitting Diode — dioda elektroluminescencyjna

liczba binarna stałoprzecinkowa — liczba binarna przedstawiona w postaci bitów znaków i bitów cyfr, przy przetwarzaniu w systemie **l.b.s.** interpretacja miejsca przecinka należy wyłącznie do programisty, komputer w tym wypadku nie wyróżnia części całkowitej i ułamkowej.
A. fixed point binary number
N. binäre Festkommazahl (f), Festpunktzahl (f)
R. бинарное число. фиксированная запятая

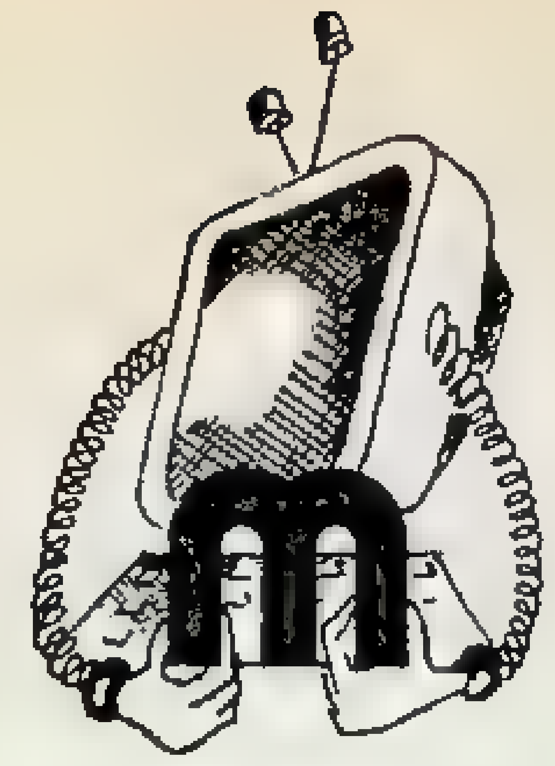
liczba binarna zmiennoprzecinkowa — liczba binarna,

która przedstawiona jest w zapisie „logarytmicznym" typu $2^W \cdot m$ w postaci wykładnika W zwanego cechą i mantysą m. Przy przetwarzaniu **l.b.z.** miejsce przecinka komputer uwzględnia automatycznie
A. floating point binary number
N. binäre Gleitkommazahl (f)
R. бинарное число с плывающей запятой
licznik programu — rejestr rozkazów — licznik służący do tworzenia i przechowywania adresu komórki pamięci, w którym znajduje się kolejny rozkaz przeznaczony do realizacji
A. program counter PC, instruction counter, command counter
N. Befehlszähler (m), Programm(schritt)zähler (m)
R. счетчик программы (комэнд)
LIFO — (memory) — Last in First Out — (memory) — pamięć kolumnowa pracująca na zasadzie logicznej → stosu
lista — 1. lista adresów (lista indeksowa) danych w pamięci, znajduje się w pamięci operacyjnej (A. directory, supscript list, N. Adressbuch (n), Indexlista (f)); 2. tablica adresów (lista adresów) języka programowania (A. adresstable, N. Adressentafel (f)); 3. lista zbiorów danych (A. data file directory, N. Datenverzeichnis (n), Dateiverzeichnis (n))
A. directory
N. Verzeichnis (n)
R. оглавление, список
LISP — LISŁ Processing language — język programowania LISP
lista rozkazów — spis repertuaru rozkazów wraz z ich opisem
A. instruction list, order list, command list
N. Befehlsliste (f)
R. список команд, состав команд, список инструкции
logiczne układy podstawowe → układ elementarny
lmp — line per minuts — wierszy (linii) na minutę
lps — line per seconds — wierszy (linii) na sekundę
LPS — Low Power Schottky — tranzystor (dioda) Schottky'ego małej mocy
LSB — Least Significant Bit → bit najmniej znaczący
LSI — Large Scale Integration → integracja wielkiej skali
LSM — Low — Speed — Moden → układ towarzyszący



ładowanie — wprowadzanie programu lub danych do pamięci komputera
A. loading
N. Laden (n)
R. загрузка, нагрузка

ładowacz pamięci trwałej — urządzenie do wprowadzania programu do pamięci stałej programowalnej (PROM), **ł.p.t.** należy do sprzętu pomocniczego-projektowego → urządzenie do programowania
A. programer, memory loading device
N. Programmer (m), Speicherladungsgerät (n)
R. программер



magistrala — zespół elementów lub układów logicznych i linii przesyłowych, do których przyłącza się kilka nadajników i/lub odbiorników celem umożliwienia wzajemnego przesyłania (komunikacji) informacji cyfrowych
A. bus, highway
N. Bus (m)
R. магистраль (шина)
magistrala adresowa — magistrala mikrokomputera (komputera), po której przesyłane są adresy
A. address bus
N. Adressbus (m)
R. магистраль адреса
magistrala danych — magistrala mikrokomputera (komputera), po której przesyłane są dane
A. data bus
N. Datenbus (m)
R. информационная линия (шина, магистраль)
makro — zamknięty odcinek programu lub sekwencja rozkazów (często mylone z makrorozkazem) w języku typu asembler, której się nie pisze lecz przywołuje (wywołuje makro), definicje **m.** i wywołania zawarte są w programie źródłowym, natomiast w trakcie asemblowania asembler makro zastępuje je odpowiednimi sekwencjami rozkazów
A. macro
N. Makro (m)
R. макро
makroasembler → asembler makro, patrz też → makro
manipulator — 1. urządzenie do ręcznego manipulowania, 2. program do kontroli i sterowania urządzeniami peryferyjnymi, 3. urządzenie do pomiarów i testowania układów scalonych — tester, 4. manipulator drążkowy, sterownik do sterowania ruchem figur w grach lub wskaźnika piszącego (rysującego) na monitorze ekranowym, 5. sterownik kursora na ekranie monitora
A. 1. i 2. handler
3. prober, 4. joystick, 5. mouse
N. 1. Handler (m), Manipuliervorrichtung (f)
2. Hantierer (m), Steuerungsprogram (n)
3. Prober (m), 4. Steuerknüppel (m)
5. Maus (f)
R. манипулятор
maska — 1. układ bitów lub znaków, który steruje eliminowaniem jednej części a zachowaniem (w pamięci lub w programie) innej części znaków maskowanego układu znaków, może służyć do eliminowania lub akceptowania określonych funkcji, np. → maska przerwań, 2. płytka z odpowiednią topografią do maskowania, stosowana w produkcji i do reprogramowania układów scalonych
A. mask
N. Maske (f)
R. маска
maska przerwań — wewnętrzne słowo sterujące, które uzależnione jest programowo i definiuje dla każdego przedziału czasowego, jakie z przerwań mają być akceptowane, a jakie ignorowane.
A. interrupt mask
N. Interrupt — Maske (f)
R. маска прерывания
maszyna matematyczna → komputer
MCGA — Multi Colour Graphics Array — układ grafi-

ki wielokolorowej (sterowanie grafiki monitora PS/2)
MCNC — Micro Computer Numerical Control → mikrokomputerowe sterowanie numeryczne
MDS — Microprocessor Development System → systemy uruchomieniowo rozwojowe.
Meritum — polski mikrokomputer, który po raz pierwszy zaprezentowano publicznie jesienią 1983 r., a w następnym roku wyprodukowano ponad tysiąc sztuk Meritum I w Zakładach Urządzeń Komputerowych „Mera-Elzab" w Zabrzu. Powstały dwie wersje Meritum I, a wkrótce — Meritum II, Meritum I wyposażone w 64kB pamięci RAM oraz polskie znaki na klawiaturze było najtańszym dostępnym mikrokomputerem na naszym rynku, umożliwiającym pewne zastosowania profesjonalne.
metajęzyk — język pośredni służący do opisu innego języka i umożliwiający jego użycie, znajduje zastosowanie w kompilatorach i asemblerach, których makrodefinicje pozwalają na rozpisywanie makrorozkazów w ciągi rozkazów komputera
A. meta language
N. Metasprache (f)
R. метаязык
metoda adresowania — charakteryzuje sposób przedstawienia adresów lub metodę szukania argumentów, w zasadzie istnieją trzy klasy metod adresowania: bezpośrednia, pośrednia i względna → adresowanie
A. addressing mode
N. Adreseierungsart (f)
R. адресация метода
metody programowania — techniki programowania, czyli podział programowania w zależności od użytych środków technicznych w odniesieniu do programowania obróbki części na OSN dla przykładu wyróżnić można następujące metody: ręczne, wspomagane maszynowo (zwane też półmaszynowym); maszynowe (zwane też automatycznym); podział technik programowania mikrokomputerów patrz pod → mikroprogramowanie komputerów pod →mikroprogramowanie
A. programming technique, programming method
N. Programmierungsverfahren (n)
R. метод программирования
MIDI — Musical Instruments Digital Interface — złącze do elektronicznych instrumentów muzycznych
mikroarchitektura — architektura mikrokomputera
A. microcomputer architecture
N. Mikrocomputerarchitektur (f)
R. архитектура микрокомпютера
mikro — asembler — asembler mikroprogramowalny mikrokomputerów
A. micro code assembler — micro language assembler
N. Mikroassembler (m), Assembler für Mikroprozessorsysteme
R. микроассеблер
mikroelektronika — dział elektroniki zajmujący się mikroelektronicznymi (ekstremalnie zminiaturyzowanymi) układami elektronicznymi, realizowanymi w formie scalonej
A. microelectronics
N. Mikroelektronik (f)
R. микроэлектроника
mikrokomputer — układ scalony, lub zestaw, który pod względem funkcjonalnym odpowiada komputerowi, zawiera on jednostkę centralną w postaci mikroprocesora, pamięć programu i danych, system we/wy oraz (coraz powszechniej) urządzenia peryferyjne w postaci drukarek, monitorów itd. → podział mikrokomputerów
A. microcomputer, μC, mC
N. Mikrocomputer (m)
R. микрокомпютер, микро-ЭВМ
mikrokomputer jednostkowy (monolityczny) — mikrokomputer wykonany na jednej płytce półprzewodnikowej zawierającej: mikroprocesor, pamięci RAM i ROM, generator zegarowy, układ kontroli przerwań, układy we/wy
A. single-chip microcomputer, single-board microcomputer-all-in-one chips
N. Einplatinen — Komputer (m)
R.
mikrokomputer wspomagający → komputer wspomagający
mikrokomputerowe sterowanie numeryczne — sterowanie numeryczne z zastosowaniem mikrokomputerów
A. MCNC, microcomputer numerical control
N. MCNC, Microcomputersteuerung (f)
R. управление с микро-ЭВМ
mikromoduł — modułowy układ przełączający o dużej gęstości pakowania (do 20 elementów podstawowych na 1 cm³) dzięki zastosowaniu półprzewodnikowych elementów podstawowych (mikromoduły wypierane są coraz bardziej przez układy scalone)
A. micromodule
N. Mikromodul (m)
R. микроблок
mikroprocesor — 1. układ scalony, który pod względem funkcjonalnym porównywalny jest z jednostką centralną komputera, zawiera jednostkę sterującą (cu) i arytmometryczno-logiczną (ALU); 2. popularne określenie każdego sprzętu kontrolnego, sterującego lub do przetwarzania danych — opartego na mikroprocesorach → podział funkcjonalny **m.**; jednoznacznych definicji i kryteriów podziału **m.** dotychczas nie opracowano.
A. microprocessor, μP, mP, MPU
N. Mikroprozessor (m)
R. микропроцессор
mikroprogram — składa się wyłącznie z mikrorozkazów, definiując w jaki sposób ma być wykonany rozkaz komputera
A. microprogram
N. Mikroprogram (n)
R. микропрограмма
mikrooprogramowanie → programowanie sprzętowe
mikroprogramowanie — 1. przygotowanie albo 2. użycie mikroprogramów, przy tym **ad.1.:** może to być a) „ręczne" programowanie opierające się na instrukcji z użyciem repertuaru mikroprogramów (niezwykle uciążliwe i na ogół stosuje się cross assemblery lub cross compilatory) lub b) wspomagane sprzętem pomocniczym, **ad. 2.:** dotyczy użycia oprogramowanych pamięci ROM w mikrokomputerze lub rozkładu rozkazów komputera na mikrorozkazy
A. microprogramming
N. Mikroprogrammierung (f)
R. микропрограммирование
mikroprogramowalny — przymiotnik sprzętu (np. mikrokomputera, pamięci), który ma możliwość mikroprogramowania
A. microprogrammable
N. Mikroprogrammierbar
R.
mikrorozkaz — 1. operacja logiczna (mikrooperacja), na której rozkładany jest rozkaz komputera w trakcie jego realizacji. 2. rozkaz mikrokomputera, który ten może realizować bezpośrednio, ponieważ jest to element jego repertuaru rozkazowego ciąg **m.** stanowi mikroprogram.
A. microinstruction, microcommand
N. Mikrobefehl (m)
R. микрокоманда
minidysk → dysk elastyczny
minikomputer — komputer zbudowany z układów scalonych, przeważnie o 16-bitowym formacie i rzadko

większej pojemności słów od 16 K, typowy do prac obliczeniowych, sterowania procesami i przetwarzania danych problemów mało i średnio złożonych, jako jednostka centralna jest bardziej sprawny od systemów mikrokomputerowych; **m.** są wypierane coraz bardziej przez mikrokomputery, które je naśladują, a czasami przewyższają
- A. minicomputer
- N. Minicomputer (m), Minirechner (m)
- R. мини-ЭВМ, миникомпютер

miniprocesor — 1. mikroprocesor naśladujący rozwiązania (architektura, lista instrukcji) minikomputerów SSI i MSI, 2. modularna jednostka centralna komputera charakteryzująca się krótkimi słowami rozkazu i szybkimi operacjami podstawowymi.
- A. miniprocessor
- N. Miniprocessor (m)
- R. минипроцессор

MIPS — Milion Instructions Per Second — milion operacji na sekundę — umowna jednostka prędkości przetwarzania

mnemo (tech)nika — 1. metody i środki służące do wspierania pamięci człowieka na zasadzie skojarzeń, 2. o nazwie lub skrócie instrukcji (rozkazu) , z której jej (jego) funkcji można łatwo się domyśleć — mówi się, że jest „wykonana" w **m.**
- A. mnemonic
- N. Mnemo (tech)nik (f)
- R. мнемотехника

modem — urządzenie (galwaniczne — połączone trwale lub akustyczne — przykładane do słuchawki telefonicznej) do modulacji (modulator i demodulator) danych cyfrowych (sygnałów dyskretnych) w trakcie teletransmisji łączami telefonicznymi
- A. modem
- N. Modem (m)
- R. модем

moduł — zunifikowany podzespół złożony ze standardowych elementów podstawowych
- A. module
- N. Modul (m), Boustein (m)
- R. модул

moduł mikroprocesorowy — N-bitowy (na ogół 4-bitowy) układ scalony w postaci modułu stanowiący element mikroprogramowalny mikroprocesora, który może być rozbudowywany kaskadowo w układy n×N-bitowe
- A. microprocessor element, bit slicemicroprocessor, μPe
- N. Mikroprozessorelement (n), Bit — Slice
- R. модул (блок) микропроцессорный

monitor — 1. urządzenie, które zawiera urządzenia we/wy i na ogół ekran (→ monitor ekranowy), pozwala (łącznie z 2) na obserwowanie, kontrolę, sterowanie i weryfikowanie przebiegów przetwarzania; 2. → program — monitor; 3. system operacyjny
- A. monitor, supervisor
- N. Monitor (m)
- R. монитор

monitor ekranowy — urządzenie peryferyjne do wprowadzania i wyprowadzania danych za pomocą ekranu i klawiatury, **m.e.** alfanumeryczne zwane są też alfaskopami, a graficzne → grafoskopami
- A. VDU — Visual Display Unit, display (— unit, video) CRT-Display (-Terminal), alphanumeric display
- N. Datensichtgerat (m), Sicht(anzeige)gerat (m), Schirmbildgerat (m), Anzeige (f), Bildschirm (m)
- R. дисплей

monitor uruchamiania — program, który jednoczy w sobie cechy programu monitora (sterującego) i programu usuwania błędów w trakcie — uruchamiania
- A. debug monitor
- N. Debug — Monitor (m)
- R. монитор отладки

MP/C — Multi Point (Control) — sterowanie ciągłe

MPU — Micro Processing (Processor) Unit → mikroprocesor

MSB — Mest Significant Bit → bit najbardziej znaczący

multirozkaz → rozkaz wielosłowny

multisłowo — zbiór symboli o długości odpowiadającej wielokrotności długości słowa czy formatu danych
- A. multilength word
- N. mehrstelliges Wort (n)
- R. м слово многократной длины

μC → mikrokomputer

μP → mikroprocesor

μPe → moduł mikroprocesorowy



nadmiar arytmetyczny — liczby przetwarzane nie mieszczą się w rejestrach, wskaźnikach itp.
- A. arythmetic overflow
- N. arithmetischer Überlauf (m)
- R. арифметическое переполнение

nadmiar → redundacja

nadzorowanie — śledzenie wykonywania programów mikrokomputerowych pod kontrolą programów wyszukiwania błędów lub symulujących, które wymuszają takie działanie, że po każdym wykonanym rozkazie następuje wydruk (wyświetlenie) zawartości pamięci
- A. trace
- N. Verfolgung (f), Ablaufverfolgung (f)
- R. слежение

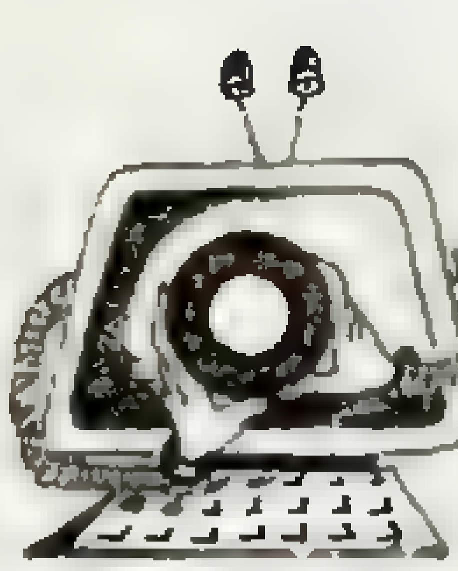
NC — Numerical Control — sterowanie numeryczne (SN); No Connection — brak połączenia

NDP — Numeric Data Processor — procesor do przetwarzania danych numerycznych

NOP — No Operation → rozkaz pusty

nośnik danych — materiał, w którym lub na którym określona zmienna wielkość fizyczna może reprezentować dane
- A. data carrier, data medium, recording medium
- N. Datenträger (m), Aufzeichnungsträger (m), Speichermedium (n)
- R. носитель данных (записей)

notacja dziesiętna → zapis dziesiętny



obliczanie adresu — działania arytmetyczne na adresach lub częściach adresów celem uzyskania → adresu aktualnego
- A. address arithmetic
- N. Adressrechnung (f)
- R. сосчитание адреса

obróbka danych → przetwarzanie danych

obszar pamięci danych — obszar pamięci przewidziany lub zarezerwowany dla danych

A. data area
N. Datenbereich (m)
R. область (зона) данных

obszar pracy (roboczy) — obszar pamięci systemu mikrokomputerowego, z którym mikroprocesor przeważnie współdziała
A. working space
N. Arbeitsbereich (m)
R.

obszar wprowadzania pierwotnego — obszar pamięci stale zajęty przez (lub zarezerwowany na) program wprowadzania pierwotnego
A. bootstrap are
N.
R.

obwód logiczny — obwód zbudowany z elementów podstawowych, którego działanie można opisać za pomocą operacji logicznych pod względem funkcjonalnym. **O.l.** może być identyfikowany z → układem logicznym
A. logic (al) circuit
N. logische Schaltung (f), logische Verknüpfung (f), Logikschaltung (f)
R. логическая схема

obwód przełączający — logiczna jednostka przełączająca złożona z elementów podstawowych. Pod względem funkcjonalnym **o.p.** może być identyczny z elementami lub układami przełączającymi
A. switching circuit
N. Schaltkreis (m), Schaltungsverknüpfung (f)
R.

obwód drukowany, p.c. — printed circuit → karta obwodu drukowanego

obwód scalony → układ scalony

ochrona pamięci — cecha mikrokomputera (komputera) zapobiegająca przypadkowym zapisom lub odczytom w określonych obszarach pamięci operacyjnej
A. memory protection
N. Speicherschutz (m)
R. защита памяти

opcja → wariant

operacja — umownie wyróżniona czynność, np. czynność komputera określona za pomocą pojedynczego rozkazu
A. operation
N. Operation (f)
R. операция

operand → argument

operator — 1. obsługujący sprzęt komputerowy, obrabiarkę sterowaną numerycznie, itp.; 2. część (symbol) zależności matematycznej, która określa jak daną operację mat. wykonać
A. operator
N. Operator (m), Bediener (m)
R. оператор

opis pola — dane o ukształtowaniu (formatu, długości, organizacji, zawartości) określonego pola
A. field discription
N. Feldbeschreibung (f)
R. описание поля (зоны, массива)

oprogramowanie — software, pojęciem tym określa się wszystkie rodzaje programów i dokumentacji przynależnej, dostarczanej przez producenta sprzętu
A. software, immaterial ware
N. Software (f), immaterial Ware (f), Weichware (f)
R. софвер

oprogramowanie mikroprogramowe → oprogramowanie sprzętowe

oprogramowanie sprzętowe — oprogramowanie standardowe stanowiące stały składnik sprzętu, np. programy zawarte w pamięciach typu ROM (w tym również postaci mikroprogramów)
A. firmware
N. Firmware (f), Festware (f), Standardsoftware (f)
R. опрограммирование хардверове

opróżnione — stan pamięci (rejestru) po przetworzeniu danych (zrealizowaniu programu)
A. empty
N. leer, abgearbeitet
R. пустой

OS/2 — Operating System/2 — system operacyjny PS/2

OSN — obrabiarka sterowana numerycznie
A. numerical control machine, NC machine
N. numerisch geteuerty Maschine (f), NC Maschine (f)
R. цифровое (числоговые) управление [УПУ] станка

overlay — 1. metoda podziału programu na segmenty w celu lepszego wykorzystania pamięci (wczytuje się je metodą nakładania, tzn. kiedy jeden segment stopniowo, opróżnia pamięć, na to miejsce wczytuje się następny) 2. nakładane segmenty programu, 3. nakładkowe przetwarzanie instrukcji
A. overlay
1. operating system,
2. program part in the memory,
3. pipelinning overlap
N. Überlagerung (f), Überlappung (f), Overlay
1. Betriebssystem overlay
2. Überlagerungssegment,
3. Überlappung
R. оверлей, перекрытие, суперпозиция

oznacznik zgodności — bit lub inny znak wskazujący na spełnienie warunku
A. tag
N. Kennzeichen (n), Markierung (f)
R. меток, признак



PA → PLA

pakiet — 1. zestaw kart obwodów drukowanych określonego standardu (np. European standard size p.c. board) stosowany w budowie modułowej układów; 2. pakiet programów przeznaczonych do określonego komputera lub do rozwiązania określonego problemu
A. 1. flat (-pack) subassembly
2. software package
N. 1. Flachbaugruppe (f)
2. Programmpacket (n)
R. пакет

pakowanie danych — zapisywanie danych w pamięci komputera w sposób zwarty (patrz też: upakowany, format upakowany, gęstość upakowania, kumulacja danych)
A. packing
N. Packen (n)
R. упаковка данных

PAL — Programmable Array Logic — programowalne układy logiczne

pamięć — układy i środki do przechowywania informacji w określonym czasie i udostępniające je, bez zmiany ich treści, na każde odpowiednio określone żądanie
A. store, storage, memory
N. Speicher (m)
R. память

pamięć danych — pamięć, w której przechowuje się wyłącznie dane w mikrokomputerze realizowane przez pamięć RAM
A. data storage
N. Datenspeicher (m)
R. память для данных

pamięć domenowa, pęcherzykowa — pamięć półprzewodnikowa (typu RAM lecz o stopniu trwałości typu ROM), działająca na zasadzie efektu pęcherzyków magnetycznych, o architekturze blokowej (np. 2048 stron po 512 bitów) z sumaryczną pojemnością np. 1 Mbit
A. Bubble (domains) Memory, BM
N. Magnetblasenspeicher (m), Blasenspeicher (m)
R.

pamięć dyskowa — zewnętrzna pamięć na → dysku
A. magnetic disc memory, magnetic disc store
N. Magnetplattenspeicher (m)
R.

pamięć ferrytowa → pamięć rdzeniowa

pamięć kolumnowa → LIFO

pamięć laserowa — urządzenie, w którym zapis i odczyt odbywa się za pomocą promieni laserowych (duża szybkość — 300 tys. znaków na sekundę, trwałość i gęstość zapisu — pojemność około 500 MB (**p.l.** jest stosowana w mikrokomputerach w postaci → CD

pamięć magnetyczna rdzeniowa → pamięć rdzeniowa

pamięć masowa — bliżej nie sprecyzowane określenie pamięci o dużych możliwościach, nazwy tej używa się często do określenia pamięci zewnętrznych
A. mass storage, bulk storage
N. Massenspeicher (m)
R. запоминающие устройство болшой емкости

pamięć nietrwała — pamięć, w której przechowywane informacje zanikają po wyłączeniu zasilania
A. volatile story (memory)
N. flüchtiger Speicher (m)
R. память с разрушением информации (при выключении питания)

pamięć „notatnikowa" — pamięć małej pojemności lub tylko obszar pamięci lub rejestrów jako baza do przechowywania wyników pośrednich
A. scratch pad memory
N. Notitzblock-Speicher (m)
R. память-блокнот

pamięć operacyjna — pamięć typu zapis/odczyt o dostępie swobodnym (bezpośrednim) do programów, wyników pośrednich, ewentualnie danych, z którą współpracuje głównie jednostka centralna
A. main storage, operating storage, working storage, main memory, central storage, general storage, primary storage
N. Hauptspeicher (m), Arbeitsspeicher (m), Primarspeicher (m) Zentralspeicher (m)
R. основная память, оперативное запоминающее устройство озу

pamięć pęcherzykowata → pamięć domenowa

pamięć półprzewodnikowa — pamięć w postaci półprzewodnikowych układów scalonych, o dużych i ciągle rosnących pojemnościach w jednym układzie (1983 r. — 64 k; 1986 r. — 256 k; w opracowaniu pamięci RAM wykonywana metodą litografii laserowej — 16 MB)
A. semiconductor storage, semiconductor memory
N. Halbleiterspeicher (m), mikroelektronischer Speicher (m)
R. полупроводниковая память

pamięć programu — pamięć (lub obszar pamięci), w której przechowuje się wyłącznie programy
A. program storage
N. Programmspeicher (m)
R. запоминание программы

pamięć robocza — pamięć operacyjna lub zarezerwowana przez program część pamięci
A. working store, working memory
N. Arbeitsspeicher (m)
R. рабочее запоминающее устройство

pamięć rdzeniowa — pamięć o bezpośrednim i stosunkowo szybkim dostępie, zbudowana z rdzeni ferrytowych
A. core memory, magnetic core storage, ferrite storage
N. Kernspeicher (m), Magnetkernspeicher (m) Ferritspeicher (m)
R. запоминающее устройство на ферритовых сердечниках

pamięć trwała — urządzenie do przechowywania informacji niezanikających przy wyłączeniu zasilania
A. non-volatile store (memory)
N. nichtflüchtiger Speicher (m)
R. постоянная память

pamięć ulotna — urządzenie do przechowywania informacji zanikających po wyłączeniu zasilania
A. velatile store
N. flüchtiger Speicher (m)
R. непостоянная память

pamięć wirtualna — pamięć nominalna, obszar pamięci wyznaczany programem, wykorzystywany jak pamięć operacyjna działająca na zasadzie tzw. stronicowania
A. virtual story
N. Wirtueller Speicher (m)
R. виртуальная память

pamięć zewnętrzna — każda pamięć, która nie jest pamięcią operacyjną
A. peripheral storage (device), external storage
N. peripherer Speicher (m), externer Speicher (m), äusserer Speicher (m)
R. периферийная (бнешняя) память

PC — 1. Programmable Controller — programowalny sterownik (układ sterowania) → PLCs; 2. Programmcounter — licznik programu; 3. Personal computer → podział mikrokomputerów

PCB — Printed Circuit Board — płytka (karta) obwodów drukowanych

PCU — Program Control Unit — jednostka sterująca

pętla rozkazów — ciąg rozkazów, który jest tak długo wykonywany aż spełniony zostanie warunek przerwania (skoku warunkowego)
A. loop
N. Schleife (f), Befchlsschleife (f)
R. петля

PGC — Profesional Graphic Controler — sterownik profesjonalny grafiki

PIA — Peripheral Interface Adapter — zunifikowany moduł mikrokomputera zapewniający łączność we/wy pomiędzy mikroprocesorem a urządzeniami peryferyjnymi

pisak cyfrowy → urządzenie rysujące

pixel — picture element — element obrazu (najmniejsza jego cząstka)

pióro świetlne — urządzenie do współpracy operatora z komputerem przez monitor ekranowy → grafoskop
A. Loght pen
N. Lichtgriffel (m), Lichtstift (m)
R. цветовое перо

PL/1 — Programming Language 1 — język programowania zorientowany problemowo, dostosowany do rozwiązywania zarówno problemów gospodarczych, jak i naukowo-technicznych (uniwersalny)

PL/M — Programming Language for Microcomputers — bazujące na PL/1 języki programowania mikrokomputerów

PLA — Programmable Logic Array — programowalne

zespoły logiczne w postaci układów scalonych oparte na pamięci ROM
PLCs — Programmable Logic Controllers — programowalne sterowniki logiczne → programowalne układy sterowania
ploter → urządzenie rysujące
płytka obwodu drukowanego → karta obwodu drukowanego
PMC — Programmable Machin Control — układ sterowania maszyn (obrabiarek) typu CNC (Computeriesed Numericale Control) z możliwością dostosowania do różnych typów obrabiarek i zadań technologicznych przez wczytanie odpowiedniego programu do pamięci stałej
N. programmierbare Maschinensteuerung (f)
R. програмного управления УПУ
podprogram — program, który może być włączony do różnych programów lub może być przywoływany kilkakrotnie w tym samym programie, **p.** może być otwarty („open") i stanowić tylko fragment programu lub zamknięty („closed") i stanowić odrębną całość programu
A. subroutine
N. Unterprogramm (n)
R. подпрограмма
podprogram przerwania — podprogram, który zostaje włączony i zrealizowany w wypadku wystąpienia → przerwania
A. interrupt-service routine, interrupt handler
N. Interrupt-Unterprogramm (n), Unterbrechungs behandlungs-programm (n)
R. прерывающая подпрограмма

podręcznik — 1. popularne określenie instrukcji, przejęte z języka angielskiego, np. instrukcja programowania — podręcznik programowania (programming manual), instrukcja użytkowania (user manual); 2. popularne określenie opisu (dokumentacji) kompleksowych programów
A. manual
N. Handbuch (n)
R. книга описания программы
podręcznik programowania — podręcznik zawierający instrukcję oprogramowania mikrokomputera (komputera)
A. programming manual
N. Programmierungsbuch (n)
R. книга программирования
podręcznik użytkowania — podręcznik opisujący sposób użytkowania mikrokomputera (komputera) niezbędny przy projektowaniu systemów cyfrowych opartych na mikroprocesorach
A. user manual
N. Anwendungsbuch (n)
R. книга пользователя
podwójna precyzja — oznacza, że do przedstawienia argumentu lub wyniku użyta zostanie podwójna długość słowa lub formatu danych
A. double precision, double precision arithmetic
N. doppelte Genauigkeit (f), doppelte Genaue Arithmetik
R. двойная прецизийность
podzespół — układ składający się z wielu → układów elementarnych, wchodzący w skład większego zespołu i/lub spełniający określone zadanie w danym urządzeniu
A. subassembly
N. Baugruppe (f), Untergruppe (f)
R. подузел, субблок, подсистема
Podział czasu — system pracy, w którym część urządzeń może być potrzebna jednocześnie wielu programom lub użytkownikom, z tych względów ich czas pracy zostaje podzielony wg przyjętego schematu (np. cyklicznie, w zależności od potrzeb, wg hierarchii potrzeb), a nazywa się to „podział czasu" lub „praca z podziałem czasu"
A. time sharing (system)
N. Zeitteilung (f), Tieme — sharing, Teilnehmersystem (n)
R. разделение времени
podział funkcjonalny mikroprocesorów — wskazujący na obszar ich zastosowań (aplikacji):
— **sterowniki-kontrolery** — o stałej liczbie instrukcji, 4- i 8-bitowe
A. microcontroller
N.
R.
— **mikroprocesory** (uniwersalne) o stałej liczbie instrukcji, 1- 4- 8- 16- i 32-bitowe
A. microprocessor
N.
R.
— **mikrokomputery** — jednoukładowe — o stałej liczbie instrukcji 4- 8- 16- i 32-bitowe, wszystko na jednej płytce oraz peryferii porównywalnej do średnich komputerów profesjonalnych
A. microcomputer, single — board computer
N.
R.
— **mikroprocesory segmentowe** (mikroprogramowalne — modułowe), 2- i 4-bitowe
A. processor slice (bit-slice), single board computer
N.
R.
podział mikrokomputerów — przyjęty **p.m.** wynika bardziej z zakresu przeznaczenia i oprogramowania niż architektury i uwarunkowań sprzętowych, w związku z tym jest dość dowolny i z grubsza daje się usystematyzować następująco:
1. **mikrokomputery domowe** — przeznaczone głównie do celów zabawowych (gry komputerowe) i edukacyjnych, a w tej grupie odbiornik telewizyjny spełnia na ogół zadanie monitora
A. home-computer
N.
R.
2. **mikrokomputery przenośne** (podręczne) — z możliwością używania nawet w trakcie podróży (zasilanie z baterii lub akumulatora), o przeznaczeniu profesjonalnym
A. portable (hand-held) — computer
N.
R.
3. **mikrokomputery biurowe** (biurkowe) — mikrokomputery o przeznaczeniu profesjonalnym → komputer biurowy
A. office (desk top) — computer
N.
R.
4. **mikrokomputer osobisty** — przeznaczony dla konkretnej osoby (inżyniera, lekarza, adwokata itp.) może obejmować zarówno grupę 2 jak 3, a nawet 1 — w zależności od wyposażenia i oprogramowania
A. personal computer, PC, PCS — PS System
N.
R.

podział pamięci półprzewodnikowych z uwzględnieniem ich podstawowych funkcji użytkowych:
1. **pamięć typu „wyłącznie odczyt" — ROM** — układ do przechowywania informacji cyfrowych zapisanych na stałe przez producenta układu, w trakcie eksploatacji można z niej tylko pobierać („czytać") informacje bez możliwości ich zmiany. Odmianami tej pamięci są:

PROM — programmable ROM — pamięć stała programowalna

REPROM — reprogrammable PROM — pamięć stała reprogramowalna

EPROM — UVEPROM — erasable programmable ROM — pamięć stała, kasowalna (promieniami ultrafioletowymi), która może być przeprogramowana

EAROM, EEPROM — Electrically Alterable ROM — Electrically Erasable PROM — pamięć stała reprogramowalna elektrycznie

A. ROM — Read Only Memory

N. ROM, Restwertspeicher (m), ausschliesslich lesbarer Speicher (m)

R. постоянная память, постоянное запоминающее устройство выборкой

2. **pamięć typu „zapis/odczyt"** — **RAM** — układ do przechowywania informacji cyfrowych z możliwością wpisu i odczytu, oznaczony również jako pamięć R/W od angielskich słów: read/write memory — RWM

A. RAM — Random Acces Memory, read/write memory

N. RAM, Schreib-/Lesespeicher (m)

R. память (запоминающее устройство) ц произвольной

3. **pamięć typu „stos"** — układ do przechowywania informacji cyfrowych z odwrotną kolejnością odczytu w stosunku do zapisu.

A. LIFO — last in first out memory

N. Kellerspeicher (m), Stapel (m)

R. стек пакет

4. **pamięć typu „silos"** — układ do przechowywania informacji cyfrowych, które można odczytać w kolejności zapisu

A. FIFO — first in first out memory

N. Silosspeicher (m)

R.

pole — 1. określony obszar zarezerwowany do przyjmowania określonego rodzaju danych, np. obszar pamięci lub miejsc bitów w rozkazie na argumenty czy adresy; 2. określony obszar kartoteki, formularza itd.

A. field

N. Feld (n)

R. поле

pole adresów — określone pole (np. pamięci, formatu rozkazu), które może zawierać tylko adresy

A. address field

N. Adressfeld (n)

R. поле адресов

pole komentarzy — znaki zawarte w takim polu stanowią komentarz dla asemblera lub kompilatora

A. comment field

N. Kommentarfeld (n), Bemerkungsfeld (n)

polling — 1. patrz praca z wywołaniem; 2. oprogramowanie systemu przerwań i zapytań

A. polling

N. Polling (m)

R. опрос

półsłowo — ciąg bitów lub znaków o połowie długości słowa, który może być adresowany jako całość

A. half word

N. Halbwort (n)

R. полуслово

port — równoległe rejestry buforowe ułatwiające współpracę dwukierunkową z magistralą danych

A. port

N.

R.

potwierdzenie zwrotne przerwania — sygnał zwrotny komputera (mikrokomputera), że dopuszcza się przerwanie

A. interrupt acknowledge

N. Interrupt — Quitung

R.

praca autonomiczna — praca w trybie autonomicznym

praca bezpośrednia — praca w trybie bezpośrednim

praca interakcyjna — praca konwersacyjna

praca konwersacyjna — rozwiązywanie zadania przez wielokrotną wymianę pytań i odpowiedzi między użytkownikiem i komputerem poprzez urządzenie dialogowe

A. interactive mode, conversational operation (processing)

N. Dialogbetrieb (m)

R. диалоговая работа

praca pojedynczymi krokami — system pracy mikrokomputera (komputera) lub innego sprzętu, w którym wykonanie każdego rozkazu wymaga nowego sygnału ze strony operatora obsługującego

A. single step operation

N. Einzelschrittbetrieb (m)

R.

praca równoległa — rodzaj przetwarzania lub transmisji danych w zespołach, w systemie tym bity słowa przenoszone są równocześnie w oddzielnych przewodach celem zwiększenia szybkości pracy urządzenia

A. parrellel operation

N. parallele Arbeitsweise (f), Parallelwirkung (f), Parrallelbetrieb (m)

R. параллельная работа, параллельное действие

praca szeregowa — rodzaj przetwarzania lub transmisji danych, w którym bity słowa przenoszone są w jednym przewodzie jeden za drugim. Jest to metoda wolniejsza od równoległej, ale wymaga mniej elementów elektronicznych

A. serial operation

N. serielle Arbeitsweise (f), Serien-Betrieb (m)

R. последовательная работа, последовательное действие

praca w trybie autonomicznym — forma łączności z komputerem przez nośnik danych, a nie sprzętowo

A. off-line

N. Off-line, Abgefrent

R. „оф-лайн, автономное

praca w trybie bezpośrednim — bezpośrednia forma łączności z komputerem (sprzętowo)

A. on-line

N. On-line, gekoppelt

R. „он-лайн", непосредственно

praca w ograniczonym czasie — podział czasu

praca z wywołaniem — system pracy, w którym stacje abonenckie są wywoływane do przekazywania danych

A. polling mode (procedure)

N. Abrufbetrieb (m), Abrufverfahren (n)

R. метод опроса

prober — próbnik stanów logicznych — analizator obwodów logicznych — sonda układów logicznych

procedura — 1. sekwencja instrukcji w językach zorientowanych maszynowo; 2. reguła transmisji danych, procedura transmisji

A. procedura, routine

N. Prozedur (f), Routine (f)

R. процедура

procedura rekursywna — procedura, która się sama wywołuje

A. recursive procedure

N. rekursive Prozedur (f)

R. рекурсивная процедура

procesor — 1. w zakresie sprzętowym (procesor centralny): układ elektroniczny, który funkcjonalnie odpowiada jednostce centralnej (CPU), tj. jednostce sterującej i arytmometrowi (ALU); 2. w zakresie oprogramowania: a) program, który spełnia funkcje translatora, kompilatora, asemblera dla określonego języka programowania, b) program pierwszego etapu kompilowania w niektórych językach maszynowego programowania OSN (mogą być trzy etapy: preprocesor, procesor i postproc.) lub drugiego etapu kompilowania w języku PL/1

A. processor (-unit)

N. Prozessor (m)

R. процессор

procesor centralny — jednostka centralna

PROM — Programable Read Only Memory — podział pamięci, programowalna przez użytkownika pamięć stała (bez możliwości zmiany)

program diagnostyczny — program testowania, określenie stosowane głównie w zakresie testowania sprzętu

A. diagnostic program

N. Diagnoseprogramm (n)

R. диагностическая программа

program docelowy — program wynikowy

program drugoplanowy — przetwarzanie w drugiej kolejności przetwarzania wieloprogramowego

A. background program

N. Hintergroundprogram (n)

R. программа с низким приоритетом, фоновая программа

program dyspozytor — specjalny program sterujący, stosowany w systemach wielodostępnych, np. w prze-

twarzaniu partiowym, który wywołuje programy użytkowników lub przydziela im czas
A. dispatcher
N. Verteiler (m), Abfertiger (m)
R. диспетчер
program dyrygent — program sterujący i kontrolujący wykonanie innych programów (część systemu operacyjnego)
A. executive program, executive routine
N. Executive — Routine (f), Exekutivprogram (n)
R. основная программа, программа-диспетчер
program generujący → generator
program interpretujący → interpreter
program komputera — program napisany w języku komputera, bądź przez programistę — sposób bardzo rzadko stosowany z uwagi na dużą pracochłonność, bądź też w wyniku wykonania translacji
A. machine program, computer program, machine routine
N. Maschinenprogram (n), Computerprogramm (n) Rechnerprogramm (n)
R. программа вычислительной машины, машинная программа
programy korezydentne — programy, które zanikają z chwilą wyłączenia zasilania
A. co-resident
N. Koresident
R. корезидентный
program ładujący — program umożliwiający ładowanie programu w mikrokomputerze (komputerze), tzn. wczytanie i wprowadzenie do właściwego obszaru pamięci
A. loader, program loader, loader routine
N. Lader (m), Programmloader (m), Ladeprogramm (n)
R. программа ввода, программа загрузки
program łączący — program, za pomocą którego z kilku programów użytkowych i ewent. łącznie z programami usługowymi lub systemowymi można stworzyć jedną funkcjonalną całość w postaci programu wynikowego
A. linker
N. Binder (m)
R.
program łącząco-segmentujący — program, który obok właściwości programów łączących ma możliwość tworzenia struktury segmentów nakładkowych (overlay-structure) do poszczególnych programów i ich segmentów
A. linkage editor
N. Programmverbinder (m), Linkage-Editor (m)
R. редактор (межпрограммных) связей
program maszynowy → program komputera
program monitor — 1. program sterujący i koordynujący inne programy lub ich części; 2. część programu organizacyjnego, od którego programy systemowe uzależnione są całkowicie, a programy użytkowe korzystają z niego fakultatywnie (np. przy wprowadzaniu)
A. monitor program
N. Monitor-Programm (n)
R. программа-монитор
program nadzorczy → program dyrygent
program organizacyjny — zespół programów systemu operacyjnego, które zapewniają właściwe współdziałanie poszczególnych zespołów komputera oraz zarządzanie wykonywanymi programami i zadaniami
A. executive program, master program, control program
N. Organisationsprogramm (n), Steuerprogramm (n), Dachprogramm (n), Leitprogramm (n), Hauptprogramm (n)
R. основная (ведушая) программа
program pomocniczy — program do bardzo specyficznych i stosunkowo prostych zadań, pod wzgl. funkcjonalnym porównywalny do programu usługowego
A. utility program
N. Utility-Program (n)
R. вспомогательная программа
program powtarzalny — 1. program lub zamknięta funkcjonalna część programu; 2. w publikacjach zagranicznych słowo „routine" używa się czasem w znaczeniu — podprogram, zamiast słowa „subroutine"
A. routine
N. Routine (f)
R. программа
program przetłumaczony → program wynikowy
program przemieszczalny — program o adresach przemieszczalnych, którego miejsce w pamięci ustalone jest w trakcie ładowania
A. relocatable program
N. relokatibles Program (n), verschiebbares Program (n)
R. настраиваемая (перемещаемая) программа
program redagujący → editor
program rezydentny — programy osadzone na stałe w mikrokomputerach, które nie zanikają również przy wyłączeniu zasilania (pamięć typu ROM)
A. resident
N. resident
R. резидентный
program sprawdzający — program, który bada programy lub dane pod względem oczywistych błędów, jak wady dziurkowania (błędy formalne i syntaktyczne)
A. checking program
N. Prüfprogram (n)
R. программа контроля
program sterujący → program organizacyjny
program systemowy — każdy program będący częścią składową systemu operacyjnego
A. system program
N. System-programm (n)
R. системная программа
program testowania — 1. program do sprawdzania prawidłowego funkcjonowania komputera (program diagnostyczny); 2. program do testowania programów (często z symulacją) przy ich opracowywaniu
A. test program
N. Testprogramm (n)
R. тест программа
program tłumaczący → translator i → kompilator
program usuwania błędów — 1. program do lokalizacji i korekty błędów syntaktycznych (formalnych) i semantycznych (logicznych) w programach mikrokomputerów (komputerów); 2. program do lokalizacji i napraw uszkodzeń (awarii) w mikrokomputerach (komputerach)
A. debung program, debugger
N. Debug-Program (n), Fehlersuchprogramm (n)
R. программа отладки
program usługowy — program, który rozszerza funkcje komputera w określonym obszarze, jednak w przeciwieństwie do programów systemowych nie jest konieczny dla systemu operacyjnego
A. service routine, service program, utility program
N. Dienstprogramm (n), Service-Programm (n), Hilfsprogramm (n)
R. служебная программа. вспомогательная программа
program użytkowy — program przetwarzania danych dla jednego lub wielu użytkowników
program wewnętrzny → program komputera
program wykonawczy → program dyrygent
program wynikowy → patrz translator, program utożsamiany często z programem komputera, jeżeli translatorem był asembler

A. object program
N. (maschinen) übersetztes Programm (n), Zielprogramm (n), Objektprogramm (n)
R. конечная программа, объективная программа

program współ- i wieloużywalny — program lub podprogram mający cechy współ- i wielоużywalności, tzn., że nie ulega on zniekształceniom („zaśmieceniom") i nie gubi danych, które mogą być potrzebne przy ponownym wywołaniu
A. re-entrant program, re-entrable program
N. wiedereintrittsinvariant Programm (n)
R.

program wyszukiwania błędów → program testowania
A. error search program
N. Fehlersuchprogramm (n)
R. программа поиска ошибок

program wprowadzania pierwotnego — program inicjujący działanie mikrokomputera (komputera) na stałe w nim zainstalowany, który umożliwia przyjmowanie innych programów (przede wszystkim programu ładującego)
A. bootstrap loader, bootstrap input program
N. Bootstrap-Lader (m), Urlader (m), Ureingabeprogramm (n)
R. программа первичного ввода (методика предввода)

program zarządzający → program dyrygent

program źródłowy — program napisany w dowolnym języku symbolicznym (→ język źródłowy)
A. source program
N. Quellenprogramm (n)
R. исходная программа, оригинальная программа

programowanie — zespół czynności prowadzących do ułożenia programu realizującego założony cel (zadanie) przez: a) układ logiczny lub system mikroprocesorowy (komputerowy); b) system lub urządzenie robocze (OSN, roboty, zautomatyzowane gniazda, ciągi i oddziały produkcyjne). W **p.** wyróżnia się na ogół następujące etapy: wypracowanie koncepcji, opracowanie sieci działań dla zadanego (wypracowanego) algorytmu lub formuły (wykresu przejść narzędzi w wypadku OSN), kodowanie (instrukcje, rozkazy), testowanie i usuwanie błędów, opracowanie dokumentacji dla użytkownika. Program mikroprocesora może być zapisany w języku problemowym, w kodzie mnemotechnicznym języka typu asembler lub kodzie binarnym, a następnie po przetestowaniu (translacja i usuwanie błędów) zapisywany jest w pamięci za pomocą urządzenia we/wy lub sprzętu pomocniczo-projektowego
A. programming
N. Programmierung (f), programmieren
R. програмирование

programowanie automatyczne → programowanie wspomagane komputerowo

programowanie maszynowe → programowanie wspomagane komputerowo

programowanie wspomagane komputerowo — na wielu etapach tworzenia programu korzysta się z komputera, dla którego program jest opracowywany lub komputera wspomagającego. Szczególnym przypadkiem **p.w.k.** jest programowanie obróbki części na OSN, program źródłowy jest tu przetwarzany na program sterowania automatycznego obrabiarki za pomocą kompilatora zwanego procesorem (pre-, post- procesor). Najbardziej znane języki programowania OSN, to: APO, APT, EXAPT, MS--RWPG, NARWIK
A. automatic programming, computer aided programming
N. maschinelle Programmierung (f), maschinemmterstütztes Programmieren (n)
R. машинное програмирование

programator — 1. urządzenie do programowania; 2. popularna nazwa sterownika, typu sekwencyjny układ sterowania, masowego zastosowania (np. w sprzęcie domowym). Współczesne rozwiązania **p.** oparte są na ogół na układach PLA, FPLA lub mikroprocesorach typu kontroler; **p.** dzielą się na: generatory sekwencyjne z pamięcią ROM (A. sequence generators) oraz programowane sterowniki (A. programmable logic controllers) z pamięcią PROM
A. 1. programmer, 2. controller,
N. 1. Programmer (m), 2. Kontroller (m)
R. 1. программиер, 2. контроллер

programowanie układowe — zmiana programu pracy (sterowania) przez zmianę części połączeń układu sterowania przeprowadzana przez użytkownika, przy czym może to być programowanie za pomocą: 1. a) wtyków diodowych (na tablicy programowania) i zderzaków nastawczych oraz b) wtyków kodujących, jako odmiana sterowań NC (low cost NS); 2. lub bezpośrednio na zespołach lub elementach układu sterowania (nie mylić z oprogramowaniem sprzętowym)
A. 1. pinboard programming, 2. programming with wired-in
N. Stecktafelprogrammierung (f), 1.a) Kreuzschienenverteilersteuerung (f), 1.b) Nockenlose-Program, Steuerung (f), 2. Programmierung mit Festverdrehtung
R. програмирование на коммутационным панелу

programowalny zegar impulsów — zegar impulsów, którego częstotliwość może być zmodyfikowana za pomocą programu
A. programmable timer (clock), PIT — Programmable Intervole Timer
N. programmierbarer Zeitgeber (m), programmierbarer Taktgeber (m)
R.

programowalne układy logiczne → programowalne układy sterowania

programowalne układy sterowania (cyfrowego) — układy logiczne służące do sterowania różnymi procesami i urządzeniami (zastępujące sterowanie z elementami dwupołożeniowymi, jak: styczniki, przekaźniki, wyłączniki) z programem zapisanym w pamięci. Współczesne **p.u.s.** są na ogół sterownikami typu PLA, FPLA lub mikroprocesorami typu kontroler i występują pod różnymi nazwami
A. PC Programmable Control (ler), PLCs — Programmable Logic Controllers, PCS — Programmable Control System
N. PS — Programmiesbare Steuerungen (pl), Prinzip speicher — programmierte Steuerungen (pl), FPS — Freiprogrammierbare Steuerungen (pl), SPS — Speicher Programmierbare Stenrungen (pl)
R. числового программного управлении

przełączenie programu — czynności, które mikrokomputer (komputer) wykonuje w czasie realizacji jakiegoś podprogramu wzgl. powrotu do programu głównego, jeżeli podprogram używa innego niż dla programu głównego zespołu rejestrów lub obszaru działania.
A. contex switch
N. Programmumschaltung (f)
R. переключание программы

przenoszenie danych — przesyłanie danych z jednego miejsca do innego (miejsca, systemu, miejscowości)
A. data communication
N. Datenaustausch (m), Datenübertragung (f)
R. обмен данными

przepływ danych — przepływ danych w trakcie ich przesyłania
A. data flow
N. Datenfluss (mm)

R. поток данных

przesyłanie danych — przenoszenie danych tylko w obrębie systemu mikrokomputerowego (komputerowego)
A. data transfer, data transmission, data traffic
N. Datentransfer (m), Datentransport (m), Datenverkehr (m)
R. передача данных

przerwanie — przerwanie aktualnie przez mikrokomputer (komputer przetwarzanego programu, przerwanie może być wywołane sygnałem zewnętrznym (przez operatora) lub wewnętrznym (przez komputer) przy stwierdzeniu np. błędu. W trakcie przerwania przetwarzany jest odpowiedni podprogram przerwania, po czym komputer może powrócić w sposób automatyczny do przetwarzania programu przerwanego. Przerwanie występuje często w systemie → przetwarzania w czasie rzeczywistym
A. interrupt
N. Interrput (n), Unterbrechung (f), Programmunterbrechung (f)
R. прерывание

przerwanie priorytetowe — priorytet przyznawany niektórym źródłom sygnału przerwania (patrz też przerwanie wielopoziomowe), może się odnosić również do sprzętu
A. priority interrupt, prioritized interrput
N. Prioritäts-Interrupt (m), Vorgangsunterbrechung (f)
R. прерывание по приоритету

przerwanie wielopoziomowe — 1. przerwanie wykonywanego podprogramu przerwania, przerwaniem wyższego priorytetu; 2. elementy lub układy, które umożliwiają 1. nazywane są elementami lub układami **p.w.**
A. multi level interrupt
N. Mehrebenen — Interrupt (n)
R. многоуравнительное прерывание

przerzutnik — pamięciowy element bistabilny (np. w rejestrach) lub układ przełączający
A. flip-flop
N. Flip-Flop
R. триггер

przesunięcie — przesunięcie adresowe w adresie względnym dwuczęściowym (adres bazowy + przesunięcie adresowe)
A. displacement
N. Verschiebung (f), Distanz (m)
R. сдвиг, смещение

przesunięcie arytmetyczne — operacja przesunięcia (ciągu znaków, bitów). W rejestrach lub pamięci z zachowaniem właściwego dla tego ciągu znaku liczby
A. arithmetic shift
N. arithmetisches Schieben (Verschieben) (n), Schieb (n)
R. арифметическое смещение, сдвиг

przesunięcie logiczne — przesunięcie całego ciągu znaków lub bitów, łącznie z pozycją znaku liczby
A. logic shift
N. logisches Schieben (n)
R. логический сдвиг

przesuw wiersza — przesuw papieru o wiersz w dziurkarkach, tabulatorach, maszynach do pisania itp.
A. Line feed
N. Zeilenverschieb (m)
R.

przetwarzanie danych — wykonywanie usystematyzowanego ciągu operacji na danych w postaci cyfrowej lub analogowej
A. data processing
N. Datenverarbeitung (f)
R. обработка данных

przetwarzanie danych lokalne — zdecentralizowane przetwarzanie danych w miejscu ich powstawania lub istnienia potrzeby natychmiastowych wyników przetwarzania
A. distributed processing
N. dezentralisierte Datenverarbeitung (f)
R. дистрибутивная обработка

przetwarzanie drugoplanowe — w systemie pracy komputera z możliwością przerywania typu interrupt, np. w trakcie przetwarzania wieloprogramowego lub w czasie rzeczywistym, przetwarza się „równolegle" pierwszoplanowy z tzw. najwyższym priorytetem i „jednocześnie" (rzeczywiście w przerwach) program (lub programy) drugoplanowe
A. background processing
N. Hintergrundverarbeitung (f)
R. обработка программы с ниским приоритетом

przetwarzanie nadążne — przetwarzanie w czasie rzeczywistym

przetwarzanie na bieżąco — przetwarzanie w czasie rzeczywistym

przetwarzanie partiowe — przetwarzanie wsadowe — system pracy komputera, w którym zadanie, jakie ma on realizować, musi być całkowicie sprecyzowane i wszelkie dane zgromadzone przed rozpoczęciem jego realizacji, a realizacja przebiega aż do całkowitego zakończenia (lub przerwania) zadania, przy czym dane wprowadzane są partiami w postaci tzw. wsadów
A. batch processing, batch mode
N. Batch — Betrieb (m), Batch — Verarbeitung (f), Stapelverarbeitung (f), Schubverarbeitung (f)
R. пакетная обработка данных

przetwarzanie w czasie rzeczywistym — nadążny system przetwarzania, w którym napływające do mikrokomputera — (komputera) dane muszą być przekazane do przetwarzania i przetworzone, a wyniki natychmiast zadysponowane (system stosowany często w układach sterowania), system taki musi dysponować odpowiednimi (do zadań) zdolnościami przetwarzania, a przy współpracy z kilkoma stacjami abonenckimi (np. OSN) odpowiedni program decyduje o hierarchii kolejności
A. real time processing
N. Echtzeit verarbeitung (f), Realzeitbetrieb (m), schritthaltende Datenverarbeitung (f))
R. обработка данных в истином масштабе времени

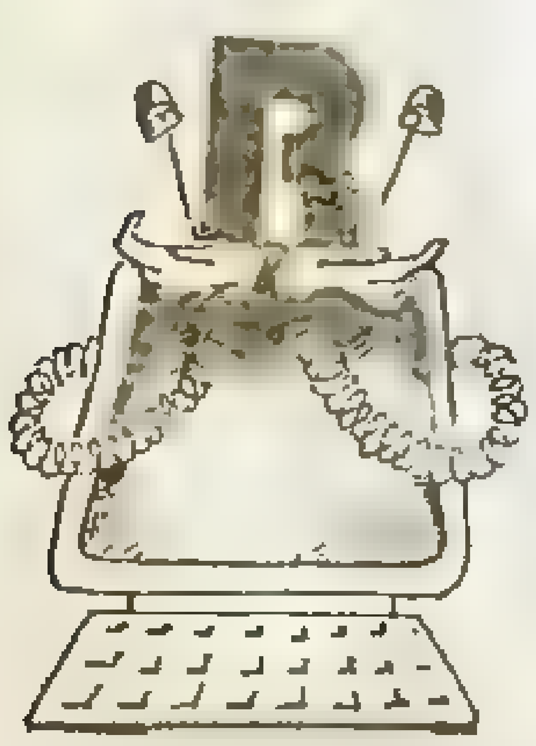
przetwarzanie wieloprogramowe — jednoczesne przetwarzanie wielu programów w jednym komputerze. Właściwość tę mają komputery o szybkich jednostkach centralnych o wolnych urządzeniach peryferyjnych oraz komputery wieloprocesorowe. Metoda ta zapewnia lepsze wykorzystanie sprzętu. Całością kieruje program organizacyjny
A. multiprogram processing, multiprogramming
N. Multiprogrammverarbeitung (f), Multiprogramming (n))
R. мультипрограммная обработка

przeznaczenie — oznacza adres, pod którym można znaleźć przesyłaną informację po wykonaniu rozkazu przesyłania
A. destination
N. Ziel (n), Empfänger (m))
R. цель, назначение

PS/2 — Personal System/2 — system komputerów osobistych IBM

pseudorozkaz — 1. rozkaz, który należy do repertuaru rozkazów mikrokomputera (komputera), ale nie wywołuje w nim żadnych działań, np. instrukcja pusta; 2. rozkaz, który należy do repertuaru danego mikrokomputera (komputera) lecz zostaje przez asembler zamieniony na inny rozkaz lub serię rozkazów; 3. rozkaz (instrukcja), który nie należy do repertuaru rozkazów mikrokompu-

tera (komputera) i stosowany jest tylko w języku symbolicznym, a nie ma odpowiednika w języku komputera np. komentarze asemblera, makrodefinicje
A. pseudo instruction, pseudo command
N. Pseudobefehl (m), symbolischer Befehl (m))
R. символическая команда
pulpit operatora — część jednostki centralnej przeznaczona do ręcznego sterowania i kontroli pracy komputera
A. operator panel, operator control panel
N. Bedienungsfeld (n), Bedienungspult (n))
R. пультъ оператора
punkt przerwania — 1. celowo zaprogramowany punkt, w którym można przerwać przetwarzanie programu (np. dla sprawdzenia częściowych wyników), jest on jednocześnie punktem wznowienia; 2. punkt przypadkowego przerwania obróbki programu → punkt wznowienia
A. breakpoint
N. Anhaltepunkt (m), Zwischenhalt (m))
R. точка остановки, местоостановки
punkt wejścia — miejsce w programie, do którego można wykonać skok z zewnątrz
N. entry point)
N. Einsprungpunkt (m), Eingangsstelle (f))
R. входная точка, точка входа
punkt wznowienia — punkt określony już w trakcie programowania, od którego należy wznowić przetwarzanie programu w wypadku przerwania. Jeżeli **p.w.** nie określono, program należy przetwarzać od nowa
A. restart point, retur point
N. Fixpunkt (m), Wiederauflaufpunkt (m))
R. точка повторного запуска



RAM — Random Access Memory — pamięć typu zapis/odczyt o dostępie bezpośrednim (swobodnym) → podział pamięci
redagowanie — 1. przygotowanie danych (wyników) wewnątrz urządzenia przed ich wydaniem; 2. modyfikacja formatu lub danych w przygotowywanym programie
A. edit editing
N. editieren, Druckkaufbereitung (f))
R. редактирование
redundancja — 1. rozwlekłość informacji lub taka możliwość, np. nadmiar znaków w kodzie (nie wykorzystywane możliwości), nadmiar znaków w przysyłanej informacji, które do niej nic nie wnoszą, 2. odnośnie sprzętu — powielanie implementacji funkcjonalnie równoważnych komponentów, np. dla zwiększenia niezawodności całego systemu
A. redundancy
N. Redundanz (f))
R. избыточность информации
rejestr — funkcjonalnie lub konstrukcyjnie wyróżniany element pamięciowy o bezpośrednim i szybkim dostępie zdolny do przechowywania pojedynczych słów lub bitów, wchodzi w skład różnych zespołów komputerowych i mikrokomputerowych, jak: pamięci, akumulatory itd.
A. register
N. Register (n))
R. регистр
rejestr adresu bazowego, rejestr bazowy, B-rejestr — rejestr, w którym przechowywany jest ciągle lub okresowo adres bazowy
A. base address register, base register, B, register
N. Basisadressregister (n), Basisregister (n), B-register (n))
R. основной (базовый) регистр, регистр базы
rejestr A → akumulator
rejestr B → rejestr indeksowy (B box), → rejestr adresu bazowego
rejestr F — flagowy → rejestr statusowy
rejestr → licznik programu
rejestr indeksowy — rejestr służący do zmiany (modyfikacji) adresu (względnego) rozkazu, do którego dodaje się lub odejmuje zawartość **r.i.**
A. index register
N. Indexregister (n)
R. регистр индекса
rejestr modyfikacji → rejestr indeksowy
A. modifier register
N.
R.
rejestr rozkazów — rejestr przechowujący aktualnie realizowany rozkaz aż do jego pełnego wykonania
A. instruction register
N. Befehlsregister (n))
R. регистр команд
rejestr stanu → rejestr statusowy
rejestr statusowy (znaczników) — składa się z poszczególnych sygnalizatorów i wskaźników (przerzutników zwanych również „flags", a zawiera dane o stanie mikrokomputera (komputera) i wpływa na sposób realizacji programu
A. status register
N. Statusregister (n)
R. регистр состояния
rejestr translacyjny → port
rejestr wskaźników →rejestr statusowy
rejestracja danych → zbieranie danych
repertuar rozkazów →zbiór dopuszczalnych rozkazów określonego maszynowo zorientowanego języka programowania, tzn. takich rozkazów, które dany mikrokomputer (komputer) realizuje bez opisywania ich w programie za pomocą innych
A. instruction set, instruction repertoire, instruction repertory
N. Befehlsvorrat (m), Instruktionssatz (m))
R. система комахд
REPROM — Reprogrammable PROM — pamięć stała reprogramowalna → patrz podział pamięci
reprogramowalna pamięć →REPROM, np. EPROM, EAROM
rezydentne → programy rezydentne
RISC — Reduced Instruction Sed Computer — procesor o zredukowanej liczbie instrukcji
ROM — Read Only Memory — pamięć typu „wyłącznie odczyt", pamięć stała o bezpośrednim dostępie → podział pamięci
robot — 1. system techniczny zdolny samodzielnie wykonywać czynności zaprogramowane przez człowieka (definicja ogólna robota); 2. programowalny wielofunkcyjny manipulator przeznaczony do operowania materiałami, częściami lub wyspecjalizowanymi urządzeniami, przez różne programowane ruchy, w celu zrealizowania różnych zadań (definicja Amerykańskiego Instytutu Robotyki RIA)
A. robot
N. Roboter (m))
R. робот
robot przemysłowy — robot przeznaczony do zadań przemysłowych, jest to urządzenie manipulacyjne sterowane programowo, wyposażone w uchwyty lub narzędzia, przeznaczone do zastosowań przemysłowych
A. industrial robot

N. Industrieroboter (m))
R. промышленный робот

rozkaz — 1. instrukcja dla komputera do wykonania określonej operacji, specyfikuje ona zarówno operację, jaka ma być wykonana jak również argumenty lub ich adresy; 2. instrukcja, która w danym języku programowania, w którym jest stosowana, nie da się rozdzielić na takie części, które mogą być samodzielnymi instrukcjami
A. instruction, order, command
N. Befehl (m), Kommando (n))
R. команда

rozkaz jednoadresowy — 1. rozkaz komputera jednoadresowego; 2. rozkaz komputera wieloadresowego, który zawiera tylko jeden adres
A. single address instruction, one-address instruction
N. Ein-Adress-Befehl (m))
R. одноадресная команда

rozkaz jednowyrazowy — rozkaz, którego długość jest identyczna z długością słowa komputera
A. one word instruction, one word command
N. Einwortbefehl (m))
R. команда состоящая одного слова

rozkaz komputera — rozkaz maszyny — rozkaz w języku wewnętrznym mikrokomputera (komputera), stanowiący element repertuaru rozkazów, na ogół jest to reprezentacja ciągu 0/1 [zerojedynkowego] lub liczby heksadecymalnej
A. machine instruction, machine command
N. Maschinenbefehl (m))
R. машинная команда (инструкция)

rozkaz makro — 1. instrukcja w języku źródłowym, która odpowiada określonej sekwencji rozkazów komputera (odpowiednik instrukcji wywołania makro w języku typu asembler); 2. określenie rozkazu komputera dla podkreślenia, że będzie on przez mikrokomputer interpretowany i wykonywany jako sekwencja mikrorozkazów
A. macrocommand, macroorder, macroinstruction
N. Makrobefehl (m), Makroinstruktion (f))
R. макрокоманда, макроинструкция

rozkaz maszyny → rozkaz komputera

rozkaz „nic nie rób" → rozkaz pusty

rozkaz pozorny → rozkaz pusty lub → pseudorozkaz
A. dummy instruction
N. Scheinbefehl (m))
R. фиктивная (пустая) команда

rozkaz pusty — rozkaz nie powodujący w mikrokomputerze (komputerze) żadnego działania poza przejściem do następnego rozkazu
A. no-operation instruction, no-op instruction, NOP
N. Leerbefehl (m), Nulloperation (f))
R. команда, пропуска

rozkaz skoku — rozkaz powodujący wykonanie skoku
A. jumb instruction
N. Sprungbefehl
R. команда перехода

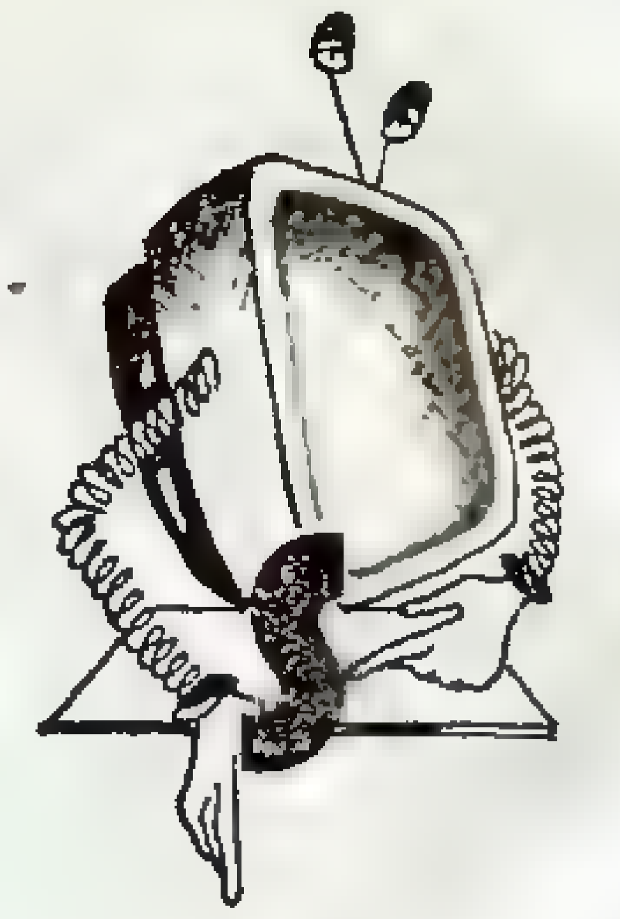
rozkaz wieloadresowy — rozkaz , który zawiera więcej niż jeden adres argumentu
A. multi address instruction
N. Mehradressbefehl (m))
R. многоадресная команда

rozkaz wielosłowny — rozkaz o długości stanowiącej wielokrotność długości słowa czy formatu danych — multisłowo, w pamięci **r.w.** zajmuje miejsce kilku słów
A. multiple word instruction
N. Mehrwortbefehl (m))
R. команда многократной длины

rozszerzony repertuar rozkazów — możliwość wprowadzenia przez użytkownika zdefiniowanych we własnym zakresie rozkazów (instrukcji) i w ten sposób powstaje **r.r.r.**
A. extended operation
N. erweiterte Operation (f))
R. расширенная операция

RPROM → REPROM

RPN — Reverse Polish Notation — odwrócony zapis polski, notacja postfiksowa (zapis wg logiki prof. Jana Łukasiewicza)
A.
N. inverfierte polnische Notation, Postfixnotation



SAA — System Aplikation Architektur — architektura zastosowań systemowych — nowa koncepcja IBM umożliwiająca przenoszenie rozwiązań (i współpracę w sieciach) pomiędzy trzema rodzinami wyrobów firmy IBM, serii: PC, 3X i 370

schemat czynnościowy — 1. sieć działań systemu mikroprocesorowego w postaci graficznej, którą opracowuje się w celu napisania programu działania systemu mikroprocesora w trakcie jego projektowania; 2. schemat stanowiący zobrazowanie algorytmu, definicji lub rozwiązań jakiegoś problemu, w którym używa się symboli do przedstawienia: rozkazów, danych, przepływu danych, przebiegu programu, sprzętu itd.
A. flow chart
N. Flussdiagramm (n), Anlaufdiagramm (n), Ablaufschema (n), Flussplan (m))
R. графическая схема, рабочая диаграмма, блоксхема программы

Schneider → Amstrad

SCSI — Small Computer System Interface — złącze małych systemów komputerowych (standard USA)

sekwenter — moduł mikroprocesorowy, stosowany w mikroprocesorach typu kontroler, który steruje realizacją mikroprogramu, przez generowanie w takt zegara adresów mikroinstrukcji
A. sequencer
N. Sequenzer (m), Ablaufsteuerung (f))
R.

semantyka — nauka o znaczeniu grup znaków języka (np. programowania)
A. semantics
N. Semantik (f))
R. семантика

S.I. — sztuczna inteligencja
A. artificial intelligence

sieć wieloprzetwarzania → system wieloprocesorowy

sieć działań → schemat czynnościowy

Sinclair — mało znana brytyjska firma zajmująca się produkcją minikalkulatorów i miniaturowych odbiorników radiowych, wypuściła na rynek w 1980 r. pierwszy mikrokomputer. Miał on pamięć tylko 1 k RAM, niewielką ROM, procesor Z-80. Ten prymitywny mikrokomputer rozpoczął rewolucję komputerową w Wielkiej Brytanii. ZX-80 — bo tak go nazwano — skonstruowany był głównie w oparciu o układy scalone małej skali integracji (oprócz procesora i pamięci) i wkrótce okazał się przestarzałym. W 1981 r. pojawił się nowy mikrokomputer ZX-81, który do końca następnego roku był najpopularniejszym mikrokomputerem w Europie. Pod koniec 1982 r. firma Sinclair wprowadziła ZX-Spectrum, który miał: 16 k RAM, 16 k ROM, Z-80A. Szybko pojawiła się wersja 48 k,

a w połowie 1984 r. Sinclair-QL, zbudowany w oparciu o 16-bitowy procesor firmy Motorola 68008, pamięci 32 k ROM i 128 k RAM oraz koprocesor INTEL. Nie było to jednak udane rozwiązanie i pod koniec 1984 r. pojawił się Spectrum Plus, czyli dawny ZX w nowej obudowie i od początku 1985 r. także o nowym rozwiązaniu konstrukcyjnym.

skala integracji, stopień scalenia — skala (wielkość) scalenia elementów podstawowych w układach scalonych (przy obecnym rozwoju technologii **s.i.** rośnie bardzo szybko, np. w 1978 r. do 20 tys. elementów podstawowych w jednym układzie scalonym lub 3 tys. elementów podstawowych na 1 mm^2, a w latach 1980—1990 przewiduje się 10^6 el. podst. i 250 tys. el. log. w jednym układzie. Dla porównania **s.i.** zamiast elementów podstawowych podaje się również liczbę tranzystorów (również elementów logicznych) przyjmując cztery stopnie **s.i.**
— mała (standardowa) **s.i.** (SSI — Small Scale Integration) do 20;
— średnia **s.i.** (MSI — Medium Scale Integration) od 20 do 100;
— wielka **s.i.** (LSI — Large Scale Integration) od 100 do 1000;
— bardzo wielka **s.i.** (VlSI — Very LSI) od 1 tys. do 100 tys.
— super wielka **s.i.** (SLSI — SUPER LSI) powyżej 100 tys. tranzystorów w jednym układzie
A. integrated scala
N. Integrationsskale (f))
R. шкала интегрирования

skok — kontynuowanie programu nie w kolejności rozkazów zapisanych w pamięci programu
A. jump, branch
N. Sprung (m))
R. переход

skok powrotny w podprogramie — rozkaz podprogramu powrotu do rozkazu wywołania podprogramu w programie głównym (wywołującym)
A. return instruction
N. Unterprogrammrücksprung (m))
R. команда возврата в подпрограмме

skok warunkowy — skok, który wykonywany jest tylko wówczas, jeżeli spełnione są odpowiednie warunki
A. conditional jump, conditional banch
N. bedingter Srung (m), bedingte Verzweigung (f))
R. условный переход, условное разветвление

skok wyliczany — zastosowany adres skoku wylicza się dopiero w trakcie przetwarzania programu na podstawie określonych warunków
A. computed go to
N. berechneter Sprung (m))
R.

składnia — reguły tworzenia określonego języka (np. oprogramowania)
A. syntax
N. Syntax (f))
R.

słowo — ciąg bitów lub znaków, które w określonym języku, również zależnym sprzętowo (rodzaj mikrokomputera czy układu sterowania) traktowane są jako całość
A. word
N. Wort (n))
R. слово

słowo kluczowe — słowo wybrane do charakteryzowania treści dokumentu
A. Keyword
N. Schlüsselwort (n))
R.

służba serwisowa — techniczna służba obsługi użytkowników (klientów)
A. field service
N. technischer Kundendienst (m))
R. обслуживание абонента

SNA — System Network Architektur — architektura systemów sieciowych (umożliwia pracę systemów PS/2 w sieci z dużymi komputerami)

software — oprogramowanie

sonda układów logicznych — sprzęt serwisowy i pomocniczo-projektowy do kontroli stanów układów logicznych → analizator stanów obwodów logicznych
A. logic (digital) probe
N. Logiksonde (f), Logik-Prüfstift (m))
R.

sprzęg → złącze

sprzęt komputerowy — fizyczne składniki (hardware) systemu przetwarzania danych bez oprogramowania (software)
A. hardware
N. Hartware (f), materielle Ware (f))
R. аппаратная частъ, хартвер

sprzęt pomocniczo-projektowy — oparte na mikro- lub minikomputerach zestawy sprzętu, jak: analizator, wideoskop, urządzenie programujące (programer), emulator i inne, służące do oprogramowywania, testowania i projektowania systemów cyfrowych opartych na mikroprocesorach; **s.p.p.** stanowi integralną część systemów uruchomieniowo-rozwojowych
A. development equipment, development tools
N. Entwicklungsgeräte (pl), Entwicklungswerkzeuge (pl))
R.

stały program pracy — układ sterowania skonstruowany dla konkretnego programu roboczego (zastosowanie w robotach i prostych urządzeniach); do jego ewentualnej zmiany należy zaprojektować i wykonać nowy układ sterowania
A. constant (non programmable) working program
N. fester (nicht-programmierbarer) Arbeitsprogramm (m))
R.

stan — aktualny stan poszczególnych układów mikrokomputera (komputera) → bit stanu → rejestr statusowy
A. status
N. Status (m), Zustand (m))
R. состояние, режим

status → stan lub sygnał stanu (→ bit stanu)

sterowanie automatyczne — ogół sterowań, w których algorytm sterowania jest realizowany bez udziału człowieka
A. automatic control
N. automatische Steuerung (f))
R. автоматическое управление

sterowanie cyfrowe → sterowanie numeryczne

sterowanie mikroprocesorowe — układy sterowania oparte na mikroprocesorach (mikrokomputerach)
A. microprocessor control, microprocessor based controller, microcontroller, microcomputer numerical control
N. Mikroprocessorsteuerung (f), Mikroprocessorsteueranlage (f), Mikrokomputersteuerung (f))
R. микропроцессорный контролер

sterowanie numeryczne — system programowalnego sterowania automatycznego działający w oparciu o → układy cyfrowe (numeryczne) wg programu zakodowanego binarnie bezpośrednio w układzie sterowania lub na nośniku wprowadzonym sekwencyjnie do pamięci układu
A. numerical control
N. numerische Steuerung (f), digitale Steuerung (f))
R. цифровое управление, дискретное управление числовое програмное управление (ЧПУ)

sterowanie programowe — takie, w którym impulsy sterowania zmieniają się wg z góry określonego → stałego lub → zmiennego programu pracy: np. sterowanie cyklem pracy robota w tym pojęciu odbywa się zawsze wg programu, a sterowanie ręczne służy tylko funkcjom pomocniczym (ustawienie, regulacja, nauka programu)
A. program controller (control unit)
N. Programmsteuerung (f))
R. програмное управление

sterowanie sekwencyjne — takie, które każdą następną czynność (określonego procesu) wywołuje wg z góry określonej kolejności, po zakończeniu poprzedniej
A. sequencing control
N. Folgesteuerung (f), Ablaufsteuerung (f))
R. последовательное управление

sterownik → układ sterowania automatycznego → układy cyfrowe → programatory → programowalne układy sterowania

stos — zespół rejestrów lub miejsc pamięci, których zawartość jest wywoływana w kolejności odwrotnej do ich zapisywania
A. stack, push-down storage, nesting store
N. Kellerspeicher (m), Stapel (m))
R. стек, накет

stopień scalenia → skala integracji

struktura rozkazu → budowa rozkazu

superkomputer — komputer dużej mocy obliczeniowej, specjalnego przeznaczenia (wojskowość, meteorologia itp.), o dużych pamięciach operacyjnych (wewnętrznych) rzędu 64—256 MB i szybkości obliczeń rzędu 100 mln — 1 mld operacji/s

sygnalizacja (zgłoszenie) błędu — 1. informacja, że wykryto błąd; 2. reakcja komputera na błąd w programie lub w danych, na którą składa się wykazanie rodzaju lub przyczyny błędu przez drukarkę, terminal lub inny rodzaj sygnalizacji
A. error message, error signal
N. Fehlermeldung (f), Fehlersignal (m))
R. сигнал ошибка

sygnalizator →flaga

symulacja — modelowe naśladowanie na mikrokomputerze lub częściej na komputerze dowolnego systemu innego mikrokomputera za pomocą odpowiedniego oprogramowania
A. simulation
N. Simulation (f))
R. моделирование

symulator — 1. program, który umożliwia symulację; 2. interpreter, który interpretuje programy napisane w języku komputera (maszynowo zorientowanym)
A. simulator
N. Simulator)
R. моделирущая программа

syntaktyka → składnia

system — 1. sprzęt komputerowy: a) kompleksowy zestaw (komputer, urządzenia peryferyjne, oprogramowanie systemowe i użytkowe), b) pewna liczba modeli określonego typu jednego wytwórcy; 2. określony zakres oprogramowania całego komputera lub jego składników (np. system wejścia/wyjścia); 3. komplet programów użytkowych (lub jeden całościowy) określonego problemu (zadania)
A. system
N. System (n))
R. система

system dziesiętny — taki, którego podstawą jest liczba 10
A. decimal system'
N. Dezimal-system (n))
R. десятичная система

system generacji oprogramowania → systemy uruchomieniowo-projektowe

system liczbowy — sposób zapisywania i nazywania liczb
A. number system
N. Zahlensystem (n))
R. система счисления

system mikrokomputerowy — system, który pod względem funkcjonalnym odpowiada komputerowi, składa się nań kilka układów scalonych, które zawierają takie grupy funkcjonalne, jak: jednostka centralna, pamięć programu i danych we/wy oraz pełnego oprogramowania systemowego i użytkowego
A. microcomputer system
N. Mikrocomputersystem (n))
R. система микро ЭВМ

system modularny — 1. system oprogramowania o budowie modułowej; 2. system mikrokomputerowy o budowie modułowej;3. → budowa modułowa
A. modular system
N. Modularsystem (n), Baukastensystem (n), Bausteinsystem (n)
R. модулярная система

system operacyjny — zespół programów, które zapewniają eksploatację komputera do przetwarzania danych, a w szczególności sterują i nadzorują wprowadzanie i wykonywanie innych programów, a poza tym umożliwiają: administrowanie programami i danymi, tłumaczenie programów, wykrywanie błędów w programach i usterek (awarii) w sprzęcie itd.
A. operating system, OS
N. Betriebssystem (n), Operationssystem (n)
R. операционная система

system operacyjny dyskowy — system operacyjny komputera, w którym m.in. programy systemu, programy użytkowe i zbiory danych zawarte są w pamięci dyskowej lub bębnowej
A. disk operating system, DOS
N. Platten-Betriebssystem (n)
R.

system prototypowy → systemy uruchomieniowo-rozwojowe

system przenoszenia i przetwarzania zdalnego — 1. system przetwarzania danych przy ich transmisji (teledacji) przez współpracę ze stacjami abonenckimi (A. remote data processing, N. Datenfernverarbeitung (f); 2. system transmisji danych (teledacji) A. data transmision, N. Datenübertragung)
A. data communication system
N. Datenfernverarbeitungssystem (n), Datenübermittlungssystem (n)
R. 1. система дистанционной обработки данных
2. система передачи данных

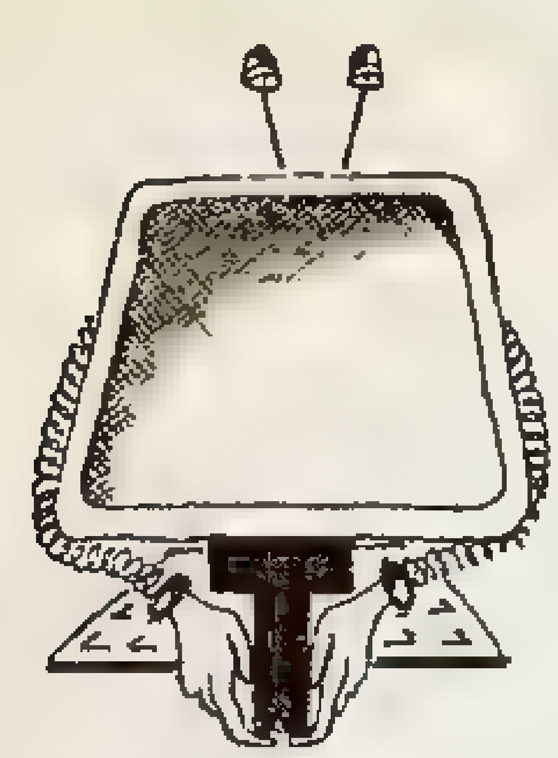
system przetwarzania danych — kompletny zestaw do przetwarzania danych składający się z komputera, urządzeń peryferyjnych, oprogramowania systemowego (operacyjnego) i użytkowego
A. data processing system
N. Datenverarbeitungssystem (n)
R. система для обработки данных

system szesnastkowy → zapis szesnastkowy

system użytkowy — mikrokomputerowy, który jest projektowany do stosowania (dla użytkownika), termin stosowany dla rozgraniczenia od systemów projektowych (uruchomieniowo-rozwojowych)
A. target system
N. Zielsystem (n), Anwendersystem (n)
R. система результата (цели)

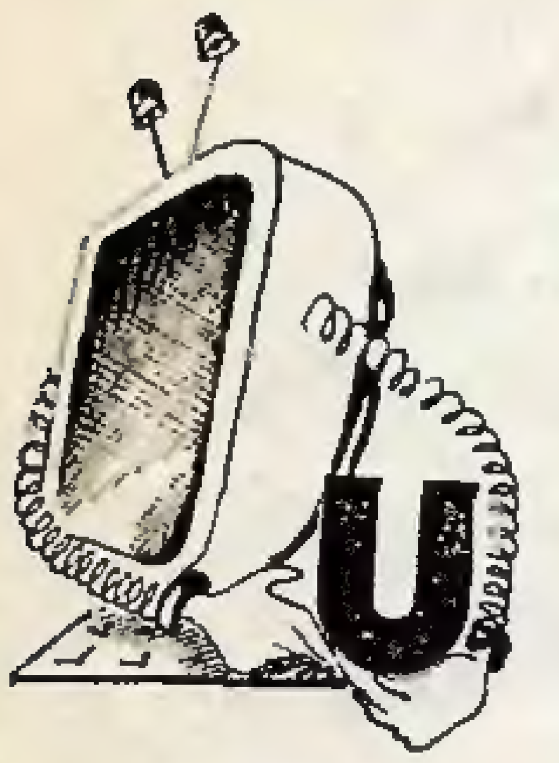
system wieloprocesorowy — system mikrokomputerowy (komputerowy), którego jednostka centralna składa się z kilku mikroprocesorów (procesorów), które współpracują ze sobą na zasadzie równorzędności lub jeden z nich nadzoruje pozostałe. Generalnie, jednostka

sterująca może być tylko jedna, a arytmometrów kilka, urządzenia we/wy i pamięci mogą być wspólne lub podzielone. System taki umożliwia przetwarzanie wieloprogramowe.
A. multiprocesor (multiprocessing) system
N. Mehrprozessorsystem (n), Mehrrechner-verarbeitung (f)
R. мультипроцессорная система
system wspomagania → systemy uruchomieniowo-rozwojowe
system zarządzania bazą danych — procedura manipulowania danymi, (zbiór programów związanych z: tworzeniem zasobów, zarządzaniem, wyszukiwaniem, zabezpieczaniem) nieodłączny element banku (zbioru) danych
A. data base file management
N. Datenbankdatei-Verwaltung (f)
R. система управления банком данных
system zunifikowany → budowa modułowa
A. modular system
N. Bausteinsystem (n), Baukastensystem (n)
R. модулярная система
systemy uruchomieniowo-rozwojowe (projektowe) — systemy oparte na → sprzęcie pomocniczo-projektowym, stanowiące podstawę do opracowywania i testowania oprogramowania (software) mikrokomputerów. W skład programów systemowych **s.u.r.** wchodzą m.in. programy: sterujący — monitor, wydawniczy — editor, uruchamianie — debugger (testowania) lub symulacyjny, translacyjny — asembler, na ogół możliwe też jest testowanie (w czasie rzeczywistym) systemu lub sprzętu (hardware) przez dodatkowy sprzęt lub programy do emulacji
A. development system, software development system, MDS — Microprocessor Development System
N. Entwicklungssystem (n), Software-Entwicklungssystem (n), Mikrocomputerentwicklungssystem (n)
R.
systemy wspomagania → sprzęt pomocniczo-projektowy i systemy uruchomieniowo-rozwojowe
szyna → magistrala
tablica połączeń → tablica programowania



tablica programowania — tablica połączeń służąca do programowania przez łączenie poszczególnych gniazd (obwodów) za pomocą wtykowych przewodów połączeń, diod lub zwieraczy wg określonego szablonu (formularza), stosowana głównie do sterowania programowego urządzeń i układów technologicznych (np. robotów, kodopisów) spotykana również w drobnym sprzęcie komputerowym (np. kalkulatorach elektronicznych)
A. plug board, plug-in board, patchpanel
N. Stecktafel (f), Schalttafel (f), Steckfeld (n)
R. наборное поле, коммутационная доска
tablica symboli — tablica wszystkich symboli zawartych w programie źródłowym i przynależne do nich adresy względne lub bezwględne, **t.s.** wykonuje asembler lub kompilator
A. symbol table
N. Symboltabelle (f)
R. таблица символов
tablica wtykowa → tablica programowania
taśma mangetyczna — rodzaj nośnika w pamięciach zewnętrznych o dostępie sekwencyjnym, używana w postaci szpul i kaset **t.m.**
A. magnetic tape
N. Magnetband (n) MB
R. магнитная лента
teledacja → transmisja danych (patrz też system przenoszenia)
teleinformatyka → system przenoszenia i przetwarzania zdalnego
teleprzetwarzanie → system przenoszenia i przetwarzania zdalnego
teletransmisja danych, teledacja → transmisja (patrz też system przenoszenia)
terminal → urządzenie końcowe
TTY — Teletype (unit) — dalekopis
test porównawczy — test porównawczy wydajności mikrokomputerów (komputerów) za pomocą jednego lub wielu programów testujących, jak np. „mieszanka" GAMM, XIX itp., niestety wszystkie te metody mają wiele wad i tzw. bezwzględne porównanie jest na obecny stan wiedzy i środków problematyczne
A. Benchmark comparison
N. Benchmarkvergleich (m)
R.
tester układów — sprzęt do testowania układów scalonych lub całych produktów na płytach obwodów drukowanych, często samoprogramujący się wg płyt wzorcowych
A. i-c tester, PCB-tester
N. i-s Tester (m), Leiterplattentester (m)
R. прибор для проберки и-с
tłumaczenie programu → translator
TOS — Tramiel Operating System — system operacyjny mikrokomputera „Atari" (opracowany przez Jacka Tramiela-Trzmiela)
translator — program służący do translacji (tłumaczenia) programów źródłowych napisanych w języku programowania (na ogół „wyższego poziomu") na język komputera, **t.** języków programowania zorientowanych (Algol, Fortran, Cobol, PL 1 itp.) nazywa się również kompilatorami, **t.** przeznaczone do maszynowego („automatycznego") programowania OSN nazywa się procesorami. Program po przetłumaczeniu nazywa się programem wynikowym
A. translator
N. Übersetzer (m)
R. транслятор
translator asemblera → asembler
A. assembler, assembly program
N. Assembler (m), Assemblierer (m)
R. транслятор
transmisja danych, teledacja — jest to przenoszenie danych za pomocą → systemów przenoszenia
A. data transmision
N. Datenübertragung
R. передача данных
transmisja dupleksowa → transmisja jednoczesna
transmisja jednoczesna — jednoczesne pełnodupleksowe przesyłanie i odbieranie danych, tzn. po obu stronach pracuje jednocześnie i nadajnik i odbiornik (łącze czteroprzewodowe), natomiast w transmisji naprzemiennej (półdupleksowej) pracuje zawsze na przemian odbiornik i nadajnik
A. duplex, duplex operation
N. Duplex, Vollduplexbetrieb (m)
R. дуплексная передача
transmisja półdupleksowa — patrz transmisja jednoczesna
A. half duplex operation
N. Halbduplexbetrieb (m)
R.

transputer — szybkie 32-bitowe, µP z ROM i RAM w jednym układzie, dające się łączyć w równoległe struktury wieloprocesorowe, o krótkiej liście rozkazów, RISC, których duża część jest realizowana sprzętowo (nazwa TRANSPUTER powstała z połączenia ang. słów TRANSistor — comPUTER)
TSIC — CTSIC — Combinational Test Station (for) Integrated Circuit — uniwersalna stacja testowania obwodów scalonych



UART — Universal Asynchronous Receiver Transmitter — uniwersalny moduł lub narzędzie we/wy do asynchronizacji seryjnej transmisji danych
ujęcie danych — część urządzenia do przetwarzania lub teletransmisji danych, które może sygnały danych przyjmować
A. data sink
N. Datensenke (f)
R. приемник данных
układ elementarny — zbudowane z elementów podstawowych proste układy logiczne, jednostki przełączające (elementy przełączające) i pamięciowe, służące jako zespoły (moduły) w budowie układów sterujących, komputerów itp.
A. elementarny circuit
N. elementare Schaltung (f), Baustein (m), elementaler Schaltkreis (m)
R. элементарная система
układ elektroniczny — układ, którego działanie jest oparte na wykorzystaniu właściwości elektronicznych elementów podstawowych
A. electronic system
N. elektronisches System (n)
R. электронная система
układ funkcjonalny — układ realizujący określoną funkcję
A. functional — block device, functional assembly
N. Funktionsblock (m), Funktionsbaugruppe (f)
R. функциональный блок
uklad hybrydowy — 1. układ (lub cały moduł) wykonany zarówno z układów scalonych, jak i elementów podstawowych; 2. układ (moduł) analogowo-cyfrowy
A. 1. hybrid technology, 2. hybrid circuit (unit)
N. 1. Hybridtechnologie (f), Hybridtechnik (f), 2. Hybridschaltung (-einheit) (f)
R. 1. гибридная техника 2. гибридная схема
układ logiczny — układ zbudowany z elementów logicznych i pomocniczych, realizujący funkcje logiczne
A. logical circuit
N. logische Schaltung (f)
R. логиская система
układ przełączający — układ zbudowany z elementów przełączających; **u.p.** dzielą się na kombinacyjne i sekwencyjne (pamięciowe), których stan wyjść jest zależny nie tylko od stanu wejść, ale i od stanu wewnętrznego; **u.p.** należą do ogółu układów logicznych o działaniu dyskretnym
A. switching circuit (element)
N. Schaltwerk (n), Schaltglied (m)
R. переключающая схема
układ sprzęgający → złącze
układ scalony, us — układ elektroniczny, którego elementy i połączenia są wytworzone wewnątrz i/lub na wspólnym podłożu w sposób nierozdzielny, w mikroelektronice (mikroprocesory, mikropamięci) są to monolityczne → **u.s.** półprzewodnikowe, zawierające elementy podstawowe czynne (bipolarne lub unipolarne) i bierne; **u.s** dzielą się według wielu kryteriów (funkcjonalnych, technologicznych, wielkości integracji). Powyższe ujęcie **u.s.** odnosi się do cyfrowych układów scalonych, w pojęciu ogólnym mogą to być również układy analogowe i hybrydowe
A. integrated circuit, IC
N. integrierte Schaltung (f), integrierter Schaltkreis (m)
R. интегральиая схема

układ scalony półprzewodnikowy — układ scalony wytworzony w płytce półprzewodnikowej technologią epiplanarną (planarną) — nakładaniem lub wytrawianiem warstw i dyfuzyjnym domieszkowaniem
A. semiconductor integrated circuit
N. integrierte Halbleiterschaltung (f)
R. полупроводниковая интегральная схема
układ sterujący → jednostka sterująca, również popularna nazwa części sterującej → układu sterowania automatycznego
układ sterowania automatycznego — zespół funkcjonalny składający się z obiektu sterowania i oddziałującego na niego automatycznego (o programie stałym lub reprogramowalnym) układu sterującego
A. automatic control unit (circuit)
N. automatischer Steuerungskreis (m)
R. схема автоматического управления

układ złączowy — na ogół układ mikroprocesorowy o budowie modułowej (wymienne moduły) lub programowany, zwany również złączem inteligentnym, umożliwiający współpracę mikrokomputera z urządzeniami peryferyjnymi i (lub siecią teledacji) → modem
A. PIA — peripheral interface adapter
PIO — Programmable Input-Output-chip
ACIA — asynchronous communication interface adapter
UART — Universal Asynchronous Receiver (Transmiter)
GPIB — General Propouse Interface Base — znany pod symbolem IEFE — 488, a w Europie jako IEC-625, wykorzystywany głównie w automatyce pomiarów
N.
R.
układy cyfrowe — układy zbudowane z elementów podstawowych lub układów elementarnych wykonane w postaci scalonej lub hybrydowej, wykonujące operacje logiczne na zbiorach sygnałów dyskretnych dwuwartościowych (binarnych). Ze względu na wykonywane funkcje dzielą się na kombinacyjne i sekwencyjne, te zaś na asynchroniczne i synchroniczne → układy numeryczne
A. digital circuit
N. Digitalschaltung (f)
R. цифровые схемы
układy dyskretne → układy cyfrowe, termin **u.d.** używa się często (a bezzasadnie) do określenia ogółu → elementów podstawowych (przeciwstawnie do układów scalonych)
układy numeryczne — układy sterowania cyfrowego (lub analogowo-cyfrowego, AC, CA). Nazwę „numeryczne" stosuje się głównie do systemów sterowania obrabiarek (dawniej zwane sterowaniem liczbowym), patrz też → układy przełączające, układy scalone

A. numerical circuit
N. numerische Schaltkreise (f)
R. схема (система) числовного управления
upakowany — 1. sposób zapisu danych (patrz też: format upakowany); 2. sposób lub stan zagęszczenia elementów w urządzeniu (patrz też: gęstość upakowania)
A. packed
N. gepackt
R. упакованный

uruchamianie (programu) z usuwaniem błędów — sprawdzanie poprawności działania programu z kolejnym usuwaniem wykrytych w nim błędów, może się również odnosić do uruchamiania sprzętu
A. debug, debugging
N. debugging, „entwanzen"
R. отладка программы, отлаживать

urządzenie dialogowe — urządzenie do komunikacji z komputerem przy pracy konwersacyjnej
A. interactive terminal
N. Dialoggerät (n)
R. диалоговой терминал
urządzenie do programowania (programujące) urządzenie do programowalnych i reprogramowalnych pamięci typu „tylko odczyt"; kasowanie uprzedniej zawartości jest dokonywane za pomocą promieni X, promieni ultrafioletowych lub na drodze elektrycznej przez tzw. lawinowe wstrzykiwanie dziur. Zapis dokonywany jest z „MASTER-PROM" metodą duplikowania tzw. → ładowaczem pamięci lub z klawiatury **u.p.**
A. data I/O programmer, programming device, PROM programmer
N. Programmiergerät (n), PROM — Programmiergerät (n)
R. программер

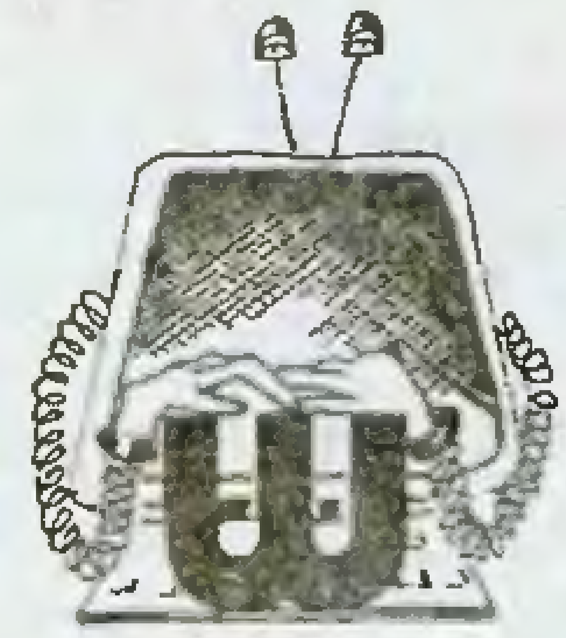
urządzenie końcowe — urządzenie przeznaczone do kontaktowania się z komputerem z oddalonego miejsca
A. terminal
N. Terminal (n), Datenendgerät (n)
R. терминал
urządzenie liczące → arytmometr
urządzenie peryferyjne — takie, które współpracuje z komputerem, lecz nie stanowi jego integralnej części
A. peripheral, peripheral equipment (unit, device)
N. Peripheriegerät (n), Anschlussgerät (n), Anschlusseinheit (f)
R. внешнее устройство, периферийное устройство

urządzenie rysujące, rejestrator XY — urządzenie peryferyjne umożliwiające wyprowadzenie wyników w postaci punktów, krzywych (wykresów) i rysunków
A. plotter, digital plotter, graph plotter, graphical device
N. Plotter (m), Zeichengerät (n), Digitalplotter (m)
R. самописец, графопостроитель, координатор
urządzenie sterujące → jednostka sterująca
urządzenie wejścia/wyjścia — 1. urządzenie do wprowadzania i wyprowadzania danych; 2. określenie ogółu składników związanych z we/wy danych
A. i/o, input/output (unit)
N. Ein — (Ausgabe) (f)
R. устройство ввода — вывода
urządzenie zewnętrzne → urządzenie peryferyjne
urządzenie zwrotne → urządzenie wejścia/wyjścia
U.S — układ scalony i → układ sterowania
USART — Universal Synchronous Receiver/Transmitter — uniwersalny moduł (lub urządzenie) we/wy do synchronicznej seryjnej transmisji danych
używalność wielokrotna → wieloużywalność



VGA — Video Graphies Array — układ sterowania grafiką (sterownik monitora)
VHSIC — Very High Speed Integrated Circuit — układy scalone bardzo szybkiego działania

VLSI — Very Large Scale Integration — bardzo wielka skala integracji układów scalonych

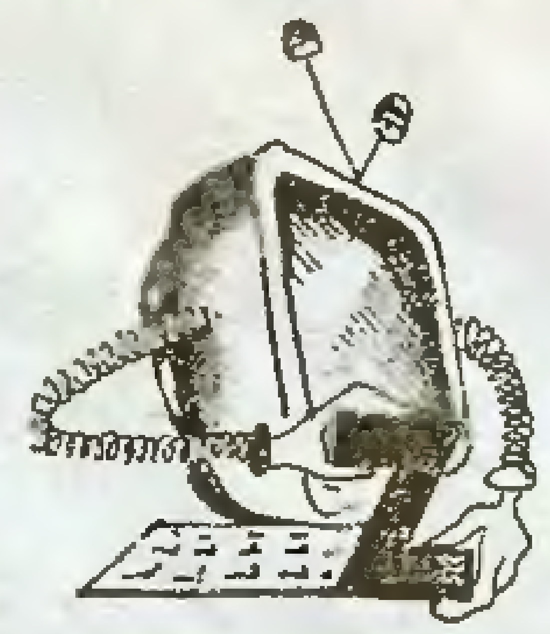


wariant — 1. warunek (wariant) w instrukcji, z którego może skorzystać programista; 2. możliwość otrzymania (zakupu) dodatkowego wyposażenia, którego normalny zakres dostawy nie obejmuje
A. option
N. Option (f), 1. Auswahlbedingung (f), 2. Zusatz (m)
R. выбор, вариант
wartości standardowe → dane standardowe
wejście — wyrażenie „input" odnosi się do takich pojęć, jak: 1. urządzenie techniczne wejścia (wprowadzania) danych; 2. czynność wyprowadzania danych; 3. dane lub sygnały wprowadzane
we/wy → urządzenia wejścia/wyjścia
wieloprzetwarzanie — system wieloprzetwarzania
A. multiprocessing, paraller processing
N. Multiprozessing (n), Mehrrechnerverarbeitung (f)
R. многократная (параллельная) обработка
wieloprogramowość → przetwarzanie wieloprogramowe
A. multiprogramming
N. Multiprogramming (f), Mehrprogrammbetrieb (m)
R. мультипрограммирование
wieloużywalność — możliwość wielokrotnego szeregowego użycia programu lub podprogramu przy przetwarzaniu wieloprogramowym, bez potrzeby jego ponownego ładowania
A. reusability
N. reusability
R. повторно используемость

WIMP — Windows, Icons, Mouse (Mice) and Pull-down menus — okna, ikony, mysz i zwijane menu — elementy systemów operacyjnych (porównaj GEM) ułatwiających obsługę mikrokomputerów (tzw. systemy przyjazno-życzliwe)
WS Word Star — edytor tekstów
wskaźnik pozycji — migający punkt lub przecinek na monitorze ekranowym
A. cursor
N. Kursor (m), Positionsanzeiger (m)
R. курсор
wskaźnik stanu → flaga
wskaźnik stosu — rejestr, w którym znajduje się adres słowa stosu (ostatni zapisany lub w trakcie przetwarzania kolejny nie przykryty)
A. stack pointer
N. „Stapel" — Zeiger (m), Kellerspeicherzeiger (m)
R.
wprowadzanie danych — umieszczanie danych w pamięci wewnętrznej

A. data input
N. Dateneingabe (f)
R. ввод данных
współużywalność — możliwość wielokrotnego, jednoczesnego (reenterable) użycia programu lub podprogramu
A. reentry
N. reentry
R. реентерабельнный
wsuwka — popularna nazwa wsuwanej karty (płyty), pakietu lub kasety stosowanych w budowie modułowej
A. slide-in chassis
N. Einschub (m)
R. сьемный (сменный) блок
wyciąg zawartości pamięci — kopią zawartości pamięci
wyjście — wyrażenie „output" odnosi się do wielu pojęć i w kontekście może mieć bardzo różne znaczenie dające się sklasyfikować jako: 1. urządzenia techniczne wyjścia danych (sygnałów); 2. czynność wyprowadzania danych; 3. dane (sygnały) po wyprowadzeniu (wyjściu)
A. output
N. Ausgabe (f)
R. выход
wyjście — np. wyjście z pętli, tzn. warunki przerwania pętli
A. exit
N. Ausgang (m), Ausgabe (f)
R. выход, вывод
wykonanie rozkazu — odcinek cyklu maszynowego, w trakcie którego wykonywany jest rozkaz
A. instruction execution
N. Befehlsausführung (f)
R. выполнение команды
wymiana danych – przenoszenie danych między dwoma miejscami naprzemiennie
A. data exchange
N. Datenaustausch (m)
R. обмен данными
wymienność — urządzenia, nośniki danych i oprogramowanie, które bez procedur dostosowujących mogą być między sobą wymieniane albo mogą ze sobą współdziałać, nazywane są wymiennymi, czyli mają one cechę wymienności
A. compatibility
N. Kompatibilität (f)
R. совместимость
wymienność oprogramowania — wymienność dotycząca tylko oprogramowania
A. software compatibility
N. Software — Kompatibilität (f)
R. софтвар (компатибильность) сменяемый
WYSIWYG — What You See Is What You Get — co widzisz, to otrzymasz, czyli wierne odzwierciedlenie treści z ekranu monitora
wywołanie — rozkaz lub sekwencja rozkazów do zainicjowania w komputerze określonej funkcji, np. przerwania obróbki programu i rozpoczęcia podprogramu
A. call
N. Aufruf (m)
R. вызов
wywołanie makro — instrukcja w języku typu asembler powodująca zastąpienie przez asembler — makro definicji — makro odpowiednią sekwencją rozkazów
A. makro call
N. Makroaufruf (m)
R. вызов макро

wywołanie podprogramu – wywołanie
A. subroutine call
N. Unterprogrammaufruf (m)
R. бызов подпрограммы
wywołanie rozkazu — faza cyklu wykonania rozkazu, w czasie której następny rozkaz jest przenoszony z pamięci programu do jednostki sterującej
A. instruction fetch (look a head)
N. Befehlsabruf (-phase) (m), „Befehl hole"
R. вызов команды



zadanie — 1. sekwencja programów do opracowania zdefiniowanego zadania (np. wykonanie list płac); 2. każdy z programów użytkowych przewidziany do realizacji na komputerze — stanowi dla komputera zadanie (pracę)
A. job, fask
N. 1. Auftrag (m), 2. Arbeit (f), Stellung (f)
R. задание
zagęszczenie → pakowanie danych
zakładkowe wykonywanie instrukcji — w czasie jednego cyklu zegarowego mikroprocesor może jednocześnie wykonywać mikroinstrukcję i zinterpretować następną
A. pipelining
N. Überloppung (f)
R.
zapis dziesiętny — zapis utworzony przy zastosowaniu dziesięciu cyfr od 0 do 9 (o podstawie 10), umieszczonych na pozycjach, którym są przyporządkowane kolejne potęgi liczby 10
A. decimal notation
N. Dezimaldarstellung (f)
R. десятичное представление
zapis szesnastkowy — zapis systemu liczbowego o podstawie 16 przy wykorzystaniu 16 cyfr 0...9, A = 10, B = 11, C = 12, D = 13, E = 14, F = 15 umieszczonych na pozycjach, którym przyporządkowane są kolejne potęgi liczby 16
A. hexadecimal
N. Hexadezimaldarstellung (f), Sedezimaldarstellung (f)
R. представление чисел в шестнадцатеричний
zapytywanie — za pomocą programu zapytywania można przez urządzenia konwersacyjne (monitor) sprawdzić aktualny stan komputera lub jego poszczególnych urządzeń (bajt stanu) lub wywołać pożądane informacje o wynikach przetwarzania zawarte w pamięci
A. interrogation
N. Abfrage (f)
R. спрос
zarządzanie danymi — ogólne pojęcie do określenia czynności systemu operacyjnego i programów systemowych, które dotyczą dostępu do zbioru danych, a więc wprowadzania i wyprowadzania danych, ich przechowywania i zabezpieczenia
A. data management
N. Datenmanagement (n), Datenverwaltung (f)
R. управление данными
zawieszenie — przerwanie nieprzewidziane lub niepożądane w wykonywaniu programu spowodowane nieprawidłowym zapisem instrukcji lub danych, zawieszeniem programu w pętli bez wyjścia („zapętlicowanie") itp.

A. hang-up
N. Aufhängen (n)
R. неожиданная остановка

zbieranie danych — działania w celu zebrania danych z kilku miejsc lub nośników w jedno miejsce lub nośnik
A. data collection, data gethering
N. Datenerfassung (f)
R. набор данных, запись данных

zbiór danych — uporządkowany zbiór danych w pamięci zewnętrznej
A. data set, data file
N. Datei (f), Datensatz (m)
R. сбор данных

zbiór rozkazów → repertuar rozkazów

zbiór znaków — zestaw znaków, umowna skończona liczba znaków
A. character set
N. Zeichenvorrat (m), Zeichnsatz (m)
R.

zegar — zespół funkcjonalny (na ogół → generator zegarowy z licznikiem), który emituje bezwzględne, względne lub przyrostowe sygnały czasowe; niezbędny w synchronicznych układach sekwencyjnych
A. clock, timing (signal) generator timer
N. Zeit (impuls) geber (m), Taktgeber (m)
R. программный регулятор

zegar przerwań — zegar (generator impulsów z licznikiem) inicjujący przerwanie w określonym czasie

zero nieznaczące — zera znajdujące się na najbardziej znaczących pozycjach liczby, wypełniają one puste miejsca przed liczbą i nie mają znaczenia
A. leading zero, left, hand zeroes
N. führende Null (f)
R. ведущие нули. нули в старших разрядах

zespół znaków — grupa znaków lub bitów w postaci ciągu, który może być traktowany jako jednostka
A. string
N. Zeichenfolge (f)
R.

zespół sterowania → jednostka sterująca

zgodność → wymienność

złącze, interfejs — 1. sprzętowo: a) układ elektroniczny (lub jego właściwości), który dwa urządzenia lub zespoły w miejscu połączeń do siebie dostosowuje, b) oznaczenie sygnałów, które przez przewody złączowe są wymieniane; 2. w dziedzinie oprogramowania: obszar pamięci lub rejestrów, z którego korzystają dwa lub więcej programów
A. interface
N. Interface (n), Schnittstelle (f), Nahtstelle (f)
R. интерфейс

złącze inteligentne → układ towarzyszący złączowy

zmienna — dowolny element określonego zbioru
A. variable
N. Variable (f)
R. переменная (величина)

znak — element zestawu znaków (litery, cyfry, znaki, interpunkcje, znaki sterowania itp.)
A. character
N. Zeichen (n)
R. знак

znak pusty — znak wypełniający bez znaczenia
A. dummy
N. blind, Attrape (f)
R. фиктивный знак

znak sterujący — znak, który w przesyłanych danych pełni funkcje sterujące
A. control character
N. Steuerzeichen (n)
R. знак управления

znacznik rozgraniczający — jakikolwiek przyjęty znak specjalny lub ich kombinacja do rozdzielenia nazw stałych, rozkazów lub danych
A. delimiter, limiting mark
N. Begrenzungszeichen (n), Begrenzungamarke (f)
R. ограничитель

źródło — wyrażenie „source" jest stosowane przez niektórych autorów w dość szerokim (dowolnym) znaczeniu, jak np: 1. źródło danych; 2. adres, pod którym znajdują się dane, które należy zastosować (przynieść); 3. program źródłowy
A. source
N. Quelle (f)
R. источник информации

żądanie — 1. żądanie komputera w stosunku do obsługi (wypisywane na ogół na monitorze) lub 2. żądanie jako sygnał urządzeń w stosunku do siebie (np. urządzenie wejścia do jednostki centralnej, przydział miejsca w pamięci)
A. request
N. Anforderung (f)
R. запас, требование

żądanie przerwania — sygnał z jednostki peryferyjnej do mikrokomputera (komputera) żądającej przerwania
A. interrupt request
N. Interrupt Anforderung (f)
R. требование прерывания



Repr. J. Zelman

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Werszawa ul. Grzybowska 77. telefon centrali 20-12-61 w. 486. Telex 313664.
Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prewo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zem. 2107. Nr ind. 396281. A-47